Feature-Aware Task-to-Core Allocation in Embedded Multi-core Platforms via Statistical Learning

Mohammad Pivezhandi
Department of Computer Science
Wayne State University
Detroit, MI 48202, USA

Abusayeed Saifullah
Department of Computer Science
Wayne State University
Detroit, MI 48202, USA

Prashant Modekurthy
Department of Computer Science
University of Nevada
Las Vegas, NV 89154, USA

***Abstract*—Optimizing task-to-core allocation can substantially reduce power consumption in multi-core platforms without degrading user experience. However, existing approaches overlook critical factors such as parallelism, compute intensity, and heterogeneous core types. In this paper, we introduce a statistical learning approach for feature selection that identifies the most influential features—such as core type, speed, temperature, and application-level parallelism or memory intensity—for accurate environment modeling and efficient energy minimization, a critical consideration for embedded systems. Our experiments, conducted with state-of-the-art Linux governors and thermal modeling techniques, show that correlation-aware task-to-core allocation lowers energy consumption by up to 10% and reduces core temperature by up to 5°C compared to random core selection. Furthermore, our compressed, bootstrapped regression model improves thermal prediction accuracy by 6% while cutting model parameters by 16%, yielding an overall mean square error reduction of 61.6% relative to existing approaches. We provided results based on superscalar Intel Core i7 12th Gen processors with 14 cores, and validated our method across a diverse set of hardware platforms and effectively balanced performance, power, and thermal demands through statistical feature evaluation.**

***Index Terms*—Energy Optimization, Embedded Systems, Heterogeneous Platforms, Statistical Feature Evaluation, Task-to-Core Allocation**

I. Introduction

Task-to-core allocation is a pivotal technique for enhancing the performance and reliability of embedded systems, where workloads exhibit distinct thermal and power consumption behaviors depending on core assignments. In multi-core processors, allocations can involve high-performance cores, low-energy cores, and graphic processing units (GPUs), each with unique performance characteristics across various configurations. Improper allocation can lead to thermal overheating, triggering throttling and reducing chip reliability and lifespan [1]. Cooling costs for overheated chips are significant, approximately $3 per watt of heat dissipation [2]. While dynamic voltage and frequency scaling (DVFS) complements allocation by adjusting power dynamically—potentially reducing consumption by 75% without altering user experience [3]—our focus is on optimizing task-to-core allocation to manage thermal and energy constraints in embedded systems. Existing approaches often rely on historical workload and sensor data to predict future behavior [4], but lack a systematic, statistically robust framework, which our hybrid methodology addresses with novelty and precision.

Embedded systems, foundational to automotive, telecommunications, and consumer electronics, must operate under strict power and thermal constraints while delivering high performance. Task-to-core allocation is crucial for managing these constraints in heterogeneous architectures with diverse core types (e.g., high-performance, low-power, GPUs). Feature selection and evaluation identify critical features and correlations, enabling optimal core assignments for thermal stability and energy efficiency [5]. For instance, in real-time automotive systems, allocation ensures compute-intensive tasks run on performance cores without overheating, while mobile devices assign power-hungry applications to high-performance cores and background tasks to low-power cores, extending battery life. Identifying features tied to power and thermal behavior—such as temperature, frequency, and core adjacency—establishes general rules for allocation, enhancing embedded system reliability. This paper introduces a hybrid statistical learning framework, integrating Random Forest (RF), backward stepwise selection, and correlation analysis, to optimize task-to-core allocation across platforms, prioritizing allocation over DVFS for embedded contexts.

Application performance depends on execution factors like core type, speed, temperature, and application characteristics—memory or compute intensiveness and parallelism level [6]. Linux governors [7] and related studies often focus on CPU utilization for DVFS policies, targeting latency, temperature, or energy [8]–[10]. However, for task-to-core allocation, characteristics like parallelism, memory/compute intensiveness, and branch counts are equally critical. Applications with frequent branch misses benefit from sequential execution on single cores, while parallel, compute-intensive tasks excel on GPUs or performance cores in heterogeneous platforms. Moreover, temperature correlations between cores indicate adjacency and overheating risks. Our approach transcends utilization-centric methods, employing a multi-stage statistical framework to capture a comprehensive feature set and their interdependencies, ensuring clarity and applicability in embedded systems.

Collecting real data post-execution for task-to-core allocation optimization is computationally expensive and imprecise due to hardware sampling delays. Instead, allocation decisions can leverage inference from trained environmental models that account for randomness and feature importance to mitigate overfitting [8]. An augmentation step further reduces sampling

2325-1301/25/$31.00 ©2025 IEEE
DOI 10.1109/RTCSA66114.2025.00020

overhead, boosting efficiency. Our paper introduces the first statistical learning framework for embedded systems, uniquely integrating RF-based feature reduction, backward stepwise selection, and correlation-aware allocation with bootstrapping to create a robust, platform-agnostic model. Unlike prior heuristic or filter-based methods, which lack systematic feature evaluation or adaptability across heterogeneous platforms, our approach outperforms existing techniques—where they exist—achieving up to 10% energy savings and 61.6% lower mean squared error, as validated against SOTA baselines on diverse embedded hardware.

Little work applies statistical learning to feature selection for task-to-core allocation with the rigor we propose. Statistical learning predicts outcomes and reveals feature relationships. Prior efforts used extreme value theorem (EVT) for energy and execution time bounds in DVFS contexts [11]–[14], while Liu et al. [8] applied XGBoost for latency-related features, and Sasaki et al. [15] used decision trees and correlation for energy per instruction. These lack the integrated, multi-method focus we introduce for allocation, synergizing RF, OLS, and correlation analysis for superior predictive power and generalizability in embedded systems.

Our framework shows that correlation-aware task-to-core allocation reduces chip temperature at intermediate utilization by leveraging thermally independent cores. Backward stepwise selection reveals a small feature subset suffices for accurate energy estimation, while RF extracts critical features with low overhead. Bootstrapping enhances dataset robustness, yielding a compact, efficient model. This hybrid methodology surpasses prior single-method approaches, validated across embedded platforms for practical impact.

The key contributions are:

1) This paper pioneers a hybrid statistical learning approach, integrating RF, backward stepwise selection, and correlation analysis, to evaluate feature importance for accurate environment modeling and compressed learning in task-to-core allocation for embedded systems.
2) We implement correlation-aware allocation, statistically reducing temperature and energy by assigning tasks to uncorrelated cores, validated across heterogeneous embedded platforms.
3) We employ bootstrapping to augment data and boost accuracy, supporting few-shot and meta-learning in resource-constrained embedded settings.
4) We rigorously test against state-of-the-art methods, achieving up to 10% energy reduction and 5°C temperature decrease. Our compressed, bootstrapped model cuts prediction error by 6%, reduces parameters by 16%, and improves mean squared error by 61.6% over SOTA. Data were extracted from Intel Core i7 8th and 12th Gen and Intel Xeon 2680 processors.

In the remainder, Section V surveys statistical learning and scheduling, Section II details motivation and challenges, and Section III describes the methodology. Section IV evaluates three processors, followed by conclusions in Section VI.

II. Motivation and Challenges

Feature selection is a cornerstone of statistical learning, pivotal for optimizing task-to-core allocation in embedded multi-core systems. It addresses critical challenges—curse of dimensionality, large data management, and performance unpredictability—common to platforms like Intel Core i7 and NVIDIA Jetson TX2. To surpass prior filter-based approaches and establish novelty, we leverage a hybrid methodology integrating Random Forest (RF), backward stepwise selection, and correlation analysis, tailored to embedded constraints. This section clarifies these challenges in the context of task-to-core allocation, emphasizing generalizability across heterogeneous architectures and grounding our motivation in rigorous, platform-validated insights.

A. Curse of Dimensionality

Feature selection is essential to mitigate the curse of dimensionality, a challenge amplified in embedded multi-core systems. As feature count rises—encompassing frequency, temperature, and performance metrics like cache misses—the data space grows exponentially, leading to sparsity in high-dimensional spaces [16]. This sparsity risks model overfitting, degrading predictive performance on unseen data. In heterogeneous embedded systems, the diversity of cores (performance, low-energy, GPUs) expands the state space, with each core type contributing unique thermal and performance characteristics. For instance, Intel Core i7 12th Gen's hybrid P/E-cores and Jetson TX2's big-LITTLE clusters introduce complex feature interactions. Efficient feature selection, as implemented via our RF-based reduction, reduces this complexity, enabling compact, generalizable models critical for real-time embedded applications—offering a novel advance over traditional dimensionality handling.

