A Dynamic GPU Power Prediction Framework for Production HPC Applications

Fatih Acun[1], Zhengji Zhao[2], Ermal Rrapaj[2], Brian Austin[2], Ayse K. Coskun[1], and Nicholas J. Wright[2]
[1]Boston University, Boston, USA, {acun, acoskun}@bu.edu
[2]Lawrence Berkeley National Laboratory, Berkeley, USA, {zzhao, ermalrrapaj, baustin, njwright}@lbl.gov

***Abstract*—As exascale systems increasingly rely on hardware overprovisioning to sustain throughput under strict power limits, accurate, fine-grained power prediction becomes critical. We present a machine learning framework leveraging lightweight, always-on NVIDIA DCGM telemetry from the Perlmutter supercomputer to predict runtime GPU power consumption, designed with real-time deployment constraints in mind. Using a month-long production telemetry dataset for training and evaluation, we demonstrate that a generic, application-agnostic model achieves an average Root Mean Squared Error (RMSE) of 50 W. Specialized application-aware models for dominant workloads (e.g., Chroma, VASP) further reduce prediction error by up to 48% compared to generic models, while combining DCGM metrics with historical power data improves accuracy by up to 15.8% over power-only models with negligible inference latency. Our framework provides a scalable building block for dynamic power management systems that allow the hardware-overprovisioned systems to reclaim performance otherwise lost to static power caps.**

***Index Terms*—HPC power consumption, GPU power prediction, machine learning**

I. Introduction

With data center power usage projected to rise sharply in the coming years [1], HPC centers are increasingly compelled to operate under strict power limits. To sustain improvements in system throughput under these constraints, centers are adopting the hardware-overprovisioned system model [2], where the aggregate peak power of the hardware intentionally exceeds the site's available power capacity.

In hardware-overprovisioned systems, Dynamic Power Management (DPM) is crucial: by quickly directing power to where it is most needed, it maximizes system throughput while staying within the power limit. DPM has proven effective on CPU systems, for example, PowerSched [3] reduces power consumption of the Hawk [4] cluster by 20% with no performance loss [5]. However, GPU-accelerated systems have yet to see documented production-scale DPM deployment despite their dominance in HPC power consumption. This motivates the need for methods that can support future real-time power-aware control, including frameworks for fast and accurate prediction of dynamic power usage.

Prior work often formulates power consumption prediction as estimating aggregated metrics such as average or peak power before job execution [6], [7]. While these approaches offer improvements over imposing a system-wide, uniform power limit, they fail to capture fine-grained runtime power dynamics and consequently oversimplify the highly variable power consumption patterns of HPC workloads [8], [9]. Previous attempts at fine-grained power prediction, while advancing the state-of-the-art [10], [11], often rely on complex, deep-stack telemetry or are not validated against the highly diverse scientific workloads found in production HPC environments, making them infeasible for real-time deployment. To address this critical gap, we propose a machine learning (ML) prediction framework that performs short-term (e.g., 10 to 30 seconds) per-GPU power consumption predictions. We train and evaluate prediction models using a month-long production telemetry dataset from Perlmutter [12], an HPE Cray EX system based on NVIDIA A100 GPUs at the National Energy Research Scientific Computing Center (NERSC). Our main contributions are as follows:

- We develop an ML-based approach for runtime GPU power prediction designed for real-time deployment, using only lightweight, always-on telemetry collected via NVIDIA Data Center GPU Manager (DCGM) [13].
- We provide a generic model training approach, using application-agnostic telemetry data to predict future power consumption for arbitrary applications, achieving an average Root Mean Squared Error (RMSE) of 50 W across different application categories.
- We introduce an application-specific model training methodology, further reducing prediction error by up to 48% compared to generic models, using isolated telemetry datasets across five production HPC applications, Chroma, VASP, E3SM, ATLAS, and Espresso, which collectively account for over one-third of Perlmutter's total GPU time [14].
- We demonstrate that incorporating DCGM metrics, beyond historical power consumption alone, improves power prediction accuracy, achieving up to a 15.8% reduction in RMSE over power-only models.

Utilizing standard NVIDIA DCGM metrics, our method is readily applicable to all NVIDIA GPU systems and can be extended to other vendors' GPUs where lightweight activity metrics are available. With negligible inference latency and reliance on lightweight telemetry, our method is well-suited for real-time deployment and provides a key building block for DPM on hardware-overprovisioned systems.

The remainder of this paper is organized as follows. Section II reviews related work and highlights how our approach differs. Section III describes the data collection process and

analysis, followed by a description of our methodology in Section IV. Section V presents the evaluation results, and Section VI discusses future directions for further enhancing our prediction framework. Finally, Section VII concludes the paper.

II. Related Work

A substantial body of prior work has addressed power optimization across the HPC system stack, beginning with coarse-grained, system-level modeling that estimates average or peak job power for capacity planning and resource provisioning [7], [15]. In production environments, power management is typically enforced through static mechanisms such as uniform power capping or Dynamic Voltage and Frequency Scaling (DVFS), often at the cost of reduced performance. Several large HPC centers have explored these approaches to stay within prescribed power budgets and to understand their system-wide performance impact. For example, CINES (the National Computer Center for Higher Education in France) reduced GPU frequencies on Adastra from 1.7 GHz to 1.5 GHz [16] to lower power consumption. The Flatiron Institute [17] has experimented with reducing peak GPU power in one of its clusters from 400 W per GPU to 225 W. Similarly, system-wide power-capping experiments have been conducted on MIT's Supercloud HPC system, where GPUs were limited to 60% of their maximum power [18]. While effective at constraining total power usage, these system-level, static policies do not adapt to workload dynamics and therefore cannot optimize performance under a fixed power envelope.

More recent application-aware efforts improve upon uniform power capping by tailoring power limits to individual jobs, but they largely remain static in nature and focus on predicting aggregate metrics such as mean or maximum job power rather than runtime behavior [6], [7]. As a result, they still leave performance headroom unexploited, particularly during phases where jobs operate below their allocated power budget. Studies on leadership-scale systems such as Summit [19] and Polaris [20], provide valuable insights into cluster- or job-level power characteristics, yet they do not capture the fine-grained power fluctuations required for effective DPM. These limitations highlight the need for accurate runtime power prediction to enable dynamic, application-aware power management in modern HPC systems.

To capture the complex dynamics of GPU-accelerated workloads, recent work has advanced fine-grained, telemetry-based prediction frameworks. These models enable application-aware power management by utilizing low-level data, involving complex techniques like Graph Neural Networks (GNN) [10], employing adaptive incremental learning [21], ensemble methods [22], or leveraging architectural performance counters to forecast unseen workloads [11], [23], [24]. Despite these significant advances, the complexity and variability of production workloads introduce high prediction error, which has prevented these methods from being widely integrated into production DPM systems.

We present an ML-based, runtime GPU power prediction framework, designed with real-time deployment constraints in mind, using standard NVIDIA DCGM telemetry, eliminating the need for intrusive profiling or application-specific probes. Trained on a month-long production dataset, our generic predictor achieves an average RMSE of 50 W. Application-aware variants—distinguished only by their training dataset—further reduce error by 48% over the generic models. We also demonstrate that incorporating DCGM metrics beyond historical power usage reduces prediction error by up to 15.8% compared to power-only models. By leveraging standard DCGM metrics and a negligible inference latency of just 3 µs, this framework provides a robust building block for dynamic power management that allows hardware-overprovisioned systems to maximize throughput while maintaining sustainable operation.

III. Data Collection and Analysis

A. Perlmutter System Architecture

This work was performed on the Perlmutter [12] supercomputer at NERSC, a heterogeneous HPE Cray Shasta system comprising 1,792 GPU-accelerated nodes and 3,072 CPU-only nodes. Each GPU-accelerated node includes one AMD EPYC 7763 "Milan" processor, 256 GB of DDR4 memory, four NVIDIA A100 Tensor Core GPUs, and four HPE Slingshot 11 "Cassini" NICs. Each A100 GPU offers peak performance of 9.7 TFLOPS (FP64) and 19.5 TFLOPS (FP64 tensor operations), with 40 GB of HBM2 memory and 1.5 TB/s peak bandwidth. Among the GPU-accelerated nodes, 256 have 80 GB of HBM, while 1,536 have 40 GB.

The estimated Thermal Design Power (TDP) of a 40 GB GPU node is 2,350 W, accounting for 280 W for the CPU, 400 W per GPU, and roughly 470 W (20%) for peripherals including memory and NICs. Perlmutter uses Slurm as its batch system. Additional details are available online [12].

B. DCGM Telemetry Data

NVIDIA DCGM provides a rich set of lightweight GPU metrics, enabling always-on, system-wide monitoring. On Perlmutter, Lightweight Distributed Metric Service (LDMS) [25] runs the DCGM sampler at 10-second intervals on all GPU nodes, forwarding the collected data to the database server via the Kafka bus [26]. This data can be accessed either by querying the backend database or subscribing to Kafka bus through Redfish [27] endpoints for reduced overhead and scalability. While real-time deployments can use Kafka subscriptions to minimize overhead, in this work, we rely on the backend database. The DCGM metrics capture GPU functional unit activities, including FLOPS, memory, and communication operations. Table I lists the metrics used in our prediction models.

