Machine Learning based Memory Load Value Predictor for Multimedia Applications

Alain Aoun[1], Mahmoud Masadeh[2], and Sofiène Tahar[1]
a_alain@ece.concordia.ca, mahmoud.s@yu.edu.jo, tahar@ece.concordia.ca
[1] Department of Electrical and Computer Engineering, Concordia University, Montreal, Quebec, Canada
[2]Computer Engineering Department, Yarmouk University, Irbid, Jordan

***Abstract*—Approximate computing (AC) has gained traction as an alternative computing method for energy-efficient processing. This paper proposes the exploitation of AC to address the memory wall. The proposed model predicts the load value using machine learning (ML). Subsequently, the ML model is a load value approximator (LVA) where the generated value is accepted as-is. The proposed LVA was tested under various approximate conditions, where 50% to 95% of the load instructions were approximated using multimedia applications. The peak signal-to-noise ratio (PSNR) exceeded 100 *dB* in several scenarios.**

***Keywords*—Approximate Computing, Approximate Memory, Machine Learning, Multimedia Processing**

I. Introduction

Approximate Computing (AC), also known as *inexact computing*, has gained renewed interest as an alternative to exact computation in error-resilient applications. AC provides inaccurate outputs to save area, power, and delay [1]. AC is applicable in domains where errors are acceptable due to factors such as the absence of a definitive answer, noisy input data, imperfect human perception of noisy output, and algorithms with error attenuation patterns, e.g., machine learning, multimedia, and search engines. While existing AC research has mainly concentrated on arithmetic units, e.g., approximate full adders (FAs) [2], we focus in this paper on approximating the *memory access* process. This involves replacing traditional memory accesses with a machine learning-based predicted value to tackle the challenges posed by the *memory wall* in memory-intensive applications such as deep learning [3].

Existing efforts to overcome the *memory wall* include process-in-memory, load value speculation, approximate memory, along other techniques. Process-in-memory relocates redundant computational operations, e.g., multiply and accumulate, from the CPU to the memory or its proximity [4]. Load value speculation adds a unit in the processor that estimates the value when a load occurs without altering the Von Neumann architecture significantly [5]–[7]. In case of incorrect speculation, the CPU rolls back to the pre-speculation state and flushes the pipeline. Load value approximation extends this concept by accepting the estimated (speculated) value regardless of its correctness, as explored in [8]–[10]. Another approach to address the memory bottleneck is approximate memory, as explored in [11], where the authors proposed an energy-efficient dynamic random access memory (DRAM) with various refresh rates, reducing energy consumption and enabling concurrent loading of the most significant bits of multiple values.

Machine Learning (ML) finds broad application across domains, especially in scenarios when outcomes follow a sequence of events [12]. Exploiting the principle of locality in computers, both *spatial* and *temporal* localities, have been extensively utilized as well as the concept of *value locality* introduced in [5]. The authors of [5] posit that *value locality* is similar to branch prediction, where subsequent branch outcomes correlate with previous outcomes. This paper delves into load value approximation using ML, capitalizing on the *value locality* feature.

In the remainder of this paper, we present a literature review in Section II. We describe the proposed model of load value approximation in Section III and the experimental results in Section IV. We conclude this paper in Section V.

II. Related Work

Numerous publications in the literature have proposed strategies to alleviate memory bottlenecks. In this section, we will discuss methods that are most pertinent to our research, focusing on: 1) Load Value Speculation, 2) Load Value Approximation, and 3) Approximate Memory.

A. Load Value Speculation

Load value speculation (or prediction) involves speculating the value to hide memory latency. The work in [5] introduced *value locality*, suggesting comparable magnitudes in neighboring memory locations, exemplified by adjacent image pixels. The authors of [5] proposed a dynamic lookup table for load value speculation, aiming to conceal memory access latency. In case of incorrect speculation, the processor initiates a rollback and flushes the pipeline. The lookup table undergoes updates after each memory access. This concept is extensively investigated, with alternative techniques explored in the literature such as [6] and [7]. All load value speculation methods consist of hiding the memory access latency where memory accesses are required to verify speculation correctness.

B. Load Value Approximation

To circumvent the hardware and clock cycle costs associated with rollbacks, some researchers explored load value approximation (LVA) strategies. These techniques speculate values and eliminate rollbacks in the event of incorrect predictions, resulting in approximations. For instance, the approach in [8] suggests LVA based on a dynamic predictor. Predictor accuracy is enhanced by learning from recent load values.