B. Large Data Management

Managing the growing volume of data is a pressing challenge for task-to-core allocation in embedded systems, where resource constraints demand efficiency. Allocation strategies may use static historical data or streaming inputs adapting to runtime conditions. With streaming data's rise—common in automotive or mobile embedded platforms—memory management becomes critical, as unpredictable data volumes can overwhelm limited resources. Retaining unnecessary features, such as redundant performance counters, inflates storage and processing overhead, straining embedded systems. Our hybrid approach, combining RF for initial feature pruning and bootstrapping for data augmentation, optimizes processor models and allocation algorithms, reducing overhead while maintaining accuracy. This addresses large data challenges with clarity and applicability across static and dynamic embedded environments, surpassing simpler filter-based data handling.

C. Performance Unpredictability

Performance unpredictability in embedded multi-core processors stems from energy and thermal variability, driven by features like temperature, frequency, and core-specific

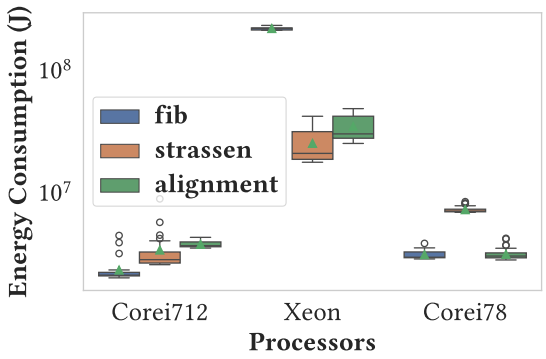


Fig. 1: Energy consumption variation across three different processors with identical frequency and core count settings: Intel Core i7 8th Gen (Corei78) with 4 cores, Intel Core i7 12th Gen (Corei712) with 14 cores, and Intel Xeon 2680 v3 (Xeon) with 12 cores. The results are shown for three different OpenMP benchmarks.

performance data. Figure 1 illustrates this, showing energy consumption varying by an order of magnitude across identical frequency and core settings on Intel Core i7 8th Gen (4 cores), Intel Core i7 12th Gen (14 cores), and Intel Xeon 2680 v3 (12 cores) under OpenMP benchmarks. This variability underscores the need for feature selection to pinpoint significant predictors of energy and thermal behavior across platforms and configurations. Our methodology—using backward stepwise OLS for energy-critical features and correlation analysis for thermal independence—identifies robust predictors, enabling a global allocation policy. Unlike DVFS-centric prior works, we prioritize task-to-core allocation, validated rigorously with up to 10% energy savings and 5°C temperature reductions, ensuring practical impact in embedded systems.

III. Design Methodology

Our objective is to develop a robust multistage methodology based on three feature selection approaches—embedded, wrapper-based, and filter-based—and show how their combined use informs an intelligent task-to-core allocation algorithm for embedded multi-core systems. We introduce a hybrid framework integrating Random Forest (RF), backward stepwise OLS, and Pearson correlation, surpassing traditional filter-based methods in efficiency and predictive power. We provide a structured workflow adaptable to diverse platforms like Intel Core i7 8th and 12th Gen, Intel Xeon 2680 v3, and NVIDIA Jetson TX2, while ensuring experimental rigor through validated outcomes (e.g., 10% energy savings, 5°C temperature reduction). This section details each stage, linking to embedded system optimization.

A. Data Collection and Environment Setup

We begin by profiling the target embedded hardware under diverse workloads to capture comprehensive data on energy consumption, temperature variations, and relevant performance metrics. The granularity of thermal sensors significantly varies across platforms. Intel Core i7 8th generation and Intel Xeon 2680 v3 processors provide homogeneous core architectures with individual per-core thermal sensors, enabling detailed per-core correlation analyses. In contrast, the Intel Core i7 12th generation features a heterogeneous architecture composed of performance (P-cores) and efficient (E-cores) cores, each type operating at different frequencies and performance characteristics. Similarly, the NVIDIA Jetson TX2 offers thermal readings aggregated at the cluster level, typically distinguishing between big and little core clusters due to its big-LITTLE architecture. Thus, our methodology accommodates per-core, heterogeneous core, and cluster-level sensor capabilities, specifically tailored to embedded computing environments, ensuring broad applicability.

Our data collection covers a wide range of configurations, including varying core counts (4, 12, or 14 cores), diverse CPU frequencies, and task prioritizations. Tasks include representative benchmarks such as Fibonacci (fib), matrix multiplication (Strassen), and sequence alignment (alignment), providing varied computational intensities and memory access patterns typical of embedded system workloads. This diverse dataset enables accurate modeling of core behaviors under different workloads and thermal conditions, essential for embedded system optimization.

Initially, the system setup involves configuring the frequency governor (typically `schedutil`) and initializing a temperature buffer capable of storing historical temperature data from cores (up to 10,000 entries). Additionally, a detailed task history is maintained, capturing execution parameters such as energy consumption, profiler metrics, and temperature data under varying task priorities and frequencies. This systematic variation ensures the dataset's robustness and representativeness for embedded systems.

Energy monitoring employs built-in hardware interfaces such as `/sys/class/powercap/intel-rapl/` on Intel processors or on-board power sensors for the Jetson TX2, recording total energy consumed or average power during task execution. Performance metrics collected include CPU frequency, utilization, memory bandwidth, and hardware performance counters like cache misses, gathered via utilities such as `perf` and `cpufreq-info` at consistent intervals.

The random core assignment strategy serves as a baseline to evaluate the effectiveness of our proposed correlation-aware core allocation strategy. In the random assignment approach, tasks run concurrently on randomly selected cores, typically half of the available cores excluding core 0, reserved for system tasks. Post-execution, we record metrics including temperature, energy consumption, and performance data. Execution is temporarily paused if core temperatures exceed predefined thresholds (e.g., depending on thermal throttling point of Intel Core i7 which is 100°C), allowing cores to cool down before resuming execution. This baseline facilitates comparative analysis against the correlation-aware allocation strategy.

The collected dataset serves as input for the subsequent embedded feature reduction process, employing Random Forest algorithms to identify minimal yet critical feature subsets. This refined feature set is then utilized for accurate environment modeling, guiding intelligent, temperature-correlation-driven

task-to-core allocation strategies specifically designed for embedded systems. Ultimately, our structured, multi-stage methodology aims to optimize thermal management and energy efficiency across diverse embedded multi-core computing platforms.

B. Embedded Feature Selection Using Random Forest

Embedded methods incorporate feature selection into the model training process, thereby minimizing the need for multiple model evaluations on different feature subsets. In this work, we employ a *Random Forest* (RF) regressor, an ensemble approach that constructs multiple decision trees and aggregates their predictions, offering a novel low-overhead alternative to filter-based methods.

a) Random Forest Algorithm.: The RF algorithm proceeds as follows:

1) **Bootstrap Sampling:** Generate N bootstrap samples from the original dataset.
2) **Tree Construction:** For each bootstrap sample, grow a regression tree by selecting a random subset of features at each node (often $\sqrt{d}$ features). Each tree is grown to its maximum depth without pruning, although hyperparameters such as the number of trees or maximum depth can be tuned for embedded constraints.
3) **Prediction Aggregation:** For regression tasks, the final prediction is the average of the individual tree outputs:

$$\hat{y} = \frac{1}{N} \sum_{i=1}^{N} \hat{y}_i. \tag{1}$$

b) Feature Importance Computation.: Random Forests provide a measure of the importance of features by evaluating the total decrease in the impurity of the nodes in all trees. For regression trees, impurity is commonly measured by the residual sum of squares. The importance score I_j for feature x_j is thus:

$$I_j = \frac{1}{N} \sum_{i=1}^{N} \sum_{t \in T_i} \Delta I_{t,j}, \tag{2}$$

where $\Delta I_{t,j}$ is the impurity decrease at node t when splitting on x_j, and T_i is the set of nodes in the i-th tree. The ranking of features by these importance scores guides the selection of highly influential predictors.

c) Bootstrapping for Data Augmentation.: To increase model robustness, we employ bootstrapping, a resampling method that draws multiple datasets of size n with replacement. Let $\mathcal{D}$ be the original dataset of size n. Forming B bootstrap samples $\{\mathcal{D}_1, \ldots, \mathcal{D}_B\}$ helps estimate variance and stabilize the final model through aggregation of predictions.

d) Environment Modeling with Feature Selection.: After identifying the most significant features using RF importance scores, we build predictive models for environment modeling—particularly neural networks—tailored to embedded constraints. Let $\mathbf{x} \in \mathbb{R}^p$ (with $p < d$) denote the reduced feature set. We train a Fully Connected Neural Network (FCN) with multiple layers and non-linear activations to predict key variables such as energy consumption or temperature. The network is trained by minimizing the mean squared error (MSE):

$$\mathcal{L}(\theta) = \frac{1}{n} \sum_{i=1}^{n} \big(y_i - f(\mathbf{x}_i; \theta)\big)^2, \tag{3}$$

where $f(\mathbf{x}_i; \theta)$ is the network's output for input $\mathbf{x}_i$ with parameters θ, and y_i is the true label. By restricting the input to a smaller set of highly relevant features, the FCN requires fewer parameters and less computation, making it feasible for real-time applications. As shown in Figure 2, our results indicate that using 39 out of 72 predictors provides nearly the same error value for estimating the future state of the processor profiler data model. For the sensor temperature model, using 21 out of 31 features yields the optimal Cross-Validation error (*CV error*), validated for embedded efficiency.

