Edge learning for energy-aware resource management

Nasser Alkhatani, Ioan Petri
School of Engineering
Cardiff University
Cardiff, UK
{AlkhataniNM, petrii}@cardiff.ac.uk

Omer Rana
School of Computer Science and Informatics, Cardiff University
Cardiff. UK
ranaof@cardiff.ac.uk

Manish Parashar
School of Computing
University of Utah
Utah, US
parashar@sci.utah.edu

***Abstract*— As the demand for intelligent systems grows, leveraging edge learning and autonomic self-management offers significant benefits for supporting real-time data analysis and resource management in edge environments. We describe and evaluate four distinct task allocation scenarios to demonstrate the autonomics for edge resources management: random execution, autonomic broker-based scheduling, priority-driven execution, and energy-aware allocation. Our experiments reveal that while prioritization-based scheduling minimizes execution times by aligning with task criticality, the energy-aware approach presents a sustainable alternative. This method dynamically adapts task execution based on renewable energy availability, promoting environmentally conscious energy management without compromising operational efficiency. By harnessing renewable energy signals, our findings highlight the potential of edge autonomics to achieve a balance between performance, resource optimization and sustainability. This work demonstrates how intelligent edge-cloud integration can foster resilient smart building infrastructures that meet the challenges of modern computing paradigms.**

Keywords—edge learning, autonomics, energy aware scheduling, buildings, sustainability

I. INTRODUCTION

Increase in the number of edge devices and their decreased production costs are now a compelling incentive for industry applications looking to cut costs by switching from a fully cloud-based resource-dependent model to a less expensive hybrid model that incorporates edge devices. Similarly, machine learning (ML) has assumed a pivotal role across various industries and applications, fundamentally transforming decision-making processes, automating tasks and unearthing valuable insights from extensive datasets [1]. For instance, use of ML in a building energy management system to facilitate real-time data analysis and generate predictive insights enables autonomous regulation of energy consumption [2]. The effective integration of autonomics in governing and fine-tuning machine learning models has become indispensable for addressing the challenges associated with varying computational environments and workload demands [3]. By dynamically managing computational resources and enabling real-time adjustments, an autonomic system ensures efficient utilization and enhances the adaptability of ML models across different deployment platforms [4].

Edge devices can be part of computing frameworks closer to an end user and can expose more control and administration in data gathering and generation. In recent years, there has been a rise in the number of applications that can be run directly on edge devices, particularly those that require in situ analysis and event processing along with a transfer to cloud-based execution.

We assess the efficacy of ML algorithms operating on different edge resource configurations, within building energy optimization. We perform "what-if" analyses of ML tasks executed on edge environments to enable insights into performance, optimization of such ML tasks and adaptation to different applications. We evaluate actuation and sensing to support ML orchestration and execution on edge systems. We investigate how an autonomously controlled energy system can reduce building energy consumption while ensuring optimized use of the computing infrastructure. A comparative analysis is carried out on the use of multiple ML models and their behavior when deployed across different platforms, considering the influence of the integrated autonomic system on their adaptability and real-time performance. The subsequent sections of this paper are structured as follows: Section 2 provides an extensive review of relevant literature on machine learning models and previous comparative studies. Section 3 delineates the methodology, encompassing data preparation and model implementation. Section 4 presents the findings of the models along with a comprehensive discussion. Finally, Section 5 concludes the paper, summarizing key findings and suggesting potential avenues for future research.

II. RELATED WORK

With the development of edge computing and the growth of Internet of Things (IoT) infrastructures, users have the ability to customize applications and improve the overall performance of their workflows. Edge computing has become necessary to reduce the load on cloud systems infrastructure and enables processing data in the vicinity of a data source. This capability also supports security/ data privacy, performance, and reduces communication latency and delays. Several studies [5] report that using autonomic techniques alongside edge-based deployment can enhance the effectiveness of interactions between devices and applications whilst accelerating and cutting processing costs. The functional requirements of an edge infrastructure include:

(1) Self-configuration, which refers to automatic device initialization during operation.

(2) Self-optimization referring to the system's ability to continuously raise the value of non-functional resources in response to a given set of incoming application needs or corporate goals.

(3) Self-restoration, which refers to the system's ability to identify and fix issues on its own while in use.

(4) Self-protection, measuring the system's ability to respond to and stop harmful attempts or intrusions by putting various trust and protection techniques in place.

A novel perspective on the advancement of Internet of Things (IoT) technologies and intelligent mobile edge computing (MEC) has been explored by several related studies demonstrating that moving computer tasks and complex analysis from cloud to the edge can improve the performance and the accuracy of the workflows leading to fast data processing, low latency, and advanced intelligence [7].

Artificial Neural Networks (ANNs) have gained widespread attention in energy applications due to their ability to model complex relationships in data. ANNs consist of interconnected nodes that simulate the functioning of neurons [6] and are applied for image and speech recognition, natural language processing, and financial forecasting [7]. Recent innovations in machine learning have led to the development of deep neural networks, such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), which have achieved remarkable results in tasks like image classification and sequence prediction [8]. However, ANNs require large datasets and substantial computational resources for training and can be prone to overfitting [9].

Support Vector Machines (SVMs), on the other hand, are powerful supervised learning models known for their ability to handle classification and regression tasks [10]. SVMs help to maximize the margin between different classes, making them effective for both linearly separable and non-linearly separable data [11]. SVMs have been successfully applied in various domains, including text classification, image recognition, and bioinformatics [12]. They offer robust generalization performance and are less susceptible to overfitting, but they can be sensitive to the choice of kernel function and parameter tuning [13]. Random Forest is an ensemble learning technique that combines multiple decision trees to improve predictive accuracy and reduce overfitting [14]. It has gained popularity for its versatility and effectiveness in classification and regression tasks [15]. Random Forest models are less prone to overfitting compared to individual decision trees and can handle both categorical and numerical data [16]. They have been applied in diverse fields, such as remote sensing, bioinformatics, and finance [17]. The model's ability to capture complex relationships in data while providing insights into feature importance makes it a popular choice in machine learning [18].

In relation to the performance of different machine learning models, specifically SVM, ANN, and Random Forest, across various applications, several authors [21] havefound that ANN achieved comparable or superior results to SVM in document-level sentiment analysis, particularly outperforming SVM in the context of unbalanced data. On the other hand, other authors [20]demonstrate that SVM generally outperformed Random Forest in microarray-based cancer classification. A similar study [21] has compared Random Forest with SVM in remote sensing classification and found that both models performed equally well in terms of classification accuracy and execution time, with Random Forest requiring fewer user-defined parameters. Lastly, other authors [24] have proposed a method to improve the speed of SVM classification in sentiment analysis tasks, showing that it outperformed the standard SVM method in terms of execution time.

