GPU Power Prediction via Ensemble Machine Learning for DVFS Space Exploration

Bishwajit Dutta
Virginia Tech
Blacksburg, Virginia 24061
bdutta@vt.edu

Vignesh Adhinarayanan
Virginia Tech
Blacksburg, Virginia 24061
avignesh@vt.edu

Wu-chun Feng
Virginia Tech
Blacksburg, Virginia 24061
wfeng@vt.edu

ABSTRACT

A software-based approach to achieve high performance within a power budget often involves dynamic voltage and frequency scaling (DVFS). Consequently, accurately predicting the power consumption of an application at different DVFS levels (or more generally, different processor configurations) is paramount for the energy-efficient functioning of a high-performance computing (HPC) system. The increasing prevalence of graphics processing units (GPUs) in HPC systems presents new multi-dimensional challenges in power management, and machine learning presents an unique opportunity to improve the software-based power management of these HPC systems. As such, we explore the problem of predicting the power consumption of a GPU at different DVFS states via machine learning. Specifically, we perform statistically rigorous experiments to quantify eight machine-learning techniques (i.e., *ZeroR*, *simple linear regression (SLR)*, *KNN*, *bagging*, *random forest*, *sequential minimal optimization regression (SMOreg)*, *decision tree*, and *neural networks*) to predict GPU power consumption at different frequencies. Based on these results, we propose a hybrid ensemble technique that incorporates SMOreg, SLR, and decision tree, which, in turn, reduces the mean absolute error (MAE) to 3.5%.

1 INTRODUCTION

Power and energy efficiency have emerged as first-order design constraints in high-performance computing (HPC) systems. For the DOE Office of Science, an exascale supercomputer needs to operate under 20 MW [6] while meeting or exceeding its performance requirements. The increasing prevalence of *dark silicon* [13] and the emerging *hardware-overprovisioned* supercomputers[1] [26] have made it harder to safely operate these systems under their respective power budgets. This has necessitated the introduction of *power-management* systems such as Intel's Running Average Power Limit (RAPL) [11] and several research prototypes [9, 14, 21, 31] that are capable of enforcing strict *power caps*. Central to the functioning of such a power management system is the ability to predict the power consumption of an application at different processor configurations (typically different operating frequencies or DVFS states, but not necessarily limited to that). The idea is to configure the system for best performance while ensuring that *power caps* are not violated.

While several models for DVFS-based power prediction have been previously proposed [12], two emerging trends motivate the need for our work. First, graphics processing units (GPUs) are increasingly common in HPC. The latest November 2017 ranking of the Top500 has 101 GPU-accelerated systems on the list [2]. Second, recent interest and advances in machine-learning techniques (including neural networks) has necessitated a re-examination of data-driven modeling techniques in many areas of computing, including system design. As such, we investigate the applicability of several machine-learning techniques in predicting the power consumed by a GPU at different voltage-frequency settings (or P-states). Our contributions in this paper include the following:

- **Accurate power prediction of a GPU at different DVFS states.** We explore eight (8) prediction techniques in order to predict the GPU's power consumption at different DVFS states. To the best of our knowledge, this study is broader than any other power prediction study, including those carried out for CPUs.
- **Statistically rigorous comparison of different machine-learning techniques**. Unlike previous studies that *only* compared the *mean error* of a few modeling techniques, we employ *Tukey's HSD* test for comparing multiple approaches in a statistically rigorous manner at once.
- **Design of an ensemble approach for GPU power prediction**. Based on our analysis, we propose different ensemble designs of machine-learning techniques and evaluate them to leverage the relative strengths of each. While such ensemble designs are popular in other domain areas, we are among the first to use it for DVFS prediction.

Our statistically rigorous experiments resulted in several findings. We highlight the most prominent ones below.

- *Simple linear regression (SLR)*, *sequential minimal optimization (SMO) regression*, *K-nearest neighbors (KNN)*, and *reduced error pruning (REP) tree* can be statistically verified to produce better prediction accuracy than the baseline method *ZeroR*.
- *Neural networks* consistently showed lower accuracy than other methods. This is due in part to the few data points being used in the training set used for modeling. *Composite methods*, which combine results from several regression trees, produce consistent results (i.e., about the same error for different configurations). Thus, we believe these methods are more appropriate for power management systems that are required to provide guarantees.
- Our ensemble method, which combines *SMO regression*, *SLR*, and *REP tree* shows the best accuracy. The average error for this method is 3.5% with a maximum error of 11%.

The rest of the paper is organized as follows. In §2, we motivate our power prediction work via several use cases. Next, we present our methodology to evaluate existing machine-learning techniques in §3, and experimental results and analysis in §4. Based on our analysis, we propose an ensemble method for GPU power prediction and present and analyze the associated results in §5. In §6, we

[1] In an overprovisioned supercomputer, more compute nodes are added than what the nameplate capacity can support.

distinguish our work from other related work. Finally, we present our conclusions in §7.

2 MOTIVATION

Our paper aims to predict the power consumption of an application at different system configurations (namely, DVFS states). Accurately predicting the power consumption of a computing system is useful in several scenarios, some of which are described below.

2.1 Use Cases

Power Capping. Modern computing systems can draw more power than they can safely sustain. At the node level, the processor may end up consuming more power than its rated thermal design power (TDP) for some applications [13]. At the supercomputer level, machines may be built that can draw more power than its nameplate capacity (i.e., hardware overprovisioned) [26].