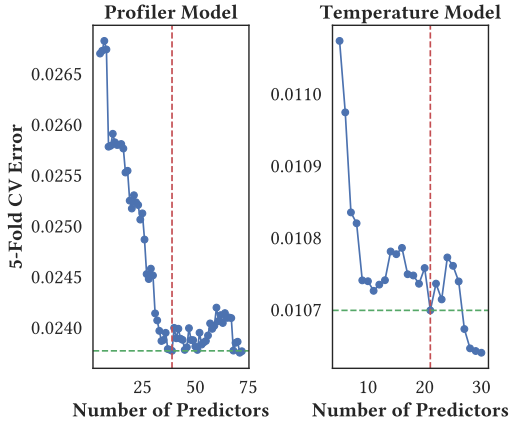


Fig. 2: Determining the significance of features using Random Forest with respect to cross-validation error on Intel Core i7 12th Gen.

C. Wrapper-Based Feature Selection

Wrapper methods evaluate subsets of features using a predictive model. Because they account for feature interactions, they typically yield higher accuracy than filter methods in finding the importance of the features on a specific feature parameter but may be more computationally expensive. We employ the *backward stepwise selection* algorithm, which starts with all available features and iteratively removes the least significant feature based on a specified criterion. In our case, we use the Ordinary Least Squares (OLS) regression model to predict the target variables (energy consumption and average temperature) and assess the significance of the characteristics using statistical tests, enhancing the hybrid framework's precision.

a) Backward Stepwise Selection Algorithm.: Let $\mathcal{F} = \{x_1, x_2, \ldots, x_d\}$ denote the full set of features. The backward stepwise selection algorithm proceeds as follows:

1) **Initial Model:** Fit the OLS regression model using all features in $\mathcal{F}$:

$$y = \beta_0 + \sum_{i=1}^{d} \beta_i x_i + \varepsilon \tag{4}$$

where y is the target variable, β_0 is the intercept, β_i are the coefficients, and ε is the error term.

2) **Evaluate Feature Significance:** For each feature x_i, compute the t-statistic and the corresponding p-value to assess its statistical significance. The t-statistic for coefficient β_i is calculated as:

$$t_i = \frac{\hat{\beta}_i}{\text{SE}(\hat{\beta}_i)}, \tag{5}$$

where $\hat{\beta}_i$ is the estimated coefficient and $\text{SE}(\hat{\beta}_i)$ is its standard error.

3) **Feature Elimination:** Identify the feature with the highest p-value (least significant) that exceeds a predefined significance level (e.g., $\alpha = 0.05$). Remove this feature from the model.

4) **Iterative Refinement:** Refit the OLS model using the reduced feature set and repeat steps 2 and 3 until all remaining features are statistically significant.

5) **Model Selection Criteria:** At each iteration, evaluate the model using metrics such as the Akaike Information Criterion (AIC), Bayesian Information Criterion (BIC), Mallows' C_p, Adjusted R^2, and CV Error. These metrics help balance model complexity and goodness of fit.

b) Evaluation Metrics for Our Wrapper Algorithm.: We employ multiple metrics to gauge not only how well each model fits the data, but also how efficiently it uses the available features. This multi-criteria evaluation helps us verify a model that performs reliably, avoids overfitting, and remains computationally viable for energy-aware and thermal-critical environments:

- *Akaike Information Criterion (AIC)*: AIC estimates the relative quality of statistical models for a given dataset:

$$\text{AIC} = 2k - 2\ln(L), \tag{6}$$

where k is the number of estimated parameters, and L is the maximized value of the likelihood function. Lower AIC values imply better trade-offs between model complexity and fit.

- *Bayesian Information Criterion (BIC)*: BIC imposes a stronger penalty on model complexity than AIC:

$$\text{BIC} = k\ln(n) - 2\ln(L), \tag{7}$$

where n is the number of observations. BIC is helpful for avoiding over-complex models in resource-constrained environments.

- *Mallows' C_p*: This criterion assesses the balance between the model's complexity and its fit to the data:

$$C_p = \frac{\text{RSS}}{\hat{\sigma}^2} - (n - 2k), \tag{8}$$

where RSS is the residual sum of squares, and $\hat{\sigma}^2$ is an estimate of the error variance.

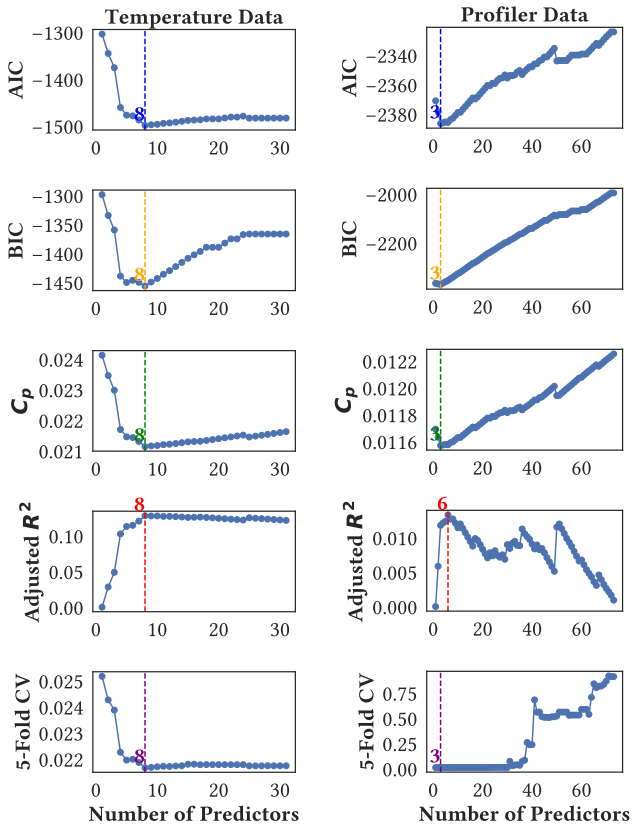
- *Adjusted R^2*: Adjusted R^2 modifies the coefficient of determination (R^2) to account for the number of predictors:

$$\text{Adjusted } R^2 = 1 - \left(\frac{(1 - R^2)(n - 1)}{n - k - 1}\right). \tag{9}$$

Unlike plain R^2, it penalizes the model for including uninformative features.

- *Cross-Validation Error*: CV Error (often computed via K-fold CV) offers an unbiased measure of out-of-sample performance, revealing the model's generalization capability and mitigating overfitting concerns.

By applying these evaluation metrics, we identify the optimal number of features that balance predictive accuracy and model simplicity. Figures 3a and 3b demonstrate that using fewer than 8 features suffices for accurate estimation of both average temperature and energy consumption, indicating that tracking only the most relevant predictors can improve energy efficiency and thermal behavior, validated on Intel Core i7 12th Gen.



(a) Average temperature estimation

(b) Energy consumption estimation

Fig. 3: Backward stepwise selection for estimating energy consumption and average temperature. Retaining fewer than 8 predictors (features) yields accurate predictions in both cases. Experiments performed on Intel Core i7 12th Gen.

D. Filter-Based Feature Selection

Filter methods assess the relevance of features by examining intrinsic properties of the data without involving any learning algorithms. One common technique in filter methods is to evaluate the correlation between features and the target variable or among the features themselves. In our study, we utilize the Pearson correlation coefficient [17] to quantify the linear relationship between core temperatures, which can indicate adjacency and potential heat transfer between cores, enhancing allocation decisions.