979-8-3503-7939-6/24/$31.00 ©2024 IEEE

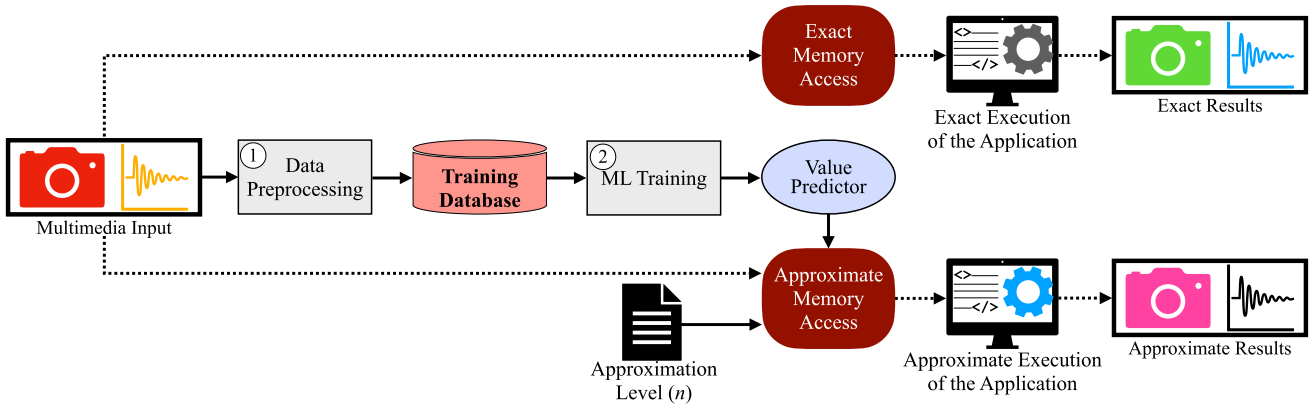


Fig. 1. Proposed Memory Load Value Approximation

This model trades bandwidth and energy for reduced quality. Another approach for LVA is delineated in [9], presenting a model tailored for GPU architecture. Both LVA techniques mandate memory access to minimize errors and require a hardware overhead to implement the predictor. Alternatively, the work in [10] eliminates all memory accesses in real-time execution. The authors of [10] proposed the profiling of an error-tolerant application to instrument load accesses and load context. Thereafter, the instrumented data is checked for the existence of *value locality* and in case of presence, the data is used to train a machine learning (ML) model which serves as a load predictor in real time.

C. Approximate Memory

As an alternative to LVA, researchers explored *approximate memory* concepts to save energy and bandwidth. In [11], the focus was on implementing DRAM with transposed data storage, allowing diverse refresh rates for rows. Rows containing the least significant bits (LSBs) are refreshed less frequently, reducing energy consumption. Storing data in a transposed manner allows the reading of the most significant bits (MSBs) of multiple values with a single load. Another approach of approximate memory in [13] proposes the compression of error-tolerant data in DRAM. Compression and decompression in distinct memory regions define varying accuracy levels, controlled by a software-hardware integration.

III. Proposed Methodology

The implementation methods of load value approximation (LVA) proposed in [8] and [9] require a hardware overhead, e.g., lookup tables, in addition to continuous memory access to update the predictor and thus maintain an acceptable quality level. On the other side, the work of [10] proposes a static predictor that eliminates all memory accesses in real-time. However, the work of [10] has the drawback of the computation overhead to evaluate and store the hash values encoding history of execution required by the ML-based LVA. Furthermore, the method proposed in [10] is hardware-specific since the ML-based LVA is trained on a given set of program counter and memory address values. Subsequently, when implemented on a new hardware the model will have to be reconfigured. With the aim of alleviating the challenges of existing methods, in this paper, we propose an LVA that combines the advantages of previously proposed approaches. The key benefits of this LVA are static predictor, quality/energy control in real-time, and simple predictor requiring minimal overhead. The LVA we propose in this paper is beneficial when the same multimedia is used repetitively, e.g., a video game where the same objects are used repetitively.

