Empirical Evaluation of ML Models for Per-Job Power Prediction

Debajyoti Halder
Stony Brook University
Stony Brook, New York, USA
dhalder@cs.stonybrook.edu

Manas Acharya
Stony Brook University
Stony Brook, New York, USA
macharya@cs.stonybrook.edu

Aniket Malsane
Stony Brook University
Stony Brook, New York, USA
amalsane@cs.stonybrook.edu

Anshul Gandhi
Stony Brook University
Stony Brook, New York, USA
anshul@cs.stonybrook.edu

Erez Zadok
Stony Brook University
Stony Brook, New York, USA
ezk@cs.stonybrook.edu

ABSTRACT

Sustainability has become a critical focus area across the technology industry, most notably in cloud data centers. In such shared-use computing environments, there is a need to account for the power consumption of individual users. Prior work on power prediction of individual user jobs in shared environments has often focused on workloads that stress a single resource, such as CPU or DRAM. These works typically employ a specific machine learning (ML) model to train and test on the target workload for high accuracy. However, modern workloads in data centers can stress multiple resources simultaneously, and cannot be assumed to always be available for training. This paper empirically evaluates the performance of various ML models under different model settings and training data assumptions for the per-job power prediction problem using a range of workloads. Our evaluation results provide key insights into the efficacy of different ML models. For example, we find that linear ML models suffer from poor prediction accuracy (as much as 25% prediction error), especially for unseen workloads. Conversely, non-linear models, specifically XGBoost and Random Forest, provide reasonable accuracy (7–9% error). We also find that data-normalization and the power-prediction model formulation affect the accuracy of individual ML models in different ways.

CCS CONCEPTS

• **Hardware → Power and energy; Power estimation and optimization; Enterprise level and data centers power issues**.

KEYWORDS

Sustainability, per-job power prediction, ML models, co-executed workloads.

ACM Reference Format:
Debajyoti Halder, Manas Acharya, Aniket Malsane, Anshul Gandhi, and Erez Zadok. 2024. Empirical Evaluation of ML Models for Per-Job Power Prediction . In *Companion of the 15th ACM/SPEC International Conference on Performance Engineering (ICPE '24 Companion), May 7–11, 2024, London, United Kingdom.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3629527.3651418

1 INTRODUCTION

The exponential growth in digital data, coupled with increasing computational demands (*e.g.*, DNN training and crypto mining), has raised significant questions about the carbon footprint of data centers [31]. Both data center providers and users often share a common interest in regulating carbon usage [12, 25]. To regulate carbon usage, an important first step is to track the power consumption of *each* workload (or job, used interchangeably). This per-job power-tracking enables informed decision-making, empowering users to make design choices that align with sustainability goals [15]. Further, in the near future, providers may consider pricing models that partly charge users based on their attributed power use (*e.g.*, carbon tax), thus incentivizing sustainable practices.

Predicting the per-job power consumption is a difficult problem due to the often time-varying utilization of the various server resources by a job at runtime. The problem is further exacerbated by OS- and device-specific scheduling intricacies when resources have to be *shared* between jobs. Machine Learning (ML) approaches, such as regression models, are well suited to the power prediction problem given their ability to infer complex relationships between variables [25]. In particular, prior works have used a variety of ML models and model settings for the power-prediction problem. However, given the large variety of ML models and their settings, *a thorough evaluation is first necessary to assess the usefulness of different ML models for job-level power prediction.*

Recent works on *per-job power prediction* have primarily focused on estimating the power consumption based on CPU and memory utilization metrics [6, 13, 14]. As we discuss in Section 2, it is not enough to account only for the power consumption of CPU and memory subsystems. The classical works in power prediction (*e.g.*, Joulemeter [21], VMeter [17]) employ linear ML models to predict power as a function of resource-utilization metrics. While such models may work well for benchmarks designed to saturate individual resources, we find that linear models have poor accuracy when predicting per-job power for workloads that stress multiple resources simultaneously, such as TensorFlow and MongoDB.

In this paper, we empirically evaluate the performance of several, diverse ML models (both linear and non-linear) to predict per-job power consumption, using several workloads and both micro- and macro-benchmarks. We also evaluate the impact on

ICPE '24 Companion, May 7–11, 2024, London, United Kingdom

ACM ISBN 979-8-4007-0445-1/24/05…$15.00
https://doi.org/10.1145/3629527.3651418

prediction accuracy of several models and system settings, such as data normalization, accounting for background processes, and factoring in idle power. To investigate the impact of training data and the deployment context, we evaluate the ML models under different settings, including testing on the training workloads and testing on unseen workloads.

Our experimental results using 8 different pairs of (co-executed) test workloads under 7 different ML models show that *non-linear models outperform linear models* in terms of per-job prediction accuracy (see Section 4). In particular, XGBoost and Random Forest provided less than 10% error when predicting the per-job power consumption of *unseen* workloads (comparing the sum of predicted per-job power values with full-server power measurements). By contrast, linear regression (LR) had much worse accuracy, with errors as high as 40–50% for some pairs of co-located workloads. Our experiments beyond two co-executed workloads show that non-linear models can predict per-job power consumption with ~10% error when the training dataset corresponds to the test co-execution scenario. However, training a model for each workload class (CPU-, DRAM-, I/O-heavy) separately did not improve the prediction accuracy significantly (3–7% difference).

We also find that the ML model settings and prediction formulation can have an impact on prediction accuracy. For example, predicting for the residual power (after subtracting idle system power) instead of total power, and including the intercept term in supported ML models, can reduce prediction error by as much 10%. We also found that data normalization techniques like standardization or min-max scaling significantly affect neural network models' accuracy. Other models, such as decision trees, are less sensitive to data scaling. Finally, we found that prediction accuracy is not much affected when we take into account the resource utilization of background processes, suggesting that ML models can capture such activities through the resource usage of foreground workloads.