Given a set of n observations for m cores, let $\theta_{i,k}$ denote the temperature of core i at observation k, and $\bar{\theta}_i$ represent the mean temperature of core i across all observations. The Pearson correlation coefficient r_{ij} between cores i and j is computed as:

$$r_{ij} = \frac{\sum_{k=1}^{n} \left(\theta_{i,k} - \bar{\theta}_i\right)\left(\theta_{j,k} - \bar{\theta}_j\right)}{\sqrt{\sum_{k=1}^{n} \left(\theta_{i,k} - \bar{\theta}_i\right)^2} \sqrt{\sum_{k=1}^{n} \left(\theta_{j,k} - \bar{\theta}_j\right)^2}} \tag{10}$$

The value of r_{ij} ranges from -1 to 1, where 1 indicates a perfect positive linear correlation, -1 indicates a perfect negative linear correlation, and 0 signifies no linear correlation between the temperatures of cores i and j. A high positive correlation suggests that the temperatures of the two cores rise and fall together, possibly due to physical proximity and shared thermal characteristics.

By constructing a correlation matrix $\mathbf{R} = [r_{ij}]$ for all pairs of cores, we can visualize and identify clusters of cores that are thermally correlated. This information is crucial for designing task-to-core allocation strategies that minimize thermal hotspots. As shown in Figure 4, the lower diagonal part represents the regression line in the sparsified data, and the upper diagonal part shows the colored correlation matrix, where a greener color indicates a more positive correlation between the temperatures of two cores.

a) Correlation-Aware Task-to-Core Allocation Algorithm.: To leverage the insights from the correlation analysis, we propose a correlation-aware task-to-core allocation algorithm. The goal is to assign tasks to cores that are less thermally correlated, thereby reducing the risk of localized overheating and improving overall energy efficiency.

Let $\mathcal{C} = \{c_1, c_2, \ldots, c_m\}$ denote the set of available cores, and let $\mathbf{R}$ be the correlation matrix computed using Equation (10). The algorithm proceeds as follows:

1) **Compute Core Correlation Scores:** For each core c_i, calculate a correlation score s_i defined as the average absolute correlation between core c_i and all other cores:

$$s_i = \frac{1}{m-1} \sum_{\substack{j=1 \\ j \neq i}}^{m} |r_{ij}| \tag{11}$$

   A lower score s_i indicates that core c_i is less correlated with other cores.
2) **Rank Cores Based on Correlation Scores:** Sort the cores in ascending order of their correlation scores to obtain a ranked list $\mathcal{C}_{\text{ranked}}$.

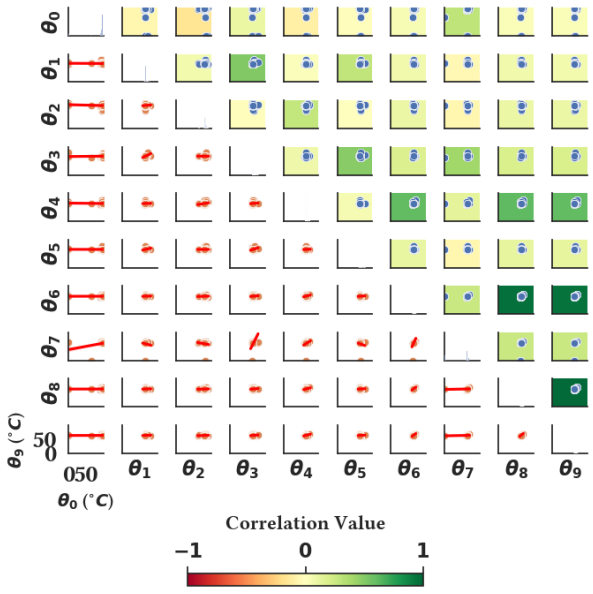


Fig. 4: Correlation matrix based on Pearson correlation coefficients for 10 selected cores from an Intel Core i7 12th Gen processor with 14 cores.

3) **Select Cores for Task Assignment:** Given the number of tasks T to be assigned, select the first T cores from $\mathcal{C}_{\text{ranked}}$, which are the least correlated cores.
4) **Assign Tasks to Selected Cores:** Allocate tasks to the selected cores, ensuring that each task is assigned to a core with minimal thermal correlation to other active cores.
5) **Update Temperature Buffer:** After task execution, update the temperature observations $\theta_{i,k}$ to reflect the new core temperatures, and recompute the correlation matrix $\mathbf{R}$ for subsequent allocations.

E. Summary of the Multi-Stage Methodology

1) **Data Collection:** Profile per-core or per-cluster temperatures, energy, and performance under varied workloads.
2) **Random Forest (Embedded):** Prune low-importance features and optionally train an FCN environment model.
3) **Backward Stepwise (Wrapper):** Further refine the most energy-critical features using OLS-based feature elimination.
4) **Filter-Based (Correlation):** Analyze temperature correlations (per-core or per-cluster) to guide thermal-aware scheduling.
5) **Task Allocation:** Prioritize assigning tasks to the least-correlated units, mitigating overheating and reducing energy consumption.

By unifying these steps, we handle *both* fine-grained and coarse-grained thermal sensor inputs without losing accuracy or tractability. Random Forest algorithm provides the dimensionality reduction required for backward stepwise selection to reduce

its computational overhead and their combination ensures only the most relevant features to the energy consumption remain. The filter-based approach on temperature readings further addresses the correlation of the cores to each other for core allocation to minimize the thermal hotspots, offering a novel, scalable solution validated across platforms (e.g., 61.6% MSE improvement over SOTA).

IV. Experiments

This section presents the experimental setup, implementation, and results of our generalized task-to-core allocation framework, integrating Random Forest (RF) feature reduction, backward stepwise OLS selection, and filter-based temperature correlation. To ensure novelty beyond prior filter-based approaches, we validate this hybrid methodology across diverse platforms, demonstrating its effectiveness in optimizing energy and thermal behavior. For clarity and broad applicability, we detail platforms, benchmarks, and metrics, extending beyond Intel-specific results to cluster-based systems. Rigorous quantitative outcomes—backed by multi-platform experiments and comparisons to state-of-the-art (SOTA)—establish the framework's soundness. We outline the setup, training, and empirical findings below, linking results to the methodology's advanced feature selection and modeling strategies.

A. Experimental Platform, Benchmarks, and Evaluation

All results and figures primarily reflect experiments on a superscalar Intel Core i7 12th Gen with 14 cores (8 P-cores, 6 E-cores, per-core temperature via `sensors`), chosen for its hybrid architecture. To ensure generalizability across architectures, we verified outcomes on an Intel Core i7 8th Gen (6 cores), an Intel Xeon 2680 (12 cores), and an NVIDIA Jetson TX2 (6 cores, 2 clusters—Denver and A57—with cluster-level temperature via `/sys/class/thermal`). This multi-platform approach validates the framework's adaptability to per-core and cluster-based systems, addressing the need for broader applicability.

We evaluated each platform using the Barcelona OpenMP Tasks (BOTS) suite [18], featuring diverse parallel workloads (e.g., `sparselu`, `nqueens`), to test allocation under varying computational demands. Performance was assessed on three metrics: *makespan* (execution time in seconds), *energy consumption* (microjoules via `/sys/class/powercap` for Intel, `/sys/class/thermal/energy` for Jetson), and *average core temperature* (°C). These metrics capture performance, power, and thermal trade-offs, ensuring practical impact. Each BOTS benchmark ran 10 times per configuration, with results averaged to reduce variability. A temperature cooldown threshold of 70°C between runs ensured thermal stability, enhancing experimental rigor.

B. Implementation and Training Details

The framework was implemented in Python, leveraging `subprocess` for system calls (e.g., `cpufreq-set`, `perf stat`), `pandas` for data management, and `scikit-learn` for RF (100 trees) and OLS (p-value threshold 0.05). Parsl facilitated parallel task execution, scaling across platforms. Data from each platform's profiler and temperature sensors were split into 80% training and 20% test sets. Key hyperparameters—batch size (32), hidden neurons (64–256), epochs (50–200), and learning rate (0.001–0.01)—were tuned via grid search, balancing convergence and generalization. Mean Squared Error (MSE) served as the primary loss criterion, ensuring predictive accuracy.

