Engineering Educators Bringing the World Together
2025 ASEE Annual Conference & Exposition
Palais des congrès de Montréal, Montréal, QC • June 22–25, 2025
ASEE
Paper ID #48820

# BOARD #111: WIP: Predictive Wear Balancing and Approximation for Efficient Non-Volatile Main Memory Management

**Dr. Marjan Asadinia, California State University, Northridge**

Marjan Asadinia is an assistant professor in the Computer Science department at California State University, Northridge, where she leads the MOCCA Lab (Modern Computing and Communication Architectures Lab). Her research interests include machine learning, deep learning, advanced computer architecture, and processing in memory/near memory.

**Dr. Sherrene Bogle, California Polytechnic State University Humboldt**

Dr. Sherrene Bogle is a Fulbright Scholar and alumna of the University of Georgia, USA, where she earned her PhD in Computer Science. She is currently an Associate Professor of Computer Science and Program Lead for the BS Software Engineering at Cal Poly Humboldt. Dr. Bogle has a passion for sharing and helping students to improve the quality of their lives through education, motivation and technology. She has published two book chapters, two journal articles and several peer reviewed conference papers in the areas of Machine Learning, Time Series Predictions, Predictive Analytics, Multimedia in Education and E-Learning Technologies.

**Rowena Quinn**

# WIP: Predictive Wear Balancing and Approximation for Efficient Non-Volatile Main Memory Management

Rowena Quinn[1], Sherrene Bogle[1], and Marjan Asadinia[2]

[1]Department of Computer Science, California State Polytechnic University, Humboldt, USA, Rowena.Quinn@humboldt.edu, Sherrene.Bogle@humboldt.edu

[2]Department of Computer Science, California State University, Northridge, USA, marjan.asadinia@csun.edu

**Abstract**

Phase Change Memory (PCM) is an emerging non-volatile memory technology that leverages the thermal properties of chalcogenide glass to transition between amorphous and crystalline states, enabling scalable data storage. PCM's non-volatility, low power leakage, and cost-efficient read operations position it as a promising alternative to Dynamic Random Access Memory (DRAM). However, high energy requirements for write operations and limited endurance ($10^7$ to $10^9$ write cycles) present challenges to its adoption as a primary memory solution. To address these challenges, we propose a novel data approximation and retrieval framework to reduce energy consumption and improve memory endurance in PCM systems. In multi-level cell applications, where intermediary resistance levels represent multiple bits, our approach employs a Neural Network (NN) model to classify the energy level of incoming data and applies dynamic approximation techniques to minimize high-energy bit patterns. High and medium energy levels are approximated by analyzing data patterns and avoiding writes that require high-resistance states. To preserve data integrity, a flag byte is inserted into the least significant byte of approximated data, enabling a Convolutional Neural Network (CNN) to reconstruct the original data during retrieval. This combination of data approximation and retrieval significantly reduces write energy, prolongs the operational lifespan of PCM, and maintains reliable system performance. Our method overcomes critical impediments to PCM adoption, particularly for applications where energy efficiency and data reliability are paramount. By integrating intelligent approximation and retrieval strategies, this work lays the foundation for PCM to serve as a scalable and dependable main memory technology.

## 1 Introduction

Phase Change Memory (PCM) is an emerging technology that leverages the thermal properties of chalcogenide glass to transition between a high-resistivity amorphous state and a low-resistivity crystalline state in order to store data in computer memory. Since PCM cells are able to achieve different levels of resistivity, each cell has the capability to represent more than one bit, resulting in the potential for significant scalability [1-5]. This attribute, coupled with non-volatility, low power leakage, and low cost to perform read operations, positions PCM as an attractive alternative to other current main memory technologies that rely on capacitors like Dynamic Random Access Memory (DRAM).

In PCM single-level cell application, a RESET write cycle applies a high-amplitude, short-width current pulse to achieve a high-resistivity amorphous state, which represents a binary '0'. A SET write cycle applies a low-amplitude, longer current pulse which achieves a low-resistivity crystalline state, which represents a binary '1'. In multi-level cell application, by changing the amplitude of write current, the chalcegonide material can be stabilized at intermediary levels of amorphousness with distinct resistances, with each resistance representing a different pattern of binary [6-9].