C. Slurm Job Submission Data

Perlmutter utilizes Slurm as its batch system. We accessed comprehensive job data, including job name, ID, node count, allocated node IDs, start/end times, duration, and the submit

TABLE I: List of collected DCGM metrics. For simplicity, the common prefixes like DCGM_FI_PROF_PIPE_ and DCGM_FI_DEV_ are omitted from the metric names.

Metric	Description
fp16_active	Ratio of cycles the FP16 pipe is active (excluding HMMA).
fp32_active	Ratio of cycles FP32 pipe is active.
fp64_active	Ratio of cycles FP64 pipe is active.
gpu_utilization	GPU utilization (percentage of active processing time).
gr_engine_active	Ratio of cycles Graphics/Compute Engine is active.
nvlink_rx_bytes	Total active NVLink receive (RX) bytes.
nvlink_tx_bytes	Total active NVLink transmit (TX) bytes.
pcie_rx_bytes	Active PCIe receive (RX) bytes (Host $\rightarrow$ Device).
pcie_tx_bytes	Active PCIe transmit (TX) bytes (Device $\rightarrow$ Host).
sm_active	Ratio of cycles at least one warp was scheduled on the Streaming Multiprocessor (SM).
sm_occupancy	SM occupancy (average ratio of active warps to max warps).
tensor_active	Ratio of cycles Tensor pipe is active (IMMA, HMMA, or DFMA).
tensor_hmma_active	Ratio of cycles Tensor HMMA pipe is active.
dram_active	Ratio of cycles DRAM memory controller is active.
mem_copy_utilization	Memory Copy Engine utilization: percentage of time the engine was active during the sampling period.
memory_temperature	Device memory temperature (°C).
fb_free	Free frame buffer memory (MiB).
fb_used	Used frame buffer memory (MiB).
gpu_temperature	GPU temperature (°C)
power_usage	GPU power consumption (Watts)

TABLE II: Summary of application-agnostic and application-specific DCGM telemetry datasets.

Dataset	# Jobs	# Rows
App-Agnostic	–	21,536,280
Chroma	155	73,992,225
VASP	33,773	39,553,021
E3SM	504	25,117,066
ATLAS	574	8,281,380
Espresso	477	3,726,367

line (identifying the executed application binaries), directly from the Slurm accounting logs using `sacct` [28].

"`sacct -o submitline`" command returns srun command lines, from which we extract the executable names. We further map executable names to application names using exact or near-exact regex matching. For example, for the VASP application, we matched 33,773 jobs, which consist only of executable names `vasp_gam`, `vasp_ncl`, or `vasp_std`. We perform labeling as post-processing, but for the real-time deployment, the same lightweight parsing can be done immediately after job submission.

D. Dataset Curation for Time Series Prediction

To train time series prediction models, we prepare six datasets collected in March 2025 on the Perlmutter supercomputer, as listed in Table II. The first dataset, *app-agnostic*, refers to the one-day-long DCGM telemetry dataset collected over 700 GPU nodes on the Perlmutter system without any job or application information included. The remaining five datasets are application-specific for Chroma [29], VASP [30], [31], ATLAS [32], E3SM [33], [34], and Espresso [35]. Figure 1 illustrates our data pipeline, which consists of three main stages: (1) preprocessing, where we clean and filter the raw DCGM and Slurm datasets, (2) merging, where we join DCGM telemetry with Slurm job records to incorporate job-level information; and (3) splitting, where we partition the merged dataset into distinct application categories. In the preprocessing stage, we filter out DCGM measurements that fall outside valid metric ranges, remove jobs with more than 10% missing data, and apply linear interpolation to fill remaining gaps. For Slurm data, we retain jobs submitted to the regular and premium GPU queues only, excluding those from shared or interactive queues. To construct application-specific telemetry datasets, we merge the DCGM and Slurm records using the Slurm job metadata, which includes job start and end timestamps and the list of allocated nodes. Finally, we divide the merged dataset into application categories, producing inputs suitable for model training. VASP ranks among the most widely used applications on Perlmutter [14] and has the highest job submission count within our selected set. In contrast, Chroma and E3SM include fewer job submissions but have a large dataset of roughly 73 and 25 million rows, respectively. This stems from the large node allocations per job for Chroma and E3SM. ATLAS and Espresso also provide sizable datasets, though smaller than the other application datasets. For application-specific datasets, we use a strict job-level temporal split (single split): jobs with both start and end times before March 24 are used for training, and jobs with both start and end times on/after March 24 are used for testing; jobs spanning the cutoff are discarded. Finally, we store the datasets in a tabular format such that each row is identified by the corresponding timestamp, node_ID, GPU_ID for app-agnostic datasets, and additionally with job_ID for application-specific datasets.

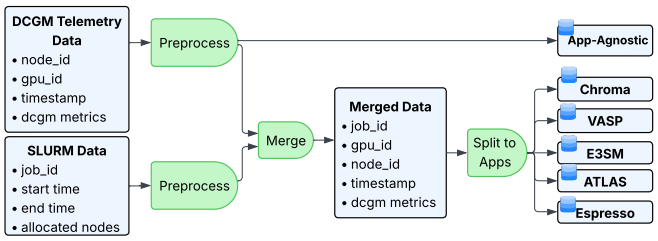


Fig. 1: An illustration of our data preprocessing pipeline, summarizing the steps from the raw DCGM and Slurm datasets to the final processed datasets.

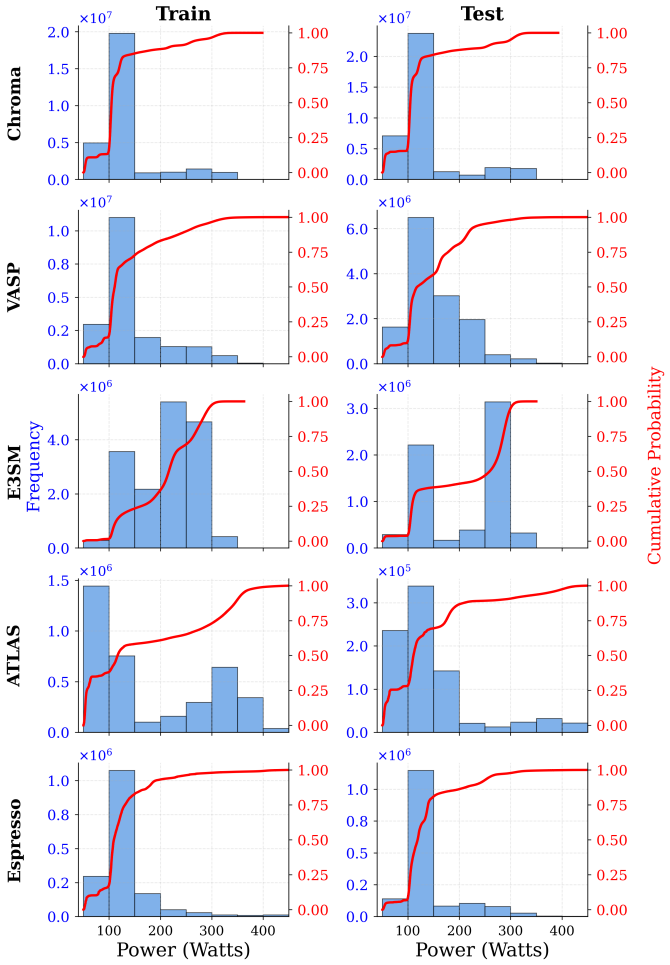


Fig. 2: GPU power consumption distributions for train and test splits across five application datasets.

E. Data Analysis

To understand the overall trends and distribution characteristics of the application datasets, we perform a data analysis that highlights both the general patterns and the unique behaviors exhibited by each application.

Power Distributions. We provide per-GPU power distributions for each dataset to identify the dense regions for power measurements across different applications. Figure 2 shows the distributions of application datasets for both train and test splits. The power distributions for Chroma, VASP, and Espresso are dominated by lower power values, with 80% of power measurements remaining below 200 W, which corresponds to half of the 400 W TDP of NVIDIA A100 GPUs. In contrast, the power data for E3SM and ATLAS span a wider range of the GPU power domain and exhibit a bimodal pattern. For E3SM, most measurements fall within the 100–150 W and 200–250 W intervals. For ATLAS, the majority of samples are less than 150 W, and also the density is moderately high in the power range 250-400 W. Although the training and test distributions are generally similar across applications, certain differences are observed. For instance, ATLAS shows noticeable differences in the proportion of high-power samples (200 W and above) between the training and test sets, while E3SM's distribution differs in the 150–250 W range. These variations highlight the importance of collecting datasets that span sufficiently long time periods, emphasizing that both large volumes of data and diverse samples are essential to capture the full range of HPC application behaviors.

Job Size Distribution. Analyzing the size of jobs is essential for informing power management strategies in HPC systems, as operators must focus on workloads that consume a substantial fraction of system-level power. Large jobs, in particular, dominate overall power usage and therefore serve as key targets for effective power predictions. In this section, we analyze our application datasets to characterize the distribution of job sizes and the total node-hours they consume on the Perlmutter system.