To illustrate the need for power capping, we present the power profile of an application on a reference processor operating at three different frequencies *f_1*, *f_2*, and *f_3* in Fig. 1. Note that this application consists of three distinct phases: a low-power phase, followed by a high-power phase, which in turn, is followed by another low-power phase. With a power cap of 120 W enforced on this system, a conservative approach would set the machine at *f_2* (or lower) where it can be guaranteed that the power cap is *never* violated. However, if a power predictor can accurately predict the power consumption at different frequencies, the power management system may choose to operate at *f_1* for phase I, *f_2* for phase II, and back to *f_3* for phase III.

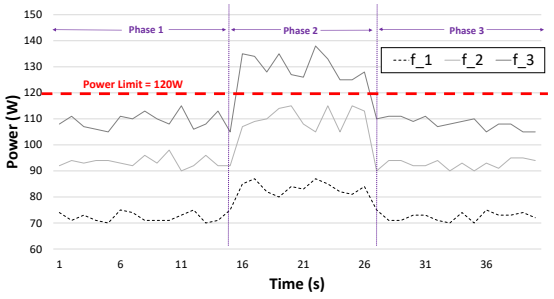

Figure 1: Power consumption of an application at different frequencies

Energy-Performance Trade-off Analysis. The energy cost of operating today's HPC systems is around 25% of their acquisition cost [16]. For systems built with older technology, the energy cost may even exceed the operating cost. Reliable prediction of power consumption and performance can help in judicious trade-offs between energy and performance to keep the total cost of operation (TCO) low. An alternative approach to operating an application at an ideal frequency is to exhaustively explore the configuration space. However, with numerous configuration options available in today's HPC system (e.g., different DVFS knobs for different components of a processor), the energy cost of finding the correct configuration itself could exceed the energy cost of running the application. Thus, automatically predicting power consumption at different configuration points via modeling becomes indispensable.

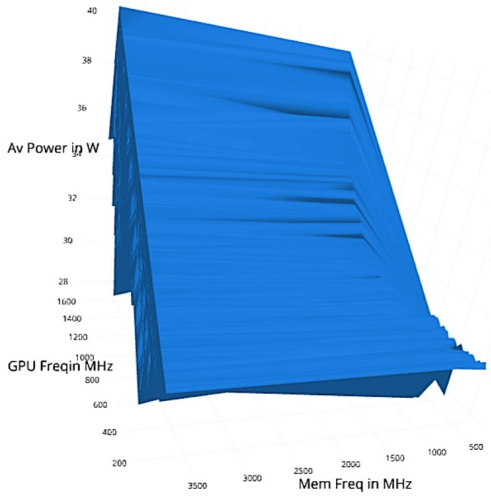

Figure 2: Scaling surface for average power on the SHOC `Triad` application

Machines without Built-in Power Meters. Even in scenarios where exhaustive exploration is viable (e.g., a mission-oriented computing system where only a handful of applications are run), a model-based exploration is useful in many cases. For example, consider a popular GPU platform used in computer vision – NVIDIA's Tegra TK1. The TK1 lacks built-in GPU power meters, as is the case with many consumer-grade GPUs. Direct techniques to measure the GPU power consumption on these platforms can be intrusive and time-consuming. On the other hand, a model-based approach requires instrumentation only once for collecting the training data set, after which commonly available profiling tools such as `nvprof` can be used to estimate the power consumption.

Design Space Exploration: The ability to accurately estimate power and performance at different hardware configurations can accelerate design space exploration, e.g., estimating the power consumption when the L2 cache size is doubled (and hence its utilization level is halved for an application) while keeping all other architectural aspects same. Low-level power estimation techniques, while accurate, are generally too slow for fast design-space pruning. Thus, research into off-line high-level prediction techniques can lead to cheaper and faster design space exploration.

2.2 Challenges

Previous work, which modeled GPU power consumption at different DVFS states via machine learning, operated *only* at the level of a GPU kernel [30]. While our machine-learning approach differs along three different fronts: (1) modeling parameters, (2) breadth of techniques evaluated, and (3) target metrics modeled, the most fundamental difference is at the level of problem formulation. Specifically, while previous work tries to predict the power consumption of individual GPU kernels only, we take on the more challenging

problem of predicting the power characteristics of the application as a whole. In [30], the individual kernels can exhibit irregular scaling properties with respect to frequencies. This scaling behavior gets more complex when an application has several kernels, each of which is scheduled in parallel and has different computational and memory bounds. For example, Fig. 2 shows the power and performance scaling surface for the `Triad` application from the SHOC benchmarks. We observe that the power and performance scaling is not linearly dependent on frequencies but is a complex interaction of frequencies and resource utilization levels. Thus, it is harder to find lines, curves, or models that fit the data, hence warranting a new look at the GPU power prediction problem.

3 METHODOLOGY

Here we present an overview of our approach, followed by details on data collection, modeling techniques, and evaluation methods.

3.1 Overview

We divide our approach into two phases – training phase and testing phase – as illustrated in Fig. 3.

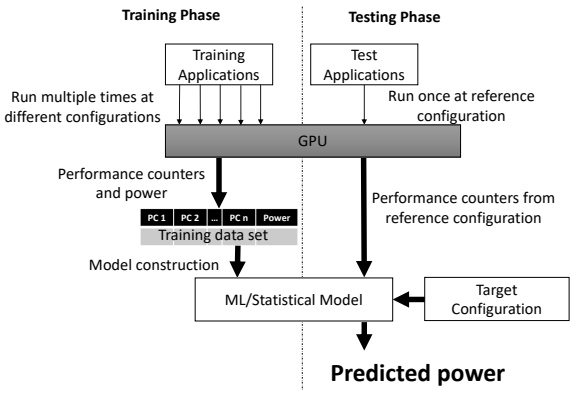

Figure 3: Overview of our approach

Training Phase. In the training phase, we run our training applications (Table 1) on the target platform (an NVIDIA Quadro P4000 GPU) and collect the average power consumption and several performance counters at different configurations. Some pertinent details of the target platform are presented in Table 2. Note that there are 288 possible combinations of GPU core frequency and memory frequency for this platform. As such, in the training phase, each application has to be run a few hundred times in order to collect the necessary data for modeling. However, this data collection is only a one-time cost. Once all the data are collected, the dataset is fed as input to the machine learning model which is initially tuned with the *training* data itself.

