Machine Learning for Value Prediction

Randy Shoemaker
Department of Computer Science
College of William and Mary
Williamsburg, USA
rwshoemaker@email.wm.edu

***Abstract*—In computer architecture, value prediction attempts to predict the value of long latency instructions in order to break up data dependencies. Predicted values are speculatively used by subsequent instructions, which reduces stalling in the pipeline. The benefits are clear when values are correctly predicted, when a prediction is incorrect, however, a costly pipeline flush is incurred, harming performance. Therefore it is important to avoid mispredictions. In this work we investigate the usefulness of machine learning techniques for value prediction with the goal of increasing predictive power. We specifically focus on predicting the value of a load instruction. The goal of this work is to lay the groundwork for future investigations of machine learning for value prediction. The techniques we analyze are ordinary least squares regression (OLS), deep neural networks (DNN) and convolutional neural networks (CNN). Using our models we also investigate which factors are most important for value prediction [1].**

I. Introduction

Load instructions require fetching a value from memory; subsequent instructions which require this value must wait to execute while it is fetched. Due to the high cost of fetching from memory, load instructions can lower the instructions per cycle (IPC), harming performance, while dependent instructions wait to be executed. Value prediction has been proposed to address this issue. Value prediction leverages the concept of value locality, introduced by Lipasti et. al. [1], which is similar to spatial and temporal locality, which are leveraged in cache design. Value locality is the likelyhood of accessing a previously seen value in a particular memory location.

The goal of value prediction is to predict the value associated with a long latency operation, such as a load instruction, and supply this value to dependent operations which can then execute speculatively. When a value is correctly predicted the rewards are high due to the fact that dependent instructions may execute without delay. Once the value is fetched from memory it is compared to the predicted value. If the prediction is incorrect all speculatively executed instructions must rerun with the correct value. This can be costly in terms of performance, so correctly predicting values is crucial. Prior work has focused on confidence mechanisms [1] [2] [3] for deciding which instructions can be confidently predicted. Prior work has also examined methods of prediction, which typically are computation based, context based, or a hybrid of the two.

[1]The code for all of our work can be found in our repository: https://github.com/shoemarw/MLforVP

In this work we investigate the ability of machine learning algorithms to predict load values. Machine learning has been utilized extensively in many fields including computer vision [4], computer graphics [5] [6], and even in areas of computer architecture [7]. To the knowledge of the author it has yet to be utilized for value prediction. Machine learning is very effective at detecting patterns, therefore it is important to investigate if it can leverage the phenomenon of value locality and be used for value prediction. We investigate the ability of various machine learning algorithms, ordinary least squares regression (OLS), deep neural networks (DNN), and convolutional neural networks (CNN) for value prediction. We also use our results from OLS to analyze which attributes (PC, effective address, previous values) are important for value prediction. Our efforts focus on determining the accuracy of our machine learning models when predicting values. We do not implement these models in a predictor and we leave such efforts to future work. We only seek to evaluate the predictive ability of our models and the efficacy of machine learning for value prediction in general. Our hope is that this work is used in future work as a starting point for utilizing machine learning for value prediction. Our contributions are as follows:

1) We lay the groundwork for future research in machine learning for value prediction. While our models are not yet ready to accurately predict values, this work takes a step in the direction of successfully using machine learning for value prediction.
2) Using our models, we analyze the relative importance of the PC, effective address, and previously seen values for value prediction.

In the following section we discuss related work, in Section III we discuss our machine learning models, in Section IV we discuss our experimental framework and in Section V we discuss our results before discussing future work in Section VI and concluding in Section VII.

II. Related Work

We now discuss related work in value prediction before discussing relevant work in machine learning.

A. Value Prediction

The concept of value locality was introduced by Lipasti et. al. [1] where value prediction was proposed to predict the values associated with load instructions. Lipasti et. al. make use of a Load Value Prediction Table (LVPT) to predict load

values. The LVPT stores commonly seen values and thus essentially stores the state of a program in terms of its load values. This means that values that have not yet been observed can not be predicted. Our models, however, learn the state of programs and interactions between the PC, effective address, and previously seen values. This means our models are capable of predicting values that have not yet been observed.