Figure 3 presents the distributions of total node-hours grouped by the number of nodes allocated for job submissions. We observe that E3SM jobs contribute the majority of node-hours at the 512-node scale, reflecting their tendency toward large, system-spanning allocations. For Chroma, most of the node-hours are spent by large jobs allocating 256 or more nodes. Similarly, ATLAS shows substantial node-hour contributions from jobs allocating 64, 256, and 512 nodes. In contrast, VASP exhibits a smaller scale of parallelism, with jobs allocating at most 150 nodes and the bulk of node-hours arising from jobs using four or fewer nodes. Espresso jobs primarily allocate 8 or 16 nodes, representing small-sized workloads relative to the other applications.

Temporal Power Consumption Analysis. Prediction of power consumption is a fundamentally challenging task when temporal fluctuations are observed. We analyze the temporal characteristics of power consumption to assess the predictability of power across different applications. We first look at the average of the differences of power consumption with the Mean Absolute Power Change (MAPC) metric defined as follows:

$$\text{MAPC} = \frac{1}{n-1}\sum_{t=1}^{n-1} |P_{t+1} - P_t| \tag{1}$$

where n is the total number of power measurements, P_t and P_{t+1} are the per-GPU power consumption at time t and $t+1$, respectively. Figure 4 shows the MAPC values of applications. E3SM has the highest MAPC with 53 W, followed by Chroma and VASP with 29 W. ATLAS and Espresso show relatively lower variability with 27 and 21 W MAPC, respectively.

The above analysis highlights substantial heterogeneity in GPU power behavior across applications, job sizes, and runtime, underscoring the need for prediction models that capture both temporal dynamics and application-specific characteristics. Motivated by these observations, we next describe our machine learning methodology for GPU power prediction.

IV. Methodology

To perform forecasts by capturing the complex temporal patterns in time-series data, ML methods provide promising

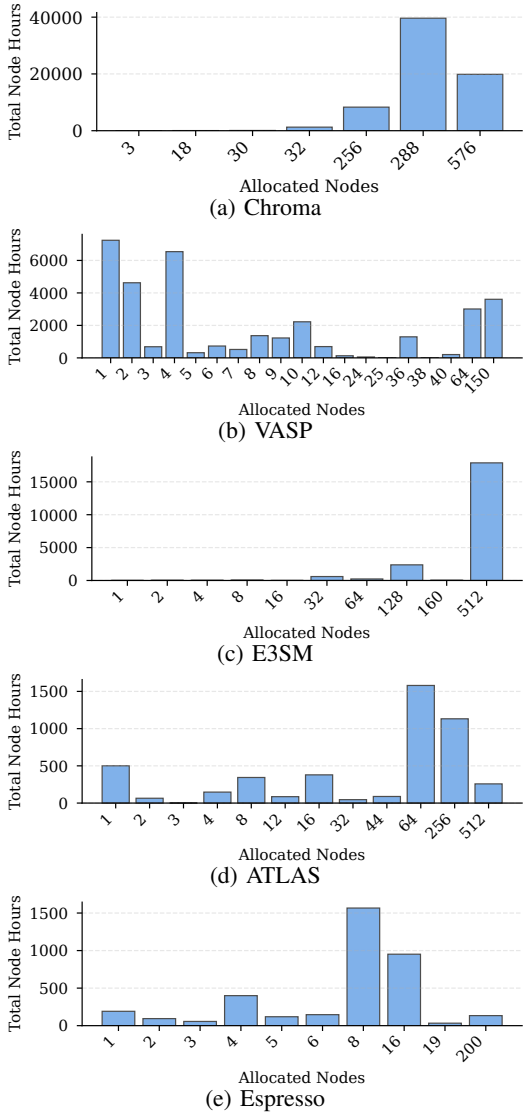


Fig. 3: Total node hours grouped by number of allocated nodes across application datasets.

solutions. In this section, we describe our ML-based approach and the formulation of the power prediction problem.

A. Background

Recurrent Neural Network (RNN) architectures are specifically designed for time-series prediction by incorporating feedback loops to utilize past data for future predictions [36]. A common issue with RNNs is the vanishing gradient problem, where gradients' magnitude decreases exponentially as they are backpropagated through time, making it difficult for the model to learn long-range dependencies [37]. In this study, we utilize Long Short-Term Memory (LSTM) [38] architecture, which includes a gating mechanism to control information flow and achieve better performance than vanilla RNNs. LSTMs introduce memory cells that replace each recurrent node in RNNs. Each memory cell stores an internal state and uses three multiplicative gates to regulate information flow [39].

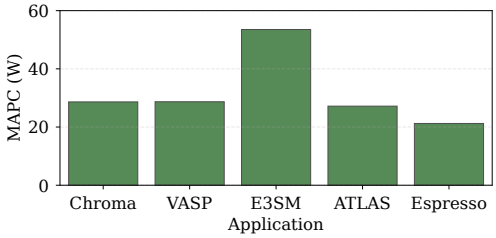


Fig. 4: Mean Absolute Power Change (MAPC) across all applications, showing the average fluctuation over consecutive power consumption measurements.

The input gate, i_t, controls how new inputs influence the internal state, the forget gate [40], f_t, determines how much of the existing state is kept or discarded, and the output gate, o_t, determines how the internal state contributes to the cell's output.

Subsequent developments include the Bi-directional LSTM and Stacked LSTM [41], to further enhance prediction performance, and the simpler GRU (Gated Recurrent Unit) model with reduced parameter count and faster training [42]. More recently, the transformer model [43] introduced the self-attention mechanism, revolutionizing deep learning and fostering the widespread integration of attention mechanisms across various neural network architectures. These models exhibit robust data analysis and strong feature extraction abilities and are the backbone of large language models (LLMs), but they can increase drastically in the number of parameters and may require large amounts of data. To keep both the model and training lightweight, here we focus on LSTMs.

B. Problem Formulation

We formulate the time series prediction problem by a prediction model f that takes a history window of length S and predicts a horizon of length H as follows:

$$\hat{y}_{t+H}, \ldots, \hat{y}_{t+1} = f(X_t^N, X_{t-1}^N, \ldots, X_{t-S}^N; \theta), \qquad (2)$$

where t represents the current time-step, θ refers to model parameters, X_t^N is the input data for time step t with N features, and $\hat{y}$ denotes the predictions. Our feature set includes the DCGM metrics that encapsulate power consumption, as previously listed in Table I, to predict the future GPU power consumption. Figure 5 illustrates the prediction task we formulate for GPU power prediction. We emphasize that the model produces predictions for a single GPU.

C. Model Training and Implementation

Our training approach is structured along two dimensions:

1) univariate or multivariate,
2) application-specific or generic.

Figure 6 illustrates our model training and evaluation pipeline based on these design choices. We develop, train, and compare models using only univariate power consumption data with models trained on the full set of DCGM features, including power. This approach allows us to assess the contribution

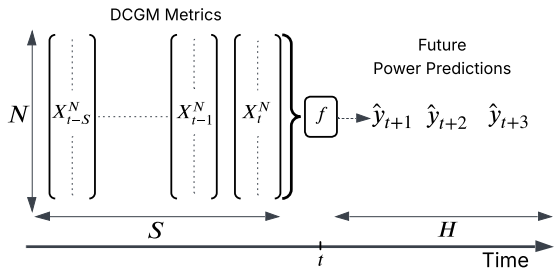


Fig. 5: An illustration of GPU power prediction by the prediction model f, over a prediction horizon H, using features of size $N \times S$, where N is the number of DCGM metrics and S is the past window length.

of additional DCGM metrics to forecasting future power consumption.

Our use of application-specific datasets is motivated by the hypothesis that models trained on isolated datasets can better learn application-specific patterns of the target applications than generic models trained on diverse application-agnostic data. To evaluate this hypothesis, we train six models: one generic model using the application-agnostic dataset and five application-specific models using the corresponding application datasets. Each application-specific model is evaluated on its respective test set. For the generic model, we assess its performance by testing it on the five application test sets, enabling a direct comparison with the application-specific models across all applications.

For the experimental setup, we set the prediction horizon to $H = 3$, providing multi-step predictions over a 30-second window with 10-second granularity. The input sequence length is set to $S = 12$, utilizing the previous 2 minutes of data, and we apply sliding windows with a stride equal to 1. Our architecture consists of a stacked LSTM network with 5 layers of 512 hidden units each, implemented in PyTorch 2.9. To prevent overfitting, we apply a dropout rate of 0.2. Optimization is performed using the Adam optimizer [44] with Mean Squared Error (MSE) as the loss function. We employ a `ReduceLROnPlateau` scheduler with an initial learning rate of 0.001, a decay rate of 0.5, and a patience of 10 epochs. The model is trained for 50 epochs using a random seed of 42 for reproducibility. We reserve the final 20% of the training data for validation to perform model selection based on the lowest validation loss. While these hyperparameters were optimized for our specific dataset, they remain adaptable to different system scales or workload characteristics.

D. Baseline Methods

In addition to the ML-based training approaches described in the previous section, we compare our method against two categories of baselines: static power estimation methods and traditional time-series models for dynamic prediction. The static baselines include *oracle_mean*, an idealized predictor that assumes the true average GPU power consumption of a job is known in advance and predicts that value exactly, and *train_mean*, which generates constant application-level power predictions by averaging GPU power consumption across the training set for each application. For dynamic prediction, we consider two time-series baselines: persistence, which uses the most recently observed value as the forecast for future timesteps, and a simple moving average (SMA), which predicts using the mean of observations over a fixed historical window. For SMA, we use a window length of 12 to match the sequence length of our LSTM models.