These write operations require a significant amount of energy to achieve the highest resistance states, which presents a drawback to using PCM as a main memory solution. The energy requirement for write cycles can exacerbate wear on PCM cells as well, with current PCM lifetime only enduring from $10^7$ to $10^9$ write cycles before losing the ability to transition states [3]. In applications with frequent write cycles, these drawbacks make PCM costly and present a question of dependability.

In order to position PCM as a viable alternative to current main-memory solutions, steps need to be taken to ameliorate the energy cost of writing data. Our paper presents an innovative data approximation and retrieval method that utilizes both Neural Network (NN) and Convolutional Neural Network (CNN) models to lower the energy required in multi-level cell operation of PCM. The NN model classifies the energy level of incoming data, after which high and medium level energy will be approximated by examining the pattern of the data and avoiding writing high-energy bit combinations. Based on the pattern used to approximate the data, a flag byte will be inserted into the least significant byte, which will be used by the CNN to retrieve the original data after reading the approximated data. By avoiding writing high energy bit patterns that require high-resistance states, the energy required to perform write cycles will be drastically reduced. This approximation method, paired with a CNN that will ameliorate errors introduced by the simplification of the written data, will overcome some of the impediments PCM faces.

This paper is sectioned into the following parts: Section 2 discusses works that are related to this paper, which is followed by the elaboration of our proposed method in Section 3. This section first details our data collection and pre-processing before discussing our energy classification model that labels incoming data as high, medium, or low energy. This segues into our data approximation method, which details the different binary patterns we examined, how they were approximated, and the flag byte that was attached to each. This is followed by a subsection on our de-approximation model, which exploits a CNN model's pattern recognition capabilities to retrieve the original pattern of the data that was approximated. This is followed by Section 4, which evaluates the effectiveness of our methodology. This is followed by Section 5 which concludes our work, and Section 6 which discusses future work that can be undertaken to improve our method.

## 2 Related Works

In recent years, several studies have explored the optimization of write cycles and power efficiency in Non-Volatile Memory (NVM), especially Phase Change Memory (PCM) using various machine learning techniques. Lim et al. [7] introduced a novel workload-aware and optimized write cycle management system for NVRAM. Their approach focused on improving endurance and performance by dynamically adjusting write strategies based on workload characteristics, which helps to distribute write operations more evenly across memory cells, thereby prolonging the lifespan of the memory and enhancing overall performance. Another significant contribution presented in [8] demonstrates the effectiveness of power-aware reinforcement learning in managing memory power consumption. Their method leverages reinforcement learning algorithms to predict and adjust power states dynamically, thus significantly reducing energy usage while maintaining performance. The proposed system learns from historical power consumption patterns and optimizes the power states accordingly, leading to substantial energy savings without compromising the performance of the memory system.

In [9], authors proposed a deep neural network model designed for accurate and efficient performance prediction in memory systems. Their model emphasizes the ability to handle complex memory access patterns and predict performance metrics with high accuracy. By utilizing a deep neural network, the model can capture intricate relationships within the memory access patterns, enabling more precise performance predictions and, consequently, better optimization strategies for memory management. The work by Chen et al. [10] highlighted the potential of distributed reinforcement learning for optimizing power consumption in large-scale memory systems. Their approach involves deploying reinforcement learning agents across a distributed memory architecture to collaboratively optimize power usage. This method not only improves power efficiency but also enhances computational performance by balancing the load and reducing bottlenecks in the memory system. The distributed nature of the solution ensures scalability and robustness, making it suitable for large-scale implementations.

Additionally, in [11], the authors present a technique to eliminate redundant writes, reducing energy use by up to 50% and doubling endurance. Their method shows minimal performance impact, enhancing PCM's reliability and efficiency for future systems. The modular reinforcement learning approach presented by Kim et al. [12] provides a flexible framework for integrating various reinforcement learning models to manage different aspects of memory systems. This approach allows for the simultaneous optimization of multiple parameters, such as power consumption, latency, and endurance, by utilizing specialized reinforcement learning agents for each task.

The self-optimizing memory controller in [13] demonstrates that by dynamically adjusting scheduling decisions based on past behaviors, memory access latencies can be significantly reduced, thereby improving overall system performance. For modern memory systems, [14] introduces Memory Cocktail Therapy (MCT), which leverages machine learning techniques and architectural insights to optimize memory system performance. By employing gradient boosting and quadratic lasso models, MCT dynamically selects the best memory management policy based on runtime conditions. This approach has been shown to significantly improve instructions per cycle (IPC), extend memory system lifetime, and reduce overall energy consumption compared to static policies, making it a highly effective method for modern memory systems.