Recent research has focused on energy-aware scheduling for edge computing and IoT environments to optimize resource allocation and minimize carbon footprint while maintaining quality of analysis. Studies have explored various approaches, including priority-based task scheduling for edge devices [23], energy-aware job scheduling for vehicular edge networks [24], and zone-oriented algorithms for serverless edge computing [25]. These methods aim to maximize the use of renewable energy sources while considering the dynamic nature of task demands and energy availability. Additionally, real-time scheduling for smart grid integration with renewable energy has been investigated, addressing challenges such as power system stability, frequency regulation, and voltage management [26]. The integration of advanced forecasting methods, energy storage systems, and optimization techniques such as Linear Programming and Dynamic Programming has shown promise in enhancing scheduling efficiency and grid resilience[29], [30].

Building on these findings, the present study investigates and compares four task allocation strategies—random execution, autonomic broker scheduling, priority-based allocation, and energy-aware orchestration—to comprehensively evaluate the impact of machine learning energy workflows on performance, resource utilization, and energy efficiency in hybrid computing environments.

III. Edge Autonomics for Energy Management

The proliferation of data and the advancement of information technology, including artificial intelligence (AI) and the Internet of Things, have enabled the rise energy management and automation in buildings and industries. In the engineering and industrial sectors, various assets at the building or city level can benefit from the recent advantages in machine learning and edge computing. Energy models for buildings are developed using sensor data where a digital simulation/twin of the asset can be enabled to gather information and analyse the energy performance of the building.

Energy applications are striving to reach a higher order of intelligence through the integration of various surrogate machine learning models and their subsequent execution required for real-time energy analysis and performance projections. This kind of intelligence can be put into practise by means of various controllers and actuators that enhance the physical asset's administration and operation by utilising the control signals or set-points derived from the energy simulation. Energy optimisation scenarios, for instance, can be used in buildings to optimise ventilation and heating by using various set-points that result from machine learning analysis. By using actuators to adjust the air ventilation temperature, these set-points improve the building facility's energy efficiency. Using the right machine learning models to generate the necessary level of intelligence around the physical asset is a crucial step in achieving energy optimisation in buildings.

These machine learning algorithms frequently possess a certain level of complexity and necessitate diverse computational resources that are contingent upon quality-of-service requirements and application-specific time constraints. Further research is required to determine how such machine learning tasks can be hosted closer to the energy data capture points within the built asset, to create an environment that can span across multiple computational

layers from edge to cloud resources. Hosting energy optimisation services and operations at the edge requires the use of autonomics to address a number of significant challenges related to the orchestration of models and their subsequent execution on the edge (Fig. 1).

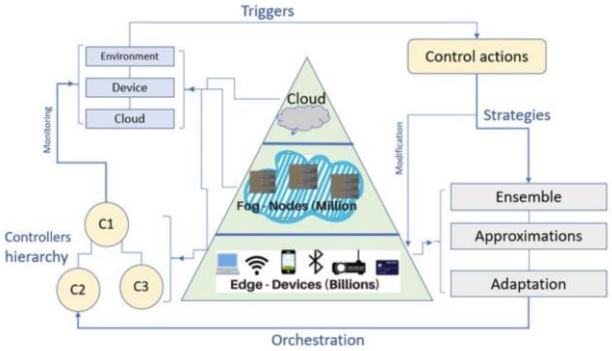

FIGURE 1: EDGE COMPUTING LAYERS

Cloud platforms like Azure Microsoft offer scalable and accessible environments for deploying machine learning models with benefits such as scalability and ease of deployment [27]. Existing models expose a serverless computing infrastructure for deploying machine learning applications in the edge layer, addressing challenges like latency and data privacy [32], accuracy ofpredictions [29] and qualitative data analysis [30].

Deploying machine learning models on resource-constrained edge devices (i.e., Raspberry Pi) presents challenges that can be addressed through ML model optimization techniques. Several solutions exist that propose a hardware-friendly pruning technique to create user-specific machine learning models directly on mobile platforms, resulting in speedups for specific models on edge CPUs [31]. A runtime adaptive convolutional neural network (CNN) acceleration framework, for example, optimized for heterogeneous IoT environments can be used to dynamically select the optimal degree of parallelism based on available computational resources and network conditions, leading to improved inference speed and reduced communication costs on Raspberry Pi devices [32]. Local deployments such as regular computers (i.e., laptops or desktop computers) can offer a comparison basis with advantages in terms of data privacy and reduced latency. Several studies highlight the need for standardized architecture to integrate and manage machine-learned components in industrial settings [33]. Such deployment brings challenges and opportunities in hardware design for machine learning applications, particularly in local embedded processing near the sensor such as energy consumption, cost, throughput, accuracy, and flexibility requirements [34].

Autonomic systems play a pivotal role in automating and optimizing the management of machine learning operations, ensuring adaptive and efficient performance in varying computing environments [35]. The models around autonomic computing aim to increase reliability and performance through self-protecting, self-healing, self-configuring, and self-optimizing mechanisms. Several authors emphasize the importance of autonomic systems in automating and optimizing machine learning operations for adaptive and efficient performance [36], [37].

Related studies highlight the potential of autonomic systems in mitigating performance bottlenecks and ensuring consistent model behavior under fluctuating workloads, with a multi-objective optimization solution to find trade-offs between model accuracy and resource consumption to enable the deployment of machine learning models in resource-constrained smart environment [38], [39]. Other studies introduce methods to find a combination of multiple models that are optimal in terms of energy efficiency and model performance [40], [41]. These studies emphasize the need for autonomous power management strategies to minimize energy consumption without compromising model performance, thus extending the operational lifetime of resource-constrained devices.

This paper shows how edge autonomics can be used to coordinate the execution of different machine learning algorithms necessary for delivering energy management in buildings and to enable autonomy and timely control response to signals received from sensor data within the building assets.

IV. METHODOLOGY

We conduct the analysis on a real case study example such as Cardiff University's Queen's building (Fig. 2), an educational building facility aiming to reduce and optimise energy consumption. The building is instrumented with sensors measuring temperature, humidity, air quality and carbon concentration and is used to demonstrate the benefits of edge autonomics for energy optimization. Based on sensor readings, a machine learning model is developed to inform energy analysis and optimization of the facility.

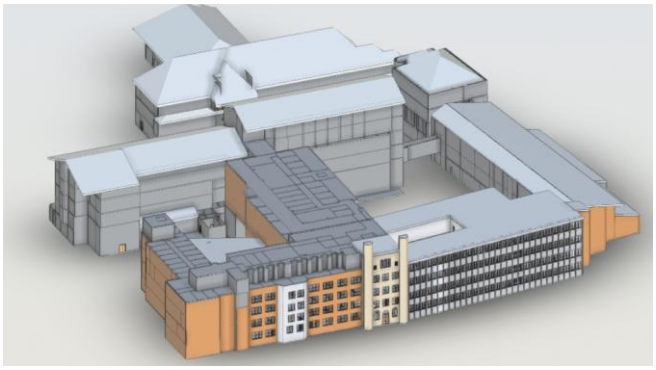

FIGURE 2: QUEEN'S BUILDING MODEL

We conducted a comparative analysis of three prominent forecasting machine learning models developed using historical energy consumption data from Queen's Building: Artificial Neural Network (ANN), Support Vector Machine (SVM), and Random Forest (RF), applied to the Forum (an open space in Queen's buildings as illustrated in Figure 1). For demonstrating edge autonomics for energy optimization, we have constructed a Node-Red model [43] to gather data from sensors and test an autonomic broker used to coordinate machine learning models on three different computational environments namely (i) cloud computing, (ii) edge computing and (iii) traditional computing environments using an MQTT Broker [44]. The MQTT Broker automates task execution based on the readings received from sensors. This implies that if the incoming sensor data is sensitive and requires immediate action, the Broker will, based on predefined autonomics rules, direct it to a specific computing environment, among the three, for immediate execution (Fig. 3).

