Modeling Performance and Power on Disparate Platforms using Transfer Learning with Machine Learning Models

Amit Mankodi, Amit Bhatt, Bhaskar Chaudhury, Rajat Kumar, and Aditya Amrutiya

Dhirubhai Ambani Institute of Information and Communication Technology, Gandhinagar, India
{mankodiamit,rajat.tech.002,amrutiyaaditya}@gmail.com

Abstract. Cross prediction is an active research area. Many research works have used cross prediction to predict the target system's performance and power from the machine learning model trained on the source system. The source and target systems differ either in terms of instruction-set or hardware features. A widely used transfer learning technique utilizes the knowledge from a trained machine learning from one problem to predict targets in similar problems. In this work, we use transfer learning to achieve cross-system and cross-platform predictions. In cross-system prediction, we predict the physical system's performance (runtime) and power from the simulation systems dataset while predicting performance and the power for target system from source system both having different instruction-set in cross-platform prediction. We achieve runtime prediction accuracy of 90% and 80% and power prediction accuracy of 75% and 80% in cross-system and cross-platform predictions, respectively, for the best performing deep neural network model. Furthermore, we have evaluated the accuracy of univariate and multivariate machine learning models, the accuracy of compute-intensive and data-intensive applications, and the accuracy of the simulation and physical systems.

Keywords: Multivariate Performance and Power Prediction, Transfer Learning, Machine Learning, Cross Prediction

1 Introduction

Performance has always been an essential factor in the selection of computer systems. However, computing resources' power consumption has emerged as a significant concern in recent times due to environmental hazards. Advancements in software-hardware co-development have increased the power consumption even further. Therefore, performance and power prediction models using machine learning have become an active area of research. Several researchers have used power prediction to reduce power consumption in computer systems. Work in [1] uses power consumption obtained through prediction as feedback to

improve power supply units used in physical computer systems. In [2] and [3] system-level power prediction is performed to understand effect of component-level usage on power consumption. By enabling efficient thread-level parallelism, work in [4] achieves power consumption reduction in heterogeneous cores.

Cross-platform prediction aid us in predicting performance and power from one platform or system to the other. For example, work in [5] shows application performance prediction for GPU from features collected from the x86-based system. Similarly, work in [6] achieves software performance and power prediction for the ARM-based hardware system by collecting performance counters on X86 systems using linear regression univariate model. Work in [7] shows that a trained machine learning model from one HPC system can be retrained on only one percent of samples from test (another) HPC system to predict ninety-nine percent of performance data. In this work, we have explored the transfer learning technique for cross-platform or cross-system prediction.

We have identified four questions from similar work in the referenced literature. 1) Which of the univariate and multivariate machine learning models predicts runtime and power with high accuracy? 2) Which type of application compute-bound or memory-bound has higher prediction accuracy? 3) Can we use transfer learning to predict physical system targets (runtime and power) from simulation systems, a cross-system prediction? 4) Can we predict the runtime and power of one type of physical system (Intel Xeon) from other physical systems (Intel Core) using transfer learning, a cross-platform prediction?

The remainder of the paper is organized as follows: Section 2 describes the computer hardware selection and dataset. Section 3 lists machine learning models and techniques that we have used. Section 4 provides results of three scenarios: (a) Prediction accuracy analysis of different machine learning models on each application type and system type dataset (b) Evaluation of results concerning compute and memory-bound applications. (c) Prediction accuracy of cross-platform and cross-system prediction implementation using transfer learning concerning physical and simulated systems. Finally, section 5 provides concluding remarks and work that we plan to continue.

2 DataSet Preparation

2.1 Algorithms Selection for Workload

We have selected applications as workloads for our performance modeling according to their computation and data access patterns categorizing them as compute-bound (CB) and memory-bound (MB). Compute-bound applications such as monte carlo and sha have a higher dependence on processor features than the memory features. On the other hand, memory-bound applications like matrix multiplication and dijkstra depend more on memory features than the processor features. Some applications, such as quicksort and stitch, depends on processor and memory features approximately equally categorizing them as compute-and-memory-bound (CB+MB). Table 1 lists the scientific applications and their respective type CB, MB, or CB+MB we have used in our experiments.