Furthermore, reinforcement learning-based methods have also shown promise in the domain of adaptive caching and refresh optimization. These methods use reinforcement learning algorithms to make real-time decisions about cache management and refresh operations, learning from access patterns to minimize refreshes and improve cache hit rates. This adaptive approach allows for a more efficient and responsive memory system that can better handle the dynamic nature of workload demands [15], [16].

## 3 Proposed Method

Within this section, we will discuss our methodology. First, we will describe our data collection procedure before discussing the cleaning and preparation undertaken to prepare our data for model loading. We then move on to our energy classification method and model, which will then transition to our data approximation method. This section will then conclude with our model that employs a pattern-learning algorithm to de-approximate the memory data to its original form.

## 3.1 Data Collection

### 3.1.1 Trace Generation

We began with generating a trace file containing one million rows of synthetic memory data. Each row contained whether it is a read or write operation, the address the data is being written to, and data consisting of 512 bits of binary. Each trace's address consisted of a randomly generated 16 bit hexadecimal value, and each trace's data consisted of a randomly generated 512 bit binary value. For our purposes, we considered each trace's data as consisting of 256 bytes that contain 2 bits each. This would mimic operation of a two-level cell in PCM.

Throughout our methodology, we focus exclusively on the data portion of the traces. By randomly generating the binary, we ensured that we would have an even distribution of ratios of ones to zeroes in the traces, with traces containing anywhere between 0 to 100 percent zeroes. This also ensured an even distribution of patterns within the bytes of data, with an even amount of 11, 10, 00, and 01 patterns appearing throughout the traces' data. This would allow us to apply approximation to data of varying levels of energy required to write to memory.

### 3.1.2 Data Labeling

After generating the synthetic data, we separated each 512 bit binary data into 8 sections of 64 bits. By separating the data into sections, we could classify each section as high, medium, or low energy, which would allow for later approximation to be tailored to patterns found in only part of the data.

Once the data was separated, we calculated the percentage of zeroes within each section as well as the percentage of high energy level patterns. We determined the binary pattern 10 as the high energy level pattern we were looking for, as it requires the most energy out of any 2 bit binary pattern to write at 547 pJ of energy.

With these percentages, we labeled each section of data as either high, medium, or low energy data. If the data had between 40 and 60 percent zeroes or more than 25 percent high energy level patterns, then the data was classified as ”high”. If the data had between 25 and 40 percent zeroes, between 60 and 75 percent zeroes, or between 10 and 25 percent high energy pattern then it was classified as ”medium”. Any left over data was then classified as low energy data.

## 3.2 Energy Classification Model

### 3.2.1 Data Preprocessing

After labeling the data, we processed the data in preparation of loading into an energy classification model. We One-Hot-Encoded the energy labels to avoid ordinality when training the model. We then split the data into training and testing sets, with 80 percent of the data set aside for training and 20 percent devoted to testing.

After this, we scaled our continuous data which consisted of the percentage of zeroes and percentage of high energy patterns within each section of data. We used the MinMaxScaler from the scikit-learn library to ensure the these percentages were scaled to fall within a range between 0 and 1. Scaling took place after splitting the data into training and testing sets to ensure that the model would not overtrain on our synthetic data.

### 3.2.2 Model Structure

In order to determine an appropriate approximating scheme for each section of data, we have to classify each section as either high, medium, or low energy. To accomplish this, we created a model that can classify the energy of a section of binary data based on observable metrics. We determined that percentage of zeroes and percentage of high energy patterns would be the most useful in categorizing this data, as due to the physical properties of PCM, zeroes and the pattern 10 take the most energy to write. Based on this knowledge, we separated our data into our input features (percentage of zeroes and percentage of high energy level patterns) and our target classes (high, medium, or low energy). We then constructed a Multi-Layer Perceptron (MLP) model that could classify the sections of data based on their input features.

The MLP model required a unique structure that was able to take the features of all eight sections of the 512 bit data at once and output their individual classifications. This entailed building the structure of a multi-output feed-forward neural network utilizing the Functional API structure of the Keras library [17]. Each sections features and targets would be fed into the neural network before splitting off into their own individual feed-forward network. Each network would then output its classification of its section. This ensured that each section was trained only on its own features and was not classified using information from the other sections. Figure 1 displays the general structure of the model.