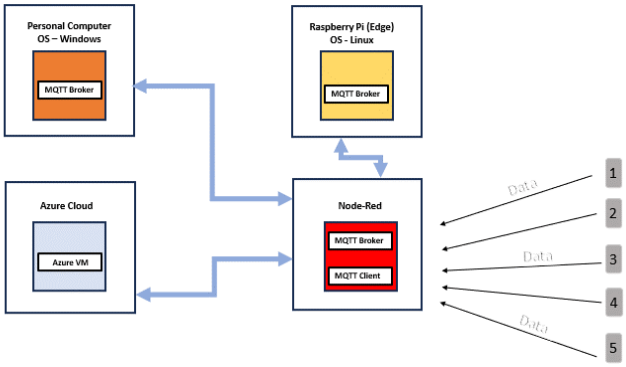

FIGURE 3: THE EXPERIMENTAL FRAMEWORK AND DATA CHANNELS

The machine learning training phase utilizes a carefully curated dataset of historical energy consumption and indoor parameters sourced from the Queen's Building – Forum Room, with a total size of 5.61 MB. This dataset was selected based on its appropriateness for training and evaluating machine learning models. To evaluate the generalization performance of the models, a separate testing dataset was created by randomly dividing the original dataset. This approach ensured that the models were assessed on unseen data to provide unbiased performance metrics. Prior to training, the dataset underwent data cleaning procedures to handle missing values, outliers, and any data inconsistencies. This step was crucial in ensuring the quality of the data used for model training. Feature engineering involves selecting relevant features from the dataset and potentially creating new ones to enhance model performance. Feature selection methods such as Principal Component Analysis (PCA), Recursive Feature Elimination (RFE), and Mutual Information were employed to identify the most informative attributes for each machine learning model. To ensure that all model features were on a consistent scale, data normalization or scaling was performed. This step was essential for models such as Support Vector Machines, which are sensitive to feature scales. Categorical variables were encoded using techniques such as one-hot encoding to convert them into a numerical format that machine learning models can process. The dataset was divided into training, validation, and testing sets. The training set was used to train the machine learning models, the validation set was used for hyperparameter tuning, and the testing set was reserved for the final evaluation of model performance. The splitting ratio was 70-30% to ensure an adequate amount of data for each phase.

V. EVALUATION

In the context of the experiment, the use of Jupyter Notebook facilitated the uniform execution of machine learning models across the different environments, ensuring the consistency of performance and execution time assessments. The implementation process encompassed various standardized steps for training and evaluating the Support Vector Machine (SVM), Artificial Neural Network (ANN) and Random Forest models with the following model properties:

- `MLPRegressor:`hidden_layer_sizes=(100,50)`,`max_iter=1000`, `random_state=42`
- `SVR: `kernel='rbf'`, `C=100`, `gamma=0.1`, `epsilon=0.1`,
- `RandomForestRegressor:`n_estimators=100`, `random_state=42`

These steps included the initial loading of the dataset from the specified file, followed by data preprocessing to extract model features and the target variable, incorporating necessary data cleaning, transformation, and scaling procedures. Subsequently, the preprocessed data was split into training and testing sets, with an 80-20% division for training and testing, respectively. The subsequent phases focused on individual model training, where each model underwent a specific set of actions, including training time measurements, prediction generation, and the computation of Mean Squared Error (MSE) and R-squared (R^2) as performance metrics. The comparison and evaluation stage allowed for a comprehensive assessment of the models' performance metrics, providing insights into their respective computational efficiencies. These findings offer valuable guidance for making informed decisions regarding model selection, considering the balance between model performance and computational resources, all tailored to meet the specific requirements of the application or problem under examination.

A. Node-RED environment

Integrating the Node-RED experiment into the study proved to be instrumental in providing a comprehensive and accessible visualization of the machine learning model performance (Fig. 4). By leveraging the dynamic capabilities of the Node-RED platform, the experiment facilitated a user-friendly and interactive representation of the complex performance metrics and execution times, allowing for a more engaging and informative analysis.

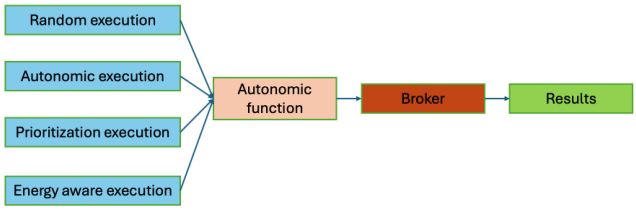

FIGURE 4: NODE-RED EXPERIMENT FRAMEWORK

Moreover, the Node-RED environment effectively portrayed the models' execution times, providing critical insights into their computational efficiency and responsiveness. By offering real-time updates and visual representations of the models' processing capabilities, Node-RED facilitated a nuanced understanding of the trade-offs between computational performance and predictive accuracy, thus enabling researchers to make informed decisions about model selection and deployment strategies.

As such, the evaluation metrics utilized in the experiements include (i) Mean Squared Error (MSE) and (ii) R-squared (R^2), along with the measurement of (iii) execution time. These metrics play a critical role in assessing the performance and computational efficiency of machine learning models across different environments.

1. *Mean Squared Error (MSE):*

- Parameters:

- `Y_test`: Actual target values from the test set.

- `Y_pred`: Predicted target values from the model.

$$MSE = \frac{1}{n}\sum_{k=0}^{n}(Y\_test - Y\_pred)^2$$

2. R^2 *Score*:

- Parameters:

- `Y_test`: Actual target values from the test set.

- `Y_pred`: Predicted target values from the model.

$$R^2 = 1 - \frac{\sum_{k=0}^{n}(Y\_test - Y\_pred)^2}{\sum_{k=0}^{n}(Y\_test - mean(Y\_pred))^2}$$

3. *Execution Time*:

- Parameters:

- `start_time`: The time when the model training or prediction starts.

- `end_time`: The time when the model training or prediction ends.

$$Execution\ Time = end\_time - start\_time$$

MSE is a commonly used metric in regression analysis, quantifies the average squared differences between predicted values and actual values. In the context of this study, MSE is calculated for each model to evaluate the accuracy of predictions, providing insights into the models' ability to capture the variability within the data and make precise estimations. Lower MSE values indicate better predictive performance and a closer fit between the model's predictions and the actual data points. R-squared (R^2) measures the proportion of the variance in the dependent variable that is predictable from the independent variables. It provides an assessment of how well the model fits the observed data, with higher R^2 values indicating a better fit and greater explanatory power. In this study, R^2 is utilized to gauge the predictive strength of the machine learning models and their effectiveness in capturing the variability in the data across different environments.