Testing Phase. After the model is constructed, it is tested using a different set of applications which are listed in Table 3. We collect performance counters for the test application on a reference configuration and predict the power consumption at any target configuration using our machine learning model.

Table 1: Training Applications

Source	Applications
SHOC	Device Memory, Sort, Scan, SpMV, Reduction
Rodinia	Huffman, HotSpot
CUDA SDK	Matrix Multiplication, Transpose, Radix Sort, Interval, Line of Sight, Merge Sort

Table 2: NVIDIA Quadro P4000 Hardware Details

Hardware Feature	Hardware Value
GPU Architecture	Pascal
Number of GPU Cores	1792
GPU Frequencies (MHz)	1708 - 139
Memory Frequencies (MHz)	3802, 810 and 405
Peak Power	105 W

Table 3: Test Applications

Source	Applications
SHOC	FFT, MaxFlops, BFS, Triad
Rodinia	Stream Cluster, Gaussian
CUDA SDK	Quad Tree, Scalar Product, Image Segmentation, Eigen Values

3.2 Data Collection

During the training phase, we profile the target GPU's power consumption every 100 ms using its built-in power meter via `nvprof`. Applications with a short execution time are repeated until they run on the GPU for at least 30 seconds. As the idle power of the GPU is known to be affected by its operating temperature [5], we let the GPU to cool to its initial temperature between each successive run.

Hardware performance counters are profiled using `nvprof` during both the training and testing phases. The list of performance counters profiled include utilization factors for the various components of the GPU: (1) DRAM (2) instruction issue slot (3) L2 cache (4) texture cache (5) texture unit (6) special functional unit (SFU) (7) load/store unit (8) control unit (9) single-precision (SP) unit (10) double-precision (DP) unit. These performance counters were chosen to represent every major component within the GPU (see Fig. 4 which shows a representative block diagram of our target GPU for comparison). The only major unit left unrepresented is the register file which does not have a distinct performance counter associated with it. However, we believe that the utilization of other units should track register file utilization as well.

While some previous studies used correlation analysis to choose performance counters [5], we believe such an approach will not help choose the best set of variables to construct the model. For instance, we found that the texture units are rarely used and hence showed a poor correlation with power. However, these units do consume a non-trivial amount of power in the limited number of cases where they are exercised. Therefore, we rely on our knowledge of the underlying architecture to select performance counters rather than employing statistical techniques.

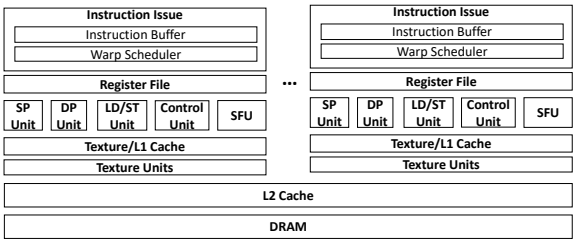

Figure 4: Simplified block diagram of NVIDIA Pascal GPU

In our experimental setup, while power is collected at the application level, the performance counters (i.e., utilization levels) are profiled at the kernel level by `nvprof`. Thus, we aggregate the kernel-level utilization metrics into an application-level metric:

Application level utilization of a resource =

$$\frac{\sum_{i=1}^{n}(Resource\ utilization\ level\ of\ kernel_i * Time\ spent\ in\ kernel_i)}{\sum_{i=1}^{n} Time\ spent\ in\ kernel_i}$$

3.3 Modeling Techniques

Initially, we study eight different machine-learning techniques—`ZeroR`, simple linear regression (`SLR`), `KNN`, `bagging`, `random forest`, sequential minimal optimization regression (`SMOreg`), `decision tree`, and `neural networks`—to predict GPU power consumption at different frequencies. These techniques cover a breadth of approaches in predicting continuous variables and forms the fundamental building blocks in other prediction problems and hence were chosen for our initial comparison study. We use these fundamental building blocks to construct ensemble models which are described in Section 5. A short summary of the base techniques is presented in Table 4. While we used `R` and `weka` [15] software for evaluating the accuracy of the various models, we use terminology from weka for the sake of consistency. The specific parameter values used for the various approaches are also presented in Table 4. These values were determined empirically in order to minimize error within the *training* dataset and *not* the *test* dataset.

3.4 Model Evaluation

We use `ZeroR` algorithm as the baseline to determine the efficacy of the other *seven* methods. This technique has no prediction capability and simply predicts the power of a new application as the mean power consumption of all test applications. The mean absolute error (MAE) for this technique is compared with the MAE value of other techniques and is calculated as follows:

$$MAE\ \% = \frac{100}{n} * \sum_{i=1}^{n}\left(\frac{|predicted\ value_i - actual\ value_i|}{actual\ value_i}\right)$$

Tukey's HSD test is then used to determine the statistical significance of the results.

4 RESULTS

Here we present the accuracy of the various machine-learning (ML) techniques with respect to the mean absolute error (MAE) percentage. A lower MAE indicates a more accurate technique.