In summary, this paper makes the following contributions:

- We empirically evaluate ***several ML models***, including a workload-specific model, for power prediction. This is in contrast to existing works that often only consider a single model. Further, we report on the impact of model formulation settings and data-processing techniques on power prediction accuracy. We have made our datasets and code available [27] for reproducibility.
- We consider the practical yet challenging problem of ***per-job power prediction*** to allow users in shared environments to assess their sustainability footprint. We considered up to four co-executed workloads. Few prior works focus on this realistic case.
- Unlike prior works, we experiment with ***diverse workloads*** that are not just CPU- or memory-bound but stress the entire system. Further, we consider the realistic scenario where a test workload *has not been observed* in training, unlike much of the prior work that focuses only on cross-validation results.

2 BACKGROUND AND PRIOR WORK

The problem of predicting the power consumption of individual jobs *in the presence of other co-executing jobs* is challenging for at least two reasons. First, the power consumption of a server when running multiple jobs simultaneously is not simply the sum of the power consumed when the jobs are run individually (see Appendix A for empirical data). This is likely due to resource saturation, sharing, and contention when multiple jobs co-execute.

The second challenge is that there is no accurate, ground truth power value that is available for *individual jobs*. Prior research has focused on predicting *total* server power [3, 32] by using resource usage metrics (*e.g.*, CPU and memory utilization). While these models are valuable for specific use cases, our goal is to model the concurrent utilization of all system resources to predict *per-job* power consumption.

Prior Work. Joulemeter [21] employs linear ML models to predict per-VM power consumption using observable power states in the hypervisor. Linear regression models have also been employed in CloudMonitor [32] and VMeter [17] to predict VM-level power. Likewise, linear regression has also been used to predict total server power (Krishnan *et al.* [23]) and process-level power (Bertran *et al.* [4]). However, we find that linear models are inadequate for predicting the power consumption of co-executed jobs in shared environment scenarios (see Section 4.2).

Several studies have employed a single, non-linear ML model for power prediction. Xiao *et al.* [35] and BitWatts by Colmant *et al.* [7] explore polynomial regression for predicting power in virtual environments. Dhiman *et al.* [11] proposed the use of Gaussian Mixture Models (GMM) for power prediction in virtualized environments. Recent works by Fieni *et al.* [13, 14] leverage Lasso and Ridge regression for power modeling. The authors also use sequential learning to calibrate their power models online by training on currently executing workloads [14]. The authors also rely on PowerAPI [18] (which in turn relies on Intel RAPL [9]) to track CPU package and DRAM power consumption values as ground truth. RAPL-reported values provide power consumption of CPU package (cores, caches, and any integrated GPU) and DRAM. Phung *et al.* [30] leverage RAPL values as additional features for learning, but focus on modeling the power use of only CPU-intensive workloads.

While RAPL power values can serve as ground truth for CPU and/or DRAM power consumption, they are not accurate indicators of full-server power (see Appendix B) as *RAPL does not include power consumed by disks, motherboard, network, or GPU(s).*

Given the importance of power consumption tracking, there have also been power modeling tools developed for consumer use. Scaphandre [28] predicts per-process power consumption by tracking the jiffies and correlating it with RAPL power values when a process is running. However, as noted by prior work [20], Scaphandre's focus is primarily on CPU-power consumption (hence the reliance on RAPL). Kepler [6] is a tool developed by Red Hat that predicts pod and node power consumption; however, Kepler is limited to only Kubernetes environments. Further, Kepler uses cgroups and sysfs to get CPU and memory usage statistics, and thus only focuses on the power of these two resources. Similar tools have also been developed for user-facing purposes, but these tools are not accurate enough for tracking the power of workloads that stress multiple resources. For example, Apple provides an "Energy Impact" metric with their Activity Monitor tool [2], but the estimates reported are only relative values (with no units) that are based on a job's CPU usage [26].

There are other prior works that focused on specific scenarios or workloads (see survey paper by Lin *et al.* [25]), such as approaches

Table 1: Features used to train ML models.

Entity	Features
CPU	Cycles, Ref-cycles, Instructions
DRAM	LLC-load-misses, LLC-loads, LLC-store-misses, LLC-stores
Disk	Bytes, Blocks (# of reads and writes)
RAPL	Package power, RAM power

that estimate the power usage of HPC servers [19, 34] or predict the power consumption of DNN training systems [1, 24, 31]. However, they rely on the specifics of the workload or system, and are thus not easily generalizable.

3 POWER MODELING

For all ML models we considered, the ground truth for server power, P_{server}, was obtained via a power meter (see Section 4.1). In general, the ML models estimate server power at time t, say $\hat{P}^t_{server}$, as a function of some feature vector, $\vec{x}^t$, as $\hat{P}^t_{server} = f(\vec{x}^t)$. The hat notation denotes predicted values (as opposed to ground truth values). For ease of notation, we drop the t superscript by implicitly considering the formulations as being specific to a given time.

3.1 Features

The ML models aim to predict power consumption as a function of resource utilization and other metrics, referred to as features (the $\vec{x}$). Rather than determining these features from scratch, we built on existing studies to obtain features for our power-prediction problem; note that we are not considering networked systems in this paper, so we do not include network features, though they could be easily added as needed.

Based on prior works [17, 21, 32], we arrived at the feature list shown in Table 1. We believe this list is short yet representative enough to capture the important resource-utilization values. While RAPL power values may not track full-server power, RAPL power values may still serve as useful *features*, as we explore in Section 4. For CPU and DRAM features, we used `perf-stat` to obtain performance event counts, which are reported per TID (Thread Identifier). We used `pstree` to track all TIDs (including for child threads) pertaining to a given workload to aggregate the features from `perf-stat` *per workload*. We used `blktrace` to track per-process disk reads and writes.