Furthermore, the execution time is measured to evaluate the computational efficiency of the models in each environment. Execution time refers to the time taken for the machine learning models to process the input data, train the models, and generate predictions for the energy optimization scenarios. Longer execution times may signify higher computational demands and resource requirements, potentially highlighting challenges related to model scalability and real-time processing capabilities. Conversely, shorter execution times indicate more efficient processing and quicker response times, emphasizing the suitability of the models for deployment in time-sensitive applications or resource-constrained environments. These evaluation metrics have been meticulously measured and determined through the standardized implementation of the machine learning models in the respective environments, ensuring consistency and reliability in the assessment of the models' performance and computational efficiency.

B. The Autonomic Broker

The following structured algorithm outlines the autonomics process for efficient task allocation in a machine learning environment. The algorithm is designed to optimize the distribution of tasks based on specific parameters, ensuring effective utilization of resources. This represents a key advantage for the energy optimization problem that relies on data collected from sensors and sends actuation signals to various controlling devices within the building facility.

```
# AUTONOMIC BROKER ALGORITHM
1.  ExecutionTimes = {}  ← Ø     // create
empty dictionary
2.  Models = ['ANN', 'SVM', 'RF'] ←   create
list that contains the machine learning
models 'ANN', 'SVM', and 'RF'.
3.  Environments = ['Azure Cloud',
'Raspberry Pi', 'Personal Computer (PC)']  ←
create list that contains the available
environments 'Azure Cloud', 'Raspberry Pi',
and 'Personal Computer (PC)'.
4.  AutonomicsRules = {

                              'ANN': 'Cloud
                (Azure Cloud)',
                               'SVM': 'Edge
                (Raspberry Pi)',
                                    'RF':
     'Traditional (Personal Computer (PC))'
                          } ← // create empty
 dictionary (to specify the allocation rules
 for each machine learning model to different
                environments).
5.  FOR EACH Model IN Models DO   ← repeat
// Execution of tasks for each model based
on the rules specified in (AutonomicsRules).
a.   AllocatedEnvironment =
AutonomicsRules[Model]  autonomics rules
b.   ExecutionTime = Random.Uniform(1, 10)
← // generate a random execution time
between (1, 10) seconds
c.   IF Model NOT IN ExecutionTimes THEN   ←
condition statment // Store the execution
time for the current model

                      ExecutionTimes[Model] =
                      {}
                    END IF  ←  // the end of
           the condition statement.
d.  ExecutionTimes[Model][AllocatedEnvironme
nt] = ExecutionTime
e.    PRINT "Executing " + Model + " model
in " + AllocatedEnvironment + " with an
execution time of " + ExecutionTime + "
seconds."

     END FOR  ←  // the end of the loop.
6.  PRINT "\nPerformance Evaluation
Results:" ←// display Performance evaluation
results.
7.  FOR EACH Model IN ExecutionTimes DO  ←
repeat
a.  PRINT "Model: " + Model
b.   FOR EACH Environment, Time IN
ExecutionTimes[Model] DO   ← repeat
i. PRINT "  Executed in " + Environment + " in
" + Time + " seconds."

                END FOR  ←  // the end of the
                     loop.
     END FOR  ←  // the end of the loop.
```

ALGORITHM 1: PHASES OF THE AUTONOMIC BROKER EXECUTION

The phases involved in the execution of the edge autonomics broker from Algorithm 1 are:

1. *Initialization*: The algorithm begins by initializing the necessary variables, including the tasks to be executed, the corresponding data sizes, and the anticipated execution times.
2. *Environment Setup*: The predefined environments, namely (i) Cloud (Azure Cloud), (ii) Edge (Raspberry Pi), and (iii)Traditional (Personal Computer (PC)) are established to facilitate the allocation process.
3. *Task Allocation*: For each task in the system, the algorithm follows a decision-making process based on specific conditions. If the data size is less than 40 and the expected execution time exceeds 5 units, the task is allocated to the Traditional computing environment (PC). Similarly, if the data size is less than 40 and the execution time is within or equal to 5 units, the task is assigned to the Edge Computing Environment (Raspberry Pi). Alternatively, if the data size surpasses 40 or the execution time remains indeterminate, the task is directed towards the Cloud computing environment (Azure Cloud) for processing.
4. *Task Execution*: Upon the allocation of tasks to their designated environments, the algorithm proceeds to execute each task within the respective environment.
5. *Performance Measurement*: To facilitate performance evaluation, the algorithm meticulously monitors and records the execution time for each task, providing essential data for subsequent analysis and optimization.

Using the edge autonomics algorithm, the *Autonomics Broker* intelligently allocates tasks to diverse environments, ensuring an optimal balance between computational resources and task requirements. This systematic approach enhances the overall efficiency of task execution within machine learning environments, contributing to the seamless integration of edge resources with machine learning models for energy management and optimization in buildings.

Environment	Model	MSE	R^2	Execution time (seconds)
PC	ANN	9.295204	0.723583	1.385199
	SVM	10.260298	0.694883	0.203290
	RF	9.506779	0.717291	0.638744
Azure	ANN	9.295204	0.723583	3.425744
	SVM	10.260298	0.694883	0.247475
	RF	9.506779	0.717291	0.998159
Raspberry Pi	ANN	9.295204	0.734867	6.491278
	SVM	10.260342	0.694882	0.222211
	RF	9.550808	0.715982	0.651005

TABLE 1: MODELS RESULT COMPARISON

VI. RESULTS

This section describes the machine learning models' performance across different environments, including the (i) Cloud, (ii) Edge and (iii) Traditional computing (i.e. regular PC) settings. The primary aim of this section is to identify the implications of these diverse computational frameworks on the efficiency and adaptability of models, furthering the understanding of the interplay between computational resources and model performance. By exploring the varying impacts of these environments on the execution times and predictive accuracies of the models, this analysis aligns with the central focus of the paper, i.e. developing edge autonomics for energy interventions in buildings through the integration of AI/ML models. These findings can be used for optimizing energy usage and decision-making in real-world applications. By considering performance metrics, such as Mean Squared Error (MSE) and R-squared (R^2), alongside execution times across different platforms, the discussion sheds light on the unique strengths and limitations of each environment in the context of the building energy management framework.

A. Performance comparison

Based on data in Table 1, model selection depends on application requirements and the resources available in each environment. While the ANN model performs relatively well across all environments, the choice of model may vary depending on the trade-offs between computational complexity, predictive accuracy and resource constraints. The selection of a model should also consider the potential for optimization and fine-tuning within each environment to achieve the best possible performance. Preliminary results highlight the importance of computing framework considerations in the context of model performance and execution time. The choice of a framework depends on the specific requirements of the application, including computational resources, scalability, and deployment constraints. The Azure environment stands out as a reliable and efficient option, particularly for applications requiring scalability and reduced execution times. The Raspberry Pi environment, despite its resource constraints, proves to be a viable choice for scenarios where low-power and compact computing solutions are essential. The PC environment remains a feasible option for local computational tasks, although it may not be the most efficient choice for resource-intensive machine learning applications.

