Project Report
Read and write operation prediction in memory traces

Zuzana Skubeňová

January 2025

1 Abstract

In this project, we aim to predict memory access operations (read/write) using machine learning models. We first preprocess a dataset (memory traces of addresses) and compute additional features such as address delta, access frequency, and so on.
Three models were evaluated: Random Forest Classifier, Neural Network, and Support Vector Machine (SVM). The Random Forest Classifier achieved the highest accuracy of 95.4%, while the Neural Network and SVM achieved accuracies 76.9% and 75.3%, respectively.

2 Introduction

The classification of read and write operations in memory traces is important for optimizing memory systems. In my master's thesis, I conduct research on analyzing memory patterns of addresses.

The dataset for this project consists of memory trace records parsed from the file "hw_echo_pinatrace.txt." This file is the output of the Intel Pin Tool, which uses the "pintrace.so" plugin to transparently capture memory accesses (read and write) performed by a program (in this case, the program is 'echo "Hello Machine learning world!"') during its execution. The data in `echo "Hello Machine learning world!"` contains 72 696 records (4.24 MB) with information such as memory addresses accessed, type of access (read/write), instruction pointer (address of the instruction causing the access), size of the access (in bytes), and the next address. We also compute additional features from this data to capture spatial, temporal, and access frequency patterns of memory accesses.

We also tested a C++ and Python program to print `"Hello Machine learning world!"`, but the logs of pinatrace were so large (36 MB and 785 MB, respectively, representing millions of addresses) that they were not usable within a reasonable time frame for processing such a huge dataset in our project.

We split our dataset into training and testing sets. The training set includes 80% of the data, while the remaining 20% is used for testing.

The dataset consists of the following features:

1. address
2. operation (read/write)
3. instruction pointer
4. size (in bytes)
5. next address
6. address delta (difference between the address and the preceding one)
7. access frequency (how often each memory address is accessed)
8. instruction pointer reuse (how often each instruction pointer is reused during program execution)
9. temporal gap (how many times a specific memory address has been accessed up to and including the current row)
10. spatial locality (addresses that are within the same page)
11. next address presence

3 Context

Predicting memory access patterns is a well-researched area with applications in memory management and system optimization. Previous work has used statistical and deep learning approaches to classify access operations. State-of-the-art methods often use Random Forests or Neural Networks due to their flexibility and performance in handling complex patterns.

4 Machine Learning Methods

4.1 Models

We used the following machine learning models:

- **Random Forest Classifier**: Used 100 estimators and a maximum depth of 20. This model provided the best performance.
- **Neural Network**: A model with three hidden layers (64, 32, and 16 neurons). Class weights were used to try to improve performance, but the results were still not good enough.

- **Support Vector Machine (SVM)**: SVM with a radial basis function (RBF) kernel, C=1, and gamma set to 'scale.' The results were similar to those of the previous Neural Network model, with still room for improvement.

4.2 Data Preparation

- Converted raw text data into a structured DataFrame.
- Normalized features using StandardScaler for Neural Network and SVM models.
- Applied `train_test_split` with an 80:20 ratio to create training and testing datasets.

5 Issues

During the project, we faced several issues. The first one was the shortage of time. At the beginning, we had more goals we wanted to achieve; however, we did not have enough time (or GPU resources) to complete everything we intended.

We also faced issues with setting the architecture of the neural network, such as optimizing the learning rate and determining the appropriate number of layers to achieve acceptable performance.

Another problem we encountered was class imbalance. We found that write operations were underrepresented, which affected our models. Class weighting partially helped with this issue.

Unexpectedly large memory addresses introduced another problem with overflows, which required handling the maximum possible value.

6 Experimental Evaluation

6.1 Random Forest Classifier

This model correctly predicts the operation (read or write) for approximately 95.4% of the total test cases. A precision of 0.97 indicates that predicting a read operation is correct 97% of the time. A recall of 0.97 means the model successfully identifies 97% of all read operations.

While slightly lower than the precision for read operations, predicting write operations was correct 92% of the time and identified 91% of all write operations.

```
Accuracy: 0.9540577716643741
Classification :
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     10524
           1       0.92      0.91      0.92      4016

    accuracy                           0.95     14540
   macro avg       0.94      0.94      0.94     14540
weighted avg       0.95      0.95      0.95     14540
```

Figure 1: RFC results

According to Figure 2, the feature `addr_delta` had the highest importance score among all features. Additionally, `spatial_locality` and `addr` had the lowest importance scores in our RFC model.

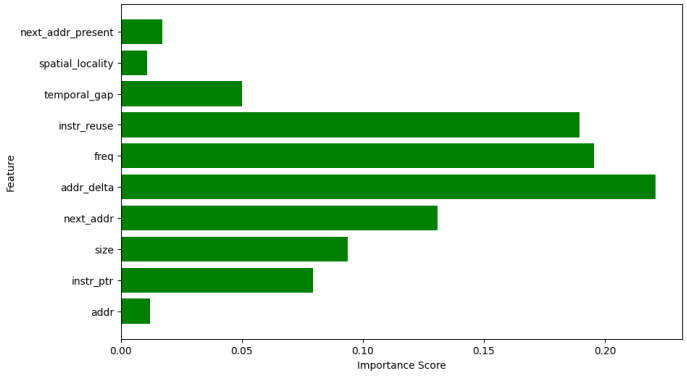

Figure 2: Feature Importance

From the learning curve in Figure 3, we can see that we achieved a quite good accuracy of approximately 95% relatively quickly.