The proposed LVA consist of two steps as shown in Fig. 1. The proposed LVA requires three inputs: *i*) Multimedia, e.g., images or audio; *ii*) Error-Tolerant Application; and *iii*) Approximate Level (n). The multimedia is used by the error-tolerant application and the proposed LVA. The error-tolerant application is an assembly program while the approximation level defines the ratio of approximated load instructions. Based on these three inputs, the LVA produces an approximate execution of the provided application. The proposed LVA consists of data preprocessing in step ① which will generate a training database as shown in Fig. 1. In step ②, the database is used as a training data for machine learning (ML) where a value predictor is produced that serves as an LVA in the proposed model. Subsequently, the value predictor, i.e., the ML-based LVA, is used in the memory access to predict the load values. The portion of memory accesses that are approximated is defined by the approximate level (n). The approximate level (n) replaces n out of $n + 1$ load instructions with a call to the ML-based LVA. For example, for $n = 19$, the LVA will consist of 1 exact load followed by 19 approximate load instructions, i.e., 1 out of 20 load instructions is exact. Thus for $n = 19$, we perform a 95% approximation. The execution of the error-tolerant application using the methodology shown in Fig. 1 results in an approximate execution of the application where some of the load instructions are approximated. The multimedia when loaded in an "*approximate memory access*" fashion results in approximate execution of the application and hence approximate results while when loaded in the conventional form, i.e., "*exact memory access*", will produce exact results. The proposed ML-based LVA must have the same multimedia in the training and in the real-time, i.e., in the approximate execution. Subsequently, it is only beneficial to use the proposed ML-based LVA if the multimedia is used repetitively, e.g., in a video game.

The advantage of the proposed model is the usage of a static predictor, i.e., memory accesses can be eliminated. Further-

more, the approximation level is adjustable and hence with more approximation, the energy consumption is reduced and potentially affects the quality. The ML-based predictor used in this paper exploits the principle of *value locality* [5], where the subsequent value is predicted by knowing the previous value, i.e., previously used value either fetched from memory or generated by the ML-based predictor. To measure the quality of the proposed methodology, we tested it using a variety of multimedia applications as described in the next section.

IV. Experimental Results

In this section we will apply the proposed LVA methodology on three multimedia applications that we test for a range of approximate levels (n) varying from 1 to 19. The applications are tested when training the ML-based LVA using two multimedia only, e.g., two images, and the same multimedia is then used in the experiment to measure the quality. Before that, we need to select the most suitable ML model for the proposed LVA. Our experiments were executed in Python 3.8.10 using a machine with Intel Xeon 4116 CPU running Ubuntu 20.04 with 192GB of RAM.

A. ML Technique Selection

We identify the most suitable machine learning (ML) technique by training various classification and regression models available in the Scikit-Learn (sklearn) library [14]. For instance, we evaluated 41 classification and 41 regression models by training on combinations of two unique images at a time using benchmark images known as "***Set5***" (with 5 images) and "***Set14***" (with 14 images) [15]. This resulted in 101 sets of training data, with 10 and 91 sets generated using ***Set5*** and ***Set14***, respectively. Each of the classification and regression models was trained and tested 101 times on the generated sets. The classification accuracy did not surpass 10%, while some regressors achieved an average root mean square error (RMSE) of 43.49. Considering the prediction is applied to one byte, ranging from 0 to 255, comparing the average RMSE (43.39) with the maximum possible value (255) deems the regression-based ML technique as the most suitable for Load Value Approximation (LVA).

We measured four metrics during the training of the various regressors, namely, training time, mean absolute error (MAE), mean squared error (MSE) and root mean square error (RMSE). We included the training time in the measurements as it indicates the model complexity, i.e., for two models with the same accuracy, the one with less training time is most suitable. The average of the four metrics for the 41 regressors that were selected are summarized in Table I. We can notice that four models achieved the best quality in terms of MAE, MSE and RMSE, namely, *Decision Tree* [16], *Extra Tree* [17], *Extra Trees* [18] and *Random Forest* [19]. On the other hand, when factoring the training time to select the best training model, we notice that the *Extra Tree* requires an average training time of 482 *ms* and hence outperforms the other three regression models. In the sequel of this section, the *Extra Tree* will be used to experiment with our proposed methodology.

B. Image Blending

In the first experiment, we test the quality of the methodology on an image blending application. All images in ***Set5*** are colored images while only 13 are colored images in ***Set14***. Since image blending consists of multiplying each of the red-green-blue (RGB) pixels with its corresponding RGB in the second image, we were able to use 13 images only. Subsequently, we were able to blend 10 and 78 combinations using ***Set5*** and ***Set14***, respectively. The 88 combinations were tested for n = 1 to n = 19, i.e., 50% to 95% approximation.

The quality of the resulting images is assessed using PSNR, MAE, MSE and RMSE. The average, maximum and minimum for each of the metrics are depicted in Fig. 2. From Fig. 2(a) we can notice that for n = 19, i.e., 95% approximation, the lowest PSNR is 67.24 *dB*. In multimedia applications, a PSNR greater than 40 *dB* is normally considered as very good [20]. Subsequently, at an approximation of 95%, the proposed LVA was able to achieve an acceptable quality. Furthermore, from