B. Cloud based performance analysis

The MSE measures the average of the squares of the errors or deviations, which represents the difference between the actual and predicted values. Lower values of MSE indicate better performance and a closer fit to the data. Based on the results, the ANN model has the lowest MSE of 9.295204, followed by the RF model with an MSE of 9.506779, and the SVM model with an MSE of 10.260298.The R^2 score, also known as the coefficient of determination, represents the proportion of the variance in the dependent variable that is predictable from the independent variables. It provides an indication of the goodness of fit of the model.

Higher values of R^2 indicate a better fit of the model to the data. The ANN model has the highest R^2 score of 0.723583, followed by the RF model with an R^2 score of 0.717291, and the SVM model with an R^2 score of 0.694883. In the given results, the SVM model has the lowest execution time of 0.247475 seconds, followed by the RF model with an execution time of 0.998159 seconds, and the ANN model with the highest execution time of 3.425744 seconds. Based on

these results (Fig. 5), the preferred model environment would depend on the specific priorities of the application.

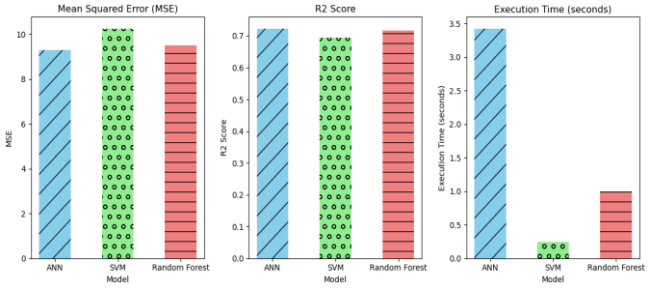

FIGURE 5: MSE, R^2, EXECUTION TIME OF CLOUD (AZURE) RESULTS

However, if a balance between accuracy and execution time is required, the RF model could be a good choice, as it provides competitive performance in terms of MSE and R^2 score, with moderate execution time.

C. Traditional computing performance analysis

For this experiment, we use a regular desktop computer to deploy the machine learning models and test their performance metrics. The MSE values for the models are as follows: ANN has an MSE of 9.295204, SVM has an MSE of 10.260298, and RF has an MSE of 9.506779. The ANN model has the lowest MSE, followed by the RF model and then the SVM model (Fig 6). The R^2 scores for the models are as follows: ANN has an R^2 score of 0.723583, SVM has an R^2 score of 0.694883, and RF has an R^2 score of 0.717291. The ANN model has the highest R^2 score, followed by the RF model, and then the SVM model. The execution times for the models are as follows: ANN took 1.385199 seconds, SVM took 0.203290 seconds, and RF took 0.638744 seconds. The SVM model has the lowest execution time, followed by the RF model and then the ANN model which might not be ideal if speed is a critical factor. The SVM model has the lowest execution time, but it has the highest MSE and a slightly lower R^2 score compared to the ANN model. However, it might be the preferred choice if time-to-execute is a critical consideration. The RF model provides a balance between accuracy and execution time.

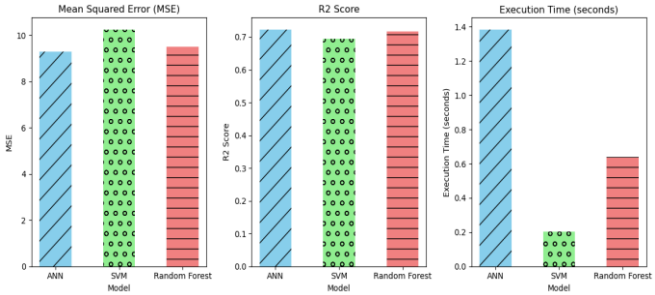

FIGURE 6: MSE, R^2, EXECUTION TIME OF TRADITIONAL COMPUTING SYSTEM

D. Edge computing performance analysis

For this edge computing experiment, we use a Raspberry Pi system to deploy machine learning models and test the associated model performance. The MSE values for the models are as follows: ANN has an MSE of 9.295204, SVM has an MSE of 10.260342, and RF has an MSE of 9.550808. The ANN model has the lowest MSE, followed by the RF model, and then the SVM model (Fig. 7). The R^2 scores for the models are as follows: ANN has an R^2 score of 0.734867, SVM has an R^2 score of 0.694882, and RF has an R^2 score of 0.715982. The ANN model has the highest R^2 score, followed by the RF model, and then the SVM model. The execution times for the models are as follows: ANN took 6.491278 seconds, SVM took 0.222211 seconds, and RF took 0.651005 seconds. The SVM model has the lowest execution time, followed by the RF model. Based on these results the ANN model has the best performance in terms of MSE and R^2 score, but it has the highest execution time, which might not be ideal if speed is a critical factor. The SVM model has the lowest execution time, but it has the highest MSE and a slightly lower R^2 score compared to the ANN model. However, it might be the preferred choice if speed is a critical consideration. The RF model provides a balance between accuracy and execution time.

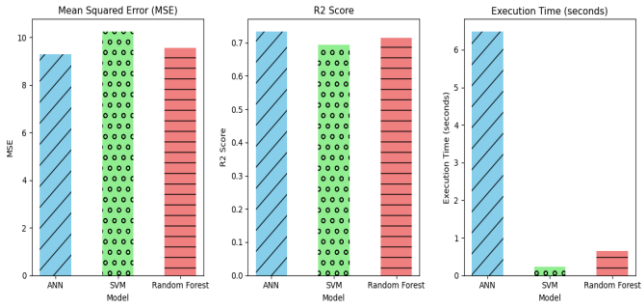

FIGURE 7: MSE, R^2, EXECUTION TIME OF EDGE (RPI) EXPERIMENT RESULTS

The traditional computing environment (i.e., desktop PC) highlights relatively higher Mean Squared Error (MSE) and lower R-squared (R^2) values for the Support Vector Machine (SVM) and Random Forest (RF) models, indicating decreased predictive accuracy compared to other settings. Notably, the Artificial Neural Network (ANN) model exhibits superior performance in this environment, showcasing adaptability despite the general computational limitations. Execution times, however, are notably longer across all models in the PC environment, with the ANN model demonstrating the lengthiest execution time among all environments.

To provide a meaningful baseline for evaluating the proposed autonomic strategies, we also assessed the performance of the machine learning models when executed exclusively in a local computing environment (i.e., desktop PC) without any scheduling mechanism or cross-platform orchestration. This represents a naive scenario where tasks are statically assigned and not adapted based on system conditions or energy availability.

Results showed that although the ANN model achieved the lowest Mean Squared Error (MSE = 9.29) and highest R^2 (0.72), execution times remained relatively high due to lack of resource coordination. More importantly, this baseline failed to exploit renewable energy production periods or task prioritization, which are key to sustainability and responsiveness. This comparison highlights the limitations of non-autonomic execution and validates the need for intelligent task scheduling, as introduced in our autonomic broker and energy-aware allocation strategies.