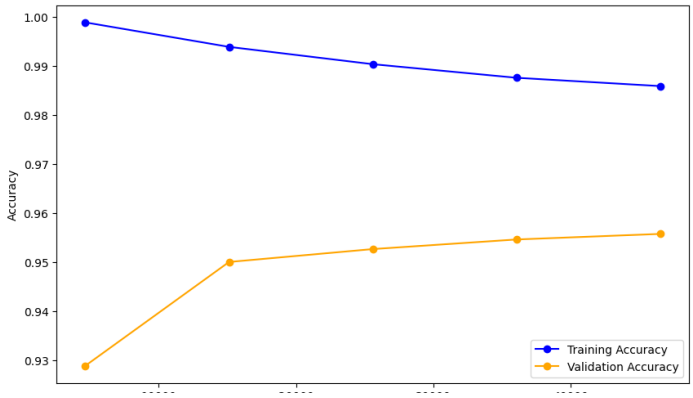

Figure 3: Learning Curve

This model demonstrated the highest effectiveness in solving our problem compared to all other models.

6.2 Neural Network (NN)

The neural network has undergone 10 epochs of training, and its performance on both the training and validation datasets was lower than that of the previous RFC model. The model starts with an accuracy of 69.1% in the first epoch and improves throughout the training, ending with 75.8% at epoch 10. Furthermore, the loss function decreases from 0.5282 to 0.4030 by the end of epoch 10.

From Figure 4, we can see that the model correctly identifies 95% of read operations. Additionally, we can observe that the precision for write operations was relatively low—only 55%, which means that many of the instances predicted as write were actually reads (false positives).

The model is able to correctly identify 72% of all actual read operations and 90% of write operations. As we mentioned previously in the Issues section, this could be due to the imbalanced dataset (the number of write operations was lower than the number of read operations).

```
Neural Network Accuracy: 0.7698074277854196
Neural Network Classification:
              precision    recall  f1-score   support

           0       0.95      0.72      0.82     10524
           1       0.55      0.90      0.68      4016

    accuracy                           0.77     14540
   macro avg       0.75      0.81      0.75     14540
weighted avg       0.84      0.77      0.78     14540
```

Figure 4: NN results

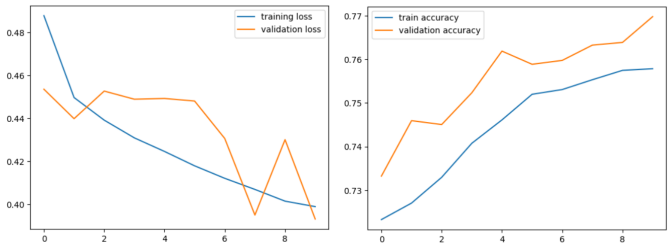

Figure 5: NN loss and accuracy

The model achieves a good overall accuracy of 75.7%, which indicates it is reasonably effective in classifying the two operations (read vs. write). However, as we have seen, the RFC model achieves significantly better results.

6.2.1 Support Vector Machine (SVM)

The SVM model achieves an accuracy of 75.39%, which is similar to the performance of the previous neural network model. This suggests that the SVM and neural network models are performing similarly in terms of overall correctness across the dataset.

```
SVM Accuracy:  0.7539202200825309
SVM Classification Report:
              precision    recall  f1-score   support

           0       0.76      0.96      0.85     10524
           1       0.67      0.22      0.33      4016

    accuracy                           0.75     14540
   macro avg       0.71      0.59      0.59     14540
weighted avg       0.74      0.75      0.71     14540
```

Figure 6: SVM results

The model performs quite well for read operations, with a precision of 76% and correctly identifying 96% of all instances. However, for write operations, the SVM struggles significantly. The low recall (0.22%) suggests that the model misses most of the true write operations. This is likely again due to class imbalances—the model tends to focus more on the read operations.

7 Conclusion

The project successfully predicted memory access read and write operations, achieving a peak accuracy of 95.4% using the Random Forest Classifier. Random Forest significantly outperformed Neural Networks and SVM, maintaining

balanced precision and recall across classes, while Neural Networks and SVM struggled with class imbalance.

Key success factors included appropriate model selection. If I were to start again, I would begin a little earlier and consider additional hyperparameter tuning for Neural Networks and SVMs, which could potentially improve their performance. I would also test the models on even larger datasets (as mentioned earlier). More importantly, I would finish what I initially planned at the beginning of the project—classifying the memory access pattern (sequential/random), which is also why the additional features were computed.

8 References

Link for source code and dataset of the project: `https://github.com/zuzanaSKB/machine-learning-project`.