4.1 Overview of Results

Fig. 5 shows the error distribution for all the techniques across all test applications and frequencies. For readability, we present this information in two different graphs. The first graph compares `SLR`, `SMOreg`, and `NeuralNet` against the baseline `ZeroR`. The peak of the distribution gives the most commonly seen errors. The baseline technique, `ZeroR`, often produces an error around 20%. This means, if we design a *power capping* system based on this method, we need to provide a safety margin of at least 20% which can be very inefficient. `SMOreg` and `SLR` both perform better with most common errors around 12% and 14%, respectively. The `NeuralNet` method produces significantly worse results with a common error close to 30%. This is to be expected given that the neural network technique needs numerous data points to train the model, which is not the case with our training dataset. Looking at the tail of the distribution, we find that the maximum prediction error is around 46% for `SMOreg`, with other models not far behind with maximum errors around 40%. Analyzing further, we found that most of these high error points come from outlier applications described in §4.2.

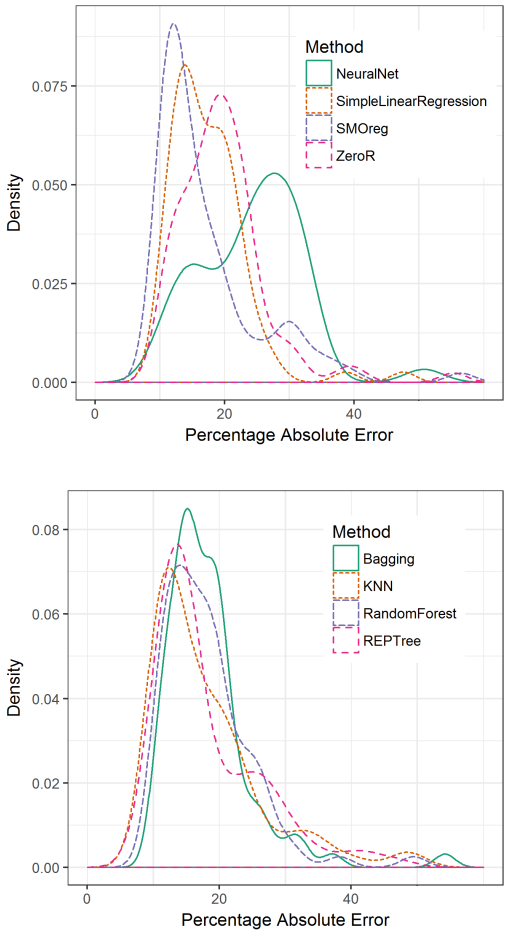

Figure 5: Error distribution for ML techniques

Table 4: Base machine learning (ML) and statistical techniques and their parameter values

Method	Description	Parameters
ZeroR	Predicts the *mean* power consumption across all applications used in the training data set. Used as a baseline for comparison.	N/A
SLR	Finds a straight line that predicts the *response* variable Y as a function of *predictor* variable X.	N/A
SMOreg	Finds a curve (unlike SLR which finds a straight line) that explains the relationship between the predictors and the response variable.	C=1.0, PolyKernel, RegSMOImproved
KNN	Identifies K closest data points in the training dataset and calculates a distance-weighted average of these data points.	K=5, CrossVal=true, 1/distance weighting, Algorithm=LinearNNSearch
REPTree	Constructs a decision tree and makes prediction based on that.	maxDepth=-1, minVarianceProp=0.001
Bagging	Multiple decision trees are created by splitting the training data. Each tree is used to make a distinct prediction and the final value is determined by a voting mechanism.	bagSizePercent=50, Classifier = REPTree
RandomForest	Similar to *bagging*, prediction is based on voting from different decision trees. However, here the training dataset is split by attributes initially before constructing the decision trees.	bagSizePercent=50, maxDepth=0
NeuralNet	Learning system modeled on human brain, where many highly interconnected artificial neurons work together to make predictions. Each neuron takes input $Y = (w^T X + b)$, where w is the weight vector and b is the bias. The weights of the network are learned via the gradient descent technique.	layers=3, learningRate=0.3, momentum=0.2

The second graph shown in Fig. 5 compares the error distribution of Bagging, KNN, RandomForest, and REPTree methods. The graphs show that the Bagging technique performs slightly poorly compared to the three other methods. KNN, RandomForest, and REPTree show significant overlap in their distribution with a most common error of around 14% which is the same as SLR from the first graph. RandomForest appears to produce more uniform predictions as it contains only one peak in the distribution. The other methods contain one peak and one or more ridges in comparison. The tail distribution is largely indistinguishable from one another.

Fig. 6 shows the mean absolute error (MAE) percentage and the error bars, not including outlier applications described in Section 4.2. SMOreg shows the best accuracy with respect to MAE at 4.5%, followed by REPTree and KNN, each with a MAE of 5.5%. SLR has a MAE of 6.0%, and RandomForest and Bagging each have a MAE of about 7.5%. NeuralNet performs the worst with a MAE of 15% due largely to the need for a huge training data set for it to be an effective predictor. With respect to maximum error, SLR is best with 15% error, followed by Bagging at 17% and SMOreg at 20%. KNN delivers the worst results as it only considers nearest points for prediction, which can be inaccurate due to the complex relationships between the predictors and response variables.

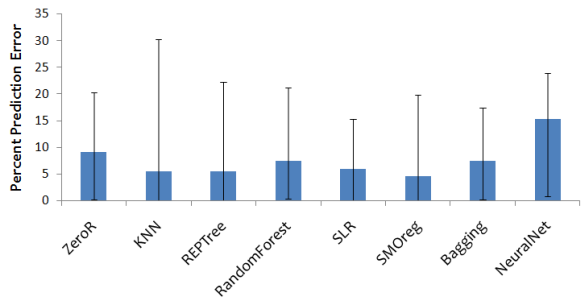

Figure 6: Percent prediction errors for different algorithms

4.2 Results in Detail

Next, we delve into the application results and the accuracy of the different methods with different GPU configurations.