All performance event counts from `perf-stat` are sampled at 200ms intervals. A higher sample rate increased the power consumption overhead of tracing by 5W. We aggregate performance event counts for every 1s interval and align them with the full-system power values obtained from the power meter every 1s. For workload-specific resource utilization, we combine the performance event counts for each workload separately. We obtain RAPL power values from `turbostat`; RAPL power values are reported as an aggregate for the CPU package and DRAM, and not per TID.

3.2 Power Prediction

We start with a simple setting where a single workload is running on a server. In this case, the power prediction can be formulated as:

$$\hat{P}_{server} = f(\vec{x_{w1}}) \tag{1}$$

where $\vec{x_{w1}}$ is the feature vector, say of size n, obtained for the (single) workload. For example, for Linear Regression (LR) with intercept term, Eq. (1) takes the form $\hat{P}_{server} = \beta_0 + \sum_{i=1}^{n} \beta_i \cdot x_{w1.i}$, where the β terms denote the coefficients of the LR model that are learned during training and $x_{w1.i}$ is the i^{th} feature of the $\vec{x_{w1}}$ feature vector. For LR (and other ML models that support the intercept term, β_0), one can also set up the ML model without intercept. We evaluate both options in our experiments.

Since the server has some baseline idle power, say P_{idle}, which can be considered as a constant (for that server), another formulation is to predict the *residual* power (or dynamic power), which is $P^{res}_{server} = P_{server} - P_{idle}$. In this case, the prediction takes the form $\hat{P}^{res}_{server} = f(\vec{x_{w1}})$, and so we predict full-server power as:

$$\hat{P}_{server} = P_{idle} + f(\vec{x_{w1}}) \tag{2}$$

Another variant of Eq. (2) is to also subtract the feature values obtained for an idle system, say $\vec{x_{idle}}$ (*e.g.*, CPU cycles of an idle system spent on background processes) to better correlate with residual power as:

$$\hat{P}_{server} = P_{idle} + f(\vec{x_{w1}} - \vec{x_{idle}}) \tag{3}$$

Co-Executed Workloads. When we have two workloads (can be extended beyond two) executing concurrently, as is the focus of this paper, we can separately predict the power consumption of each workload as $\hat{P}_{wi} = f(\vec{x_{wi}})$, and predict full-server power as:

$$\hat{P}_{server} = f(\vec{x_{w1}}) + f(\vec{x_{w2}}) \tag{4}$$

For ML models that have an intercept term (*e.g.*, Linear Regression with β_0 in $f()$), we subtract the intercept once from Eq. (4) to avoid double-counting the intercept. Note that we still use full-server power (P_{server}) as the dependent variable in the final prediction formulation since we have ground truth for only full-server power (and not for the power consumption of individual workloads).

For co-executed scenarios, we separately track the feature values (*e.g.*, CPU cycles) for each workload process and their children to obtain $\vec{x_{w1}}$ and $\vec{x_{w2}}$. Feature values that do not belong to either of the processes can be attributed to background or kernel processes, denoted as $\vec{x_{bg}}$. As such, another variant of Eq. (4) that we consider is with $f(\vec{x_{bg}})$ added to the right-hand side.

For the residual power formulation, we similarly have:

$$\hat{P}_{server} = P_{idle} + f(\vec{x_{w1}}) + f(\vec{x_{w2}}) \tag{5}$$

with the possibility of $f(\vec{x_{bg}})$ added to the right-hand side. We also consider variants of the above formulations where $\vec{x_{idle}}$ is subtracted from each feature vector on the right-hand side.

Power accounting. For the co-executed formulations, Eqs. (4) and (5), and their variants, the power contribution of each workload can be estimated as $f(\vec{x_{wi}})$, for $i = 1, 2$. If P_{idle} (or the intercept term or $f(\vec{x_{bg}})$) also must be accounted for, then we can charge each workload with a fraction of P_{idle} proportional to its estimated power. For example, in Eq. (5), we estimate workload 1's total power contribution as:

$$f(\vec{x_{w1}}) + \frac{f(\vec{x_{w1}})}{f(\vec{x_{w1}}) + f(\vec{x_{w2}})} \cdot P_{idle} \tag{6}$$

Table 2: Workloads employed in our experiments.

Resource	Workload
CPU	7zip, Cypto++, CP2K, Gzip
DRAM	Stream, MBW, Tinymembench, RAMSpeed SMP
Disk	Unpack Linux, LevelDB, SQLite, FIO
System	Stress-ng, Tensorflow, Mobile Neural Network, Sysbench, Memcached, Filebench, MongoDB

Using the proportion of predicted power to account for P_{idle} is preferable to, say, using the proportion of a resource usage metric, since predicted power is a function of all features.

4 EVALUATION

In this section we discuss the results and observations from the evaluation of the power models on various workloads and model formulations. In Section 4.1 we discuss our experimental setup, the benchmarks used for evaluation, the ML models, their model formulation settings, and the metrics used for evaluation. We conducted experiments for multiple scenarios like predicting per-job power when only a single workload is executed, or when two or more workloads are co-executed. The evaluation results for every scenario are discussed in Section 4.2. We also discuss 5-fold cross-validation results and feature importance.

To make our results reproducible, we have made available our datasets and code for power-prediction model evaluation [27].

4.1 Experimental Setup and Methodology

We conduct all our experiments on a server with two Intel Xeon E5 CPUs with Haswell architecture (has RAPL support). The server has 24 cores total and 256GB of memory. We disabled speedstep (DVFS), hyperthreading, and turboboost (overclocking) to minimize power consumption uncertainties due to dynamic system/OS behavior. To obtain ground truth, we use an external wall power meter, WattsUp Pro [33], attached to the server, which provides full-server power readings once per second.