Since the model is a multi-class classification model, the categorical cross-entropy loss function was chosen. In order to ensure an accurate model, we utilized the Keras tuner to optimize the following hyperparameters: number of hidden layers, number of neurons per hidden layer, the L1 and L2 values of the kernel regularizer, batch size, and optimizer. The tuner was given different options to choose from for each parameter and ran trials with different combinations of each until the least possible validation loss was achieved. The tuner landed on the combinations of hyperparameters as shown in Table 1.

This structure was used for each data section's branch of the model, as the classification task given to each branch was the same.

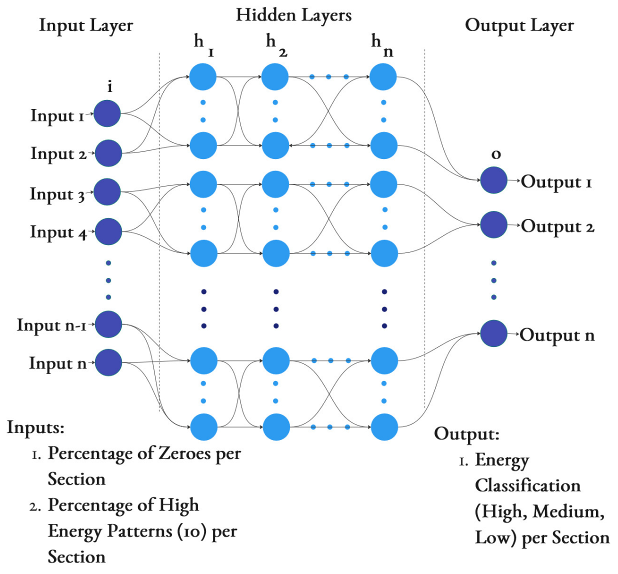
Input Layer
Hidden Layers
Output Layer
h 1
h 2
h n
i
o
Input 1
Input 2
Input 3
Input 4
Input n-1
Input n
Output 1
Output 2
Output n
Inputs:
1. Percentage of Zeroes per Section
2. Percentage of High Energy Patterns (10) per Section
Output:
1. Energy Classification (High, Medium, Low) per Section

Figure 1: MLP Neural Network structure with input, hidden, and output layers.

| **Hyperparameter** | **Value Chosen** |
|---|---|
| Optimizer | Adam |
| Number of Hidden Layers | 6 |
| Neurons per Layer | 16,36,8,20,20,24 |
| L1 Weight | 0.001 |
| L2 Weight | 0.001 |
| Batch Size | 128 |

Table 1: Neural Network Tuned Structure

## 3.3 Data Approximation Method

After the model classified the data, we moved on to approximating high and medium energy level data in order to reduce the amount of energy required to write this data. If the energy level of the data was already low, we would avoid approximation as this would be redundant and introduce unneeded error into the data.

With the knowledge that we would need to de-approximate the data when retrieving it from storage, we were careful to narrow our approximation methods to four unique patterns. With only four patterns, we could insert a flag byte in place of the least significant byte of each section that would signify the pattern used to approximate it later. This would allow our de-approximation model additional context with which to train itself.

### 3.3.1 First Pattern

The first pattern we examined was data that contained mostly zeroes. While it does take more energy to write a zero than it takes to write a one in PCM, in 2-bit level cells the pattern ”00” is the second lowest data pattern to write. This means that if the data was entirely zeroes, the energy level would still be relatively low when compared to data that has ones mixed throughout it. This narrowed our field to look at data that has between 60 and 75 percent zeroes, which is a range that contains data with enough ones to spike the energy higher while still being mostly zeroes.

With the understanding that writing an all zero pattern takes less energy than one mixed with ones, we then flipped all of the one bits to zeroes. This would drastically reduce the energy level of the data while changing as little data as possible. In order to ensure that our model would understand which pattern the original data followed, we then inserted the pattern ”00” in the least significant byte as our flag byte.

### 3.3.2 Second Pattern

The second pattern we tackled, similarly to the first, entailed data that contained mostly ones. Due to similar logic as the first pattern, we know that an all one pattern takes the lowest amount of energy to write, as the pattern ”11” in 2-bit level cells takes the least amount of energy to write. This means that we focused on a range of 25 to 40 percent zeroes contained within the data pattern to approximate. This data would contain enough zeroes to spike the energy while still being mostly ones.