TABLE I
Training of Various Regressors

Model	Training Time	MAE	MSE	RMSE
ARD	112ms	37.07	2,354.90	47.23
Ada Boost	25s 361ms	38.90	2,321.16	47.15
Bagging	146ms	32.72	1,997.03	43.49
Bayesian Ridge	133ms	37.07	2,354.90	47.23
CCA	178ms	37.29	2,932.34	52.27
Decision Tree	1s 259ms	32.72	1,996.93	43.49
Dummy	13ms	58.75	4,727.59	68.41
Elastic Net	71ms	37.07	2,354.90	47.23
Elastic Net CV	5s 822ms	37.08	2,354.90	47.23
Extra Tree	482ms	32.72	1,996.93	43.49
Extra Trees	41s 441ms	32.72	1,996.93	43.49
Gradient Boosting	1m 8s 246ms	33.31	2,026.65	43.84
Hist Gradient Boosting	4s 473ms	32.93	2,010.08	43.64
Huber	5s 422ms	35.47	2,429.47	47.98
Isotonic	620ms	34.46	2,134.20	44.96
KNeighbors	1m 17s 637ms	35.01	2,379.33	47.48
Linear SVR	3m 7s 531ms	35.84	2,643.08	50.07
Lars	84ms	37.07	2,354.90	47.23
Lars CV	274ms	37.07	2,354.90	47.23
Lasso	65ms	37.07	2,354.90	47.23
Lasso CV	5s 441ms	37.08	2,354.90	47.23
Lasso Lars	45ms	37.07	2,354.90	47.23
Lasso Lars CV	237ms	37.07	2,354.90	47.23
Lasso Lars IC	220ms	37.07	2,354.90	47.23
Linear	67ms	37.07	2,354.90	47.23
Matching Pursuit	32ms	37.07	2,354.90	47.23
PLS Canonical	110ms	37.29	2,932.34	52.27
PLS	105ms	37.07	2,354.90	47.23
Passive Aggressive	2s 218ms	57.78	8,339.24	75.60
Poisson	1s 242ms	39.18	2,545.69	49.66
RANSAC	346ms	36.24	2,823.08	51.43
Radius Neighbors	3m 18s 551ms	33.53	2,052.20	44.13
Random Forest	1m 2s 529ms	32.72	1,996.93	43.49
Chain	9m 9s 537ms	52.68	5,444.58	71.60
Ridge	59ms	37.07	2,354.90	47.23
Ridge CV	217ms	37.07	2,354.90	47.23
Stacking	11m 52s 746ms	32.76	2,001.55	43.54
Theil Sen	6m 18s 380ms	35.51	2,568.31	49.29
Transformed Target	66ms	37.07	2,354.90	47.23
Tweedie	456ms	37.07	2,354.90	47.23
Voting	1m 11s 636ms	33.92	2,077.84	44.37
Minimum	**13ms**	**32.72**	**1,996.93**	**43.49**
Maximum	**11m 52s 746ms**	**58.75**	**8,339.24**	**75.60**