Table 1. Dataset Information and Type

Scientific Applications	System Type	Data Point	System Type	Data Point	Memory/Compute Bound (MB/CB)
Dijkstra	Physical	52	Simulated	362	MB
Matrix Mult.	Physical	519	Simulated	1780	MB
Montecarlocalcpi	Physical	260	Simulated	1365	CB
MSER	Physical	52	Simulated	430	CB
Quick Sort	Physical	672	Simulated	2730	MB+CB
SHA	Physical	52	Simulated	367	CB
Stitch	Physical	52	Simulated	425	MB+CB
SVM	Physical	52	Simulated	390	CB
Tracking	Physical	52	Simulated	425	MB

Table 2. Computer Hardware Models Built in Gem5 Simulator[16]

H/W Class	ISA	CPU Speed	Cores	Mem Type	Mem Access MHz*	L1,L3 Cache Size**	Cnt
1	X86	2-3.5	2-18	DDR4	2400-2666	32,64	50
2	X86	2.8-4.7	1-8	DDR3	1600-1866	16,8	60
3	ARM	1.7-2	4,8	DDR4	1866	32,8	15
4	ARM	1-2.7	2-8	LPDDR2	400-1866	4,3	70
5	ARM	1.1-2.34	1-4	LPDDR3	1600-1866	32,4	35
6	X86	1.3-3.5	2,4	DDR3	1600	32,8	60
7	X86	1.7-3.5	2-18	DDR4	1866-266	32,16	95
8	X86	1.3-3.5	2,4	LPDDR3	1600-2133	32,8	90

*Memory Size range 1GB to 8GB

*L1 Cache Size is in kB and L3 Cache Size is in MB

2.2 Computer Hardware Selection

First, we have built a performance dataset by executing the selected application on simulation-based systems. For the execution of applications on simulated-systems, we have built 475 simulated system models in the Gem5 simulator[16], a widely accepted simulator for architectural research. Out of 475 simulated systems, 355 systems have been built based on x86 instruction set architecture and 120 hardware models based on the ARM instruction set architecture. Table 2 shows the feature set (configuration) of all 475 Gem5 systems. The feature values for each Gem5 system have been used from real hardware systems available today, indicated by H/W class column. Each application is executed on all simulated-systems, as shown in columns three and four in table 1. For each execution, we have collected runtime from Gem5 execution logs and power consumption using McPAT tool[17].

Similarly, we have built a physical systems performance dataset by executing applications on the physical systems with features set shown in table 3. Our selection of six systems includes server systems and general-purpose systems commonly used today. To take advantage of hyper-threaded systems, we have executed processes from one up to two times the number of cores in the system; that is, we have executed 24 processes on a system with 12 cores. We have modified code for each application to integrate it

Table 3. Physical Computer Hardware Models

Sr	ISA	CPU Speed GHz	Cores	Mem Type	Mem Access MHz	Mem Size GB	L1-L3 Cache Size*
1	X86	3	2	DDR3	1600	4	32,4
2	X86	3.2	4	DDR3	1600	4	32,6
3	X86	3.2	4	DDR3	1600	4	32,6
4	X86	3.2	4	DDR4	2400	4	32,6
5	X86	3.168	6	DDR4	2666	16	32,12
6	X86	2.4	12	DDR4	1866	16	32,15

*L1 Cache Size is in kB and L3 Cache Size is in MB
Configuration taken from following models
1. Intel Core i76500U 2. Intel Core i56500 3. Intel Core i53470 4. Intel Core i56500 5. Intel Core i78700 6. Intel Xeon E52620

Table 4. Machine Learning Models

Models Used	Abbreviations	True Multivariate
Linear Regressor	lr	No
Ridge Regr.[8]	rr	No
K-Nearest Neighbor Regressor[9]	knn	No
Gaussian Process Regressor.[10]	gpr	No
Decision Tree Regressor[11]	dt	Yes
Random Forest Regressor[12]	rf	Yes
Extra Trees Regressor[13]	etr	Yes
Deep Neural Network[14]	dnn	Yes

with PAPI tool API[18] to collect runtime and power consumption upon execution on each physical system.