With this pattern, we performed the opposite operation as the first by flipping all of the zero bits to ones. This also achieves a drastic reduction of energy while performing minimal modifications. We then inserted the pattern ”11” in place of the least significant byte as our flag byte.

### 3.3.3 Third Pattern

The third pattern we examined was data containing an even split of ones and zeroes, specifically focusing on data that fell between 40 and 60 percent zeroes. This data had the potential to be the highest energy data possible, as it would potentially contain the highest number of ”10” and ”01” patterns which take the most energy to write.

When approximating this data, we converted every ”10” pattern to ”11” and every ”01” pattern to ”00”, which would convert the highest energy patterns to the lowest energy patterns. We then inserted a ”01” pattern in the least significant byte as our flag byte.

### 3.3.4 Fourth Pattern

The final pattern examined a pattern of data that does not follow the previous patterns but does have a significant enough amount of ”10” energy patterns to make it medium or high energy. We flipped all ”10” patterns within this data to be ”11”, the lowest energy pattern. We then inserted a ”10” pattern in the least significant byte as our flag byte.

## 3.4 De-Approximation Model

Once we had approximated data, we needed to be able to retrieve what the data originally was after reading it from memory. To accomplish this, we designed a Convolutional Neural Network (CNN) that treats the sections of binary data as a one dimensional time series. This method ensures that the model can identify important patterns, or features, in the data when applying filters across it. The model would treat the process as classifying each approximated byte as its original byte pattern, with the four possible classes being ”10”, ”11”, ”01”, and ”00”.

### 3.4.1 Data Preprocessing

In a real-world use case, when retrieving the data from memory the model would only have the approximated data itself. This means that we have to rely only on the string of data and the flag bytes to teach the model what the data originally was. We first separated the original sections of data and approximated sections of data into arrays of 2-bit bytes. Each byte inside the array is separated into a list of two bits. We then copied the flag bytes into separate arrays, which would become a training feature. Within the CNN, the model would take the array of approximated bytes and the flag bytes as the input features and the array of original bytes as the output target. The model would classify each individual approximated bit within each byte list as its original bit.

We then split the data into separate training and testing sets, with 80 percent of the data set aside for training and 20 percent of the data set aside for testing. Since each bit is already in binary format, we did not need to perform additional scaling or encoding.

#### 3.4.2 Model Structure

When designing the CNN we kept the structure simple, with two, one-dimensional, convolution layers with a one-dimensional max pooling layer following each. We then ran the data through a flatten layer before feeding it through a dense layer for classification. This was followed by dropout for regularization and then an output dense layer using the softmax activation for multi-class classification. The hyperparameters we chose for the model are displayed in Table 2, and the general structure of the CNN is displayed in Figure 2.

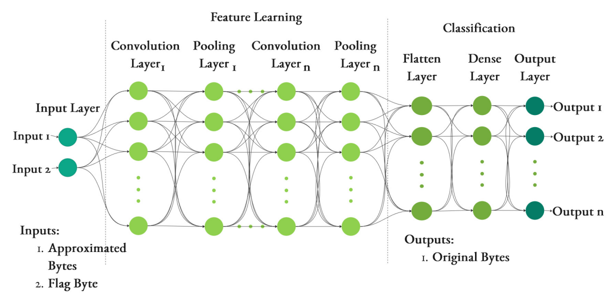

Figure 2: CNN Neural Network structure with input, feature learning, and classification layers.

| **Hyperparameter** | **Value Chosen** |
|---|---|
| Optimizer | Adam |
| Convolution Filters | 64, 128 |
| Neurons in Dense Layer | 128 |
| Dropout | 0.5 |
| Loss | Binary Crossentropy |
| Batch Size | 512 |

Table 2: CNN Tuned Structure

## 4 Evaluation

Throughout this section, we present the results of the evaluation of our energy classification model, approximation method, and de-approximation model.

### 4.1 Energy Classification Model Evaluation

Since this model was built for classification, in order to evaluate it we focused on precision, recall, and the F1 score of each section of binary fed into it. Precision in this case is the proportion of correct predictions of all predictions made. Recall is the measure of the ability of the model to capture true positives, or in this case, how many binary ones have been classified as binary ones. The F1 Score provides the harmonic mean of precision and recall, which provides a balanced view of the model that takes into account false positives as well as false negatives.