Value predictors can typically be classified as computation based predictors, context based predictors, or a hybrid of the two [2] [8], some even predict values indirectly via address prediction [3]. A commonly used computation based predictor is a stride predictor. Stride predictors predict that the next value associated with a memory location is the sum of the previous value and some fixed stride. Context based predictors predict the next value associated with a memory location based on the the most frequently seen values associated with that memory location. Our machine learning approach can be viewed as a hybrid approach. Our models learn and utilize the context associated with memory locations like a context predictor. They also have properties of a computation based predictor because they learn a function of the PC, effective address, and previous values to make predictions. The deep learning models in particular can learn non-linear relationships between these attributes and utilize them for prediction. We now discuss relevant work in the area of machine learning.

B. Machine Learning

Machine learning is well known for its ability to detect patterns in datasets [9]. Since our work focuses on Ordinary Least Squares Regression (OLS) and deep learning we briefly discuss prior work with respect to those methods. For a detailed introduction to, and survey of, deep learning we refer the reader to Goodfellow et. al. [9].

1) Ordinary Least Squares Regression: Ordinary Least Squares Regression (OLS) seeks to fit a hyperplane to the points in a dataset such that the cumulative distance from each point to its nearest point on the hyperplane is minimized. As such it is a simple machine learning model, but regardless of its simplicity it is used to model many complex phenomena such as predicting asset prices in the market [10]. When load value prediction is formulated as a regression problem we seek to compute a linear function of the relevant attributes (PC, effective address, etc.) whose value is, hopefully, the value we wish to predict. Since each attribute has a linear weight we can use OLS to see which values are more important for, or have a larger impact on, value prediction. In the context of empirical asset pricing, Gu et. al. [10] use OLS to determine which factors are most important for pricing a financial instrument, they do this by analyzing which factors are weighted more heavily in the OLS solution. In our work we use this method to determine which attributes are most important for value prediction.

In the following section we discuss some relevant work in the area of deep learning.

2) Deep Learning: Deep learning has been used extensively to detect patterns and learn complex phenomena in many fields, particularly computer vision [4]. It has been noted for its ability to detect high level features in data and learn nonlinear relationships between data points [9]. While deep learning has not been used for value prediction, it has been used in the field of computer architecture, particularly it has been utilized by Shi et. al. [7] to tackle the cache replacement problem. A major issue with deep learning with respect to value prediction is the computational costs associated with training and utilizing a neural network on the hardware level. This is a major road block for using deep learning and machine learning for value prediction. The goal of this work is to demonstrate the usefulness of machine learning for value prediction and encourage future researchers to utilize it for VP. Once values can be accurately predicted by a machine learning model, the model needs to be scaled down so that the computational costs are mitigated. Afterwords, the model needs to be simulated in hardware to determine its benefits and costs. Shi et. al. give have done this for the problem of cache replacement. Shi et. al. use a Recurrent Neural Network (RNN) to aide in cache replacement and demonstrate the efficacy of deep learning in computer architecture. They use the insights from their successful but costly model to design a simpler and less costly model, a Support Vector Machine (SVM), which is then simulated in hardware. Their model yields considerable performance gains, thus demonstrating that deep learning can be used in computer architecture. It is our hope that future researchers will use our work as a starting point for the usage of machine learning for value prediction, and that this eventually leads to a working model that yields performance gains.

In the following section we discuss our machine learning models. In Section IV we discuss or experimental framework, in Section V we discuss our results, and in Section VI we discuss possible avenues of future work before concluding in Section VII.

III. Machine Learning Models For Value Prediction

We now discuss our machine learning models used for value prediction. The models used in this work are Ordinary Least Squares (OLS), Deep Neural Networks (DNN), and a sequential Convolutional Neural Network (CNN). For each model we scale all values into the interval $(0, 1)$, predictions are then made by scaling the output of a model back into its natural range.

A. Ordinary Least Squares

Our OLS models formulate value prediction as a regression problem. Let $X \in \mathbb{R}^{n \times m}$ where n is the number of load instructions and m is the number of attributes used in prediction. We examined three OLS models, each with a different value for m, which are discussed below. $X_{i,j}$ represents the j^{th} attribute associated with the i^{th} load instruction. Let $\hat{y} \in \mathbb{R}^n$ be a vector containing the value associated with each load such that y_i is the value associated with the i^{th} load. We then seek $\hat{b} \in \mathbb{R}^m$ that minimizes

$$|X\hat{b} - \hat{y}| \tag{1}$$

The solution to Equation 1 is also the solution of the normal equation

$$X^T X\hat{b} = X^T\hat{y} \tag{2}$$

In our OLS models we find $\hat{b}$ by solving Equation 2 and then predict the values associated with the loads as $X\hat{b}$. When using our OLS models to predict the loads in a trace, we divide the loads into two sets. The first 80% are used to find $\hat{b}$, then $\hat{b}$ is used to predict the values associated with the remaining 20% of loads. We now discuss the specifics of each of our OLS models.