Workloads. For our evaluation, we employed workloads from stress-ng [22], YCSB [8], and Phoronix Test Suite [29], as shown in Table 2. The resource-specific workloads were primarily used for training whereas the System workloads were used for testing; a similar methodology was adopted by prior works that modeled the power consumption of individual (not co-executed) workloads [7, 13, 16]. Training on microbenchmarks allows the ML models to learn the impact of resource utilization on power consumption under controlled stress-test conditions. Every workload ran for around 20 minutes either independently, or co-executed with other workloads.

ML Models. We used a variety of ML models to evaluate power prediction: Linear Regression (LR), Decision Tree (DT), Random Forest (RF), Support Vector Regressor (SVR), XGBoost, Lasso, and Neural Network (NN). All ML model hyper-parameters were tuned via Grid Search; see Appendix C for details.

Data Processing Techniques. For data processing, we experimented with three popular techniques: (i) de-mean, whereby the mean of the dataset is subtracted, (ii) standardization, which additionally divides by the standard deviation of the data, and (iii) min-max normalization, which scales data to the $(0, 1)$ range. In general, all techniques provided better results than no processing as raw values for different features have different magnitudes. For example, CPU cycles/second is usually in the billions, whereas for non-disk-intensive workloads, the reads/second or writes/second during workload execution is typically in the thousands.

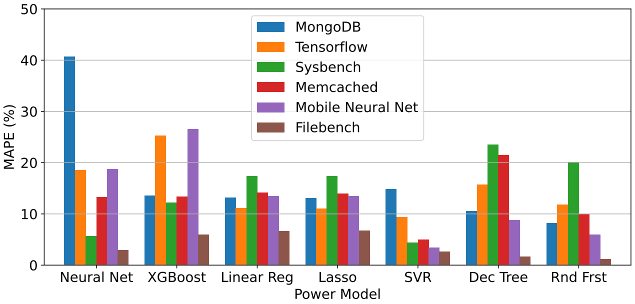


Figure 1: MAPE values for power modeling when a single application is run on the server.

Prediction Metrics. We used Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) as our error metrics; ground truth was obtained from the power meter. MAE values showed the same trend as MAPE, so we report MAPE values in our results.

4.2 Experimental Results

For evaluation, unless otherwise stated, we use the System workloads (as listed in Table 2) for testing the ML models, while using the others for training.

4.2.1 Single Workload Execution. We start by evaluating the ML models for the single workload scenario using the residual power prediction formula with idle features removed (see Eq. (3)). Figure 1 illustrates the MAPE values obtained for predicting the server power on the y-axis and the ML models on the x-axis. All the models perform well, with an average MAPE value of less than 17% across all workloads. Support Vector Regression (SVR) outperforms the others, with an average MAPE of 6.6%, followed by Random Forest with an average MAPE of 9.6%. Even Linear Regression (LR) provides satisfactory results, achieving an average MAPE of 12.7%.

For the power prediction in Figure 1, we experimented with different data-processing techniques. We found that the de-mean approach provided the best results for all models, except NN. For all models except NN, other approaches like standardization result in a 1–2% increase in MAPE. With min-max normalization, MAPE increased by 1–3%. For NN, min-max scaling worked the best, improving the prediction accuracy significantly (76%); standardization only provided some improvement (11%) in prediction accuracy. The reason for this is the NN model's sensitivity to input scale and reliance on gradient-based optimization methods [5]. Techniques like min-max scaling ensure a consistent scale for all features, facilitating efficient learning and stable convergence. Without this consistent scaling, the NN model's learning could be inefficient due to skewed gradients. Other models such as Decision trees, Random Forest, and XGBoost, are less sensitive to data scaling because their splitting criteria and/or ensemble nature focuses on the relative ordering of feature values rather than their specific scales. These models make decisions based on feature relationships, making them

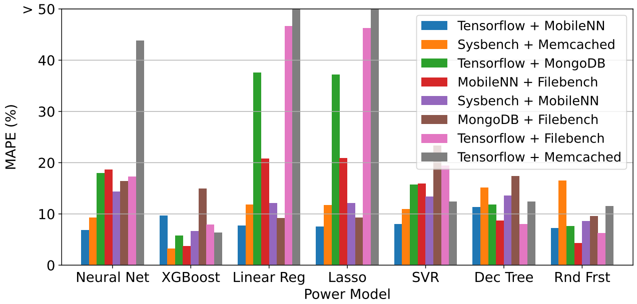


Figure 2: Prediction results when workloads are co-executed and residual server power is predicted as the sum of predicted residual power of each workload.

inherently robust to variations in feature scales. For subsequent evaluations, we used de-mean for all models except NN, for which we used min-max scaling.

Among the different model variants considered, the residual power prediction approach (Eq. (2)) offered better accuracy compared to directly predicting full-server power (Eq. (1)). This improvement is seen particularly in the case of Linear Regression (LR) and XGBoost, resulting in a 10% reduction in MAPE. Excluding idle features ($\vec{x_{idle}}$) from the prediction process did not affect prediction accuracy. Furthermore, including the intercept term in supported models led to a slight improvement in prediction accuracy. However, for LR, the improvement was substantial, reducing MAPE by approximately 20%. Unless specified otherwise, we considered these variations in our subsequent results.

Predicting residual power, where we isolate the active system usage by subtracting idle power, proved effective in improving accuracy. This is because predicting residual power allows the model to capture only the specific resource patterns associated with active jobs, providing a better understanding of how power usage is affected by resource utilization of jobs. At the same time, incorporating the intercept term in the models was crucial for considering baseline power, representing the constant power consumption of the server (which includes idle power) when no active jobs are running. *These two steps (predicting for residual power and including the intercept term) are important for power prediction as they ensure that the model does not unintentionally miss or attribute variations, preventing bias in predictions.*

Additionally, we conducted experiments without utilizing the two RAPL power features. This resulted in a slight increase (1-2%) in MAPE values across all ML models. RAPL power features cannot be obtained on a per-thread or Thread ID (TID) basis. Therefore, it is not possible to track the power contribution of individual workloads using RAPL power features. Moreover, RAPL values exhibit inconsistencies for certain server models, as reported by Desrochers *et al.* [10]. Therefore, we opted not to incorporate RAPL power features in the remainder of our evaluation.