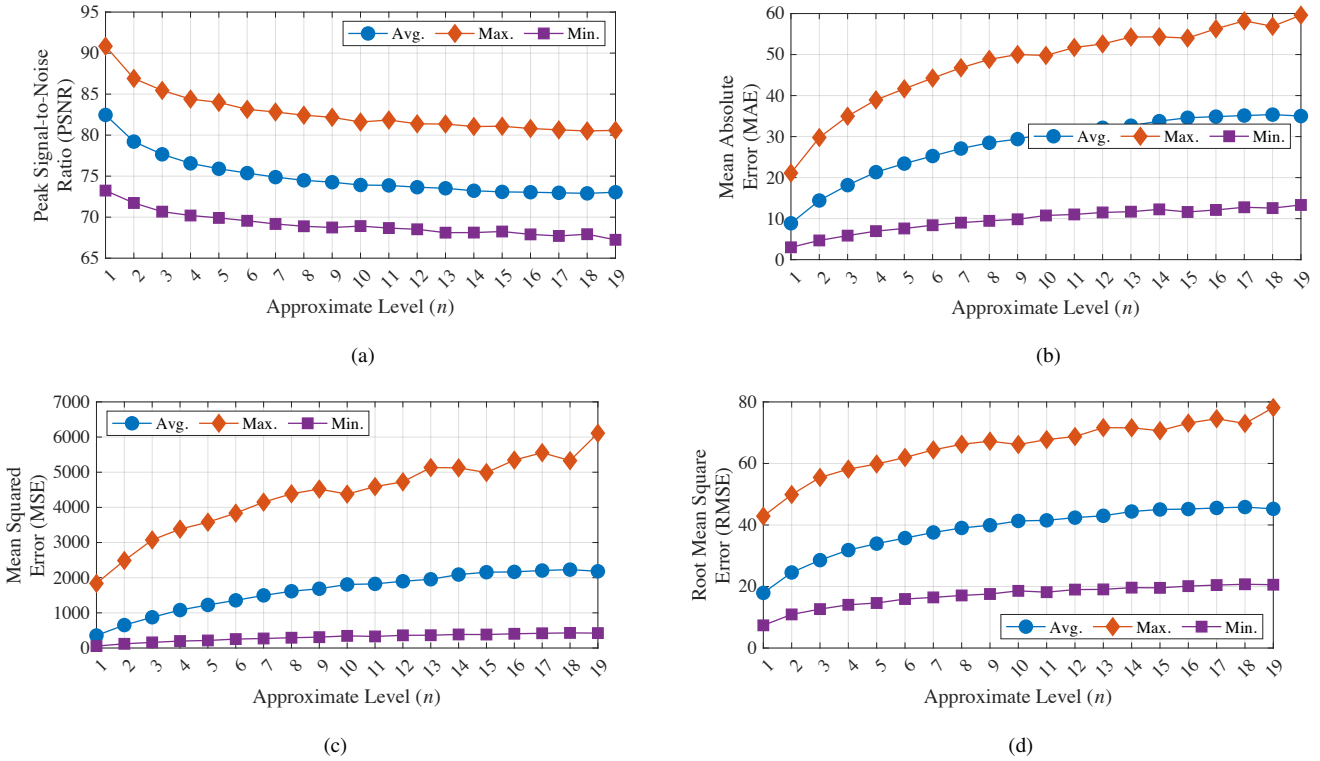


Fig. 2. (a) PSNR, (b) MAE, (c) MSE, and (d) RMSE of Image Blending for Various Approximate Levels

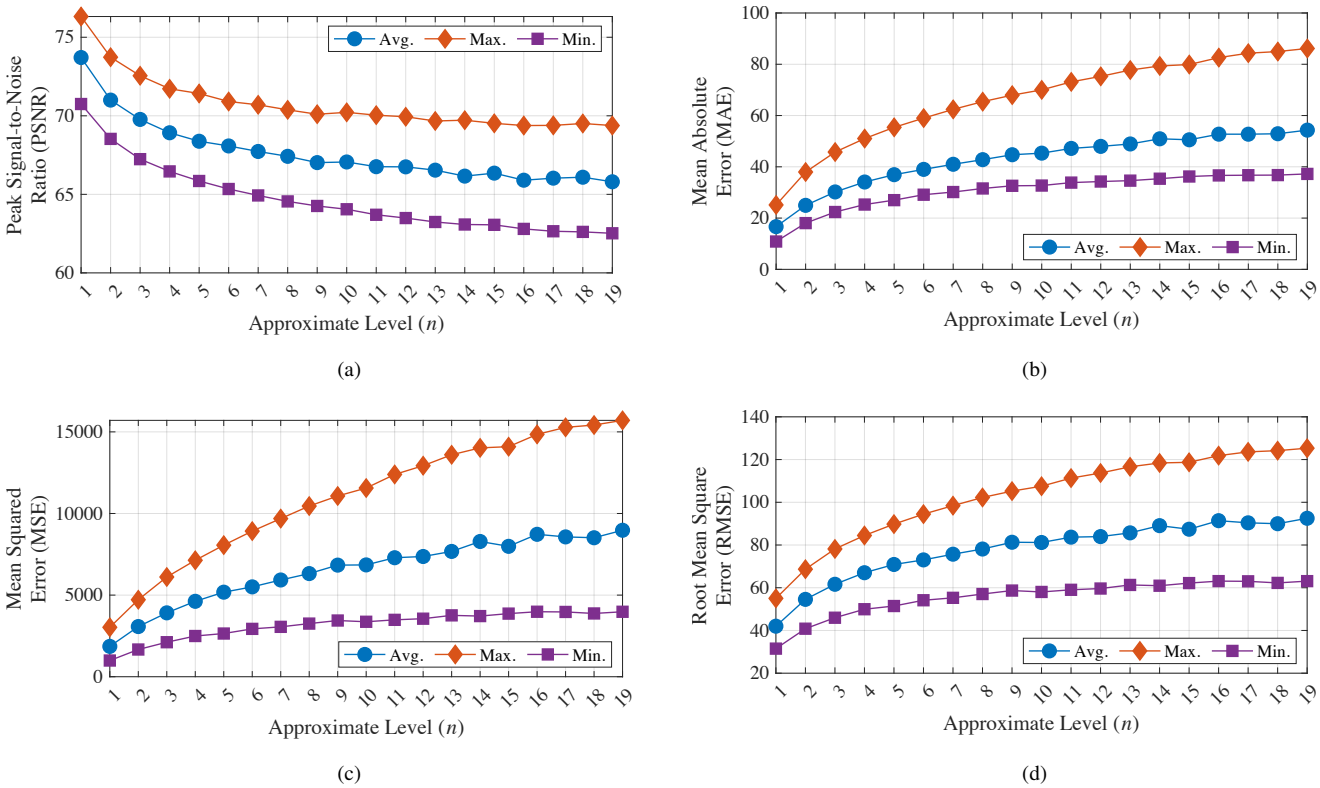


Fig. 3. (a) PSNR, (b) MAE, (c) MSE, and (d) RMSE of Edge Detection using *Sobel* Filter for Various Approximate Levels