In our simplest OLS model we use the PC and effective address associated with each load to predict the value of the load. We call our second model the split OLS model, where we split each 64 bit value into two 32 bit values and use the first and second 32 bits of the PC and effective address to predict the first and second 32 bits of the load value. In the split model we solve two regression problems, one for finding the first 32 bits of the load value and another for the second 32 bits. In our third model we use the PC, effective address and the previous 16 load values to predict the next load value, we call this model OLS 16.

In the next section we discuss our Deep Neural Network (DNN) models.

B. Deep Neural Networks

Our DNN models seek to learn a function f which maps the attributes associated with a load to the value of the load. We developed two DNN models to learn f, the first uses the PC and the effective address to predict the load value. We call the second model the split DNN model, similar to the split OLS model, every value is split into two 32 bit values. Two sub-networks are used to make the predictions, the first predicts the first 32 bits of the load value and the second predicts the second 32 bits. The inputs to each network are the same as the split OLS model: the first and second 32 bits of the PC and effective address.

We use the following architecture for our the split DNN and regular DNN models. Each model has two fully connected hidden layers which use a rectified linear unit as the activation function. The width of the hidden layers are 50 and 75 respectively. The activation function for the output is a sigmoid, which ensures all values are non-negative.

In our DNN models the first 80% of the loads of a trace are used to learn f and the remaining 20% of loads are used to test the accuracy of f for value prediction. In the next section we discuss the sequential CNN model (CNN).

C. Sequential CNN

Our CNN model seeks to learn a function f which maps the effective address of a load, along with the previous 5 effective addresses, to the value of the load. The goal CNN is to detect temporal or sequential patterns between the effective addresses and load values.

1M	compute_fp_1	compute_fp_4
compute_fp_9	compute_fp_10	compute_int_0
compute_int_4	compute_int_7	compute_int_11
compute_int_15	srv_10	

Fig. 1: The datasets used in our experiments. These can be obtained from CVP1 website.

The architecture of the CNN is as follows, the first layer is a convolutional layer with a kernel size of 4 and 20 filters. The second layer is a max pooling layer with a pool size of 4. The next two layers are fully connected layers with widths of 75 and 50 respectively and a rectified linear unit for the activation function. The output of the last layer has a sigmoid as an activation function to ensure all values are non-negative.

As with the other models the first 80% of the loads of a trace are used to learn f and the remaining 20% of loads are used to test the accuracy of f. In the next section we discuss our experimental framework.

IV. Experiments

We now discuss our experimental framework. We use trace data from the Championship Value Prediction tournament[2] (CVP1). Eleven Traces were selected from the CVP1, their file names are shown in Figure 1. We extract the load instructions from each trace, the extracted data is formatted so that it can be used in our machine learning models. All of our models are implemented in Python 3 and data is formatted using the Pandas [11] and Numpy [12] modules. The normal equations for the OLS models are solved using `numpy.linalg.solve`. The DLL and CNN models are implemented using the Keras framework [13]. Each deep learning model, including the sequential convolutional neural network, uses a learning rate of 0.001 and are trained using the adam optimizer [14] with root mean square error as the loss function. Our implementation can be found in our github repository[3].

In the following section we discuss our results.

V. Results

We now discuss our results, first we discuss the accuracy of our models, we then use the OLS models to analyze the relative importance of particular attributes for value prediction.

A. Performance

We now discuss the performance of our machine learning models in terms of accuracy. We measure performance as the root mean square error across the testing set for each trace, which is the last 20% of load instructions. Since all values are scaled into the interval $(0, 1)$ these error measures can be interpreted as the error as a percentage of the range of load values for a trace. As such, while the errors are quite small, if they are scaled into the range of load values for a trace,

[2]The dataset is available via https://www.microarch.org/cvp1/cvp1/index.htm

[3]https://github.com/shoemarw/MLforVP

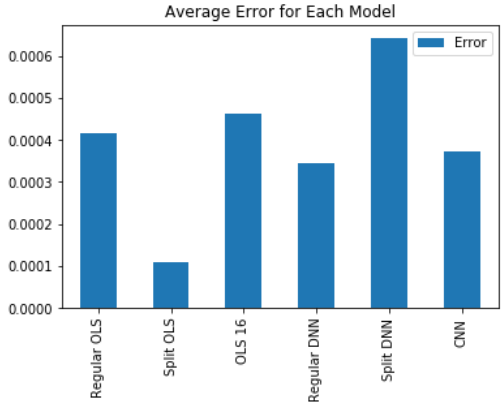

Fig. 2: The error averaged across 10 traces for each model.

the error is very large. This illustrates that these models, and machine learning models in general, are not yet ready to be implemented in a value predictor. Regardless, we may use these results to compare the relative ability of each model to predict load values. We remind the reader that our goal is to lay the groundwork for future explorations of machine learning for value prediction. While our models are imperfect, we believe that they take a step in the right direction.