V. Evaluation

In this section, we evaluate our ML-based prediction framework by comparing application-specific and generic models, as well as models trained using univariate power data versus multivariate DCGM telemetry data. Beyond providing aggregated accuracy metrics for evaluation and performance comparison across models, we provide an analysis of prediction error distributions and temporal prediction behavior in comparison to the actual GPU power consumption.

A. Metrics

We evaluate our method using aggregated error metrics, including Root Mean Squared Error (RMSE), Mean Absolute Percentage Error (MAPE), and the regression score R^2 defined as follows:

$$\mathrm{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2}, \tag{3}$$

$$\mathrm{MAPE} = \frac{100}{n}\sum_{i=1}^{n}\left|\frac{\hat{y}_i - y_i}{y_i}\right|, \tag{4}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}, \tag{5}$$

where y_i and $\hat{y}_i$ represent the actual and predicted GPU power consumption of i^{th} sample, respectively, and n denotes the total number of prediction samples, $\bar{y}$ denotes the average of the actual power consumptions. RMSE is particularly informative because it penalizes larger prediction errors more heavily than Mean Absolute Error (MAE), making it well-suited for assessing deviations in power forecasts. On the other hand, MAPE provides a percentage level error, useful for assessing the prediction quality by providing a normalized metric. R^2, often referred to as the coefficient of determination, calculates the proportion of variance in the actual GPU power consumption that is explained by the prediction model, relative to a constant mean predictor.

B. Generic and Application-Specific Model Performance

In production HPC environments, job submissions can typically be associated with particular applications through user-provided job names or the executables invoked. This enables power prediction frameworks to select between application-specific models when application identity is known, and generic models for cases where the application is not identifiable. Accordingly, our evaluation mirrors this usage such

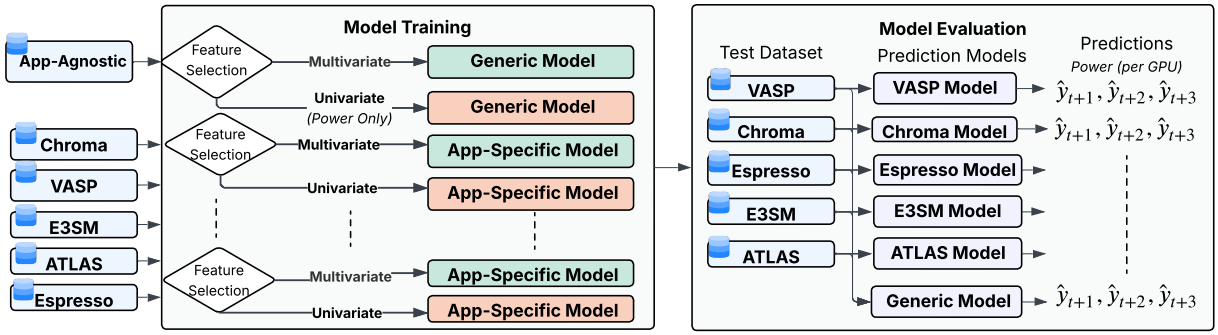


Fig. 6: Model training and evaluation procedure. We train generic and application-specific models with single-feature (power) and multi-feature input. The performance is evaluated on test data from five different production workloads.

that application-specific models are tested on their respective application test sets, whereas the generic model is evaluated on all test sets to provide a consistent basis for comparison.

Figure 7 summarizes the performance of different model configurations across the three evaluation metrics. Models trained with the full feature set are shown in blue, while univariate power-only models are shown in green. Darker shades indicate application-specific models, whereas lighter shades denote generic models. Overall, application-specific models achieve the best performance across all metrics and consistently outperform generic models that use the same feature set. Comparing RMSE between application-specific and corresponding generic models for each application, we observe substantial accuracy gains. For Chroma, the RMSE is reduced by 48% compared to generic models, decreasing from 54.46 W to 28.22 W. Significant improvements are also observed for VASP, E3SM, and ATLAS, with RMSE reductions of 10.3%, 12%, and 12.5%, respectively. In contrast, Espresso shows no meaningful improvement. We attribute this to the relatively small dataset available for Espresso, as previously shown in Table II. A similar trend is observed for R^2, where application-specific models exhibit notable gains, with improvements of up to 0.52 compared to generic models.

C. Comparison to Baseline Methods

Figure 7 shows the performance of static and dynamic baseline methods (last 4 red and orange bars in each panel). The static *train_mean* predictor, which uses the training-set mean power, consistently performs worst across applications. This mirrors conventional static approaches, such as nvidia-smi's power-profiles [45], which estimate power from application identity rather than runtime behavior. Its limitations are most pronounced when training and test power distributions differ (Figure 2), with RMSE exceeding 80 W. The *oracle_mean* predictor, which assumes perfect knowledge of the test-set mean, performs better as expected. Yet, our application-specific runtime power models outperform *oracle_mean* for most workloads, especially those with substantial temporal fluctuations, demonstrating the advantage of dynamic, fine-grained prediction. The gap is smaller for applications like Espresso and ATLAS, whose power is relatively flat (Figure 4), making mean-based estimates more effective.

Among the dynamic baselines, SMA provides a stronger point of comparison than the static predictors, indicating that short-term temporal smoothing alone can already capture part

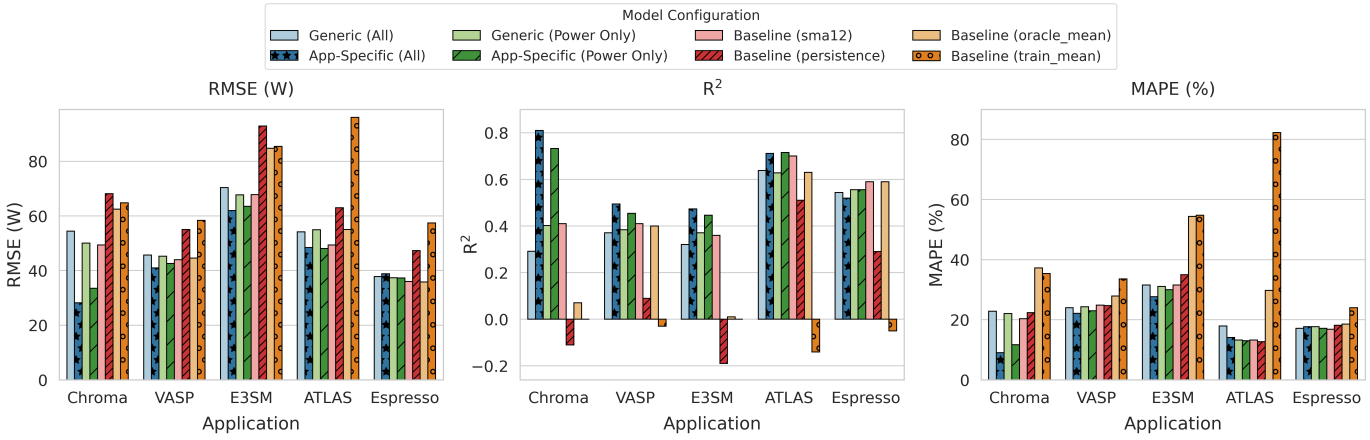


Fig. 7: Performance metrics for application-specific and generic models, using all DCGM features and only power, on test datasets across applications. Note that lower RMSE and MAPE, and higher R^2 imply better prediction performance.

of the local power dynamics. By contrast, the *persistence* baseline performs poorly overall, suggesting that simply extrapolating the most recent observation is insufficient to model the fluctuations in GPU power. SMA yields performance comparable to, and in some cases better than, the generic models, suggesting that generic cross-application predictors may not always exploit temporal structure more effectively than a lightweight history-based heuristic. This observation highlights the difficulty of learning broadly transferable power models across diverse applications without application-specific adaptation. At the same time, it reinforces our main result: once application-specific information is incorporated, the learned models consistently surpass both static and time-series baselines, including SMA. These results therefore suggest that while simple temporal baselines can remain competitive for generic prediction, application-specific learning is key to achieving the best overall accuracy.

D. Contribution of DCGM Metrics on Model Performance

We further investigate the contribution of DCGM metrics to power consumption prediction by comparing multivariate models that incorporate the full set of DCGM features with univariate models that use only historical power consumption as input. This comparison isolates the extent to which additional telemetry improves predictive accuracy beyond what can be learned from power history alone.

As shown in Figure 7, incorporating DCGM metrics yields a considerable improvement in Chroma, with a 15.8% reduction in RMSE with respect to the power-only version of its application-specific model. On the other hand, the performance improvements for VASP and E3SM are modest, with 4% and 2.5%, respectively. Models that use all DCGM features provide slightly worse results for ATLAS and Espresso. Our analysis of temporal power characteristics shows that E3SM, Chroma, and VASP exhibit the highest average power fluctuations between consecutive time steps, as indicated by the highest MAPC values among the five applications in Figure 4. These results suggest that, for workloads where power dynamics are strongly correlated with recent power history, univariate models may already capture much of the predictive structure, whereas multivariate telemetry offers additional value primarily for applications with more complex or variable behavior. In addition, incorporating the full set of DCGM telemetry expands the feature dimension from 1 to 20, substantially increasing the complexity of the learning problem. Under such conditions, ML models generally require larger training datasets to reliably learn meaningful patterns [46]. For applications with comparatively limited data, such as Espresso and ATLAS, this increased feature dimensionality may reduce the effectiveness of multivariate models.