Application Results Fig. 7 shows the MAE for the various test applications and machine-learning (ML) techniques. We observe that the MAE is higher for applications that stress hardware resources to the maximum, i.e., (1) MaxFlops, a compute-bound application [10], (2) FFT, a memory-intensive application requiring higher memory bandwidth [10], and (3) StreamCluster, a PCIe bandwidth-bound application [8]. While the relatively poor accuracy seen for these applications is due to the lack of suitable hardware counters (e.g., no performance counter for PCIe utilization in modern GPUs), most of the relatively less accurate results can be explained by coverage (or lack thereof) of the test applications. A methodological approach towards selecting test applications for modeling [4] can address this in the future (but is outside the scope of this paper). Furthermore, equipping future generations of GPUs with more appropriate counters can help in several power-management techniques.

Frequency Sensitivity Fig. 8 shows the MAE percentage for the various machine-learning techniques with different GPU configurations. We observe that NeuralNet consistently delivers the worst predictions across different GPU-memory frequency combinations. The MAE for this method exceeds 16% for *four* (4) out of the *eight* (8) combinations.

ZeroR gives the worst results at extreme frequencies with MAEs of 15% and 11%, respectively. This is because it always predicts the mean of the training data set which will correspond to the power consumption at the middle frequencies.

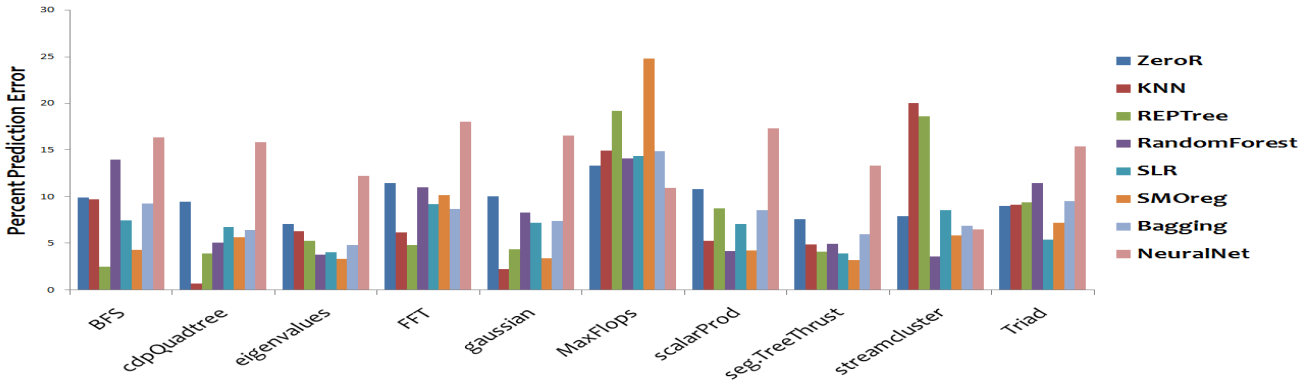

Figure 7: Mean absolute error (MAE) percentage for test applications

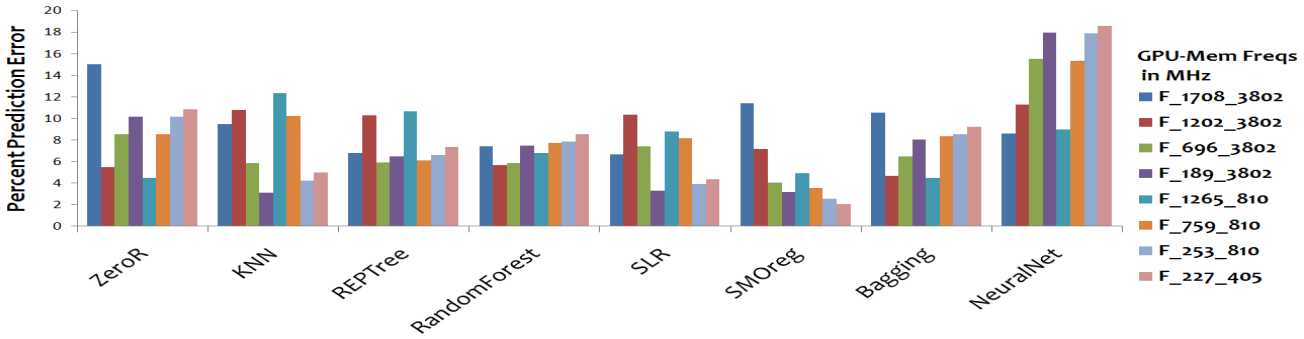

Figure 8: Mean absolute error (MAE) percentage for different GPU configurations

`RandomForest` proves to be a reliable predictor by consistently showing an average error of around 7.5% for all the configurations. This is because this method bases its predictions from multiple decision trees, each of which is specially optimized for the various subspaces of the GPU configuration space.

`SMOreg` shows greater accuracy at lower frequencies. While its average MAE is 11% at the highest frequency, it drops significantly to only 2% at the lowest frequency. Extending the idea of `RandomForest` (which combines different decision trees together), we believe that we can improve the prediction accuracy by combining the different methods together, as explored in §5.

Statistical Significance of Results. In order to statistically compare the prediction accuracy of the different ML algorithms, we use Tukey's `HSD` (honest significant difference) test. This test gives us the pairwise statistical difference between the mean error values predicted by different algorithms.