3 Machine Learning Models and Techniques Used

Supervised machine learning algorithms use to perform empirical performance modeling. The machine learning models, use the labeled dataset to learn the function $f_{(x)}$ that maps input X_i to output Y_i. Tables 2 and 3 shows processor and memory features for simulated systems and physical systems which defines input X_i for our machine learning models. The output Y_i for our machine model is two target variables runtime and power, which we collect during the application's execution on simulated and physical systems. We train machine learning models to determine the function $f_{(x)}$ by mapping hardware features X_i to runtime and power, the output Y_i. In the prediction phase, the trained model predicts the values of both runtime and power using systems with input features not utilized during training.

3.1 Models Description

There are two options to build machine learning models considering two output variables runtime and power. The first option is to build a separate machine-learning model for each output called univariate models. The second option is to build a multivariate model where one model predicts both runtime and power simultaneously. We have

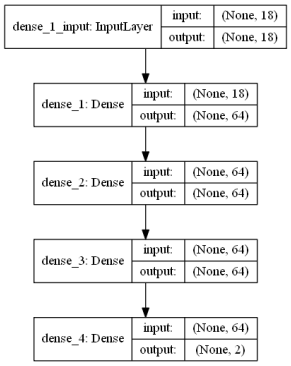


Fig. 1. DNN Model Summary

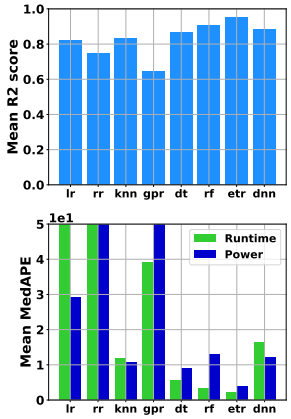


Fig. 2. Mean of R2 and MedAPE values per model across all datasets

selected different classes of machine learning models such as linear models, boosting models, tree-based models, and neural network models; some are univariate, and others are multivariate models. Sklearn library provides linear regression (lr), ridge regression (rr), k-nearest neighbor (knn), and gaussian process regressor (gpr), which can only perform univariate modeling. Linear models (lr and rr here) cannot have a single equation predicting two targets, therefore, implemented as multiple univariate models to predict multiple targets. Furthermore, knn regression returns weighted average of k-nearest neighbors corresponding to each output separately. On the other hand, the decision tree (dt), random forest (rf), extra tree regressor (etr), and deep neural network (dnn) are multivariate models. The table 4 lists all the models that we have used for the experimentation. Fig. 1 shows the configuration of our deep neural network model with the number of layers, number of neurons used at each layer, and parameter details.

3.2 Transfer Learning

Transfer Learning allows the transfer of learning from the already trained model from one problem to another similar problem. The work in [15] describes an excellent example of transfer learning, where knowledge from the pre-trained state of the art BERT model in NLP applied to a variety of tasks by just fine-tuning with an additional output layer. We have used transfer learning to predict cross-platform and cross-systems prediction. We use transfer learning in cross-platform prediction to train the model on a source dataset with one type of instruction-set to predict the target for different instruction-set while transfer-learning is used in cross-system prediction to predict the target of physical systems from simulation-based systems.