E. Error Analysis for Best Performing Models Across Applications

Model Performance Across Different Metrics. To compare model performance across applications, we jointly examine RMSE, R^2, and MAPE, which provide complementary perspectives on prediction quality. RMSE, expressed in Watts, offers an intuitive measure of absolute prediction error and is lowest for Espresso, with an RMSE of 37 W. However, because Espresso operates at a lower power range, mostly less than 200 W, than the other applications (Figure 2), RMSE alone does not fully capture relative prediction accuracy. In contrast, MAPE normalizes the error by the ground-truth power consumption, enabling a fairer comparison across applications with different power profiles. Under this metric, ATLAS exhibits the best prediction accuracy with a 13% MAPE, followed by Espresso with 17%. Looking at the R^2 metric, the best models for each application have values spanning between 0.47 and 0.81. We note that moderate R^2 values do not necessarily indicate poor predictive performance in time-series data with high intrinsic variability, where a large fraction of the signal variance is inherently unpredictable [47].

Our results suggest that analyzing the predictions with respect to various error metrics provides multiple perspectives to assess the quality of forecasts for HPC applications. While using metrics that provide errors in Watts can guide DPM methods to take into account power emergencies, percentage-based metrics give a general understanding of model prediction quality with respect to the individual power consumption range of applications.

Temporal Analysis of Predictions. As all the metrics agree, E3SM stands out as the most challenging application to predict future power consumption. Our earlier analysis of MAPC, shown in Figure 4, shows that the temporal power consumption characteristic of per-GPU power consumption in E3SM has the highest average magnitude of fluctuations between consecutive time-steps. Figure 8 presents representative timelines from production runs, comparing actual GPU power consumption with one-step-ahead predictions generated by the best-performing model for each application. As shown in Figure 8a, E3SM exhibits highly dynamic GPU power consumption, rapidly fluctuating between 100 and 300 W with no obvious repetitive patterns. Such volatility increases the difficulty of accurate power prediction; nevertheless, when latent temporal structure exists, LSTM models can still learn and exploit application-specific power dynamics. In contrast, Figure 8b illustrates a VASP workload whose GPU power varies within a narrower range of approximately 100 to 200 W and displays a more regular pattern. In this case, the prediction model closely tracks the measured power consumption despite the ongoing fluctuations. For the examples of Espresso and ATLAS shown in Figure 8c and Figure 8d, the power consumption is mostly steady, with rare spikes up to 300 W in Espresso.

To examine temporal prediction behavior in more detail, we compare application-specific and generic models for Chroma using the full set of multivariate DCGM features in Figure 9. The application-specific model closely tracks the recurring GPU power spikes exceeding 200 W, while the generic model exhibits noticeable lag and larger absolute errors, as illustrated in the lower panel of Figure 9.

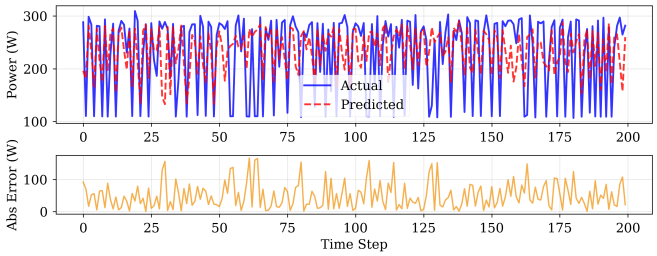
Because our model predicts GPU power up to 30 seconds

into the future at a 10-second granularity (three prediction steps), we analyze prediction accuracy at each step to assess performance degradation over the prediction horizon. Figure 10 reports the RMSE for each prediction step across all five applications. As expected, prediction error increases for later steps; however, this increase remains bounded, with RMSE rising by at most 10% from the first to the third step.

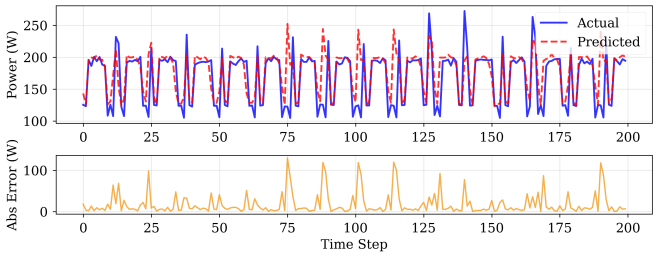
Our observations show that temporal power consumption characteristics provide critical insight into the predictability of future GPU power usage. While highly fluctuating workloads are inherently more challenging to forecast, applications that exhibit recurring temporal patterns remain predictable despite significant power variability.

Analysis of Error Distributions. Beyond aggregated accuracy metrics, we analyze the distribution of prediction errors to provide a more transparent view of model reliability. Figure 11 shows the cumulative distribution functions (CDFs) of absolute prediction errors across all applications, with the 75^{th} and 90^{th} percentiles indicated by yellow and red dashed lines, respectively. This perspective enables DPM frameworks to quantify the likelihood that prediction errors remain below a given power threshold, expressed in Watts. Among the evaluated workloads, E3SM exhibits the lowest prediction reliability, with a 90^{th} percentile error of 106 W, indicating a non-negligible risk of large errors in approximately 10% of predictions. In contrast, the remaining applications demonstrate substantially higher predictability, with absolute errors below 41 W in 75% of the predictions, and below 76 W in 90% of the predictions.

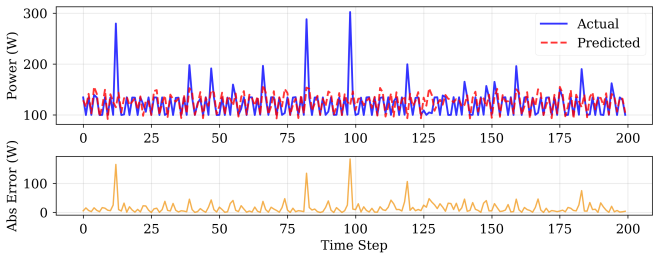
In addition to analyzing error distributions, we examine



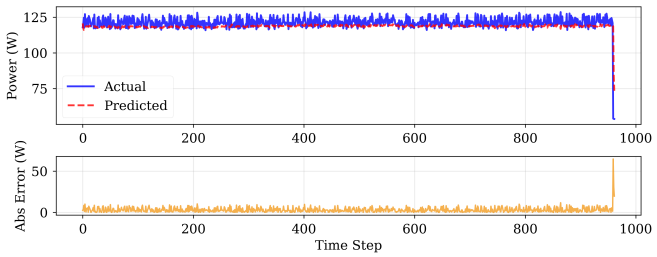
(a) E3SM: highly dynamic power consumption fluctuating between 100 and 300 W.



(b) VASP: accurate prediction of repetitive power fluctuations between 100 and 200 W.



(c) Espresso: mostly stable power with occasional spikes up to 300 W.



(d) ATLAS: steady power behavior (120–125 W) with prediction error below 10 W.

Fig. 8: One-step-ahead GPU power predictions for four representative production HPC applications. Dashed red lines denote model predictions, while solid blue lines indicate actual GPU power consumption. Each time step corresponds to a 10-second interval.

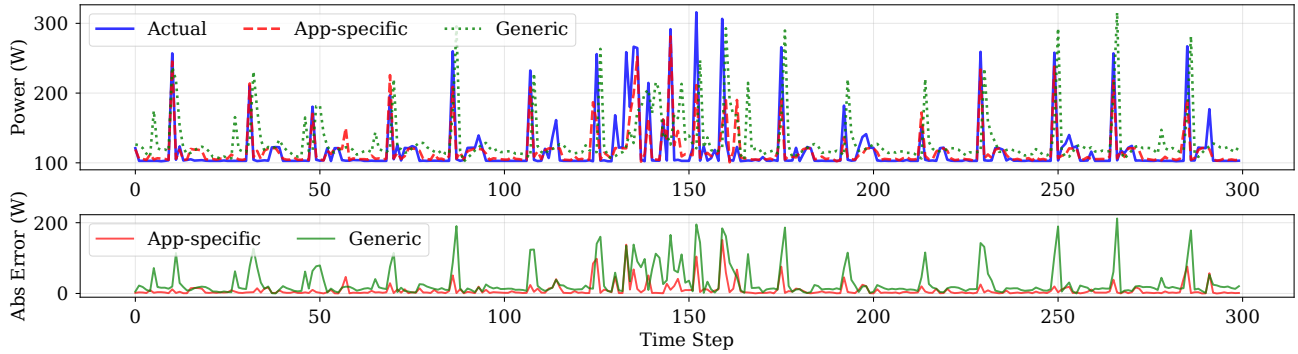


Fig. 9: Comparison of 1-step ahead predictions of generic and application-specific models, using all DCGM metrics as features, from a sample Chroma job. Each time step corresponds to a 10-second interval.