Based on the data analysis, the cloud environment (i.e., The Azure) ranks highest due to its balanced performance and significantly reduced execution times, making it an optimal choice for deploying the examined machine learning models.

The edge environment (i.e., Raspberry Pi**)** follows closely, demonstrating competitive performance and notable reductions in execution times despite its resource limitations.

E. Edge autonomics

In this section, the edge autonomics is used to demonstrate how different machine learning models can be coordinated to improve the performance of the energy optimization application and integration with the edge. To evidence the edge autonomics principles, we test three different scenarios:

(a) Random execution where the machine learning models are executed randomly of the resources available in cloud, edge and traditional computers (Fig. 8);
(b) Autonomic broker where the machine learning models are executed using edge autonomics mechanisms presented in Algorithm 1 (Fig. 9);
(c) Prioritization execution where the machine learning models are executed based on the priorities of the tasks (Fig. 10);
(d) Energy-aware allocation based on renewable production peaks (Fig. 11);

A. Random execution - In this experiment, the execution of ANN, SVM, and RF models in different computational environments (cloud - Azure Cloud, Edge -Raspberry Pi, and Traditional – PC) is performed randomly. The execution of the machine learning models (ANN, SVM, and RF) in different environments is indicated in the results presented in Fig. 8. The execution time for the models vary, with the ANN model executing in the Azure Cloud environment for approximately 4.56 seconds, the SVM model executing in the same Azure Cloud environment for approximately 6.26 seconds, and the RF model executing on a Raspberry Pi for approximately 7.60 seconds. These results are followed by the simulated performance evaluation, indicating the execution times for each model in the specific environments where they were executed (Fig. 8).

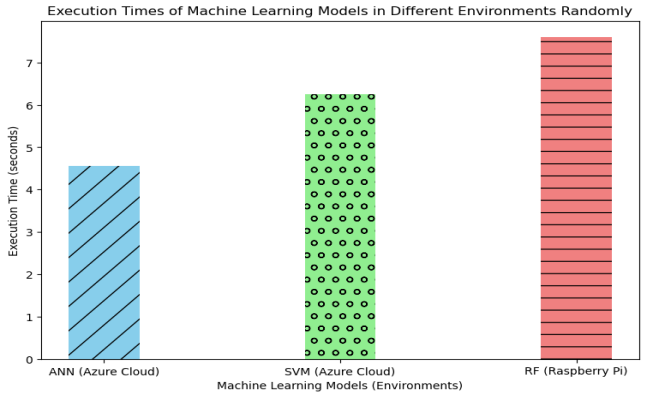

FIGURE 8: RANDOM SCENARIO RESULTS

B. Autonomic Broker – in this experiment the execution of tasks is using an *autonomics broker* (Fig. 9), whereby specific machine learning models ANN, SVM, and RF are assigned to environments *(cloud* - Azure Cloud, *Edge* -Raspberry Pi, and *Traditional* – PC*)* based on predefined autonomics rules (see Algorithm 1).

Using an autonomics broker reflects the dynamic allocation of machine learning tasks, encompassing the ANN, SVM, and RF models across disparate computing environments, specifically the Azure Cloud, Raspberry Pi, and PC. The results demonstrate varying execution times for each model, with the ANN model executing in the Cloud (Azure Cloud) in 5.56 seconds, the SVM model operating on the edge (Raspberry Pi) in 7.71 seconds, and the RF model processing on the Traditional computer (PC) in 9.84 seconds. These findings underscore the crucial role of autonomics in task distribution, effectively utilizing diverse computing resources to optimize task execution and enhance overall performance within a distributed computing paradigm (Fig. 9).

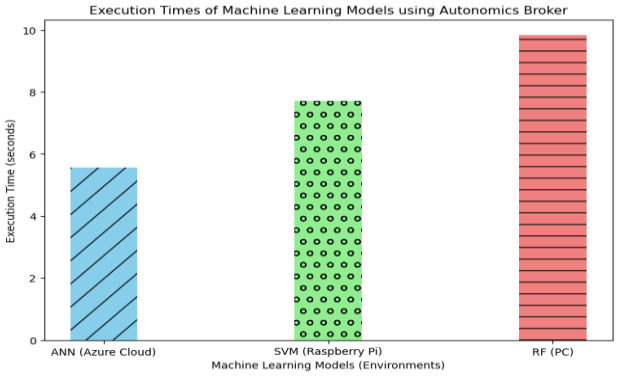

FIGURE 9: AUTONOMIC BROKER SCENARIO RESULTS

C. Prioritisation mechanisms - Employing *prioritization mechanisms* delineates the allocation of machine learning tasks, encompassing the ANN, SVM, and RF models across distinct computing environments, each with assigned performance rankings. The results presented in Figure 10 showcase vary execution times for each model, with the ANN model operating in the Cloud in 1.58 seconds, followed by the SVM model also executing in the Cloud in 3.94 seconds. Furthermore, the RF model conducts its operations on the edge (Raspberry Pi), reflecting an execution time in 6.15 seconds (Fig. 10).

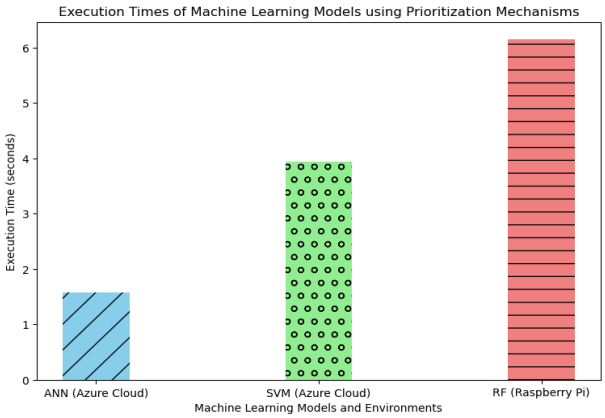

FIGURE 10: PRIORITIZATION SCENARIO RESULTS

D. Energy aware execution - In this experiment, the orchestrator continuously monitors the available renewable energy, particularly solar energy, generated by the building. A renewable energy production peak is defined as a period when the available local solar generation exceeds 80% of the maximum rated production capacity. While there is no universally established threshold in the literature for defining

renewable energy peaks, the 80% level was selected in this study as a practical and realistic operational benchmark. Defining such a threshold enables the broker to make deterministic scheduling decisions while aligning task execution with dynamic energy conditions.

When energy production exceeds the 80% threshold, computationally intensive machine learning tasks are prioritized for immediate execution across available computing environments, such as Azure Cloud or local PCs. Conversely, during periods of low renewable production (below 80%), tasks are either deferred, rerouted to low-power edge devices such as Raspberry Pi systems, or scheduled using the autonomic broker to minimize reliance on non-renewable energy resources (Fig. 11).