4.2.2 *Per-job Power Prediction for Co-Executed Workloads.*

We now consider the challenging case where each co-executed workload's power consumption is to be predicted. In particular, each co-executed workload's residual power consumption is first predicted, and then the full-server power, obtained by adding the individual workload power values and P_{idle} (via Eq. (5)), is compared with the full-server ground truth power use. We train our models on feature vectors from random pairs of non-System workloads from Table 2. We then test the models on 8 pairs of System workloads. This ensures that test workloads are separate from training.

Figure 2 shows our results for different pairs of co-executed workloads. Here, the training data included only pairs of *non-test* workloads from Table 2, representing the realistic case where test workloads may not always be available for training. XGBoost performed the best, with an average MAPE of 7.3%, followed by Random Forest (8.9%), Decision Tree (12.3%), and SVR (14.9%). LR performed poorly, with an average MAPE of 25%, highlighting the *linear model's inability to account for resource contention when workloads are co-executed.* We also explored the variation where the predicted power consumed by other processes ($f(\vec{x_{bg}})$ term from Section 3.2) was added to the predicted power of the workloads to arrive at the full-server power prediction. However, this formulation yielded slightly higher MAPE values, and was thus not considered further.

We also conducted 5-fold cross-validation for our ML models by training and testing on the dataset obtained by co-executing a pair of workloads. In general, the prediction errors are lower than those in Figure 2, since the test workloads comprise the same training data. Based on average MAPE values, all models performed well, with SVR (4.4%), RF (5.7%), NN (6.7%), Lasso (6.7%), DT (8.1%), and XGBoost (9.7%) providing less than 10% error; even LR (6.9%) resulted in low error under this cross-validation setting. *This suggests that the choice of ML model to employ also depends on the training and test data overlap assumptions.*

4.2.3 *Background Processes as Third Workload.*

As mentioned earlier, we explored another variation of our power model by considering all background processes as an additional workload co-executed during the experiments. The idea behind this approach was that the power model might be able to better segregate the power among the active workloads if the background processes are grouped separately. The power prediction formulation hence becomes:

$$\hat{P}_{server} = P_{idle} + f(\vec{x_{w1}}) + f(\vec{x_{w2}}) + f(\vec{x_{bg}}) \tag{7}$$

However, the results were not impressive. XGBoost had the least average MAPE of 20.9%, followed by RF (22.9%), NN (25.3%), and DT (25.4%). We also tried another variation of this model without the intercept term (β_0), but that resulted in much worse results (~57% average MAPE for RF, DT, and LR). We thus decided to not consider this model variation for our evaluation.

4.2.4 *Beyond Two Co-Executed Workloads.*

We next experimented by testing on four co-executed workloads by also training on the dataset obtained by co-executing four workloads. We ran 5 combinations of four workload pairs and evaluated using "leave-one-out" cross-validation (see Figure 3). For example, in Figure 3, when we test on the Tensorflow + MNN + Memcached + Filebench workload combination (blue legend), then the remaining 4 workload combinations are used for training.

All the ML models performed well except LR and Lasso. XGBoost and Random Forest had the least average MAPE of ~10%. LR performed much worse with ~24% average MAPE. This again shows that *linear models are not well suited for co-executed workloads' power prediction.* Overall, *XGBoost and Random Forest are the*

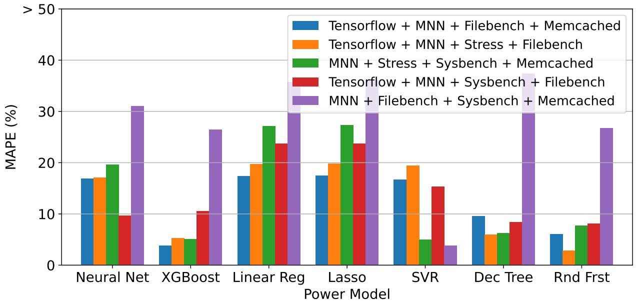


Figure 3: Prediction results when four workloads are co-executed and residual server power is predicted as the sum of predicted residual power of each workload.

best performing models with 10% or lower average MAPE in most scenarios, showing that non-linear models are effective for per-job power prediction.

4.2.5 *Workload Classification.* The workloads being executed on a server can be classified by the resource(s) they stress. It may be interesting to consider a modeling approach whereby we build a power model for each workload class separately to gain accuracy. To that end, we classified workloads into 3 classes: CPU-heavy, DRAM-heavy, and Disk-heavy. We trained a different power model for each class and tested it on a workload of the same class. We compared this new approach of "workload classification" with the original "all workloads for training" approach (except the one being tested). In this experiment, we trained and tested on a single workload execution scenario. The benchmark workloads used are the micro-benchmarks from Table 2. Figure 4 shows that the prediction error of the power model trained for specific workload classes is typically worse than the original approach (3–7% difference). *This suggests that a single model trained on multiple workload classes can improve power prediction accuracy over workload-specific models.*

4.2.6 *Feature Selection and Importance.* To evaluate our feature set, we employed the `mutual information` method from scikit-learn to estimate the significance of each feature utilized in our training, as outlined in Table 1. This method quantifies the dependence between two variables and is instrumental in assessing the information gain associated with each feature relative to the target variable. We found that DRAM and CPU features had higher importance scores (*e.g.*, LLC-loads, LLC-stores for DRAM and ref-cycles, cycles for CPU), while Disk features (*e.g.*, bytes and blocks) had the lowest scores. We repeated our power predictions by omitting the Disk features, but this resulted in slightly higher MAPE values, suggesting that our feature list is adequate. We also utilized XGBoost and Random Forest algorithms to determine feature importance, obtaining similar results.