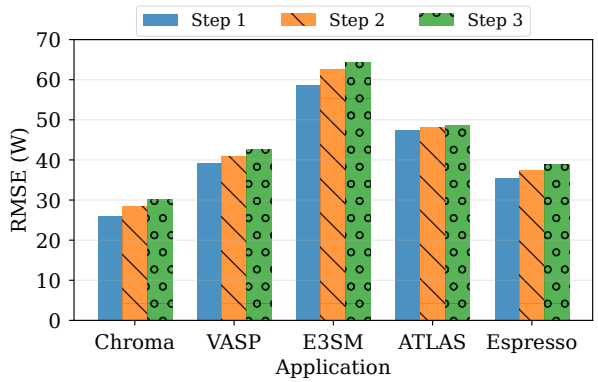


Fig. 10: RMSE of 1-, 2-, and 3-step ahead predictions of the best models across applications.

how prediction errors vary across different power regimes by computing RMSE separately for fixed intervals of actual GPU power consumption. Figure 12 reports the RMSE for each 50 W power interval across all applications. In general, RMSE increases at higher power levels, a trend that is expected and does not significantly impact overall prediction accuracy because high-power intervals contain substantially fewer samples than lower-power ranges (Figure 2). An exception is observed for E3SM, where RMSE remains elevated in the 100–150 W interval, despite this range containing a relatively large number of samples. This behavior confirms that E3SM exhibits lower prediction accuracy even in well-populated power regimes.

F. Feature Selection and Analysis

Our multivariate models use all DCGM features listed in Table I. In addition to providing models that use all features and only power, we conduct a feature-selection study to further evaluate the accuracy with a limited set of features. We use Pearson correlation to calculate the pairwise correlations across all metrics, including power. We retain metrics with correlation, $|\rho| \geq 0.3$, to power, and for any pair with inter-feature correlation $|\rho| \geq 0.8$, we keep the metric with the highest correlation to power. This yields feature sets of size 11/9/6/10/9 (including power) for Chroma/VASP/E3SM/ATLAS/Espresso. The selected features correspond to expected drivers of power (utilization/SM activity, memory activity). A detailed description of selected features and correlation matrices can be found in appendix Section IX-A.

Training and evaluating models on selected features changes RMSE by at most 0.8 W, indicating the LSTMs are largely insensitive to redundant metrics. While reduced sets remain an option to minimize overhead, using all features is practical to avoid workload-specific tuning.

VI. Discussion

A. Towards Production Deployment of Dynamic Power Management on Hardware-Overprovisioned Systems

In the exascale era, the gap between hardware's theoretical peak power and facility power budgets continues to grow, necessitating hardware over-provisioning [2], [48], where total

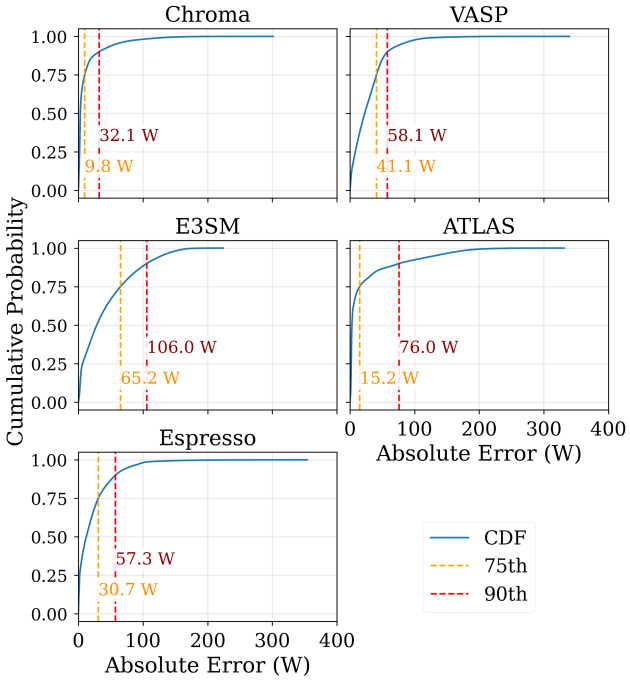


Fig. 11: CDFs of absolute prediction error for the best-performing models across applications.

TDP exceeds site capacity. Operating near physical limits reduces safety margins, making precise control essential. Our predictive framework is motivated by this setting: by forecasting power 30 seconds ahead, LSTM-based models could enable a DPM system to apply caps only when needed, thereby limiting performance loss while keeping total power draw within site constraints.

End-to-End DPM Integration. Our method addresses practical deployment requirements by relying only on NVIDIA DCGM telemetry on Perlmutter and a compact LSTM architecture. Relative to larger sequence models such as Transformers, these models use far fewer parameters and support fast inference, with an average model-only latency of about 3 µs per prediction. Although a closed-loop DPM demonstration is beyond the scope of this work, the predictor is intended as a building block for such systems. A plausible deployment strategy is a distributed design with node-local DPM agents to avoid centralized bottlenecks. The measured runtime costs

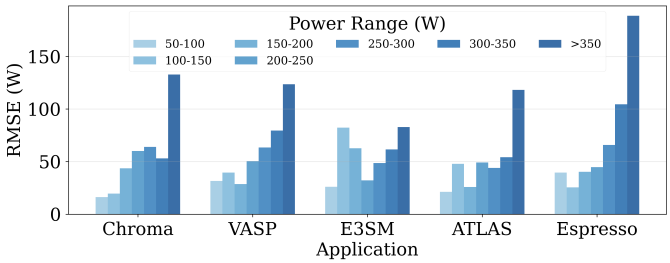


Fig. 12: Prediction error (RMSE) by power range and for each application.

on Perlmutter suggest that such a design is feasible: data acquisition takes less than 50 µs for 20 DCGM metrics read from Kafka, data preprocessing requires about 5 ms, inference takes about 3 µs, and power enforcement completes in less than 0.1 s. Altogether, the end-to-end latency remains below one second, well within the 10-second DCGM sampling interval.

Acceptable Prediction Error. Prediction quality must also be evaluated in the context of the objective. Since acceptable error depends on system- and site-level policy, we report both error distributions and summary statistics. The error CDFs in Figure 11 provide a way to estimate the probability of exceeding a given power threshold, while RMSE captures the penalty of large mispredictions. The predictor performs well for many production workloads, achieving MAPE values of 10% for Chroma and 13% for ATLAS, though highly variable workloads such as E3SM remain more challenging, with a MAPE of 27%. In production HPC environments, both the mix of deployed applications and the execution characteristics of existing workloads may change over time. Such evolution can reduce the accuracy of prediction models trained on historical data. However, the lightweight nature of our approach makes periodic retraining practical. Our model training experiments are performed using a single GPU node on Perlmutter and complete within at most ten hours for the largest datasets. As a result, retraining does not impose significant overhead in terms of either resource usage or training duration, enabling models to be updated as workload characteristics evolve.

B. Limitations and Future Directions

Job-level Prediction and Accuracy Improvement. Our current framework predicts runtime power usage for individual GPUs. While these predictions are robust, there remains a significant opportunity to further optimize accuracy. Since the job is the fundamental unit for resource scheduling and power management, a logical progression is to transition from individual device modeling to predict the spatial average power across all GPUs within a job allocation. By integrating Slurm metadata, we can synchronize and aggregate concurrent runtime data across a job's GPU pool, effectively mitigating device-level noise, as shown in the prior power analysis of the E3SM code [49]. We anticipate that this shift toward job-centric prediction will enhance the accuracy of our models.

Evaluation Across Architectures and AI Workloads. While this study uses a production dataset from an NVIDIA A100 GPU-based system, a key strength of our approach is its potential for cross-platform portability. The framework is readily applicable to any NVIDIA GPU with DCGM metrics and is expected to be straightforward to adapt to AMD (via rocm-smi [50]) and Intel (via oneAPI/intel-gpu-tools [51]) GPUs, with the same data collection and model training process. Our analysis focused on the most frequently used applications in month-long production jobs and has not yet been validated on emerging AI workloads. AI jobs can show highly dynamic power behavior, including large fluctuations during training [52], and recent studies have started exploring GPU power capping for modern AI and LLM workloads [53]. Evaluating cross-vendor platform applicability, especially on production systems dominated by AI workloads, is a primary direction for future work.

VII. Conclusion

As HPC systems increasingly rely on GPUs to meet growing computational demands, power availability has emerged as a primary operational constraint. To maximize throughput under strict power budgets, modern platforms often operate in hardware-overprovisioned regimes, increasing the risk of power excursions and underscoring the need for effective dynamic power management (DPM).

In this work, we present an ML-based GPU power prediction framework designed for production HPC workloads. By capturing fine-grained temporal dynamics, the framework delivers short-term, per-GPU forecasts suitable for real-time DPM. Using a month-long production telemetry dataset for evaluation, we demonstrate that while generic models trained on application-agnostic DCGM telemetry achieve robust performance with an average RMSE of 50 W, application-specific models further reduce error by up to 48%. We also show that incorporating DCGM metrics beyond historical power consumption improves power prediction accuracy by up to 15.8% compared to the power-only models. The framework's practical utility is further highlighted by its negligible 3 µs inference latency and reliance on lightweight, system-wide telemetry, making it promising for production deployment. Future work will focus on integrating these predictive models into system management stacks, such as batch schedulers and node-level power managers. Ultimately, this framework offers a scalable path toward the sustainable operation of overprovisioned systems, reclaiming performance otherwise lost to rigid, static power caps.