Figure 2 shows the average error, across 10 of the 11 traces, for each model. The results from one trace, `compute_int_1`, are omitted due to the OLS models suffering large prediction errors when predicting the values of this trace. If it is included the deep learning models outperform the OLS models, while this is not the case in general. We can see that the split OLS model seems to perform the best with the split DNN model having the worst performance.

Figures 3 4 5 6 7 8 show the root means square error for each model across every dataset. Results for the trace `compute_int_1` are omitted from the plots for the OLS models because their error on this dataset is several orders of magnitude larger than that of the others. Interestingly, this is not the case for the deep learning models.

From these results we can conclude that the OLS models generally outperform the deep learning models. This surprising result indicates that selecting the right architectural details for the neural network models is very important. Future efforts must focus on getting these details right. One would think that deep learning models should outperform the OLS models. A potential reason for the poor performance of the deep learning models may be due to overfitting. Even though extensive hyperparameter tuning went into building and training these models, we can not rule out overfitting.

In the following section we analyze the results of the OLS models to gauge the relative importance of each attribute for value prediction.

B. Important Factors for Value Prediction

We now analyze our OLS models to investigate the importance of different attributes for value prediction. We rate the importance of different factors for value prediction by

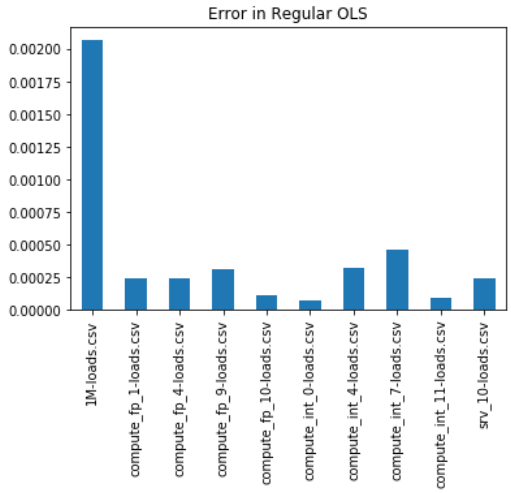

Fig. 3: Error for the regular OLS model.

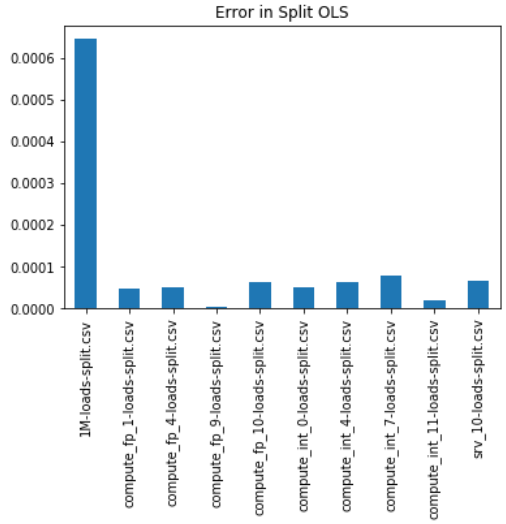

Fig. 4: Error for the split OLS model.

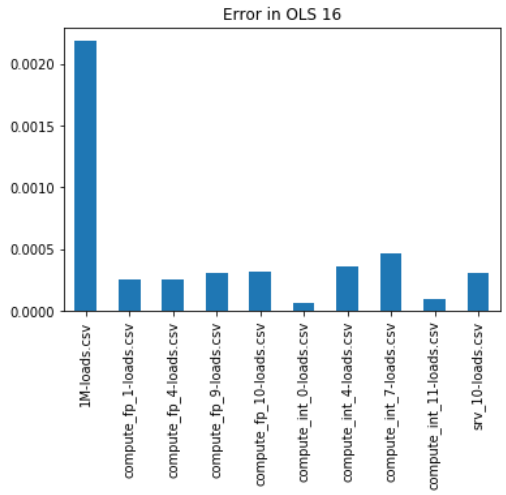

Fig. 5: Error for the OLS 16 model.