Fig. 9 presents the results of Tukey's `HSD` test. The bars show the 95% confidence interval between the lowest and highest estimate of the difference between percentage MAE of algorithm pairs. For example, for the comparison between `ZeroR` and `SMOreg`, we can say with 95% confidence that the MAE difference between `ZeroR` and `SMOreg` is 2% at the least and 7.5% at the most. When the lines in this graph cross the *zero* axis, it means that the difference in the prediction error is not significant enough to say that one method is better than the other (i.e., the result is not statistically significant). From Fig. 9, we also observe that `SMOreg`, `SLR`, `KNN`, and `REPTree` statistically exhibit better accuracy than the baseline `ZeroR` and that `NeuralNet` performs worse than every other ML technique.

5 AN ENSEMBLE METHOD FOR GPU POWER PREDICTION

Power-prediction problems in other areas (e.g., traffic forecasting [29]) have achieved accurate and more stable results by creating an ensemble algorithm from several base methods. Here we study the applicability of such ensemble methods for the GPU power-prediction problem.

5.1 Methodology

Fig. 10 illustrates our approach towards creating an ensemble predictor. In this approach, each base predictor makes a prediction (in the figure, `SMOreg`, `SLR`, and `RandomForest` predicts p_1, p_2, and p_3, respectively) and the ensemble predictor simply calculates the weighted average of the base predictions. Despite the simplicity of this approach, it has proved to be more accurate than complex voting mechanisms in many related problems [1, 29].

There are several approaches to chose the base methods to construct the ensemble predictor (e.g., lowest mean error, lowest maximum error, least variance, etc.). In this paper, we rank the base methods in increasing order of mean absolute error (MAE) and

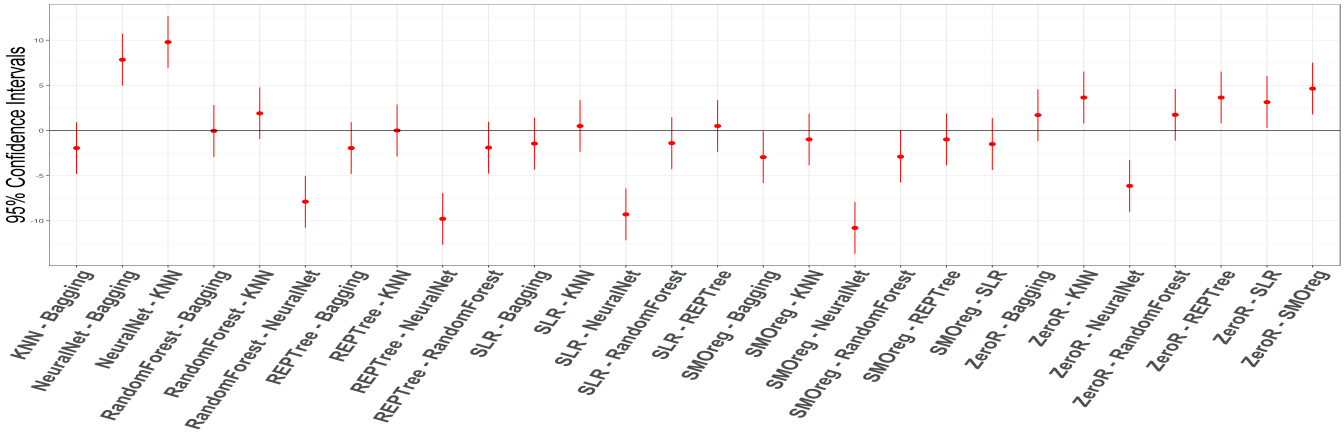

Figure 9: Tukey HSD confidence intervals for MAE difference between algorithm pairs

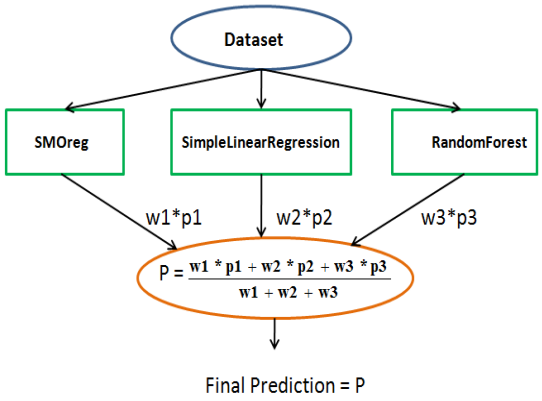

Figure 10: An ensemble approach to power prediction

progressively add more base methods to the ensemble. We also investigate unweighted and weighted averaging of the base methods. For the weighted averages, the weights are based on the reciprocal values of the MAE of the base methods. This means, predictions from base methods showing lower mean errors are weighted higher.

Ensemble's Prediction =

$$\frac{\sum_{i=1}^{n}(Algorithm_i\ Prediction * \frac{1}{Algorithm_i Percentage MAE})}{\sum_{i=1}^{n}\frac{1}{Algorithm_i Percentage MAE}}$$

Six different ensembles are created and their results compared among themselves and with SMOreg, the best single-algorithm predictor. The six ensembles we studied are as follows:

- **top2_av:** Linear average of predictions made by the *two* most accurate individual predictors – SMOreg and SLR.
- **top2_inv_weighted** Weighted average of the *two* most accurate methods. The weights are inversely proportional to their respective MAE.
- **top3_av:** Linear average of predictions made by the *three* most accurate individual predictors – SMOreg, SLR, and REPTree.
- **top3_inv_weighted:** Weighted average of the top *three* base methods.
- **all_av:** Linear average of all *seven* base methods.
- **all_inv_weighted:** Weighted average of all *seven* base methods.