4 Experimental Evaluation

In this section, we have shown the experiments performed along with the results to assess the performance of machine learning models for multivariate runtime and power prediction. Furthermore, we also include experiments for cross-platform and cross-system multivariate prediction using transfer learning. For using a dataset in machine learning models, we need to preprocess the dataset. First, we have transformed the categorical features memory-type and instruction set architecture with text data to real value using one-hot encoding. We then have standardized our real-valued dataset using a $StandardScalar()$ function from the Sklearn library, which normalizes each feature with mean of zero and unit variance used in the training and prediction phases of our multivariate models.

To ensure that our machine learning model provides high accuracy for any sample selection in the training and prediction phase, we have used 10-fold cross-validation using the $ShuffleSplit()$ function with a train-test ratio of 80:20 except for transfer learning experiments. Additionally, we have applied grid-search for tuning important hyper-parameters to improve the model accuracy. To evaluate the machine learning model's accuracy, we have used a R^2 score and a median absolute percentage error (MedAPE). We consider mean values of all 10-folds for R^2 score and MedAPE for prediction accuracy evaluation. The model with higher accuracy has low MedAPE and a high R^2 score. Also, a negative R^2 score implies that the model is not a good fit.

In subsequent subsections, we have shown model performance results comparison in different contexts. Firstly, we have shown the comparison of prediction accuracy of all models for multivariate prediction. Secondly, We examine the prediction accuracy for different application types and system types. Lastly, We show the results from cross-platform and cross-system predictions.

4.1 Model Comparison for Multivariate Prediction

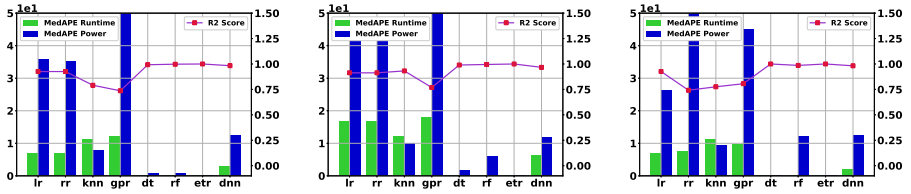
We have trained each machine learning model from table 4 with an 80% dataset and tested on the remaining 20% while measuring metric scores for each prediction. We have taken the mean of runtime and power R^2 score, while MedAPE is used separately each for runtime and power. Mean MedAPE and R^2 score is then combined across all datasets as listed in table 1. Fig. 2 shows the 10-fold cross-validation mean MedAPE and R^2 score for each machine learning model.

4.1.1 Prediction Accuracy of Univariate Models: We have built two separate univariate models, one for runtime and the other for power, for four machine learning models lr, rr, knn, and gpr using sklearn library. Fig. 2 shows the prediction accuracy of these models. We observe that the knn model with mean runtime MedAPE = 11.76%, mean power MedAPE = 10.53% and mean R^2 = 0.83 has the lowest prediction errors among the four univariate models due to its ability to map non-linear function between input hardware features and targets, runtime and power. Due to the non-linear relationship between input hardware features and the targets (runtime and power), linear models lr and rr have higher errors. The gpr model also has low prediction accuracy because the feature set does not follow gaussian curve resulting in poor prediction of the targets.

4.1.2 Prediction Accuracy of Multi-variate Models: The last four machine learning models from Fig. 2, dt, rf, etr, and dnn are truly multivariate models that predict both runtime and power simultaneously. The dt, rf, and etr are tree-based models, while dnn is a neural network model. From the results in Fig. 2, it is observed that the etr model with mean runtime MedAPE = 2.14%, mean power MedAPE = 3.94% and mean $R^2 = 0.95$ has the highest prediction accuracy among all other models including dnn. The etr higher accuracy is due to tree-based models' ability to build decision rules with optimal splitting criterion at each node rather than considering correlation like other linear models. The etr performs better than dt and rf because the splits in the etr model are random, which leads to more diversified tree formation and works well in case of noisy features. We believe that due to the availability of a limited number of samples, dnn has a lower accuracy than tree-based models. One important observation is that the multivariate models outperform univariate models.