4.2.7 *Analyzing the Per-Job Power Predictions.* Thus far we evaluated our per-job power predictions (obtained via Eq. (6)) by comparing the *sum* of per-job powers with full-server power meter readings because there is *no ground truth* for per-job power consumption in the co-executed setting. Nonetheless, we can analyze the trend in per-job power predictions. We first compared the per-job predictions obtained from the co-executed setting with the ground truth residual power when the workload is run in isolation. As expected, the former is lower than the latter, likely due to resource contention and saturation. For example, the ground truth residual power of Sysbench in isolation is about 63W. However, the predicted per-job power of Sysbench when co-executed with Memcached and when co-executed with Mobile Neural Network (MNN) is only about 46W and 50W, respectively. In both co-executions, the predicted power of Memcached and MNN is also lower than their ground truth isolated power.

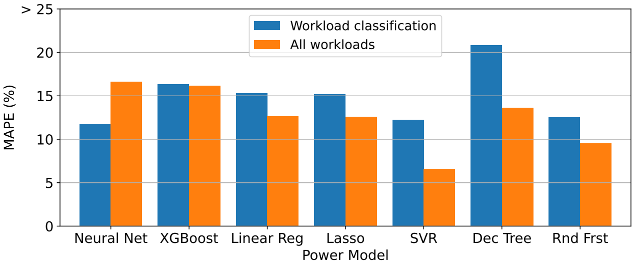


Figure 4: Performance of power models when trained on specific workload classes versus all workloads.

We also compared the per-job power predictions when TensorFlow (TF) is co-executed with MongoDB and when TF is co-executed with MNN. The predicted power for TF is 67W when co-executed with MongoDB and 85W when co-executed with MNN. Since MongoDB is I/O intensive, we expect TF's I/O to be slowed down much more when TF is co-executed with MongoDB compared to when TF is co-executed with MNN; consequently, TF may have fewer instructions to be run per second when co-executed with MongoDB, lowering its power draw. The disk and CPU features confirm this claim, providing some validation of our predictions.

5 CONCLUSION AND FUTURE WORK

Per-job power tracking is an important first step for incentivizing sustainable computing practices among consumers and providers of cloud data centers. While there is ample literature on power-prediction techniques, there is little prior work on *comparing different power-prediction models, especially under different workload and model settings.* Our evaluation results showed that non-linear ML models, specifically XGBoost and Random Forest, provided good prediction accuracy (≤ 10% MAPE). Even for SVR models, we found that a non-linear kernel provided significantly higher prediction accuracy than a linear kernel (~92% lower MAPE). By contrast, LR did not perform well (25% MAPE). As such, the choice of ML model plays an important role in power prediction. The choice of ML model also depends on the training and test data assumptions. In cross-validation settings, almost all ML models we experimented with performed quite well (less than 10% MAPE). Interestingly, workload-specific power models did not provide good accuracy; training across workload classes resulted in a non-trivial 6% accuracy gain.

ACKNOWLEDGMENTS

This work was supported in part by Dell-EMC, NetApp, Tintri, Facebook, and IBM support; and NSF awards CNS-1750109, CNS-1900706, CNS-2106263, CNS-2106434, CNS-2214980, and CCF-2324859.

REFERENCES

[1] Anthony, L., Kanding, B., and Selvan, R. Carbontracker: Tracking and predicting the carbon footprint of training deep learning models. In *ICML Workshop on Challenges in Deploying and monitoring Machine Learning Systems* (2020).

[2] Apple. View energy consumption in Activity Monitor on Mac. https://support.apple.com/en-gb/guide/activity-monitor/actmntr43697/mac.

[3] Barroso, L. A., and Hölzle, U. The Case for Energy-Proportional Computing. *IEEE Computer 40*, 12 (2007), 33–37.

[4] Bertran, R., Becerra, Y., Carrera, D., Beltran, V., Gonzalez, M., Martorell, X., Torres, J., and Ayguade, E. Accurate energy accounting for shared virtualized environments using pmc-based power modeling techniques. In *2010 11th IEEE/ACM International Conference on Grid Computing* (2010), pp. 1–8.

[5] Bhanja, S., and Das, A. Impact of data normalization on deep neural network for time series forecasting. *ArXiv* (2018).

[6] Cloud Native Computing Foundation. Kubernetes Efficient Power Level Exporter (Kepler). https://sustainable-computing.io, 2022.

[7] Colmant, M., Kurpicz, M., Felber, P., Huertas, L., Rouvoy, R., and Sobe, A. Process-level power estimation in vm-based systems. In *Proceedings of the Tenth European Conference on Computer Systems* (New York, NY, USA, 2015), EuroSys ’15, Association for Computing Machinery.

[8] Cooper, Brian. Yahoo! Cloud Serving Benchmark. https://github.com/brianfrankcooper/YCSB, 2021.

[9] David, H., Gorbatov, E., Hanebutte, U. R., Khanna, R., and Le, C. RAPL: memory power estimation and capping. In *Proceedings of the 2010 ACM/IEEE International Symposium on Low Power Electronics and Design (ISPLED)* (2010), pp. 189–194.

[10] Desrochers, S., Paradis, C., and Weaver, V. M. A Validation of DRAM RAPL Power Measurements. In *Proceedings of the Second International Symposium on Memory Systems* (Alexandria, VA, USA, 2016), pp. 455–470.