5.2 Experimental Results

Fig. 11 shows the prediction accuracy of the various ensembles we created and compares them against the most accurate base method (i.e., SMOReg with a MAE of 4.5% and a maximum error of 20%). We observe that the maximum error is considerably lower than SMOReg for all the ensemble predictors. We also observe that the mean error reduces as the number of base methods increases up to a point and then increases again. The most accurate results are obtained with three base predictors: top3_av and top3_inv_weighted show MAEs of 3.5%, and 3.4% and max errors of 10.5% and 11%, respectively. Weighting the predictors does not seem to affect the results initially when the number of base predictors is fewer and the base methods show similar MAEs (and hence, similar weights). However, as the number of base predictors increases and as the accuracy of the base predictors diverge, calculating a weighted average of the base predictions helps. For instance, the *unweighted* all_av doesn't perform as well as all_inv_weighted with a MAE of 5.5% vs. 4.5% and a maximum error of 14.3%, vs. 13.2% for the two cases. The results show that ensemble techniques lead to better predictions compared to the individual methods. The downside of ensembles, however, is that the minimum error can be marginally higher.

6 RELATED WORK

Various statistical and machine learning techniques have been explored in the past for power prediction. Specifically, techniques such as linear regression [7, 22, 28], decision trees [18, 19], clustering techniques [7], support vector machines [23], evolutionary techniques [24], and neural networks [30] have been used in the past with varying degrees of success. While much of the work in this area has been applied to CPUs (we refer to Eeckhout's survey paper [12] for an extended discussion on such work), here we summarize related work on power prediction for GPUs.

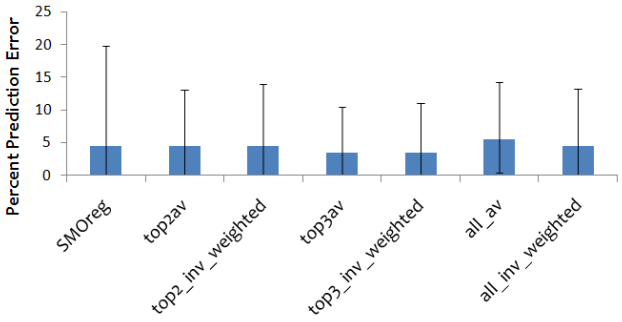

Figure 11: Percent prediction error for various ensembles. Note: top2 → SMOreg+SLR, top3 → SMOreg+SLR+REPTree

Most work on GPU power modeling [5, 17, 20, 25, 27] focuses on estimating the power consumption of a GPU in its reference configuration. Our work predicts the power consumption across many GPU machine configurations – a significantly harder problem.

Abe et al. [3] worked on a similar problem where they used a linear regression model to predict the power consumption of a GPU for different configurations. However, their approach only yielded an average error of over 20% compared to 3.5% for our ensemble method. Wu et al. [30] used K-means algorithm and neural network to predict power consumption of a GPU while achieving an accuracy comparable to ours. Our work differs from theirs in the following ways: (1) Their model requires knowledge of power consumption at a reference point even for the test application, which we do *not* need. (2) We predict the power consumption of applications rather than GPU kernels. which have simpler scaling curves. (3) Our model requires only a *fraction* of the data points for modeling compared to theirs. In addition, we present a statistically rigorous comparison of different ML techniques. To the best of our knowledge, this study is broader than any other previous work (including those carried out for CPUs) with respect to the techniques evaluated.

7 CONCLUSION

In this paper, we present a rigorous comparison of different machine-learning techniques to predict the power consumption of an application at different GPU configurations. Our evaluation showed that `SMOreg` delivers the best results with a mean error of 4.5% and `RandomForest` delivers consistent results at different frequencies. To further improve the accuracy of prediction, we designed several ensembles from the base machine learning techniques. We found that the most accurate ensemble is a combination of `SMOreg`, `SLR`, and `REPTree` methods, which reduced the mean absolute prediction error from 4.5% to 3.5% and maximum error from 15% to 11%. In the future, we plan to use our model to improve the performance of a power-capped heterogeneous system.

REFERENCES

[1] Kaggle: Your Home for Data Science. http://www.kaggle.com/.

[2] TOP500 Supercomputer Site, 2017. http://www.top500.org.

[3] Abe, Y., Sasaki, H., Kato, S., Inoue, K., Edahiro, M., and Peres, M. Power and Performance Characterization and Modeling of GPU-Accelerated Systems. In *IEEE Int'l Parallel and Distributed Processing Symposium (IPDPS)* (May 2014).

[4] Adhinarayanan, V., and Feng, W. An Automated Framework for Characterizing and Subsetting GPGPU Workloads. In *IEEE Int'l Symposium on Performance Analysis of Systems and Software (ISPASS)* (2016).

[5] Adhinarayanan, V., Subramaniam, B., and Feng, W. Online Power Estimation of Graphics Processing Units. In *IEEE/ACM Int'l Symposium on Cluster, Cloud and Grid Computing (CCGrid)* (2016).

[6] Ahern, S., et al. Scientific Discovery at the Exascale: Report from the DOE ASCR Workshop on Exascale Data Management, Analysis, and visualization. *DOE Office of Advanced Scientific Computing Research* (2011).

[7] Bailey, P. E., Lowenthal, D. K., Ravi, V., Rountree, B., Schulz, M., and de Supinski, B. R. Adaptive Configuration Selection for Power-Constrained Heterogeneous Systems. In *Int'l Conference on Parallel Processing (ICPP)* (2014).