In Table 3, we have listed the average precision, recall, and F1 score of each section of binary data. Each section's value is averaged from the values calculated for the High, Medium, and Low classes of each section.

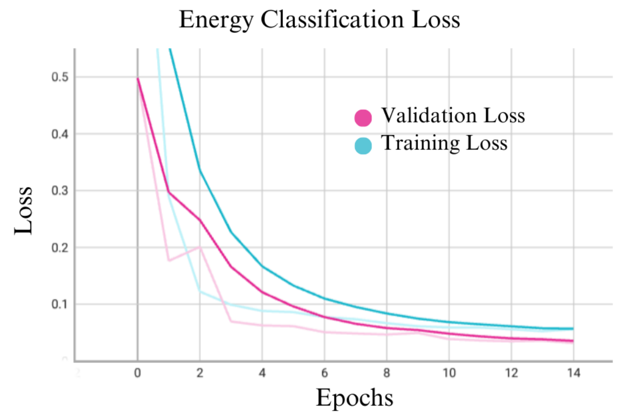

Figure 3: Validation and training loss of the model over epochs

| Data Section | Precision | Recall | F1 Score |
|---|---|---|---|
| 1 | 98% | 98% | 98% |
| 2 | 99% | 99% | 99% |
| 3 | 96% | 95% | 95% |
| 4 | 98% | 97% | 98% |
| 5 | 100% | 100% | 100% |
| 6 | 100% | 100% | 100% |
| 7 | 94% | 93% | 93% |
| 8 | 94% | 93% | 93% |

Table 3: Energy Classification Metrics

Overall, the precision, recall and F1 score of each data section was high, with an average precision of 97.39%, an average recall of 96.89%, and an average F1 score of 97.0%. This shows that our energy classification model was highly accurate.

In order to ensure the model did not overfit, we kept track of the training and validation loss of our model. In Figure 3 both are displayed over the epochs the model trained. Since both losses closely followed each other and reached below 0.05, the model did not overfit and trained equally well on the new data as the given data.

## 4.2 Data Approximation Method Evaluation

When evaluating the effectiveness of our data approximation, we focused on the amount of energy saved when writing the approximated data versus the original data. This was achieved by calculating the accumulative energy used to write one section of data by adding each pattern's energy requirement together. We used the values in Table 4 to calculate the total energy needed to write each section of data.

| Pattern | Energy (picoJoules) |
|---|---|
| 10 | 547 pJ |
| 01 | 307 pJ |
| 11 | 20 pJ |
| 00 | 36 pJ |

Table 4: Byte Pattern Write Energy Requirements

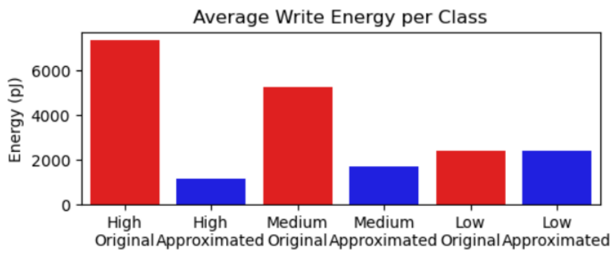

Figure 4: Average energy of data section per class before and after approximation.

After calculation, we took the average energy required to write High, Medium, and Low energy classes and compared the original data to the approximated data. Our results are shown in Figure 4. Our model reduced high energy data by an average of 84.5% and medium energy data by an average of 67.8%. With our method, the high and medium energy level data is reduced by a large amount while keeping low energy data at the same energy level.

## 4.3 De-Approximation Model Evaluation

Since the de-approximation method is a time-series classification model, the same evaluation will be performed as the energy classification model. We will focus on the precision, recall, and accuracy of each time step, which represents a byte, as shown in Table 5.

Our model was able to achieve an average precision of 69.28%, an average recall of 84.63%, and an average F1 score of 83.26% per byte. The last byte in the table are the flag byte, which were approximated without consideration to their actual value but with consideration to the overall pattern of the data section, which causes their lower scores in comparison to the other bytes.

Overall, the model's precision, recall, and accuracy at classifying ones and zeroes as their original value are shown in Table 6.

This model is highly effective at correctly identifying when data originally was a zero and moderately successful at correctly identifying data that was originally a one.