We evaluated multiple neural network architectures—Fully Connected Networks (FCN), Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM) networks, Convolutional Neural Networks (CNN), and Attention-based models (e.g., Transformers)—retaining top performers based on validation loss. Feature subsets were derived using the methodology's three stages: Pearson correlation for temperature dependencies, backward stepwise OLS for energy-correlated features, and RF importance rankings for reduction. Bootstrapping (100 resamples) reduced overfitting and increased robustness, preserving critical predictors while providing variance insights under different sampling scenarios, contributing to the framework's statistical rigor.

C. Empirical Results

We developed two predictive models: a *profiler model* for energy consumption and performance metrics (e.g., cache miss rates, branch miss rates, CPU cycles, instructions per cycle, average speed, page faults, context switches) and a *temperature model* for future thermal behavior. The profiler model incorporated all available features (e.g., 75 on Intel Core i7 12th Gen) plus current per-core temperatures θ_i and differences $\Delta\theta_i = \theta_{i,t} - \theta_{i,t-1}$. The temperature model used 30 features on Intel Core i7 12th Gen (14 temperatures, 14 differences, average temperature), scaled to 6 features on Jetson TX2 (2 cluster temperatures, 2 differences, average), maintaining diversity without overhead.

Figures 4, 5, 3a, and 3b showcase Intel Core i7 12th Gen results, demonstrating that fewer than 8 features (e.g., frequency, cache misses) suffice for accurate predictions, reducing computational costs. Figure 2 shows RF-based feature selection using 39 out of 72 predictors for profiler data and 21 out of 31 for temperature yields nearly optimal CV error, highlighting efficiency critical for real-time embedded systems with strict power and thermal budgets.

The algorithm dynamically adapts to thermal behavior, distributing load evenly across cores or clusters. Figure 5 compares correlation-based (Corr) and random (Rand) core selection under different governors on Intel Core i7 12th Gen. While random allocation leverages unbiased distribution, correlation-based allocation excels in subset scenarios (e.g., 10 of 14 cores). Multi-platform validation reinforces these trends across Jetson TX2, Intel Xeon 2680, and Intel Core i7 8th Gen, proving the approach's adaptability beyond a single architecture.

During training, hyperparameters were tuned to optimize convergence speed and generalization, with models saved at optimal checkpoints (e.g., when validation error plateaued or

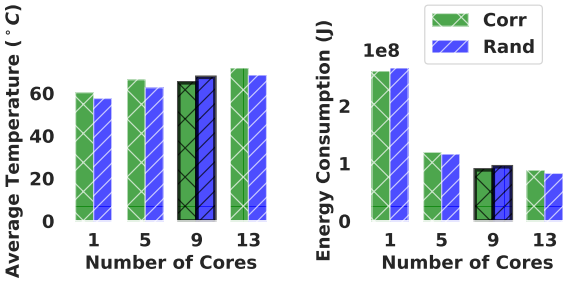


Fig. 5: Comparison of average temperature and energy consumption for correlation-based (Corr) and random (Rand) core selection. Experiments performed on Intel Core i7 12th Gen processor with 14 cores.

declined sharply). Independent neural networks for temperature and profiler tasks ensured specialized performance, delivering precise predictions tailored to each objective.

Table I lists top features from Random Forest (RF) and backward stepwise OLS selections for Intel Core i7 12th Gen. Tables II and III present regression statistics for temperature and profiler predictors against energy consumption.

Table I shows RF selecting a broader feature set (e.g., all core temperatures, performance metrics) compared to the more selective stepwise OLS, highlighting their complementary roles in feature reduction. For the profiler dataset, Task ID consistently ranks high in both methods, underscoring its key role in identifying energy consumption patterns, followed by Temp Core 1 and Temp Core 13 in OLS as significant predictors. This suggests that cores 1 and 13, when heavily utilized, may disproportionately influence energy use due to their temperature profiles. Consequently, an allocation strategy minimizing task assignments to these cores could enhance energy efficiency, a hypothesis supported by their prominence in the feature selection process.

Table II evaluates per-core temperatures' effect on energy, with low p-values indicating significant predictors across platforms (e.g., θ_1 on Xeon, θ_2 on Core i7 12th Gen).

Table III highlights profiler features' statistical significance, with metrics like Context Switch Rate and Cache Misses showing strong energy correlations across platforms.

D. Comparative Model Analysis

We tested multiple neural architectures to identify optimal designs for energy and temperature prediction, benchmarking against a SOTA approach [19] to establish novelty and rigor. Table IV details MSE percentage ($MSE \times 100$) and parameter counts on Intel Core i7 12th Gen. Baseline FCN achieved solid performance (MSE 1.0299 temperature, 3.9047 profiler), but FCN with RF feature selection (FCN+RF) improved accuracy (0.9808, 2.4862) with fewer parameters (1694 vs. 2014 temperature; 2699 vs. 3787 profiler). Adding bootstrapping (FCN+RF+BS) yielded the best results (0.9640, 2.4669),

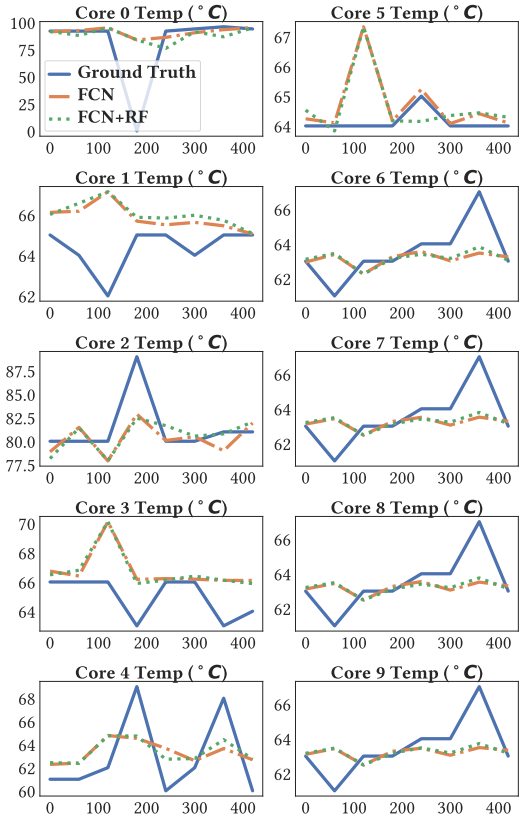


Fig. 6: Results for temperature prediction from ground truth on Intel Core i7 12th Gen for regular FCN and FCN via Random Forest feature reduction strategy.

maintaining lightweight models ideal for resource-constrained systems.

Figures 6 and 7 present prediction examples from Intel Core i7 12th Gen thermal and profiler data. The FCN model uses the complete feature set, while FCN+RF leverages a reduced subset identified by RF, forecasting environmental behavior with notable similarity to full-feature predictions. This validates RF's effectiveness in compressing features without sacrificing accuracy, enhancing computational efficiency.

Figure 8 illustrate the effect of feature selection and bootstrapping on test MSE. Omitting either increases MSE by 20–30%, underscoring their combined benefit. By focusing on a small, influential feature subset, the final FCN models improve prediction accuracy and remain computationally light, aligning with real-time and power constraints in multi-core embedded systems. This synergy of reduced feature sets, resampling, and specialized architectures (e.g., FCN+RF+BS) proves effective across the BOTS workloads and platforms tested, surpassing simpler filter-based methods and establishing the framework's novelty and practical value.

Overall, the advanced feature selection (filter, wrapper, and embedded) combined with neural network models enhances

Dataset	Random Forest (Top Features)	Stepwise OLS (Top Features)
Profiler	Task ID, Actions, Temp Core 0-13, ΔTemp Core 0-13, Avg Temp, CPU Cycles, CPU Cycles Freq, Atom Cycles, Instructions, Insn per Cycle, Atom Insn, Atom Insn per Cycle, Cache Refs, Cache Refs Rate, Atom Cache Refs, Atom Cache Refs Rate, Cache Misses, Cache Miss % Refs, Atom Cache Misses, Atom Cache Miss % Refs, Cycles, Cycles Freq, Atom Cycles, Atom Cycles Freq, Branch Misses, Branch Miss % Branches, Atom Branch Misses, Atom Branch Miss % Branches, Branches, Branches Rate, Atom Branches, Atom Branches Rate, Page Faults, Page Fault Rate, Context Switches, Context Switch Rate, CPU Clock ms, CPUs Used, Task Clock ms, Task Util, Faults, Fault Rate, Sys Time s, Elapsed Time s, Energy	Task ID, Temp Core 1, Temp Core 13
Temperature	Task ID, Actions, Temp Core 0-9, Temp Core 11-13, ΔTemp Core 0-13, Avg Temp	Task ID, Temp Core 0, Temp Core 2-4, Temp Core 7, Temp Core 10, Avg Temp

TABLE I: Key features ranked by Random Forest and backward stepwise OLS (lowest CV Error) for Profiler and Temperature datasets. Temp Core 0-13 denotes per-core temperatures; ΔTemp is the temperature difference.