[11] Dhiman, G., Mihic, K., and Rosing, T. A system for online power prediction in virtualized environments using gaussian mixture models. In *Design Automation Conference* (2010), pp. 807–812.

[12] Dutt, A., Rachuri, S. P., Lobo, A., Shaik, N., Gandhi, A., and Liu, Z. Evaluating the energy impact of device parameters for dnn inference on edge. In *Proceedings of the 14th International Green and Sustainable Computing Conference (IGSC’23)* (Toronto, Canada, 2023).

[13] Fieni, G., Rouvoy, R., and Seinturier, L. SmartWatts: Self-Calibrating Software-Defined Power Meter for Containers. In *Proceedings of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing* (2020), pp. 479–488.

[14] Fieni, G., Rouvoy, R., and Seiturier, L. SelfWatts: On-the-fly Selection of Performance Events to Optimize Software-defined Power Meters. In *Proceedings of the 21st International Symposium on Cluster, Cloud and Internet Computing* (2021), pp. 324–333.

[15] Gandhi, A., Ghose, K., Gopalan, K., Hussain, S., Lee, D., Liu, D., Liu, Z., McDaniel, P., Mu, S., and Zadok, E. Metrics for sustainability in data centers. In *Proceedings of the 1st Workshop on Sustainable Computer Systems Design and Implementation (HotCarbon’22)* (San Diego, CA, USA, July 2022), USENIX.

[16] Guo, N., Gui, W., Chen, W., Tian, X., Qiu, W., Tian, Z., and Zhang, X. Using improved support vector regression to predict the transmitted energy consumption data by distributed wireless sensor network. *EURASIP Journal on Wireless Communications and Networking 2020*, 1 (2020), 120.

[17] Husain Bohra, A. E., and Chaudhary, V. VMeter: Power modelling for virtualized clouds. In *Proceedings of the 2010 IEEE International Symposium on Parallel Distributed Processing, Workshops and Phd Forum (IPDPSW)* (2010), pp. 1–8.

[18] Inria, University of Lille. PowerAPI. https://powerapi.org, 2023.

[19] Jarus, M., Oleksiak, A., Piontek, T., and Weglarz, J. Runtime power usage estimation of HPC servers for various classes of real-life applications. *Future Generation Computer Systems 36* (2014), 299–310.

[20] Jay, M., Ostapenco, V., Lefèvre, L., Trystram, D., Orgerie, A.-C., and Fichel, B. An experimental comparison of software-based power meters: focus on CPU and GPU. In *Proceedings of the 23rd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing* (Bangalore, India, 2023), pp. 1–13.

[21] Kansal, A., Zhao, F., Liu, J., Kothari, N., and Bhattacharya, A. A. Virtual machine power metering and provisioning. In *Proceedings of the 1st ACM Symposium on Cloud Computing* (Indianapolis, IN, USA, 2010), SoCC ’10, pp. 39–50.

[22] King, Colin. stress-ng. https://manpages.ubuntu.com/manpages/xenial/man1/stress-ng.1.html, 2023.

[23] Krishnan, B., Amur, H., Gavrilovska, A., and Schwan, K. Vm power metering: Feasibility and challenges. *SIGMETRICS Perform. Eval. Rev. 38*, 3 (jan 2011), 56–60.

[24] Lacoste, A., Luccioni, A., Schmidt, V., and Dandres, T. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700* (2019).

[25] Lin, W., Shi, F., Wu, W., Li, K., Wu, G., and Mohammed, A.-A. A taxonomy and survey of power models and power modeling for cloud servers. *ACM Comput. Surv. 53*, 5 (2020).

[26] Nethercote, Nicholas. What does the OS X Activity Monitor’s “Energy Impact” actually measure? https://blog.mozilla.org/nnethercote/2015/08/26/what-does-the-os-x-activity-monitors-energy-impact-actually-measure/.

[27] PACE Lab, Stony Brook University. Replication Package. https://github.com/PACELab/sassy-metrics-data-code, 2023.

[28] Petit, B. Scaphandre. https://github.com/hubblo-org/scaphandre.

[29] Phoronix Media. Phoronix Test Suite. https://www.phoronix-test-suite.com/, 2023.

[30] Phung, J., Lee, Y. C., and Zomaya, A. Y. Modeling System-Level Power Consumption Profiles Using RAPL. In *2018 IEEE 17th International Symposium on Network Computing and Applications (NCA)* (2018), pp. 1–4.

[31] Schmidt, V., Goyal, K., Joshi, A., Feld, B., Conell, L., Laskaris, N., Blank, D., Wilson, J., Friedler, S., and Luccioni, S. CodeCarbon: Estimate and Track Carbon Emissions from Machine Learning Computing. https://mlco2.github.io/codecarbon/motivation.html, 2021.

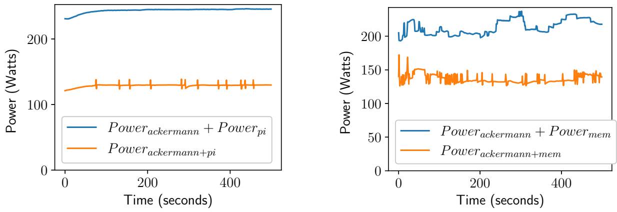
[32] Smith, J. W., Khajeh-Hosseini, A., Ward, J. S., and Sommerville, I. Cloudmonitor: Profiling power usage. In *Proceedings of the 2012 IEEE Fifth International Conference on Cloud Computing* (2012), pp. 947–948.

[33] WattsUp. WattsUp? Pro. https://arcb.csc.ncsu.edu/~mueller/cluster/arc/wattsup/metertools-1.0.0/docs/meters/wattsup/manual.pdf.

[34] Witkowski, M., Oleksiak, A., Piontek, T., and Wundefinedglarz, J. Practical Power Consumption Estimation for Real Life HPC Applications. *Future Generation Computer Systems 29*, 1 (2013), 208–217.