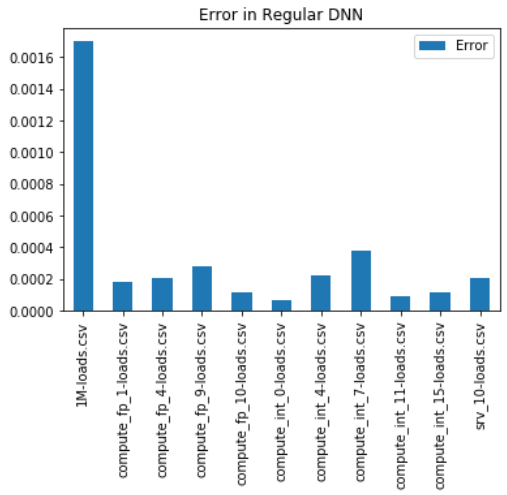

Fig. 6: Error for the regular DNN model.

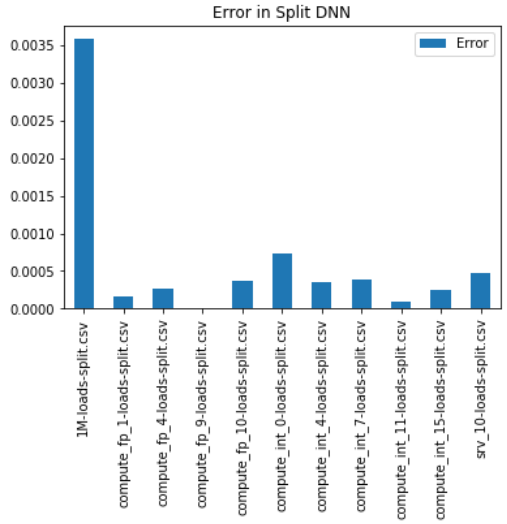

Fig. 7: Error for the split DNN model.

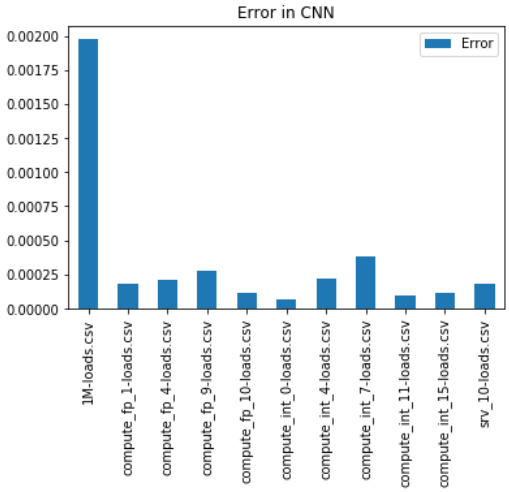

Fig. 8: Error for the CNN model.

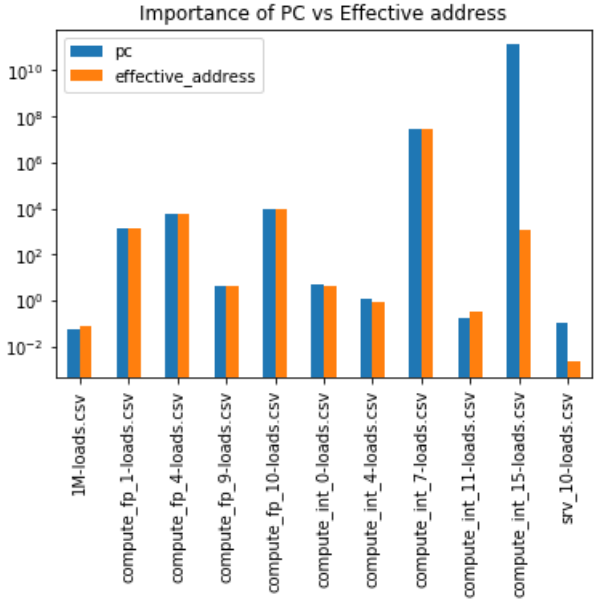

Fig. 9: Importance of the PC and Effective address for value prediction in the regular OLS model.

examining the magnitude of the coefficients of the OLS solution. A larger magnitude for a coefficient indicates that, for the particular dataset, the corresponding attribute has a larger impact on the value of the load. Figure 9 shows the relative importance of the PC and effective address attributes for predicting the value of a load in the regular OLS model. This illustrates that both factors, relatively, have the same predictive power.