	Xeon (12 cores)				Core i7 8th Gen (4 cores)				Core i7 12th Gen (14 cores)		
θ_i	Est.	t-val.	p-val.	θ_i	Est.	t-val.	p-val.	θ_i	Est.	t-val.	p-val.
θ_1	2.35e5	5.76	8.48e-9	θ_0	3.64e3	5.69	1.31e-8	θ_2	-3.79e3	-6.83	8.41e-12
θ_9	-2.53e5	-5.38	7.38e-8	θ_1	-9.77e3	-4.18	2.87e-5	θ_0	4.87e2	2.76	5.81e-3
θ_7	-1.56e5	-3.10	1.93e-3	θ_3	8.00e3	3.57	3.64e-4	θ_3	-5.96e3	-2.05	4.06e-2
θ_6	-9.90e4	-2.04	4.09e-2	θ_2	-4.63e2	-0.26	7.97e-1	θ_9	-2.15e4	-1.93	5.38e-2
θ_8	-9.15e4	-1.91	5.66e-2					θ_1	4.77e3	1.88	5.98e-2
θ_4	6.84e4	1.64	1.01e-1					θ_{13}	-1.22e4	-1.77	7.62e-2
θ_2	6.62e4	1.59	1.12e-1					θ_{10}	9.45e3	1.47	1.41e-1
θ_{11}	-4.82e4	-1.10	2.72e-1					θ_4	3.43e3	1.02	3.10e-1
θ_{10}	-2.29e4	-0.52	6.02e-1					θ_{11}	6.64e3	0.91	3.65e-1
θ_0	-2.04e4	-0.52	6.02e-1					θ_7	6.87e3	0.43	6.67e-1
θ_3	-4.50e3	-0.10	9.18e-1					θ_5	1.40e3	0.38	7.05e-1
θ_5	4.68e1	0.00	9.99e-1					θ_{12}	1.06e3	0.13	8.94e-1
								θ_6	8.30e2	0.07	9.44e-1
								θ_8	8.11e1	0.01	9.96e-1

TABLE II: Temperature predictors (θ_i) vs. energy: Estimates (Est.), t-values (t-val.), p-values (p-val.), sorted by p-value. Significant predictors (e.g., θ_1, θ_0, θ_2) show strong energy impact.

	Xeon (12 cores)				Core i7 8th Gen (4 cores)				Core i7 12th Gen (14 cores)		
Predictor	Est.	t-val.	p-val.	Predictor	Est.	t-val.	p-val.	Predictor	Est.	t-val.	p-val.
Context Switch Rate	-3.07e4	-34.90	1.34e-261	Elapsed Time	4.85e7	39.51	0.00e0	Elapsed Time	4.67e7	35.82	3.29e-275
Cache Miss % Refs	3.32e6	34.45	4.02e-255	Cache Miss	2.23e5	21.35	3.80e-100	Atom Branch Misses	1.41e0	25.10	1.21e-137
Cache Refs	2.82e-1	33.53	4.94e-242	Cache Refs	6.39e-2	16.03	1.48e-57	Cache Misses	4.18e-1	24.65	7.37e-133
Elapsed Time	6.44e7	31.52	1.52e-214	Total Cores	-5.34e6	-15.47	1.09e-53	Branch Misses	1.36e0	22.78	6.66e-114
Cycles	-1.98e-1	-16.52	5.22e-61	Branch Miss	3.37e6	10.41	2.43e-25	Atom Branches	-1.93e-2	-18.87	5.72e-79
Branches	-1.69e-1	-14.19	1.59e-45	Instructions	-1.48e-3	-6.51	7.61e-11	Atom Cycles Freq	-2.31e6	-17.85	7.14e-71
CPU Cycles	-4.33e-2	-10.94	8.14e-28	Context Switch Rate	-2.43e2	-5.42	6.10e-8	Atom Branches Rate	7.26e3	17.30	1.06e-66
Context Switches	-2.14e3	-10.15	3.53e-24	Context Switches	2.07e3	5.22	1.82e-7	Branches	1.17e4	15.79	6.30e-56
Cache Misses	-4.37e-1	-9.78	1.47e-22	Insn per Cycle	5.84e5	4.33	1.52e-5	Atom Cycles	-1.07e-1	-15.33	7.10e-53
Total Cores	7.82e4	9.69	3.43e-22	Branch Misses	-1.06e0	-4.18	2.95e-5	User Time	-1.31e8	-11.71	1.33e-31
Cycles Freq	1.74e8	9.65	5.41e-22	Cache Misses	1.04e-1	3.88	1.03e-4	Sys Time	-1.29e8	-11.59	5.34e-31
CPU Clock ms	2.63e6	8.81	1.29e-18	CPU Clock	-5.82e5	-3.68	2.30e-4	Faults	3.77e2	11.31	1.32e-29
User Time	-2.27e8	-8.63	6.37e-18	CPU Cycles Freq	1.69e5	3.56	3.65e-4	Cycles Freq	7.57e4	11.31	1.40e-29
Sys Time	-2.20e8	-8.38	5.60e-17	Task Clock	5.60e5	3.55	3.91e-4	Page Faults	-3.70e2	-11.23	3.42e-29
Cache Refs Rate	6.97e5	8.05	8.51e-16	CPU Cycles	6.89e-4	2.64	8.18e-3	Avg Freq	1.14e3	10.39	3.00e-25
Task Clock	-1.80e6	-6.17	7.12e-10	Page Fault Rate	9.68e3	2.45	1.41e-2	Atom Cache Miss	-1.29e4	-10.15	3.74e-24
Branch Misses	1.82e0	4.24	2.22e-5	Fault Rate	9.68e3	2.45	1.42e-2	Atom Branch Miss	9.16e0	9.09	1.07e-19
Insn per Cycle	-9.52e6	-4.23	2.36e-5	Task Util	9.01e6	2.28	2.25e-2	Instructions	-4.71e-4	-8.74	2.45e-18
Task Util	-1.41e8	-2.30	2.12e-2	Cycles Freq	3.52e5	1.39	1.64e-1	Cache Miss	1.25e4	7.55	4.56e-14
CPUs Used	1.25e8	2.04	4.14e-2	Branches Rate	1.56e3	1.34	1.80e-1	Branches Rate	-1.77e-1	-6.68	2.44e-11
CPU Cycles Freq	2.00e6	1.20	2.30e-1	User Time	4.06e6	1.04	2.99e-1	Cycles	-1.11e-3	-5.93	3.04e-9
Page Faults	-2.64e1	-1.18	2.36e-1	Cycles	1.15e-3	0.99	3.21e-1	Branch Miss	6.12e0	5.86	4.69e-9
Faults	-2.62e1	-1.17	2.41e-1	CPUs Used	-3.43e6	-0.87	3.85e-1	CPU Cycles	-8.56e-4	-5.71	1.14e-8
Instructions	7.52e-4	0.51	6.09e-1	Sys Time	3.21e6	0.82	4.12e-1	Atom Cycles	1.12e-3	5.67	1.47e-8
Fault Rate	-1.69e2	-0.00	9.97e-1	Faults	5.90e0	0.53	5.95e-1	Branch Miss	2.58e4	5.37	8.12e-8
Page Fault Rate	-1.69e2	-0.00	9.97e-1	Branches	-8.77e-4	-0.17	8.69e-1	Context Switch Rate	-5.68e2	-4.86	1.20e-6
				Page Faults	1.90e-1	0.02	9.86e-1	Atom Cache Miss	1.09e0	4.61	4.01e-6

TABLE III: Profiler predictors (excl. temp, freq, speed) vs. energy: Estimates (Est.), t-values (t-val.), p-values (p-val.), sorted by p-value. Key predictors (e.g., Context Switch Rate, Elapsed Time) drive energy variance.

TABLE IV: MSE percentage and total number of parameters for different architectures on Intel Core i7 12th Gen.