The 80% threshold was selected as a practical operational benchmark to capture peak solar energy periods while avoiding premature triggering during moderate generation. This value aligns with heuristics used in solar-aware scheduling in recent literature [24], [25] and balances execution performance with sustainability.

The orchestration strategy implemented in this scenario is guided by three key considerations: (i) the real-time availability of local renewable energy, (ii) the energy state of any connected battery storage systems, and (iii) the criticality and resource demand of the scheduled tasks. By integrating real-time energy monitoring into scheduling decisions, the system maximizes the use of clean energy while ensuring operational efficiency. When solar energy production was at a peak, the ANN and RF models were immediately allocated to Azure Cloud and PC environments, achieving execution times of 5.10 seconds and 4.75 seconds, respectively. In contrast, during low energy production, the SVM model was deferred and executed on a Raspberry Pi, resulting in a longer execution time of 8.35 seconds due to the limited processing capabilities of the device (Fig. 11).

While this study focuses on a single building, the proposed autonomic broker is designed to scale to larger facilities and multi-building campuses. Future work will explore deployment in heterogeneous environments with varying sensor densities, energy production variability, and network constraints. Techniques such as distributed learning and hierarchical scheduling will be investigated to maintain performance and sustainability at scale.

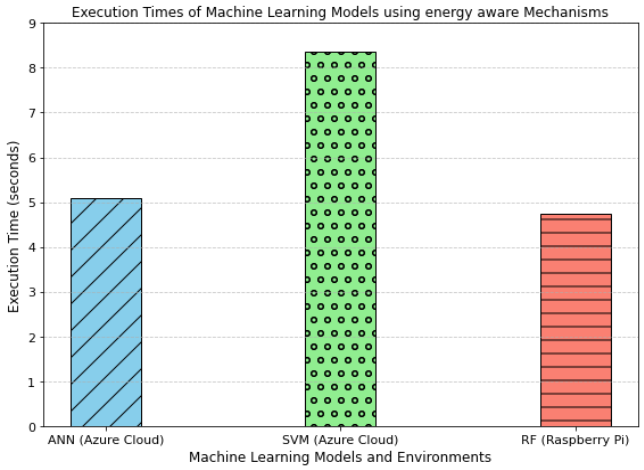

FIGURE 11: *EXECUTION TIME USING ENERGY-AWARE TASK EXECUTION*

These experiments collectively demonstrate the impact of various autonomic scheduling strategies on ML task execution efficiency and energy-conscious computing. The random execution scenario showed the limitations of uncoordinated scheduling, resulting in suboptimal execution times. The autonomic broker scenario introduced rule-based decision-making, reflecting moderate adaptability to system conditions. Prioritization mechanisms achieved the most optimized performance by aligning task criticality with computational capabilities, significantly reducing execution times for high-priority models.

In contrast, the energy-aware allocation strategy prioritized environmental sustainability by adapting execution decisions based on real-time renewable energy availability, even if it occasionally incurred longer execution times. Together, these scenarios underscore the versatility of edge autonomics in balancing performance, resource utilization, and energy efficiency across diverse operating contexts within hybrid computing environments.

VII. CONCLUSIONS

The integration of machine learning models with edge computing technology facilitates real-time data analysis, predictive insights, and edge autonomics, ultimately contributing to a more sustainable and energy-efficient computing environment. In this study, four execution strategies were systematically analyzed to assess their impact on performance, resource utilization, and energy efficiency in edge-enabled computing environments. The random execution scenario revealed the inefficiencies associated with non-deterministic task scheduling. Autonomic broker-based scheduling improved task distribution through rule-based decision making, while prioritization mechanisms achieved the best performance outcomes by aligning execution order with task criticality. Additionally, the energy-aware allocation scenario introduced an important sustainability dimension by dynamically adapting scheduling decisions based on real-time renewable energy availability. Although this energy-aware approach occasionally resulted in longer execution times for certain tasks, it successfully minimized reliance on non-renewable energy sources and highlighted the potential of edge autonomics to enable green computing practices. Overall, the findings validate the importance of multi-strategy autonomics frameworks to optimize computing performance and promote sustainability objectives in future computing infrastructures.

REFERENCES

[1] M. Bertolini, D. Mezzogori, M. Neroni, and F. Zammori, “Machine Learning for industrial applications: A comprehensive literature review,” Aug. 01, 2021, *Elsevier Ltd*. doi: 10.1016/j.eswa.2021.114820.

[2] K. Alanne and S. Sierla, “An overview of machine learning applications for smart buildings,” Jan. 01, 2022, *Elsevier Ltd*. doi: 10.1016/j.scs.2021.103445.

[3] M. Parashar and S. Hariri, “Autonomic computing: An overview,” in *Lecture Notes in Computer Science*, Springer Verlag, 2005, pp. 257–269. doi: 10.1007/11527800_20.

[4] R. International Conference on Technical Informatics (9th : 2010 : Timişoara, *ICCC-CONTI 2010 : IEEE*

International Joint Conferences on Computational Cybernetics and Technical Informatics : IEEE 8th International Conference on Computational Cybernetics and 9th International Conference on Technical Informatics : Timişoara, Romania, May 27-29, 2010 : proceedings. IEEE, 2010.

[5] Z. Lv, D. Chen, R. Lou, and Q. Wang, "Intelligent edge computing based on machine learning for smart city," *Future Generation Computer Systems*, vol. 115, pp. 90–99, 2021, doi: 10.1016/j.future.2020.08.037.

[6] "Artificial Neural Networks 12.1 Introduction."

[7] J. Kufel *et al.*, "What Is Machine Learning, Artificial Neural Networks and Deep Learning?—Examples of Practical Applications in Medicine," Aug. 01, 2023, *Multidisciplinary Digital Publishing Institute (MDPI)*. doi: 10.3390/diagnostics13152582.

[8] "Era of Deep Neural Networks: A review Poonam Sharma Akansha Singh."

[9] S. S. K. Kwok, R. K. K. Yuen, and E. W. M. Lee, "An intelligent approach to assessing the effect of building occupancy on building cooling load prediction," *Build Environ*, vol. 46, no. 8, pp. 1681–1690, 2011, doi: 10.1016/j.buildenv.2011.02.008.

[10] A. Roy and S. Chakraborty, "Support vector machine in structural reliability analysis: A review," *Reliab Eng Syst Saf*, vol. 233, p. 109126, May 2023, doi: 10.1016/j.ress.2023.109126.

[11] P. De Boves Harrington, "Support Vector Machine Classification Trees," *Anal Chem*, vol. 87, no. 21, pp. 11065–11071, Nov. 2015, doi: 10.1021/acs.analchem.5b03113.

[12] "Comparative study of hyperspectral imagery classification with SVM and ensemble machine learning methods".