Figure 10 shows the relative importance of the low bits of the PC vs the high and low bits of the effective address for predicting the lower 32 bits of the load value. Figure 11 shows the relative importance of the low bits of the PC vs the high and low bits of the effective address for predicting the higher 32 bits of the load value. We can see that the predictive power of the lower 32 bits of the PC is typically on par with that of the lower 32 bits of the effective address, which is consistent with the results from the regular OLS model. We can also see that the lower order bits typically have a larger impact than the higher order bits. This makes sense because the higher order bits have a much smaller variation than the lower order bits.

Figures 12 13 14 15 show the relative importance of the PC, effective address and previous load values for prediction. The legend of each figure shows what values are considered in the plot, where $-n$ refers to the value associated with the the n^{th} load prior to the one we seek to predict. We can see that the PC and the effective address have the most impact on the predicted value, however, in some cases the previous loads have a large impact. Theses results show the importance of previous load values and are consistent with the notion of value locality.

With the results of the OLS models we can conclude that the PC and the effective address are usually equally important

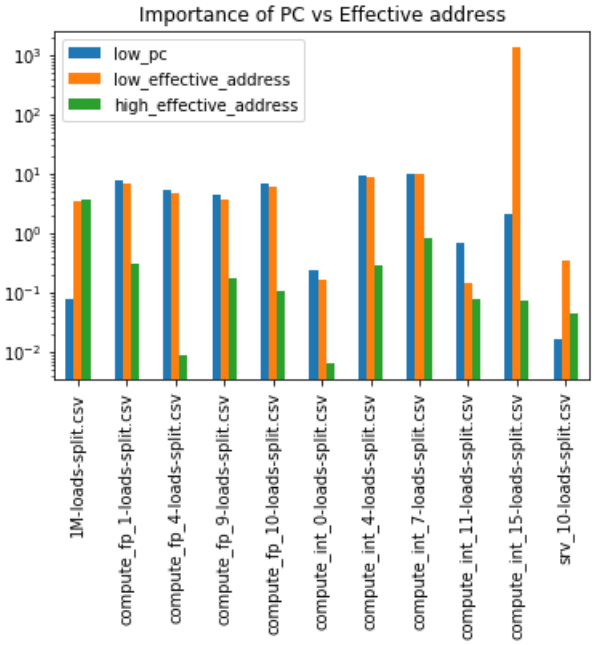

Fig. 10: Importance of the low bits of the PC vs the high and low bits of the effective address for predicting the low bits of the load value in the split OLS model.

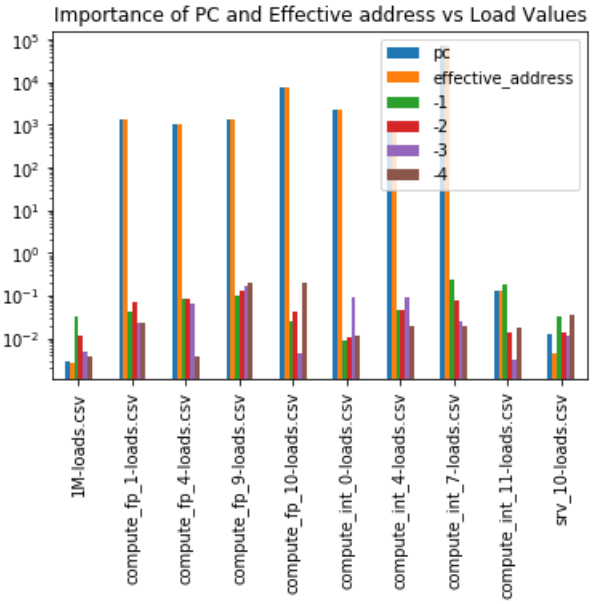

Fig. 12: The relative importance of the PC and Effective address compared to previously seen values.