Model	Temperature		Profiler	
	MSE	Params	MSE	Params
FCN	1.0299	2014	3.9047	3787
FCN+RF	0.9808	1694	2.4862	2699
FCN+RF+BS	**0.9640**	**1694**	**2.4669**	**2699**
RNN	1.0119	3070	2.8493	4843
LSTM	1.0307	9310	2.7778	15115
Conv	1.0134	5118	2.8217	6891
Attention	1.0143	6238	2.8933	8011
SOTA [19]	2.5000	-	-	-

energy and temperature prediction. Multi-platform results and comparisons to SOTA validate the approach's effectiveness, rigor, and applicability to diverse multi-core systems.

V. Related Work

a) Probabilistic Methods for DVFS and Task-to-Core Allocation: Probabilistic worst-case execution time (pWCET) and worst-case energy consumption (pWCEC) estimation in embedded real-time systems leverage measurement-based and static methods [11]–[14]. Pallister et al. [14] studied instruction-specific impacts on pWCEC, and extreme value theorem (EVT) [20] provided upper bounds for performance metrics. However, these approaches rarely quantify the statistical significance of system metrics for energy or latency bounds. Our work introduces a hybrid statistical learning framework, systematically prioritizing feature importance to optimize energy-efficient

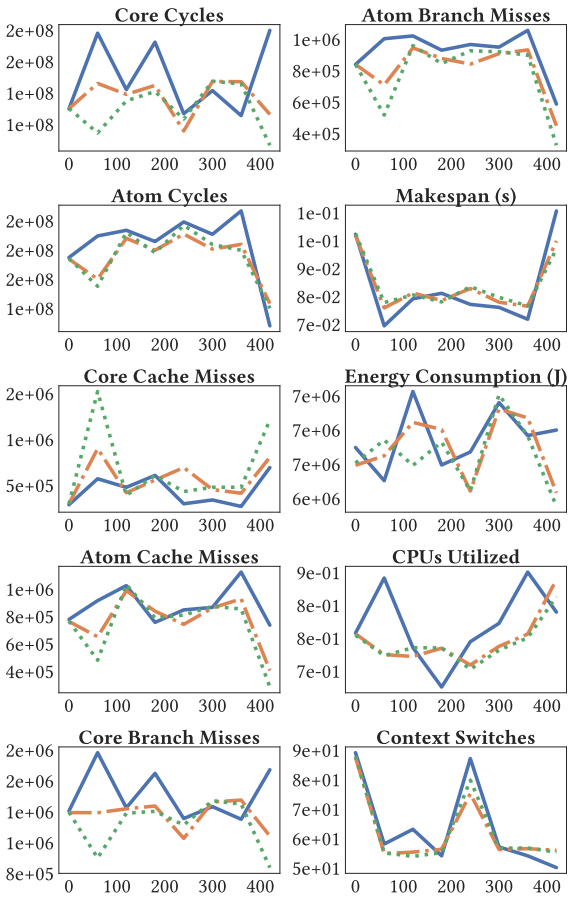


Fig. 7: Results for profiler data prediction from ground truth on Intel Core i7 12th Gen for regular FCN and FCN via Random Forest feature reduction strategy.

DVFS and task-to-core allocation across diverse platforms, surpassing traditional probabilistic bounds.

b) Statistical Learning: Statistical learning has been employed to pinpoint features for energy optimization and scheduling, often using hardware events or application traits [8], [11], [15]. Sasaki et al. [15] applied decision trees to streamline DVFS table lookups, while Cazorla et al. [11] used hardware counters for energy reduction. Liu et al. [8] correlated compiler features with latency. Yet, these efforts underexplored runtime metrics and sampling strategies critical for accuracy. We enhance this by integrating runtime performance metrics with a novel feature selection pipeline, validated across Intel Xeon and Core i7 platforms, improving parallel scheduling and environment modeling precision.

c) Low-Energy DVFS and Task-to-Core Allocation: Low-energy multicore scheduling via DVFS and task-to-core mapping is well-studied [21]–[30]. Xie et al. [21] surveyed heuristic and machine learning methods for energy-constrained scheduling. Many such ML approaches, however, require extensive datasets and computational resources, limiting practicality.

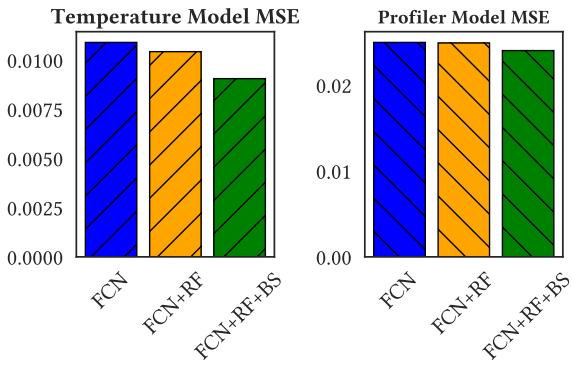


Fig. 8: Comparison of FCN models with and without feature selection and bootstrapping on Intel Core i7 12th Gen.

Our method mitigates these issues with a lightweight statistical model, reducing data needs while maintaining accuracy, as demonstrated on embedded and heterogeneous systems, offering a scalable alternative to existing high-overhead techniques.

d) Few-Shot RL: Few-shot learning, including transfer learning and meta-learning, minimizes data demands in scheduling [31]–[35]. MAML [36] adapts to new tasks with few samples, and model-based RL [37] approximates dynamics to limit real-world data use. Works like [9], [10], [38], [39] underutilize statistical resampling and feature ranking for energy-efficient scheduling. Our approach fills this gap, combining resampling with a robust feature selection strategy, validated across platforms, to enhance few-shot task-to-core allocation efficiency.

e) Feature Evaluation: Thermal-aware scheduling models often predict behavior using utilization or temperature data to prevent throttling [1], [4], [40]–[44]. Studies like [4], [9], [10] model transients and ambient conditions [19], but lack rigorous statistical analysis of feature correlations. We address this by applying a systematic correlation-based feature evaluation, rigorously tested on real hardware, to uncover interdependencies and optimize DVFS and allocation, improving energy and performance outcomes over prior heuristic-driven methods.

VI. Conclusion

We demonstrated the effectiveness of feature selection using statistical learning for environment modeling and task-to-core allocation in embedded systems. Our correlation-aware task-to-core allocation reduces energy consumption by up to 10% and temperature by up to 5°C compared to random core selection. The compressed bootstrapped regression model reduces thermal prediction error by 6% and the number of parameters by 16%. Tested on Intel Core i7 8th and 12th generation, Intel Xeon 2680 processors and Jetson TX2, our method shows a 61.6% reduction in mean squared error compared to state-of-the-art approach. This finding paves the way for future use of statistical learning methods in performance efficiency of task-to-core allocation in heterogeneous processors.

ACKNOWLEDGEMENT

The work was supported by the US National Science Foundation through grants CNS-2301757, CAREER- 2306486, CNS-2306745, and by the US Office of Naval Research through grant N00014-23-1-2151.

REFERENCES

[1] S. Hosseinimotlagh and H. Kim, "Thermal-aware servers for real-time tasks on multi-core gpu-integrated embedded systems," in *2019 IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2019, pp. 254–266.

[2] S. et al., "Temperature-aware microarchitecture," *ACM SIGARCH*, 2003.

[3] I. Ratković, N. Bežanić, O. S. Ünsal, A. Cristal, and V. Milutinović, "An overview of architecture-level power-and energy-efficient design techniques," *Advances in Computers*, vol. 98, pp. 1–57, 2015.

[4] S. Maity, R. Roy, A. Majumder, S. Dey, and A. R. Hota, "Future aware dynamic thermal management in cpu-gpu embedded platforms," in *2022 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2022, pp. 396–408.

[5] M. Shekarisaz, L. Thiele, and M. Kargahi, "Automatic energy-hotspot detection and elimination in real-time deeply embedded systems," in *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2021, pp. 97–109.

[6] S. Pagani, P. S. Manoj, A. Jantsch, and J. Henkel, "Machine learning for power, energy, and thermal management on multicore processors: A survey," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 39, no. 1, pp. 101–116, 2018.

[7] D. Brodowski, N. Golde, R. J. Wysocki, and V. Kumar, "Cpu frequency and voltage scaling code in the linux (tm) kernel," *Linux kernel documentation*, p. 66, 2013.