VIII. Acknowledgements

We thank the anonymous reviewers for their constructive feedback, which significantly improved this paper. We also thank Chris Samuel and John Stile at NERSC for providing the essential data needed to estimate end-to-end latency for inference and power actuation during the review process. This work was supported by the Office of Advanced Scientific Computing Research within the U.S. Department of Energy (DOE) Office of Science under contract DE-AC02-05CH11231. This research used the resources of the National Energy Research Scientific Computing Center (NERSC), a Department of Energy Office of Science User Facility using NERSC award ERCAP0038818.

References

[1] A. Shehabi, S. J. Smith, A. Hubbard, A. Newkirk, N. Lei, M. A. Siddik, B. Holecek, J. G. Koomey, E. R. Masanet, and D. A. Sartor, "United states data center energy usage report," report, Lawrence Berkeley National Laboratory, Dec. 2024. Published December 19, 2024.

[2] T. Patki, D. K. Lowenthal, B. Rountree, M. Schulz, and B. R. de Supinski, "Exploring hardware overprovisioning in power-constrained, high performance computing," in *Proceedings of the 27th International ACM Conference on International Conference on Supercomputing*, ICS '13, (New York, NY, USA), p. 173–182, Association for Computing Machinery, 2013.

[3] M. Marquardt, J. Mäder, T. Schiffmann, C. Simmendinger, and T. Wilde, "PowerSched: A HPC System Power and Energy Management Framework," in *Cray User Group (CUG) 2023*, (Helsinki, Finland), May 2023. Available Online: https://cug.org/proceedings/cug2023_proceedings/includes/files/pap113s2-file1.pdf.

[4] High Performance Computing Center Stuttgart (HLRS), "Hpe apollo (hawk)." https://www.hlrs.de/solutions/systems/hpe-apollo-hawk, 2025. Accessed: 2025-12-18.

[5] C. Simmendinger, M. Marquardt, J. Mäder, and R. Schneider, "PowerSched - Managing Power Consumption in Overprovisioned Systems," 2024.

[6] F. Antici, A. Borghesi, J. Domke, and Z. Kiziltan, "Uopc: A user-based online framework to predict job power consumption in hpc systems," in *ISC High Performance 2025 Research Paper Proceedings (40th International Conference)*, pp. 1–12, 2025.

[7] K. Menear, J. Acosta, and D. Duplyakin, "Classification of hpc job power consumption using a rich feature set," in *Practice and Experience in Advanced Research Computing 2025: The Power of Collaboration*, PEARC '25, (New York, NY, USA), Association for Computing Machinery, 2025.

[8] Z. Zhao, E. Rrapaj, S. Bhalachandra, B. Austin, H. A. Nam, and N. Wright, "Power analysis of nersc production workloads," in *Proceedings of the SC '23 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '23, (New York, NY, USA), p. 1279–1287, Association for Computing Machinery, 2023.

[9] Z. Zhao, B. Austin, E. Rrapaj, and N. J. Wright, "Understanding vasp power profiles on nvidia a100 gpus," in *Proceedings of the SC '24 Workshops of the International Conference on High Performance Computing, Network, Storage, and Analysis*, SC-W '24, p. 1496–1505, IEEE Press, 2025.

[10] A. Hossain, A. Abdurahman, M. A. Islam, and K. Ahmed, "Power-aware scheduling for multi-center hpc electricity cost optimization," in *Job Scheduling Strategies for Parallel Processing: 28th International Workshop, JSSPP 2025, Milan, Italy, June 3–4, 2025, Revised Selected Papers*, (Berlin, Heidelberg), p. 1–20, Springer-Verlag, 2026.

[11] J. Orteu, M. Clascà, J. Labarta, E. Jennings, S. Andersson, and M. Garcia-Gasulla, "A framework and methodology for performance prediction of hpc workloads," in *Parallel Processing and Applied Mathematics* (R. Wyrzykowski, J. Dongarra, E. Deelman, and K. Karczewski, eds.), (Cham), pp. 114–127, Springer Nature Switzerland, 2025.

[12] "Perlmutter: an hpe ex system based on nvidia a100 gpus and amd milan cpus." https://docs.nersc.gov/systems/perlmutter/architecture/. Accessed: 2025-12-12.

[13] NVIDIA Corporation, *NVIDIA Data Center GPU Manager (DCGM) Documentation*, 2025. Accessed: 2025-10-20.

[14] B. Austin, "Perlmutter machine time breakdown by applications (2023)." https://portal.nersc.gov/project/m888/shared/perlmutter_machine_time_break_down_by_application_2024.pdf, 2023. Accessed: 2025-12-12.

[15] A. M. Karimi, N. S. Sattar, W. Shin, and F. Wang, "Profiling and modeling of power characteristics of leadership-scale hpc system workloads," *arXiv preprint arXiv:2402.00729*, Feb. 2024.

[16] G. Hautreux, N. Alaoui, and E. Malaboeuf, "Optimizing gpu frequency for sustainable hpc: Lessons learned from a year of production on adastra, an amd gpu supercomputer," in *Cray User Group (CUG) 2025*, 2025.

[17] M. Johnson-Groh, "Achieving more with less: Optimizing efficiency in supercomputing," *Flatiron Scientist Spotlight*, 2023.

[18] D. Zhao, S. Samsi, J. McDonald, B. Li, D. Bestor, M. Jones, D. Tiwari, and V. Gadepally, "Sustainable supercomputing for ai: Gpu power capping at hpc scale," in *Proceedings of the 2023 ACM Symposium on Cloud Computing*, SoCC '23, (New York, NY, USA), p. 588–596, Association for Computing Machinery, 2023.

[19] C. Li, A. M. Karimi, W. Shin, H. Qi, and F. Wang, "The challenge of disproportionate importance of temporal features in predicting hpc power consumption," in *2021 IEEE International Conference on Cluster Computing (CLUSTER)*, pp. 632–636, 2021.

[20] M. Cornelius, G. Cross, S. Shilpika, M. T. Dearing, and Z. Lan, "Extracting practical, actionable energy insights from supercomputer telemetry and logs," *arXiv preprint arXiv:2505.14796*, May 2025.

[21] X. Tian, X. Li, J. Zhang, Z. Zhao, C. Wang, X. Wang, and J. Wang, "An online incremental learning framework for hpc job power consumption prediction," in *Proceedings of the 2023 7th International Conference on High Performance Compilation, Computing and Communications*, HP3C '23, (New York, NY, USA), p. 176–183, Association for Computing Machinery, 2023.

[22] B. Dutta, V. Adhinarayanan, and W.-c. Feng, "Gpu power prediction via ensemble machine learning for dvfs space exploration," in *Proceedings of the 15th ACM International Conference on Computing Frontiers*, CF '18, (New York, NY, USA), p. 240–243, Association for Computing Machinery, 2018.

[23] G. Wu, J. L. Greathouse, A. Lyashevsky, N. Jayasena, and D. Chiou, "Gpgpu performance and power estimation using machine learning," in *2015 IEEE 21st International Symposium on High Performance Computer Architecture (HPCA)*, pp. 564–576, 2015.

[24] Q. Wang, L. Li, W. Luo, Y. Zhang, and B. Wang, "Dso: A gpu energy efficiency optimizer by fusing dynamic and static information," in *2024 IEEE/ACM 32nd International Symposium on Quality of Service (IWQoS)*, pp. 1–6, 2024.

[25] A. M. Agelastos, B. A. Allan, J. M. Brandt, P. Cassella, J. Enos, J. Fullop, A. C. Gentile, S. Monk, N. Naksinehaboon, J. B. Ogden, M. Rajan, M. Showerman, J. O. Stevenson, N. Taerat, and T. Tucker, "The lightweight distributed metric service: A scalable infrastructure for continuous monitoring of large scale computing systems and applications," in *SC '14: Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 154–165, 2014.

[26] Apache Kafka, *KafkaConsumer (Kafka Clients API) — subscribe()*, 2025. Documentation showing how the Kafka consumer subscribes to topics to receive messages.

[27] "Redfish specification — eventing and subscription," 2025. See clauses on eventing and subscription creation using HTTP POST to EventService subscription collections.

[28] "sacct – slurm workload manager." https://slurm.schedmd.com/sacct.html, 2025. Accessed: 2025-12-18.

[29] Jefferson Lab, "Chroma: A lattice qcd code." https://jeffersonlab.github.io/chroma/, 2024. Accessed: 2025-01-15.

[30] J. Hafner, "Ab-initio simulations of materials using vasp: Density-functional theory and beyond," *Journal of computational chemistry*, vol. 29, no. 13, pp. 2044–2078, 2008.

[31] VASP Developers, "Openacc gpu port of vasp." https://www.vasp.at/wiki/index.php/OpenACC_GPU_port_of_VASP#Hardware, n.d. Accessed: 2025-10-20.

[32] "Atlas software documentation." https://atlas-software.docs.cern.ch/. Accessed: 2025-12-12.

[33] "E3SM code." https://github.com/E3SM-Project/E3SM. Accessed: 2025-12-12.

[34] M. Taylor, P. Caldwell, L. Bertagna, C. Clevenger, A. Donahue, J. Foucar, O. Guba, B. Hillman, N. Keen, J. Krishna, M. Norman, S. Sreepathi, C. Terai, J. White, A. Salinger, R. McCoy, L. Leung, D. Bader, and D. Wu, "The simple cloud-resolving e3sm atmosphere model running on the frontier exascale system," in *Proceedings Of The International Conference For High Performance Computing, Networking, Storage And Analysis*, 2023.