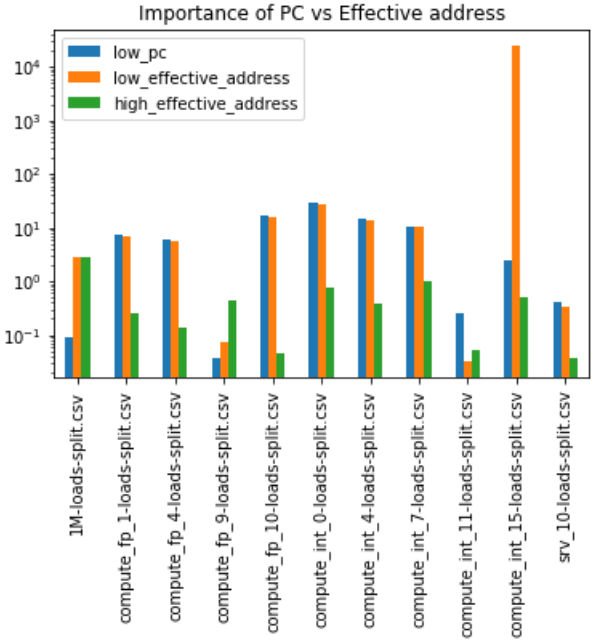

Fig. 11: Importance of the low bits of the PC vs the high and low bits of the effective address for predicting the high bits of the load value in the split OLS model.

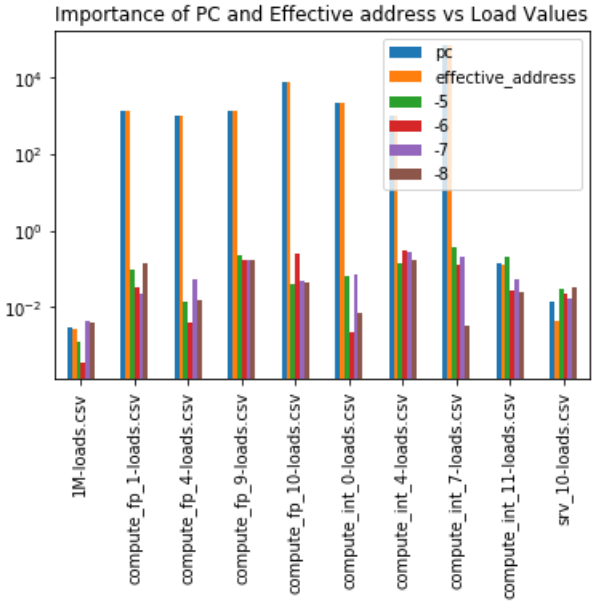

Fig. 13: The relative importance of the PC and Effective address compared to previously seen values.

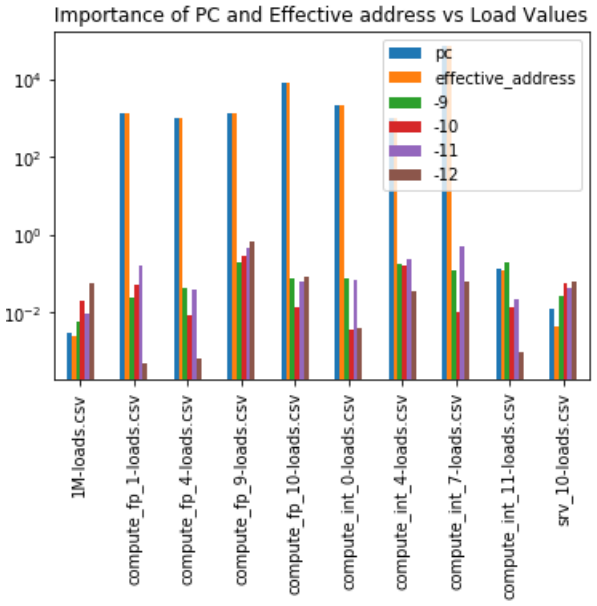

Fig. 14: The relative importance of the PC and Effective address compared to previously seen values.

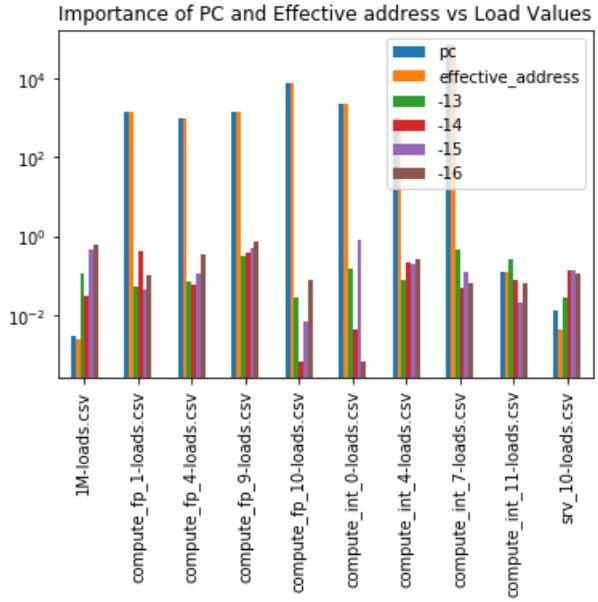

Fig. 15: The relative importance of the PC and Effective address compared to previously seen values.

for value prediction. We can also conclude that the history of previous loads is important and usually recently seen values are roughly as important as those from more temporally distant loads. In the following section we discuss potential avenues of future research.