[8] D. Liu, S.-G. Yang, Z. He, M. Zhao, and W. Liu, "Cartad: Compiler-assisted reinforcement learning for thermal-aware task scheduling and dvfs on multicores," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 2021.

[9] C. Lin, K. Wang, Z. Li, and Y. Pu, "A workload-aware dvfs robust to concurrent tasks for mobile devices," in *Proceedings of the 29th Annual International Conference on Mobile Computing and Networking*, 2023, pp. 1–16.

[10] S. Kim, K. Bin, S. Ha, K. Lee, and S. Chong, "ztt: Learning-based dvfs with zero thermal throttling for mobile devices," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 41–53.

[11] F. J. Cazorla, L. Kosmidis, E. Mezzetti, C. Hernandez, J. Abella, and T. Vardanega, "Probabilistic worst-case timing analysis: Taxonomy and comprehensive survey," *ACM Computing Surveys (CSUR)*, vol. 52, no. 1, pp. 1–35, 2019.

[12] R. I. Davis and L. Cucu-Grosjean, "A survey of probabilistic timing analysis techniques for real-time systems," *LITES: Leibniz Transactions on Embedded Systems*, pp. 1–60, 2019.

[13] F. Reghenzani, G. Massari, and W. Fornaciari, "Probabilistic-wcet reliability: Statistical testing of evt hypotheses," *Microprocessors and Microsystems*, vol. 77, p. 103135, 2020.

[14] J. Pallister, S. Kerrison, J. Morse, and K. Eder, "Data dependent energy modeling for worst case energy consumption analysis," in *Proceedings of the 20th International Workshop on Software and Compilers for Embedded Systems*, 2017, pp. 51–59.

[15] H. Sasaki, Y. Ikeda, M. Kondo, and H. Nakamura, "An intra-task dvfs technique based on statistical analysis of hardware events," in *Proceedings of the 4th international conference on Computing frontiers*, 2007, pp. 123–130.

[16] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM computing surveys (CSUR)*, vol. 50, no. 6, pp. 1–45, 2017.

[17] P. Sedgwick, "Pearson's correlation coefficient," *Bmj*, vol. 345, 2012.

[18] A. Duran, X. Teruel, R. Ferrer, X. Martorell, and E. Ayguade, "Barcelona openmp tasks suite: A set of benchmarks targeting the exploitation of task parallelism in openmp," in *2009 international conference on parallel processing*. IEEE, 2009, pp. 124–131.

[19] S. Hosseinimotlagh, D. Enright, C. R. Shelton, and H. Kim, "Data-driven structured thermal modeling for cots multi-core processors," in *2021 IEEE Real-Time Systems Symposium (RTSS)*. IEEE, 2021, pp. 201–213.

[20] S. Edgar and A. Burns, "Statistical analysis of wcet for scheduling," in *Proceedings 22nd IEEE Real-Time Systems Symposium (RTSS 2001)(Cat. No. 01PR1420)*. IEEE, 2001, pp. 215–224.

[21] G. Xie, X. Xiao, H. Peng, R. Li, and K. Li, "A survey of low-energy parallel scheduling algorithms," *IEEE Transactions on Sustainable Computing*, vol. 7, no. 1, pp. 27–46, 2021.

[22] G. Xie, G. Zeng, X. Xiao, R. Li, and K. Li, "Energy-efficient scheduling algorithms for real-time parallel applications on heterogeneous distributed embedded systems," *IEEE Transactions on Parallel and Distributed Systems*, vol. 28, no. 12, pp. 3426–3442, 2017.

[23] D. Zhu, R. Melhem, and D. Mossé, "The effects of energy management on reliability in real-time embedded systems," in *IEEE/ACM International Conference on Computer Aided Design, 2004. ICCAD-2004.* IEEE, 2004, pp. 35–40.

[24] J. Jiang, W. Li, L. Pan, B. Yang, and X. Peng, "Energy optimization heuristics for budget-constrained workflow in heterogeneous computing system," *Journal of Circuits, Systems and Computers*, vol. 28, no. 09, p. 1950159, 2019.

[25] Y. Chen, G. Xie, and R. Li, "Reducing energy consumption with cost budget using available budget preassignment in heterogeneous cloud computing systems," *IEEE Access*, vol. 6, pp. 20 572–20 583, 2018.

[26] J. Zhou, J. Yan, K. Cao, Y. Tan, T. Wei, M. Chen, G. Zhang, X. Chen, and S. Hu, "Thermal-aware correlated two-level scheduling of real-time tasks with reduced processor energy on heterogeneous mpsocs," *Journal of Systems Architecture*, vol. 82, pp. 1–11, 2018.

[27] Y. G. Kim and C.-J. Wu, "Autoscale: Energy efficiency optimization for stochastic edge inference using reinforcement learning," in *2020 53rd Annual IEEE/ACM international symposium on microarchitecture (MICRO)*. IEEE, 2020, pp. 1082–1096.

[28] S. M. P. Dinakarrao, A. Joseph, A. Haridass, M. Shafique, J. Henkel, and H. Homayoun, "Application and thermal-reliability-aware reinforcement learning based multi-core power management," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 15, no. 4, pp. 1–19, 2019.

[29] H. Shen, J. Lu, and Q. Qiu, "Learning based dvfs for simultaneous temperature, performance and energy management," in *Thirteenth International Symposium on Quality Electronic Design (ISQED)*. IEEE, 2012, pp. 747–754.

[30] Z. Wang, Z. Tian, J. Xu, R. K. Maeda, H. Li, P. Yang, Z. Wang, L. H. Duong, Z. Wang, and X. Chen, "Modular reinforcement learning for self-adaptive energy efficiency optimization in multicore system," in *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. IEEE, 2017, pp. 684–689.

[31] Y. Wang, Q. Yao, J. T. Kwok, and L. M. Ni, "Generalizing from a few examples: A survey on few-shot learning," *ACM computing surveys (csur)*, vol. 53, no. 3, pp. 1–34, 2020.

[32] D. Lee, N. He, P. Kamalaruban, and V. Cevher, "Optimization for reinforcement learning: From a single agent to cooperative agents," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 123–135, 2020.

[33] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[34] C. Florensa, Y. Duan, and P. Abbeel, "Stochastic neural networks for hierarchical reinforcement learning," *arXiv preprint arXiv:1704.03012*, 2017.

[35] S. Arora and P. Doshi, "A survey of inverse reinforcement learning: Challenges, methods and progress," *Artificial Intelligence*, vol. 297, p. 103500, 2021.

[36] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 1126–1135.

[37] T. M. Moerland, J. Broekens, A. Plaat, C. M. Jonker *et al.*, "Model-based reinforcement learning: A survey," *Foundations and Trends® in Machine Learning*, vol. 16, no. 1, pp. 1–118, 2023.

[38] T. Zhou and M. Lin, "Deadline-aware deep-recurrent-q-network governor for smart energy saving," *IEEE Transactions on Network Science and Engineering*, vol. 9, no. 6, pp. 3886–3895, 2021.

[39] Z. Zhang, Y. Zhao, H. Li, C. Lin, and J. Liu, "Dvfo: Learning-based dvfs for energy-efficient edge-cloud collaborative inference," *IEEE Transactions on Mobile Computing*, 2024.

[40] L. Yan, J. Luo, and N. K. Jha, "Combined dynamic voltage scaling and adaptive body biasing for heterogeneous distributed real-time embedded

systems," in *ICCAD-2003. International Conference on Computer Aided Design (IEEE Cat. No. 03CH37486)*. IEEE, 2003, pp. 30–37.

[41] D. Brooks, R. P. Dick, R. Joseph, and L. Shang, "Power, thermal, and reliability modeling in nanometer-scale microprocessors," *Ieee Micro*, vol. 27, no. 3, pp. 49–62, 2007.

[42] G. Singla, G. Kaur, A. K. Unver, and U. Y. Ogras, "Predictive dynamic thermal and power management for heterogeneous mobile platforms," in *2015 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2015, pp. 960–965.

[43] D. Li and J. Wu, "Energy-aware scheduling for frame-based tasks on heterogeneous multiprocessor platforms," in *2012 41st International Conference on Parallel Processing*. IEEE, 2012, pp. 430–439.

[44] A. Kassab, J.-M. Nicod, L. Philippe, and V. Rehn-Sonigo, "Green power aware approaches for scheduling independent tasks on a multi-core machine," *Sustainable Computing: Informatics and Systems*, vol. 31, p. 100590, 2021.