[35] Xiao, P., Hu, Z., Liu, D., Yan, G., and Qu, X. Virtual machine power measuring technique with bounded error in cloud environments. *Journal of Network and Computer Applications 36*, 2 (2013), 818–828.

A APPENDIX: EMPIRICAL DATA SHOWING THE POWER CONSUMPTION OF CO-EXECUTED WORKLOADS VERSUS THE SUM OF POWER CONSUMPTION OF INDIVIDUAL WORKLOADS



(a) CPU load with CPU load

(b) CPU load with MEM load

Figure 5: Residual power (after subtracting idle power) when workloads are co-executed versus sum of residual powers when workloads are run in isolation.

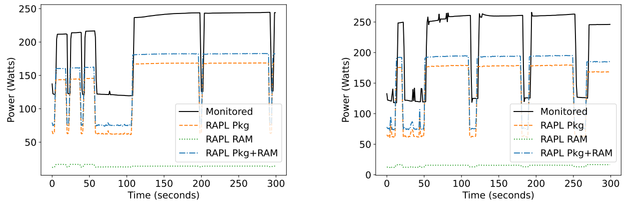
We conducted experiments using stress-ng micro-benchmarks [22] to investigate the difference in power consumption of co-executed and individual workloads. We observed a large difference (see Figure 5(a)) between the sum of residual power consumption of two CPU-bound micro-benchmarks (ackermann and pi) when run individually and the residual power consumption when these two micro-benchmarks are run together. Residual power is the server power with idle power subtracted from it.

Figure 5(a) shows, in blue, the sum of residual power consumption of two CPU-bound micro-benchmarks (ackermann and pi) when run individually; in orange, we see the residual power consumption when these two micro-benchmarks are run together. We see a similar difference (see Figure 5(b)) even if we run a CPU-bound micro-benchmark next to a memory-bound one. The large difference between the two lines shows that the power consumption profile of a job depends on other concurrent jobs as there may be resource saturation and contention when jobs co-execute.

Table 3: Illustration of our hyper-parameter tuning using Grid Search. Tuned values are highlighted in bold.

XGBoost	SVR	Decision Tree	Random Forest	Neural Network
learning_rate: [0.01, **0.03**, 0.1, 0.5] n_estimators: [100, 200, 300, **900**] max_depth: [3, 5, **6**] min_child_weight: [1, **3**, 5] subsample: [**0.6**, 0.8, 1.0] colsample_bytree: [**0.6**, 0.8, 1.0]	C: [0.1, **1**, 10, 100] kernel: [linear, **rbf**, poly] gamma: [**scale**, auto, 0.01, 0.1, 1] epsilon: [0.1, 0.2, **0.5**, 1.0]	max_depth: [None, 5, 10, 15, **20**] min_samples_split: [**2**, 5, 10] min_samples_leaf: [1, **2**, 4] max_features: [auto, sqrt, **log2**] max_leaf_nodes: [None, **10**, 20, 30] min_impurity_decrease: [**0.0**, 0.1, 0.2]	n_estimators: [100, 200, **300**, 500] max_depth: [None, 5, 10, **20**, 30] min_samples_split: [**2**, 5, 10] min_samples_leaf: [1, **2**, 4] max_features: ['auto', **'sqrt'**, 'log2'] bootstrap: [**True**, False] max_leaf_nodes: [None, 10, 20, 50]	activation: [**ReLU**, ELU, tanh] solver: [adam, **sgd**] learning_rate_init: [0.001, 0.01, **0.1**, 1]

B APPENDIX: EMPIRICAL DATA SHOWING THE DIFFERENCE BETWEEN FULL-SERVER MONITORED POWER AND RAPL POWER VALUES



(a) Sysbench

(b) Memcached

Figure 6: Power values reported by Intel RAPL [9] and the full-server power meter when running (a) Sysbench, and (b) Memcached.

While RAPL power values can serve as ground truth for CPU and/or DRAM power consumption, they are not accurate indicators of full-server power, as shown in Figure 6. We empirically show this shortcoming with Sysbench and Memcached workloads. In general, across workloads, we found that the sum of CPU package and DRAM power values for RAPL is 30–50% lower than full-server monitored power values. Further, within a workload execution, the ratio of full-server to RAPL power values varies significantly, by as much as 1.1–1.7× for the workloads we experimented with. This is to be expected as RAPL does not include power consumed by, for example, the disks, motherboard, network, or non-integrated GPU(s).

C APPENDIX: HYPER-PARAMETER TUNING

All ML models we experimented with were first tuned via Grid Search to identify the best hyper-parameter values. Table 3 shows the hyper-parameter tuning details for five ML models (XGBoost, SVR, DT, RF, and NN), along with the best values chosen. For XGBoost, parameters such as learning rate, number of estimators, and maximum depth were adjusted to find a balance between model complexity and accuracy. SVR tuning focused on the regularization parameter C, kernel type, kernel coefficient (gamma), and epsilon (for epsilon-SVR model). For kernel type, our experiments showed that the RBF (Radial Basis Function) kernel had the best prediction accuracy. RF and DT had similar tuning to prevent overfitting while maintaining model depth. For RF, we also considered the number of estimators (300 in our case) for better accuracy. For NN, we used the Multi-layer Perceptron regressor from scikit-learn with ReLU activation and three hidden layers; the three hidden layers had size of 512, 16, and 16 neurons. For LR, we experimented with and without intercept. Including intercept, as mentioned earlier, gave better results. We also tried Ridge regression as an alternative to LR and Lasso, but its prediction accuracy was worse (5–6% higher MAPE than LR and Lasso), so we do not include it in our evaluation. For regularization, since Ridge regression did not work well, we had the alpha hyper-parameter set to 0.1 for Lasso (L1 regularization).