VI. Future Work

We now disucuss future avenues of research with respect to machine learning for value prediction. The goal of this work is to lay the groundwork for future investigations of machine learning for VP, as such there is much work to do. First, more work must be done to increase the accuracy of predictions. This could involve investigating more complex models, perhaps deeper models, or different architectures such as recurrent neural networks. One could increase accuracy of the models by incorporating store instructions, since they change the state of programs and their use is known to aide value predictors [3]. Accuracy can also be increased by using previously seen values in the deep learning models, while this will significantly increase training and testing time, this is likely to result in significant gains. Our results make it clear that the architectural design of neural networks for value prediction is important and warrants a great deal of effort.

Once the accuracy of machine learning models has been sufficiently increased, future research should focus on maximizing the number of correct predictions. This can be done accomplished by using confidence mechanisms to decide which values to predict, perhaps machine learning, particularly classification, can be used here too. This could also be accomplished by using more attributes, such as additional previous load values. One could also use a load value prediction table along with machine learning models to increase the number of correctly predicted values.

Finally, once the number of correct predictions is maximized, research should focus on simplifying the complexity of the model so that it may be implemented in hardware without severe computational costs. Such simplified prediction models should then be simulated to investigate the performance gains and costs.

In the following section we summarize our results and conclusions.

VII. Conclusion

We now summarize our results. Our OLS models outperform, in general, our deep learning models, particularly, our split OLS model has the best performance on average. Analysis of our OLS models highlights the relative equal importance of the PC and the effective address for predicting load values. This analysis also shows that the values of previous loads are important when predicting the value of a subsequent load, which is consistent with the notion of value locality. Our work shows that architectural design of neural networks should be a major focus of future work in this area. This work serves as an initial investigation of the use of machine learning for value prediction. It is the hope of the authors that this work serves as a starting point for future research in this area.

References

[1] M. H. Lipasti, C. B. Wilkerson, and J. P. Shen, "Value locality and load value prediction," *SIGPLAN Not.*, vol. 31, p. 138–147, Sept. 1996.

[2] B. Calder, G. Reinman, and D. M. Tullsen, "Selective value prediction," in *Proceedings of the 26th Annual International Symposium on Computer Architecture*, ISCA '99, (USA), p. 64–74, IEEE Computer Society, 1999.

[3] R. Sheikh, H. W. Cain, and R. Damodaran, "Load value prediction via path-based address prediction: Avoiding mispredictions due to conflicting stores," in *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO-50 '17, (New York, NY, USA), p. 423–435, Association for Computing Machinery, 2017.

[4] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[5] D. Gao, X. Li, Y. Dong, P. Peers, K. Xu, and X. Tong, "Deep inverse rendering for high resolution SVBRDF estimation from an arbitrary number of images," *ACM Transactions on Graphics*, vol. 38, July 2019.

[6] N. Sharp and M. Ovsjanikov, ""pointtrinet: Learned triangulation of 3d point sets"," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[7] Z. Shi, X. Huang, A. Jain, and C. Lin, "Applying deep learning to the cache replacement problem," in *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*, MICRO '52, (New York, NY, USA), p. 413–425, Association for Computing Machinery, 2019.

[8] A. Perais and A. Seznec, "Practical data value speculation for future high-end processors," in *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*, (Los Alamitos, CA, USA), pp. 428–439, IEEE Computer Society, feb 2014.

[9] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Adaptive Computation and Machine Learning series, MIT Press, 2016.

[10] S. Gu, B. Kelly, and D. Xiu, "Empirical asset pricing via machine learning," tech. rep., National Bureau of Economic Research, 2018.

[11] T. pandas development team, "pandas-dev/pandas: Pandas," Feb. 2020.

[12] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del R'ıo, M. Wiebe, P. Peterson, P. G'erard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant, "Array programming with NumPy," *Nature*, vol. 585, pp. 357–362, Sept. 2020.

[13] F. Chollet *et al.*, "Keras." https://keras.io, 2015.

[14] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.