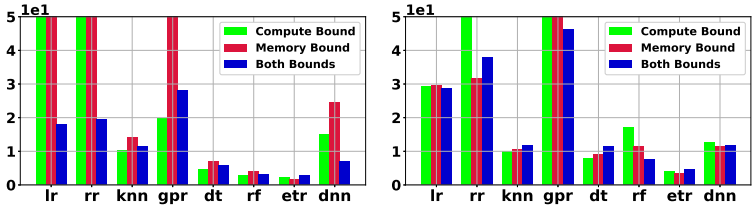
4.2 Multivariate Prediction Accuracy per Application Types and System Types

In this section, we first compare the prediction accuracy of the multivariate prediction model for different application types, compute-bound, memory-bound, and compute-and-memory-bound for the respective applications of these types shown with an example in Fig. 3. Secondly, we compare the prediction accuracy of simulation-based systems and physical systems.



(a) SHA(CB) (b) Dijkstra(MB) (c) Stitch(CB+MB)

Fig. 3. R^2 score and MedAPE for each of the example Application Types for Simulated Systems



(a) Runtime Error (b) Power Error

Fig. 4. Comparing Model Performance with respect to Application Type

4.2.1 Prediction Accuracy per Application Type: We have performed multivariate performance and power prediction for applications from table 1, and results have been combined according to their application types, compute-bound (CB),

memory-bound (MB) or compute-and-memory-bound (CB+MB). Fig. 4 shows the combined mean MedAPE across all applications per application type. We have three observations from the results. First, the tree-based multivariate models dt, rf, and etr with mean runtime MedAPE = 3.77% and mean power MedAPE = 8.47% outperform all other models, including deep neural network. Second, for tree-based models, the power prediction error is higher than the runtime prediction error due to higher variations in power values than runtime. Third, for non-linear models, the runtime prediction accuracy for compute-bound applications is higher than memory-bound applications due to the higher number of data points account for simulation data points. The simulated systems use the only supported out-of-order processor in the Gem5 simulator, resulting in lower runtime variations for compute-bound applications. On the other hand, the power prediction accuracy for compute-bound applications is analogous to memory-bound applications because the power consumption of memory units is insignificant compared to processor power consumption, which results in similar power variations in all types of applications.

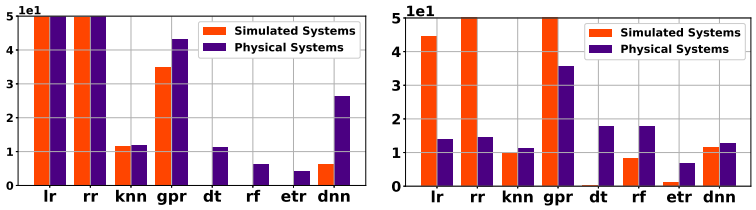


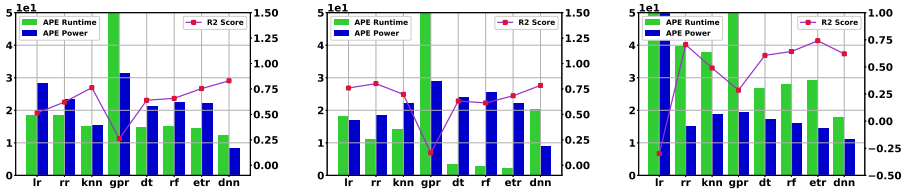
Fig. 5. Comparing Model Performance on Physical and Simulated Systems

4.2.2 Prediction Accuracy per System Type: Each scientific applications from table 1 have been executed on both simulated systems and physical systems from tables 2 and 3 respectively. In this section, we compare the prediction accuracy for both system types, simulated systems, and physical systems. Fig. 5 shows mean MedAPE for each model across all applications. We observe that for simulated systems across tree-based models, the prediction accuracy with mean runtime MedAPE = 0.096% and mean power MedAPE = 3.21% is higher than physical systems with mean runtime MedAPE = 7.31% and mean power MedAPE = 14.10%. Lower prediction accuracy in non-deterministic physical systems is because of the resource contention among multiple processes. On the contrary, the simulation environment is deterministic, resulting in fewer variations in simulation systems than physical systems. Similar to runtime and power error comparison for application types, for different systems types also the power prediction errors are higher than runtime prediction errors due to higher variance in power.