[13] M. Farsi, A. Daneshkhah, A. Hosseinian-Far, and H. Jahankhani, "Internet of Things Digital Twin Technologies and Smart Cities." [Online]. Available: http://www.springer.com/series/11636

[14] S. B. Aboul, E. Hassanien, D. Gupta, and A. Khanna, "Lecture Notes in Networks and Systems 56." [Online]. Available: http://www.springer.com/series/15179

[15] S. Consul and S. Williamson, "Differentially Private Random Forests for Regression and Classification." [Online]. Available: www.aaai.org

[16] B. Sethuraman and S. Niveditha, "Cerebrovascular Accident Prognosis using Supervised Machine Learning Algorithms," in *2023 World Conference on Communication & Computing (WCONF)*, IEEE, Jul. 2023, pp. 1–8. doi: 10.1109/WCONF58270.2023.10235122.

[17] N. Dudeni-Tlhone, O. Mutanga, P. Debba, and M. A. Cho, "Distinguishing Tree Species from In Situ Hyperspectral and Temporal Measurements through Ensemble Statistical Learning," *Remote Sens (Basel)*, vol. 15, no. 17, Sep. 2023, doi: 10.3390/rs15174117.

[18] Y. Qi, "Random Forest for Bioinformatics."

[19] R. Moraes, J. F. Valiati, and W. P. Gavião Neto, "Document-level sentiment classification: An empirical comparison between SVM and ANN," *Expert Syst Appl*, vol. 40, no. 2, pp. 621–633, Feb. 2013, doi: 10.1016/j.eswa.2012.07.059.

[20] A. Statnikov, L. Wang, and C. F. Aliferis, "A comprehensive comparison of random forests and support vector machines for microarray-based cancer classification," *BMC Bioinformatics*, vol. 9, Jul. 2008, doi: 10.1186/1471-2105-9-319.

[21] "Random forest classifier for remote sensing classification".

[22] B. A. Hamed, O. A. S. Ibrahim, and T. Abd El-Hafeez, "Optimizing classification efficiency with machine learning techniques for pattern matching," *J Big Data*, vol. 10, no. 1, Dec. 2023, doi: 10.1186/s40537-023-00804-6.

[23] V. N. Pawar and S. Mini, "Energy-Aware Priority-Based Task Scheduling for Edge Devices in Internet of Things," in *INDISCON 2024 - 5th IEEE India Council International Subsections Conference: Science, Technology and Society*, Institute of Electrical and Electronics Engineers Inc., 2024. doi: 10.1109/INDISCON62179.2024.10744294.

[24] G. Perin, F. Meneghello, R. Carli, L. Schenato, and M. Rossi, "EASE: Energy-Aware Job Scheduling for Vehicular Edge Networks With Renewable Energy Resources," *IEEE Transactions on Green Communications and Networking*, vol. 7, no. 1, pp. 339–353, Mar. 2023, doi: 10.1109/TGCN.2022.3199171.

[25] M. S. Aslanpour, A. N. Toosi, M. A. Cheema, and R. Gaire, "Energy-Aware Resource Scheduling for Serverless Edge Computing," in *Proceedings - 22nd IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing, CCGrid 2022*, Institute of Electrical and Electronics Engineers Inc., 2022, pp. 190–199. doi: 10.1109/CCGrid54584.2022.00028.

[26] N. Valizadeh *et al.*, "Edge Energy Orchestration."

[27] M. R. Mufid, A. Basofi, M. U. H. Al Rasyid, I. F. Rochimansyah, and A. Rokhim, "Design an MVC Model using Python for Flask Framework Development," in *IES 2019 - International Electronics Symposium: The Role of Techno-Intelligence in Creating an Open Energy System Towards Energy Democracy, Proceedings*, Institute of Electrical and Electronics Engineers Inc., Sep. 2019, pp. 214–219. doi: 10.1109/ELECSYM.2019.8901656.

[28] T. P. Bac, M. N. Tran, and Y. Kim, "Serverless Computing Approach for Deploying Machine Learning Applications in Edge Layer," in *International Conference on Information Networking*, IEEE Computer Society, 2022, pp. 396–401. doi: 10.1109/ICOIN53446.2022.9687209.

[29] International Conference on Advances in Information Mining and Management 6. 2016 Valencia *et al.*, *IMMM 2016 the Sixth International Conference on Advances in Information Mining and Management : DATASETS 2016 : the International Symposium on Designing, Validating, and Using Datasets : May 22-26, 2016, Valencia, Spain*.

[30] D. Pop, "Machine Learning and Cloud Computing: Survey of Distributed and SaaS Solutions," Mar.

2016, [Online]. Available: http://arxiv.org/abs/1603.08767

[31] V. Goyal, R. Das, and V. Bertacco, "Hardware-friendly User-specific Machine Learning for Edge Devices," *ACM Transactions on Embedded Computing Systems*, vol. 21, no. 5, Oct. 2022, doi: 10.1145/3524125.

[32] L. Zhou, M. H. Samavatian, A. Bacha, S. Majumdar, and R. Teodorescu, "Adaptive parallel execution of deep neural networks on heterogeneous edge devices," in *Proceedings of the 4th ACM/IEEE Symposium on Edge Computing, SEC 2019*, Association for Computing Machinery, Inc, Nov. 2019, pp. 195–208. doi: 10.1145/3318216.3363312.

[33] J. Beyerer, A. Maier, and O. Niggemann, "Machine Learning for Cyber Physical Systems Technologien für die intelligente Automation Technologies for Intelligent Automation." [Online]. Available: http://www.springer.com/series/13886

[34] V. Sze, Y.-H. Chen, J. Emer, A. Suleiman, and Z. Zhang, "Hardware for Machine Learning: Challenges and Opportunities," Dec. 2016, doi: 10.1109/CICC.2017.7993626.

[35] R. Sterritt and D. Bustard, "Towards an autonomic computing environment," in *Proceedings - International Workshop on Database and Expert Systems Applications, DEXA*, Institute of Electrical and Electronics Engineers Inc., 2003, pp. 694–698. doi: 10.1109/DEXA.2003.1232103.

[36] R. Van Renesse and K. P. Birman, "Autonomic Computing-A System-Wide Perspective *."

[37] P. Mittal, S.-125 Noida, A. Singhal, and A. Bansal, "A Study on Architecture of Autonomic Computing-Self Managed Systems," 2014. [Online]. Available: http://www.research.ibm.c

[38] D. Preuveneers, I. Tsingenopoulos, and W. Joosen, "Resource usage and performance trade-offs for machine learning models in smart environments," *Sensors (Switzerland)*, vol. 20, no. 4, Feb. 2020, doi: 10.3390/s20041176.

[39] by Josep Lluís Berral García Advisors and J. Torres Viñals Ricard Gavaldà Mestre, "Improved Self-management of DataCenter Systems Applying Machine Learning."

[40] A. Fanariotis, T. Orphanoudakis, K. Kotrotsios, V. Fotopoulos, G. Keramidas, and P. Karkazis, "Power Efficient Machine Learning Models Deployment on Edge IoT Devices," *Sensors*, vol. 23, no. 3, Feb. 2023, doi: 10.3390/s23031595.

[41] S. Iman Mirzadeh and H. Ghasemzadeh, "Optimal Policy for Deployment of Machine Learning Models on Energy-Bounded Systems."