[35] P. Giannozzi, O. Baseggio, P. Bonfà, D. Brunato, R. Car, I. Carnimeo, C. Cavazzoni, S. de Gironcoli, P. Delugas, F. Ferrari Ruffino, A. Ferretti, N. Marzari, I. Timrov, A. Urru, and S. Baroni, "Quantum espresso toward the exascale," *Journal of Chemical Physics*, vol. 152, no. 15, p. 154105, 2020.

[36] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[37] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 06, no. 02, pp. 107–116, 1998.

[38] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, pp. 1735–1780, 11 1997.

[39] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, *Dive into Deep Learning*. Cambridge University Press, 2023. https://D2L.ai.

[40] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with lstm," *Neural Computation*, vol. 12, pp. 2451–2471, 10 2000.

[41] A. Graves, A. Mohamed, and G. E. Hinton, "Speech recognition with deep recurrent neural networks," *CoRR*, vol. abs/1303.5778, 2013.

[42] K. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014.

[43] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *CoRR*, vol. abs/1706.03762, 2017.

[44] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[45] NVIDIA Corporation, *NVIDIA System Management Interface*. NVIDIA Corporation, 2023. Version 12.535.0.

[46] M. Verleysen and D. François, "The curse of dimensionality in data mining and time series prediction," in *Computational Intelligence and Bioinspired Systems* (J. Cabestany, A. Prieto, and F. Sandoval, eds.), (Berlin, Heidelberg), pp. 758–770, Springer Berlin Heidelberg, 2005.

[47] T. O. Kvalseth, "Cautionary note about r 2," *The American Statistician*, vol. 39, no. 4, pp. 279–285, 1985.

[48] T. Patki, B. Rountree, T. Wilde, A. Bartolini, S. Brink, E. Heiskanen, S. Idgunji, M. Maiterth, J. Rogers, E. Rrapaj, R. Schneider, W. Shin, K. Shoga, C. Simmendinger, N. J. Wright, and Z. Zhao, "A global perspective on supercomputer power provisioning: Case studies from united states and europe," in *Proceedings of the 39th ACM International Conference on Supercomputing*, ICS '25, (New York, NY, USA), p. 1034–1051, Association for Computing Machinery, 2025.

[49] Z. Zhao, N. Keen, O. Antepara, S. Williams, L. Bertagna, N. Mahfouz, J. B. White, III, L. Oliker, B. Austin, and N. J. Wright, "Toward energy-efficient HPC: Insights from power profiling a cloud-resolving earth system model," in *2026 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, IEEE, 2026.

[50] Advanced Micro Devices, Inc. (AMD), *ROCm System Management Interface (ROCm-SMI) Library*, 2025. ROCm Software Stack.

[51] Intel Corporation, *IGT GPU Tools*, 2025. Part of the Intel oneAPI software ecosystem.

[52] E. Choukse, B. Warrier, S. Heath, L. Belmont, A. Zhao, H. A. Khan, B. Harry, M. Kappel, R. J. Hewett, K. Datta, Y. Pei, C. Lichtenberger, J. Siegler, D. Lukofsky, Z. Kahn, G. Sahota, A. Sullivan, C. Frederick, H. Thai, R. Naughton, D. Jurnove, J. Harp, R. Carper, N. Mahalingam, S. Varkala, A. G. Kumbhare, S. Desai, V. Ramamurthy, P. Gottumukkala, G. Bhatia, K. Wildstone, L. Olariu, I. Incorvaia, A. Wetmore, P. Ram, M. Raghuraman, M. Ayna, M. Kendrick, R. Bianchini, A. Hurst, R. Zamani, X. Li, M. Petrov, G. Oden, R. Carmichael, T. Li, A. Gupta, P. Patel, N. Dattani, L. Marwong, R. Nertney, H. Kobayashi, J. Liott, M. Enev, D. Ramakrishnan, I. Buck, and J. Alben, "Power stabilization for AI training datacenters," *arXiv preprint arXiv:2508.14318*, Aug. 2025.

[53] P. Patel, E. Choukse, C. Zhang, I. n. Goiri, B. Warrier, N. Mahalingam, and R. Bianchini, "Characterizing power management opportunities for llms in the cloud," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*, ASPLOS '24, (New York, NY, USA), p. 207–222, Association for Computing Machinery, 2024.

IX. Appendix

A. Correlation Analysis of DCGM Metrics

In this section, we provide additional details for the feature selection and analysis conducted on 20 DCGM metrics for each of the application datasets. Table III outlines the selected features for each application according to the feature selection methodology described in Section V-F. Figure 13 shows the correlation matrices for each application, providing a detailed view of pairwise feature correlations computed using Pearson correlation.

TABLE III: Selected DCGM features per application after correlation-based filtering.

Feature	Chroma	VASP	E3SM	ATLAS	Espresso
fb_free					
fb_used	✓			✓	
fp16_active					
fp32_active				✓	✓
fp64_active	✓				
gpu_utilization	✓	✓	✓		✓
gr_engine_active		✓			✓
nvlink_rx_bytes					
nvlink_tx_bytes	✓				
pcie_rx_bytes	✓			✓	
pcie_tx_bytes	✓			✓	
sm_active					✓
sm_occupancy	✓	✓			
tensor_active				✓	
tensor_hmma_active		✓			✓
dram_active		✓	✓	✓	
gpu_temp	✓	✓	✓	✓	✓
mem_copy_utilization	✓	✓	✓	✓	✓
memory_temp	✓	✓	✓	✓	✓
power_usage	✓	✓	✓	✓	✓

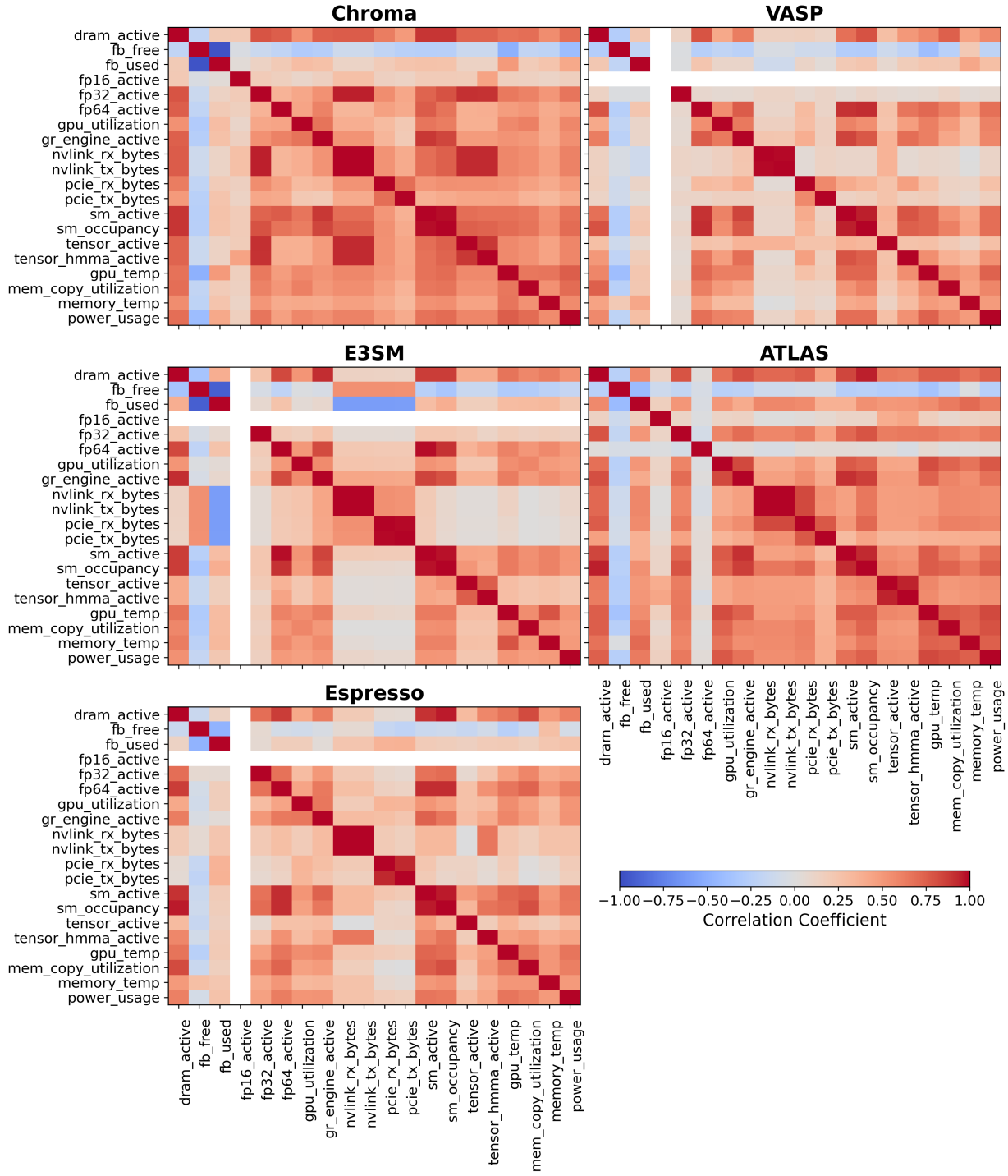


Fig. 13: Feature correlation matrices for each application.