4.3 Cross-Platform Multivariate Prediction using Transfer Learning

We have carried out cross-platform in both simulation and physical environment. We transfer the learning from the trained machine learning model from the source platform (platform-1) and apply it to the target platform (platform-2). We have selected Quick Sort application to perform cross-platform prediction due to the highest number of simulated systems and physical systems data points with 2730 and 672 respectively

among all applications, as shown in table 1. In section 4.3.2, we perform cross-platform prediction in simulation environment by separating data points into three platforms, Intel Core with 1440 data points used as source platform (platform-1) and ARM with 720 data points or AMD with 570 data points one of which is used as a target platform (platform-2). Similarly, in section 4.3.2, we have performed cross-platform prediction in the physical environment by separating data points into two platforms, Intel Core, with 360 data points used as source platform (platform-1) and Intel Xeon with 312 data points used as a target platform (platform-2).



(a) Intel to ARM (b) Intel to AMD (c) Intel-Core to Intel-Xeon

Fig. 6. Transfer Learning on Qsort Simulated Dataset (a and b) and Qsort Physical Dataset (c)

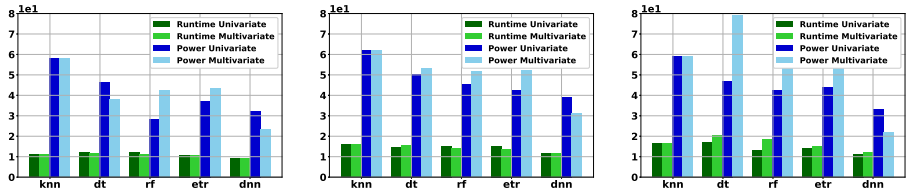
4.3.1 Intel Core to ARM and AMD Cross-Prediction: In this section, we use transfer learning to predict targets runtime and power for ARM or AMD systems (platform-2) using a trained model from Intel Core systems (platform-1) hardware features and both targets in a simulation environment. For transfer learning implementation, We have used 100% data of platform-1 and 1% data of platform-2 to train the machine learning model and predicted targets for the remaining 99% data of platform-2. We train all models except dnn using combined data of 101% from platform-1 and platform-2 due to the unavailability of partial_fit function for all machine learning models in the sklearn library. In the case of dnn, we first train the dnn model using 100% of the platform-1 dataset, and then we freeze all other layers except the last two, which we retrained using 1% of the platform-2 dataset. The results for cross-platform prediction in simulation environment are shown in Fig. 6a and 6b. From Fig. 6a, it is observed that for Intel Core to ARM prediction, dnn with R^2 score = 0.83 including runtime APE = 12.47% and power APE = 8.32% outperforms all other models. We have also achieved mean runtime APE = 14.24% and mean power APE = 18.61% for dnn and tree-based models combined. From Fig. 6b, it is observed that for Intel Core to AMD tree-based models with mean runtime APE = 2.89% outperform other models, whereas, dnn with mean power APE = 8.98% outperforms other models in case of power prediction.

4.3.2 Intel Core to Intel Xeon Cross-Predicion: In this section, we use transfer learning to predict Intel Xeon systems (platform-2) targets using Intel Core systems (platform-1) for physical systems. For transfer learning implementation, We have used 100% data of platform-1 and 10% data of platform-2 to train the machine learning model and predicted targets for the remaining 90% data of platform-2. We have used 10% data from platform-2 for training because the use of smaller than 10% data points resulted in higher errors. We train all models except dnn using combined data of 110% from platform-1 and platform-2. In the case of dnn, we first train the dnn

model using 100% of the platform-1 dataset, and then we freeze all other layers except the last two, which we retrained using 10% of the platform-2 dataset. Fig. 6c shows the results for cross-platform prediction in the physical environment. We observe that dnn with runtime APE = 18.04% and power APE = 11.08% outperforms all other models.