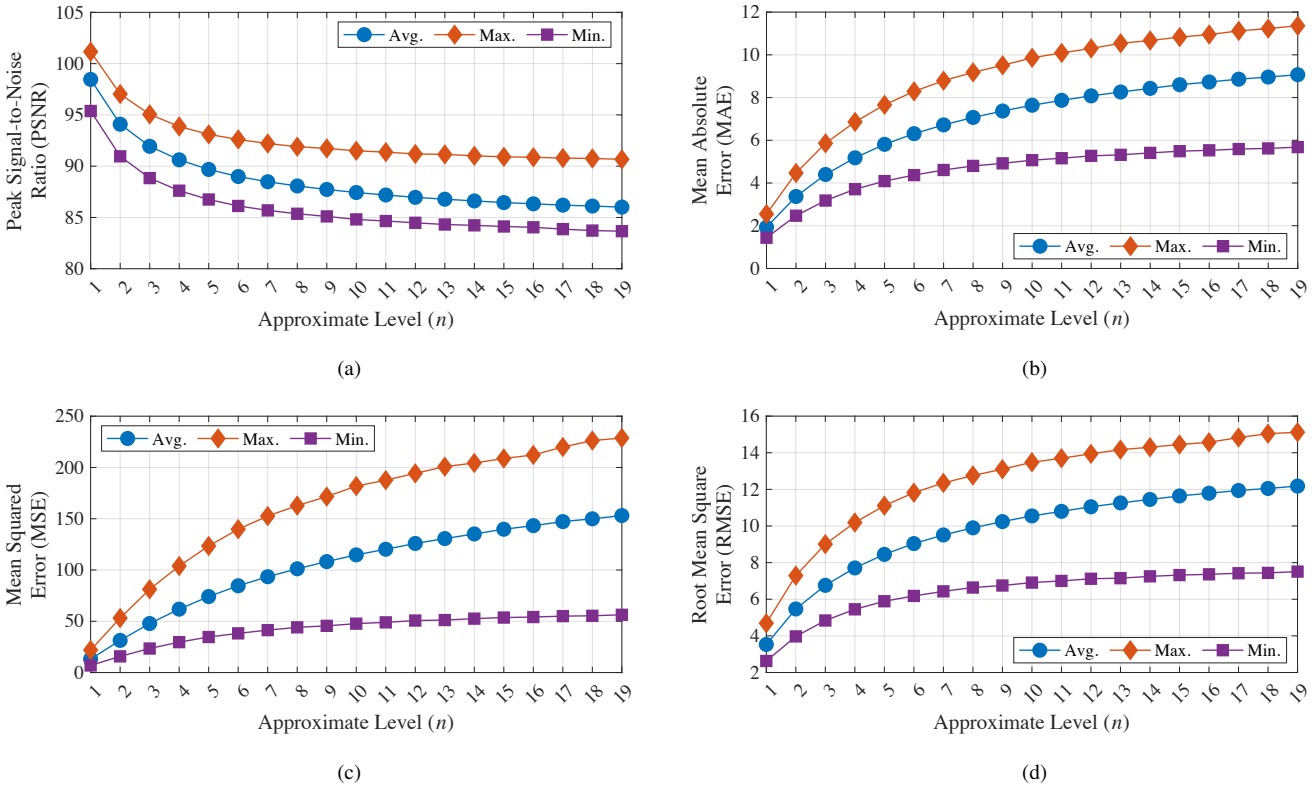
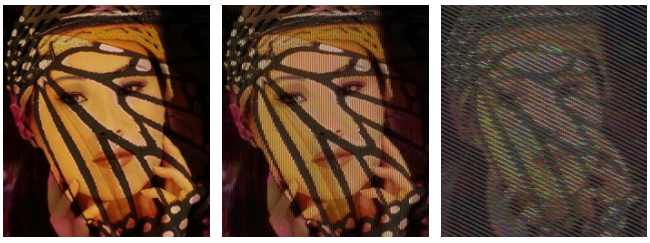