## 5 Conclusion

This work proposes a novel approach to approximating and then retrieving data written to storage in a PCM environment. By utilizing a neural network (NN) to classify the energy level of the data, approximating the data based on its pattern, and employing a convolutional neural network (CNN) to retrieve the data after its been written, we have created a comprehensive approach to memory approximation that greatly lowers the energy consumption of PCM. With a method that avoids writing high-energy patterned data, we greatly reduce the energy cost of utilizing PCM in computers over its lifetime.

Our findings show that NNs are highly accurate at classifying the energy level of incoming data when given minimal information on the characteristics of the data, such as the amount of zeroes and high energy patterns. The proposed approximation method, which avoids writing high energy patterns, was shown to be highly affective at reducing the energy consumed performing a write cycle in PCM. Our method achieved a significant reduction at an average of 84.5% in the highest energy data as well as a reduction of 67.8% in medium energy data, which promises a significant reduction in the energy required to use PCM over its lifetime. CNNs have also been shown to be effective at exploiting pattern recognition to correctly identify the original binary pattern of approximated data. Our model showed that it was highly effective at identifying data that was originally a zero and moderately successful with data that was originally a one. Per byte, the model was moderately successful at achieving the correct binary pattern of the data after approximation.

## 6 Future Work

While the Predictive Memory Wear Balancing (PMWB) framework has demonstrated significant potential in mitigating wear imbalance and extending the lifespan of non-volatile memory (NVM) systems, several areas warrant further exploration.

| **Byte** | **Precision** | **Recall** | **F1 Score** |
|---|---|---|---|
| 1 | 68.8% | 85.2% | 83.3% |
| 2 | 69.3% | 85.2% | 83.4% |
| 3 | 69.5% | 84.0% | 83.4% |
| 4 | 69.6% | 85.4% | 83.6% |
| 5 | 69.5% | 84.6% | 83.4% |
| 6 | 69.8% | 85.3% | 83.6% |
| 7 | 69.8% | 84.6% | 83.5% |
| 8 | 70.1% | 85.5% | 83.8% |
| 9 | 69.7% | 84.6% | 83.5% |
| 10 | 70.1% | 85.0% | 83.8% |
| 11 | 70.0% | 83.9% | 83.5% |
| 12 | 69.7% | 84.7% | 83.5% |
| 13 | 69.8% | 84.1% | 83.4% |
| 14 | 69.7% | 85.1% | 83.6% |
| 15 | 69.6% | 84.4% | 83.4% |
| 16 | 70.1% | 84.9% | 83.7% |
| 17 | 69.4% | 84.0% | 83.3% |
| 18 | 70.1% | 85.6% | 83.8% |
| 19 | 69.6% | 84.8% | 83.5% |
| 20 | 69.8% | 85.5% | 83.7% |
| 21 | 69.8% | 84.7% | 83.6% |
| 22 | 69.9% | 85.0% | 83.6% |
| 23 | 69.6% | 84.7% | 83.5% |
| 24 | 70.0% | 85.1% | 83.7% |
| 25 | 69.5% | 84.2% | 83.4% |
| 26 | 69.5% | 85.2% | 83.6% |
| 27 | 69.1% | 85.5% | 83.4% |
| 28 | 69.7% | 85.4% | 83.6% |
| 29 | 68.7% | 85.1% | 83.2% |
| 30 | 68.7% | 84.7% | 83.1% |
| 31 | 70.4% | 85.2% | 83.9% |
| 32 | 58.0% | 76.9% | 75.2% |

Table 5: Metrics per Byte

| **Bit** | **Precision** | **Recall** | **F1 Score** |
|---|---|---|---|
| 0 | 90.0% | 90.0% | 90.0% |
| 1 | 75.0% | 80.0% | 77.0% |

Table 6: Metrics per Bit

1. **Enhanced Wear Prediction Models:** Future research could focus on developing more sophisticated wear prediction algorithms by integrating advanced reinforcement learning techniques with additional contextual data, such as workload patterns and environmental conditions. This may improve the accuracy and robustness of wear-out predictions.

2. **Dynamic Workload Adaptation:** PMWB currently adapts to workload fluctuations dynamically, but further studies could investigate its performance under extreme workload variability or non-stationary environments. Extending the model to predict and accommodate sudden changes in data access patterns could enhance its applicability in diverse scenarios.