4.4 Cross-System Multivariate and Univariate Prediction using Transfer Learning

In this section, we have shown cross-systems prediction results in which we predict the target runtime and power of physical systems from the machine learning model trained on the simulation dataset. To compare the prediction accuracy of different application types, we have performed cross-systems experiments on svm, a compute-bound application, tracking, a memory-bound application, and stitch compute-and-memory-bound (CB+MB) application. We have performed another experiment for comparing the prediction accuracy of univariate and multivariate models implemented using the same machine learning algorithms for cross-system prediction. We have implemented univariate models using separate machine learning models for runtime and power, whereas we implement multivariate models using a single machine learning model that predicts both targets. For transfer learning implementation, We have used a 100% simulation dataset and 10% of physical dataset for each application to train the machine learning model and predicted targets for the remaining 90% physical system's data. We train all models except dnn using combined data of 110% from simulation and physical dataset. In the case of dnn, we first train the dnn model using 100% of the simulation dataset, and then we freeze all other layers except the last two, which we retrained using 10% of the physical dataset.



(a) SVM (CB) (b) Tracking (MB) (c) Stitch (CB+MB)

Fig. 7. APE error for Simulation to Physical Systems Prediction using Transfer Learning

4.4.1 Prediction Accuracy per Application Type: Fig. 7 shows runtime and power prediction errors from cross-system experiment for univariate and multivariate models of the three selected applications svm, tracking and stitch. It is observed from Fig. 7a, Fig. 7b and 7c that for all the three application types compute-bound (svm), memory-bound (tracking) and compute-and-memory-bound (stitch) dnn multivariate model outperforms all other models with svm runtime APE = 9.10% and power APE = 23.38%, tracking runtime APE = 11.81% and power APE = 31.03% and stitch runtime APE = 12.11% and power APE = 22.05%. We also observe that runtime errors between univariate and multivariate dnn implementations have a similar range, whereas power errors are lower in multivariate dnn implementation.

4.4.2 Prediction Accuracy of Univariate and Multivariate Implementation: Fig. 7 shows the results from the univariate and multivariate implementation of selected machine learning models. We observe from the plots that the univariate and multivariate implementations of dt, rf, etr, and dnn have different prediction accuracy categorizing them as multivariate models. On the other hand, knn univariate and multivariate implementation gives precisely the same prediction accuracy strengthening the argument that knn is not truly a multivariate model.

5 Conclusion And Future Work

In this paper, we have experimented with different machine learning models for a wide variety of datasets consisting of three different application types and two different system types. We have done rapid experimentation to compare and conclude important observations to give an in-depth analysis of multivariate modeling. Furthermore, we have also used transfer learning approach for cross-domain predictions. The conclusions that can be made from the experiments performed are as follows:

1. Multivariate Tree-based models namely dt, rf and etr outperform all other models including deep learning models. Also, multivariate models perform better than univariate models if the training and prediction are made on the same application dataset.
2. Among univariate models, knn gives the best performance with less than 12% percentage errors. Also, among multivariate models, etr gives the best performance with less than 4% percentage errors.
3. The errors in power prediction are more than runtime prediction due to higher variations in power.
4. Simulated systems give less multivariate prediction errors than physical systems due to the deterministic nature of simulated systems.
5. For cross-platform prediction, dnn multivariate model and tree-based models outperform all other models for power and runtime, respectively.
6. For cross-system prediction, dnn multivariate outperforms all other uni or multivariate models.

Our future work will include more experimentation for building efficient multivariate models to include the cost in addition to performance and power and contribute more data sets to the research community. We also plan to experiment with the data augmentation technique to add more data to physical systems dataset.