Fig. 4. (a) PSNR, (b) MAE, (c) MSE, and (d) RMSE of Audio Blending for Various Approximate Levels



(a) (b) (c)

Fig. 5. (a) Exact, (b) 50% Approximation (n=1), and (c) 90% Approximation (n=9) of Image Blending

(a) (b) (c)

Fig. 6. (a) Exact, (b) 50% Approximation (n=1), and (c) 90% Approximation (n=9) Execution of the *Sobel* Filter

the MAE, MSE and RMSE shown in Fig. 2 we can notice that the values are in an acceptable range where the average MAE, MSE and RMSE for n = 19 are 35.01, 2182.27 and 45.24, respectively. In addition, we can note that a higher approximation will not always result in a reduced quality. For example, from Fig. 2(d) we can see that for an approximate level of 19, i.e., n = 19, the average RMSE was 35.01 compared to 35.13 and 35.36 for n =17 and n = 18, respectively. On the other hand, assessing the quality subjectively, i.e., by visual inspection, we can notice from Figs. 5(a) and 5(b) that the difference is barely noticeable for a 50% approximation, i.e., approximate level of 1, when compared to the exact result. Furthermore, for a 90% approximation, the main features of the resulting image are still identifiable.

C. Sobel Filter

In the second experiment, the quality is measured in terms of edge detection using the *Sobel* filter where we use six benchmark images, namely, "*barb*", "*bikesgray*", "*cameraman*", "*lena*", "*mandrill*", and "*peppers*". Similar to image blending, four error metrics were measured when testing the proposed methodology for edge detection using the *Sobel* filter. The four error metrics for n varying from 1 to 19 are summarized in Fig. 3. The lowest PSNR achieved in this test was 62.52 *dB* which is acceptable for multimedia processing. Furthermore, at an approximate level of 1, i.e., 50% approximation, the average, maximum and minimum PSNR were 73.71, 76.32 and 70.75 *dB*, respectively. The average MAE, MSE and RMSE varied from 16.66 to 54.31, 1858.85 to 8972.86 and from 42.00 to 92.51, for the various approximate levels, respectively. Moreover, similar to image blending an increase in approximation did not always result in a decrease of quality. For instance, from Fig. 3(c) we can notice that the average MSE went down from 8,285.26 to 7,987.98 when the approximate level was increased from 14 to 15.

When assessing the quality subjectively, as shown in Fig. 6, we can notice that at 50% approximation of load instructions, the quality was barely affected. Furthermore, at an approx-

imation of 90%, the main features of the images are still detected. For example, the edges of the scarf, hands and legs of the lady were successfully detected. Consequently, we can deem the quality at 90% approximation to be acceptable when examining the results subjectively.

D. Audio Blending

Finally, in the third experiment, i.e., audio blending, we use five credit musical sounds from "Babylon 5", namely, "*s2cred*", "*s3cred*", "*s4cred*", "*s5cred*" and "*spocred*" as archived on the website of the University of Wyoming [21]. The retrieved audio files are in 8-bit *wav* format. The 10 combinations resulting from the five audio files retrieved from [21] were used. The PSNR, MAE, MSE and RMSE of the audio blending of these 10 combinations are shown in Fig. 4. From Fig. 4(a) we can note that the PSNR exceeded 100 *dB* in some cases and the lowest PSNR at 95% approximation was more than 83 *dB*. The average MAE, MSE and RMSE were in the range of 1.93 to 9.07, 13.15 to 153.10 and 3.53 to 12.18, respectively. Since the audio files are 8-bit, the maximum MAE and RMSE can be 255 while the MSE can be a maximum of 65025. Thus, for an approximate implementation, the measured errors were extremely low.

V. Conclusion

Approximate computing (AC) has gained traction as an alternative computing technique that offers savings in area and power consumption. Previously, explorations of AC techniques have focused on arithmetic operations which are beneficial for computation-intensive tasks. However, for memory-demanding applications, the *memory wall* would still curb the performance. To address this challenge, we proposed in this paper a load value approximator (LVA) that predicts the value using machine learning (ML) and requires minimal overhead. Unlike previously proposed ML-based LVA techniques, the model we proposed is architecture and machine independent. The proposed model was evaluated using three multimedia applications, i.e., image blending, edge detection using the *Sobel* filter, and audio blending, for various approximate levels (n), i.e., 50% to 95% of load instructions were approximated. The lowest value of peak signal-to-noise ratio (PSNR) in all the conducted experiments was 64.69 *dB*. This value of PSNR is deemed acceptable for multimedia applications. Since the proposed model was able to achieve a decent quality when 95% of the load instructions were approximated, we consider the proposed ML-based LVA as an efficient technique to be used when processing multimedia applications. The ML-based LVA we proposed in this paper is beneficial when the same multimedia is used repetitively, otherwise the training overhead outweighs the benefits.