3. **Approximation Techniques for Varied Applications:** The integration of dynamic approximation tuning in PMWB shows promise for applications like multimedia processing. Future work could expand this capability by exploring application-specific approximation strategies tailored to diverse workloads, such as scientific computing or machine learning inference, where precision requirements vary.

4. **Energy Efficiency Optimization:** Additional research could explore ways to further minimize the energy consumption of the PMWB mechanism itself. This includes optimizing the reinforcement learning process and the computational overhead associated with monitoring, prediction, and redistribution.

5. **Scalability and Multi-Level Cell Support:** PMWB's approach can be extended to handle multi-level cell (MLC) memory with more granular resistance levels. Investigating its scalability to higher memory densities and its impact on system performance in such scenarios would provide valuable insights.

6. **Integration with Emerging Technologies:** Exploring the integration of PMWB with hybrid memory systems or emerging memory technologies, such as spin-transfer torque RAM (STT-RAM) or resistive RAM (ReRAM), could enhance its adaptability and broaden its applicability.

These future directions aim to further refine and expand the PMWB framework, ensuring its continued relevance and effectiveness in addressing challenges associated with NVM wear management and performance optimization.

## 7 Acknowledgment

This work was supported by the National Science Foundation (NSF) under Grant No. [2318553]. We gratefully acknowledge the NSF for their financial support 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] 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.

[2] R. Azevedo, J. D. Davis, K. Strauss, P. Gopalan, M. Manasse, and S. Yekhanin, “Zombie memory: Extending memory lifetime by reviving dead blocks,” in *Proceedings of the International Symposium on Computer Architecture* (ISCA), 2013.

[3] 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.

[4] 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*, pp. 433–444, ACM, 2013.

[5] 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, 2014.

[6] Zhou, Y., Zhang, W., Ma, E. et al. Device-scale atomistic modelling of phase-change memory materials. Nat Electron 6, 746–754 (2023). https://doi.org/10.1038/s41928-023-01030-x

[7] J. P. Shri Tharanyaa, D. Sharmila, and R. Saravana Kumar, “A Novel Workload-Aware and Optimized Write Cycles in NVRAM,” Computers, Materials & Continua, vol. 71, no. 2, pp. 2667–2681, 2022, doi: https://doi.org/10.32604/cmc.2022.019889.

[8] K. Fatemi and Masoud, “A Power-Aware Reinforcement Learning Technique for Memory Allocation in Real-time Embedded Systems,” Dec. 22AD, Available: https://prism.ucalgary.ca.

[9] M. Shi, P. Mo, and J. Liu, “Deep Neural Network for Accurate and Efficient Atomistic Modeling of Phase Change Memory,” IEEE Electron Device Letters, vol. 41, no. 3, pp. 365–368, Mar. 2020, doi: https://doi.org/10.1109/led.2020.2964779.

[10] Z. Chen and D. Marculescu, “Distributed Reinforcement Learning for Power Limited Many-Core System Performance Optimization,” Design, Automation & Test in Europe Conference & Exhibition (DATE), 2015, Jan. 2015, doi: https://doi.org/10.7873/date.2015.0992.

[11] 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.

[12] Z. Wang et al., “Modular Reinforcement Learning for self-adaptive Energy Efficiency Optimization in Multicore System,” Jan. 2017, doi: https://doi.org/10.1109/aspdac.2017.7858403.

[13] E. Ipek, O. Mutlu, J. F. Martínez, and R. Caruana, “Self-Optimizing Memory Controllers: A Reinforcement Learning Approach,” 2008 International Symposium on Computer Architecture, Jun. 2008, doi: https://doi.org/10.1109/isca.2008.21.

[14] Z. Deng, L. Zhang, N. Mishra, H. Ho!mann, and F. T. Chong, “Memory Cocktail Therapy: A General Learning-Based Framework to Optimize Dynamic Tradeoffs in NVMs,” 2017.

[15] S. Suman and H. K. Kapoor, “Reinforcement Learning Based Refresh Optimized Volatile STT-RAM Cache,” Jul. 2020.

[16] A. Sadeghi, F. Sheikholeslami, A. G. Marques, and G. B. Giannakis, “Reinforcement Learning for Adaptive Caching With Dynamic Storage Pricing,” IEEE Journal on Selected Areas in Communications, vol. 37, no. 10, pp. 2267–2281, Oct. 2019.

[17] Chollet, F., “keras,” GitHub, Jul. 07, 2021. https://github.com/fchollet/keras.