References

1. S. Kawaguchi and T. Yachi, "Adaptive power efficiency control by computer power consumption prediction using performance counters," 2014 International Power Electronics Conference (IPEC-Hiroshima 2014 - ECCE ASIA), Hiroshima, 2014, pp. 3959-3965. doi:10.1109/TIA.2015.2466687
2. Z. Lai, K. T. Lam, C. Wang and J. Su, "PoweRock: Power Modeling and Flexible Dynamic Power Management for Many-Core Architectures," in IEEE Systems Journal, vol. 11, no. 2, pp. 600-612, June 2017. doi:10.1109/JSYST.2015.2499307
3. A. Sîrbu and O. Babaoglu, "Predicting system-level power for a hybrid supercomputer," 2016 International Conference on High Performance Computing & Simulation (HPCS), Innsbruck, 2016, pp. 826-833.

4. J. Ma, G. Yan, Y. Han and X. Li, "An Analytical Framework for Estimating Scale-Out and Scale-Up Power Efficiency of Heterogeneous Manycores," in IEEE Transactions on Computers, vol. 65, no. 2, pp. 367-381, 1 Feb. 2016.
5. Newsha Ardalani, Clint Lestourgeon, Karthikeyan Sankaralingam, and Xiaojin Zhu. 2015. Cross-architecture performance prediction (XAPP) using CPU code to predict GPU performance. In Proceedings of the 48th International Symposium on Microarchitecture (MICRO-48). Association for Computing Machinery, New York, NY, USA, 725–737.
6. Zheng, X., John, L.K. & Gerstlauer, A. LACross: Learning-Based Analytical Cross-Platform Performance and Power Prediction. Int J Parallel Prog 45, 1488–1514 (2017).
7. P. Malakar, P. Balaprakash, V. Vishwanath, V. Morozov and K. Kumaran, "Benchmarking Machine Learning Methods for Performance Modeling of Scientific Applications," 2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS), Dallas, TX, USA, 2018, pp. 33-44.
8. Arthur E. Hoerl & Robert W. Kennard (1970) Ridge Regression: Biased Estimation for Nonorthogonal Problems, Technometrics, 12:1, 55-67
9. Christopher M. Bishop. 2006. Pattern Recognition and Machine Learning (Information Science and Statistics). Springer-Verlag, Berlin, Heidelberg.
10. Rasmussen C.E. (2004) Gaussian Processes in Machine Learning. In: Bousquet O., von Luxburg U., Rätsch G. (eds) Advanced Lectures on Machine Learning. ML 2003. Lecture Notes in Computer Science, vol 3176. Springer, Berlin, Heidelberg
11. Krzywinski, M., Altman, N. Classification and regression trees. Nat Methods 14, 757–758 (2017).
12. Breiman, L. Random Forests. Machine Learning 45, 5–32 (2001)
13. Geurts, P., Ernst, D. & Wehenkel, L. Extremely randomized trees. Mach Learn 63, 3–42 (2006).
14. Wang, Haohan, and Bhiksha Raj. "On the origin of deep learning." arXiv preprint arXiv:1702.07800 (2017).
15. Devlin, Jacob, et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
16. Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. 2011. "The gem5 simulator". SIGARCH Comput. Archit. News 39, 2 (August 2011), 1–7. .
17. Sheng Li, Jung Ho Ahn, Richard D. Strong, Jay B. Brockman, Dean M. Tullsen, and Norman P. Jouppi. 2009. McPAT: an integrated power, area, and timing modeling framework for multicore and manycore architectures. In Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO 42). Association for Computing Machinery, New York, NY, USA, 469–480.
18. Terpstra, D., Jagode, H., You, H., Dongarra, J. "Collecting Performance Data with PAPI-C", Tools for High Performance Computing 2009, Springer Berlin / Heidelberg, 3rd Parallel Tools Workshop, Dresden, Germany, pp. 157-173, 2010.