As a future work, we plan to experiment with the proposed methodology using a variable approximate level (n) in a dynamic fashion for quality/energy trade-off. Furthermore, we plan to analyze the performance gain and energy saving when implementing the proposed methodology in hardware.

References

[1] H. Jiang, F. J. H. Santiago, H. Mo, L. Liu, and J. Han, "Approximate arithmetic circuits: A survey, characterization, and recent applications," *Proceedings of the IEEE*, vol. 108, no. 12, pp. 2108–2135, December 2020.

[2] V. Gupta, D. Mohapatra, A. Raghunathan, and K. Roy, "Low-power digital signal processing using approximate adders," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 32, no. 1, pp. 124–137, 2012.

[3] G. F. Oliveira, J. Gómez-Luna, L. Orosa, S. Ghose, N. Vijaykumar, I. Fernandez, M. Sadrosadati, and O. Mutlu, "DAMOV: A new methodology and benchmark suite for evaluating data movement bottlenecks," *IEEE Access*, vol. 9, pp. 134 457–134 502, September 2021.

[4] D.-I. Jeon, K.-B. Park, and K.-S. Chung, "HMC-MAC: processing-in memory architecture for multiply-accumulate operations with hybrid memory cube," *IEEE Computer Architecture Letters*, vol. 17, no. 1, pp. 5–8, May 2017.

[5] M. H. Lipasti, C. B. Wilkerson, and J. P. Shen, "Value locality and load value prediction," in *International Conference on Architectural Support for Programming Languages and Operating Systems*. ACM, 1996, pp. 138–147.

[6] G. Reinman and B. Calder, "Predictive techniques for aggressive load speculation," in *International Symposium on Microarchitecture*. ACM, 1998, pp. 127–137.

[7] L. Ceze, K. Strauss, J. Tuck, J. Torrellas, and J. Renau, "CAVA: Using checkpoint-assisted value prediction to hide l2 misses," *ACM Transactions on Architecture and Code Optimization*, vol. 3, no. 2, pp. 182–208, June 2006.

[8] J. San Miguel, M. Badr, and N. E. Jerger, "Load value approximation," in *International Symposium on Microarchitecture*. ACM, 2014, pp. 127–139.

[9] A. Yazdanbakhsh, G. Pekhimenko, B. Thwaites, H. Esmaeilzadeh, O. Mutlu, and T. C. Mowry, "RFVP: Rollback-free value prediction with safe-to-approximate loads," *ACM Transactions on Architecture and Code Optimization*, vol. 12, no. 4, pp. 1–26, January 2016.

[10] A. Aoun, M. Masadeh, and S. Tahar, "A machine learning based load value approximator guided by the tightened value locality," in *Great Lakes Symposium on VLSI*. ACM, 2023, pp. 679–684.

[11] D. T. Nguyen, N. H. Hung, H. Kim, and H.-J. Lee, "An approximate memory architecture for energy saving in deep learning applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 5, pp. 1588–1601, May 2020.

[12] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science*, vol. 349, no. 6245, pp. 255–260, July 2015.

[13] A. Ranjan, A. Raha, V. Raghunathan, and A. Raghunathan, "Approximate memory compression for energy-efficiency," in *International Symposium on Low Power Electronics and Design*, 2017, pp. 1–6.

[14] Scikit-Learn, "Machine Learning in Python," 2024. [Online]. Available: https://scikit-learn.org/stable/

[15] R. Timofte, V. De Smet, and L. Van Gool, "Anchored neighborhood regression for fast example-based super-resolution," in *International Conference on Computer Vision*. IEEE, 2013, pp. 1920–1927.

[16] Scikit-Learn, "Sklearn Tree Decision Tree Regressor," 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html

[17] Scikit-Learn, "Sklearn Tree Extra Tree Regressor," 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.tree.ExtraTreeRegressor.html

[18] Scikit-Learn, "Sklearn Ensemble Extra Trees Regressor," 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesRegressor.html

[19] Scikit-Learn, "Sklearn Ensemble Random Forest Regressor," 2024. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

[20] D. R. Bull and F. Zhang, "Chapter 4 - digital picture formats and representations," in *Intelligent Image and Video Compression*. Oxford Academic Press, 2021, pp. 107–142.

[21] J. Ward, " Sound Archive for Babylon 5 - Opening and Closing Credits," 2007. [Online]. Available: http://b5.cs.uwyo.edu/bab5/b5-them.html