[8] Che, S., Boyer, M., Meng, J., Tarjan, D., Sheaffer, J. W., Lee, S.-H., and Skadron, K. Rodinia: A Benchmark Suite for Heterogeneous Computing. In *IEEE Int'l Symposium on Workload Characterization (IISWC)* (2009).

[9] Cochran, R., Hankendi, C., Coskun, A. K., and Reda, S. Pack & Cap: Adaptive DVFS and Thread Packing Under Power Caps. In *IEEE/ACM Int'l Symp. on Microarchitecture (MICRO)* (2011).

[10] Danalis, A., Marin, G., McCurdy, C., Meredith, J. S., Roth, P. C., Spafford, K., Tipparaju, V., and Vetter, J. S. The Scalable Heterogeneous Computing (SHOC) Benchmark Suite. In *Workshop on General-Purpose Computation on Graphics Processing Units* (2010).

[11] David, H., Gorbatov, E., Hanebutte, U. R., Khanna, R., and Le, C. Rapl: memory power estimation and capping. In *ACM/IEEE International Symposium on Low-Power Electronics and Design (ISLPED)* (2010).

[12] Eeckhout, L. Computer architecture performance evaluation methods. *Synthesis Lectures on Computer Architecture 5*, 1 (2010).

[13] Esmaeilzadeh, H., Blem, E., St Amant, R., Sankaralingam, K., and Burger, D. Dark silicon and the end of multicore scaling. In *ACM SIGARCH Computer Architecture News* (2011), vol. 39, pp. 365–376.

[14] Felter, W., Rajamani, K., Keller, T., and Rusu, C. A Performance-Conserving Approach for Reducing Peak Power Consumption in Server Systems. In *ACM Int'l Conference on Supercomputing (ICS)* (2005).

[15] Frank, E., Hall, M. A., and Witten, I. H. The WEKA Workbench. Online Appendix for: Data Mining: Practical Machine Learning Tools and Techniques, Morgan Kaufmann, Fourth Edition, 2016.

[16] Gholkar, N., Mueller, F., and Rountree, B. A Power-aware Cost Model for HPC Procurement. In *IEEE IPDPS Workshops* (2016).

[17] Ghosh, S., Chandrasekaran, S., and Chapman, B. Statistical Modeling of Power/Energy of Scientific Kernels on a Multi-GPU System. In *Int'l Green Computing Conference (IGCC)* (2013), IEEE.

[18] Jia, W., Shaw, K. A., and Martonosi, M. Stargazer: Automated Regression-Based GPU Design Space Exploration. In *IEEE Int'l Symposium on Performance Analysis of Systems Software (ISPASS)* (2012).

[19] Jia, W., Shaw, K. A., and Martonosi, M. Starchart: Hardware and Software Optimization Using Recursive Partitioning Regression Trees. In *IEEE Int'l Conference on Parallel Architectures and Compilation Techniques (PACT)* (2013).

[20] Kasichayanula, K., Terpstra, D., Luszczek, P., Tomov, S., Moore, S., and Peterson, G. D. Power Aware Computing on GPUs. In *Symposium on Application Accelerators in High Performance Computing (SAAHPC)* (2012), IEEE.

[21] Lefurgy, C., Wang, X., and Ware, M. Power Capping: A Prelude to Power Shifting. *Cluster Computing 11*, 2 (2008).

[22] Ma, K., Li, X., Chen, W., Zhang, C., and Wang, X. GreenGPU: A Holistic Approach to Energy Efficiency in GPU-CPU heterogeneous architectures. In *IEEE Int'l Conference on Parallel Processing (ICPP)* (2012).

[23] Ma, X., Dong, M., Zhong, L., and Deng, Z. Statistical Power Consumption Analysis and Modeling for GPU-based Computing. In *Workshop on Power Aware Computing and Systems (HotPower)* (2009).

[24] Maghazeh, A., Bordoloi, U. D., Eles, P., and Peng, Z. General Purpose Computing on Low-Power Embedded GPUs: Has it come of age? In *Int'l Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation* (2013).

[25] Nagasaka, H., Maruyama, N., Nukada, A., Endo, T., and Matsuoka, S. Statistical Power Modeling of GPU Kernels Using Performance Counters. In *Int'l Green Computing Conference (IGCC)* (2010).

[26] Patki, T., Lowenthal, D. K., Rountree, B., Schulz, M., and De Supinski, B. R. Exploring Hardware Overprovisioning in Power-Constrained, High Performance Computing. In *ACM Int'l Conference on Supercomputing (ICS)* (2013).

[27] Song, S., Su, C., Rountree, B., and Cameron, K. W. A Simplified and Accurate Model of Power-Performance Efficiency on Emergent GPU Architectures. In *IEEE Int'l Symposium on Parallel and Distributed Processing (IPDPS)* (2013), IEEE.

[28] Subramaniam, B., and Feng, W. Statistical Power and Performance Modeling for Optimizing the Energy Efficiency of Scientific Computing. In *IEEE/ACM Int'l Conference on Green Computing and Communications (GreenCom)* (2010).

[29] Sun, S. Traffic Flow Forecasting Based on Multitask Ensemble Learning. In *ACM/SIGEVO Summit on Genetic and Evolutionary Computation* (2009).

[30] Wu, G., Greathouse, J. L., Lyashevsky, A., Jayasena, N., and Chiou, D. GPGPU performance and power estimation using machine learning. In *IEEE Int'l Symposium on High Performance Computer Architecture (HPCA)* (2015).

[31] Zhang, H., and Hoffmann, H. Maximizing Performance Under a Power Cap: A Comparison of Hardware, Software, and Hybrid techniques. *ACM SIGPLAN Notices 51*, 4 (2016), 545–559.