Indoor-Outdoor Energy Management for Wearable IoT Devices with Conformal Prediction and Rollout

Nuzhat Yamin, *Student Member, IEEE,* and Ganapati Bhat, *Member, IEEE*

***Abstract*—Internet of Things (IoT) devices have the potential to enable a wide range of applications, including smart health and agriculture. However, they are limited by their small battery capacities. Utilizing energy harvesting is a promising approach to augment the battery life of IoT devices. However, relying solely on harvested energy is insufficient due to the stochastic nature of ambient sources. Predicting and accounting for uncertainty in the energy harvest (EH) is critical for optimal energy management (EM) in wearable IoT devices. This paper proposes a two-step uncertainty-aware EH prediction and management framework for wearable IoT devices. First, the framework employs an energy-efficient conformal prediction (CP) method to predict future EH and construct prediction intervals. Contrasting to prior CP approaches, we propose constructing the prediction intervals using a combination of residuals from previous hours and days. Second, the framework proposes a near-optimal EM approach that utilizes a rollout algorithm. The rollout algorithm efficiently simulates various energy allocation trajectories as a function of predicted EH bounds. Using results from the rollout, the proposed approach constructs energy allocation bounds that maximize application utility (quality of service) with a high probability. Evaluations using real-world energy data from ARAS and Mannheim datasets show that the proposed CP for EH prediction provides 93% coverage probability with an average width of 9.5 J and 1.9 J, respectively. Moreover, EM using the rollout algorithm provides energy allocation decisions that are within 1.9–2.9 J of the optimal with minimal overhead.**

I. Introduction

Internet of Things (IoT) devices enable a wide range of exciting applications, such as remote health monitoring and digital agriculture [1–3]. Small form factor of IoT devices typically limits the battery size and capacity. Consequently, IoT devices have short operating lifetime and require frequent recharging or battery replacement [4].

Energy harvesting and management has emerged as an effective method to augment the battery lifetime of IoT devices [4–6]. Ambient energy is not available at all times, which means that IoT devices must optimally manage the harvested energy. One of the key challenges with IoT energy management (EM) is the stochastic nature of ambient energy sources [7]. The stochastic nature necessitates development of energy prediction models that aid EM algorithms.

N. Yamin and G. Bhat are with the School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA, 99164 USA e-mail: (nuzhat.yamin@wsu.edu, ganapati.bhat@wsu.edu). This work was supported, in part, by the National Science Foundation (NSF) CAREER Award CNS-2238257.

Manuscript received March 31, 2024; revised June 16, 2024; accepted July 14, 2024. This article was presented in the International Conference on Compilers, Architectures, and Synthesis for Embedded Systems (CASES) 2024 and appears as part of the ESWEEK-TCAD special issue.

A number of recent studies [7–11] have proposed methods for prediction of energy harvest (EH). Early approaches predict future energy availability as a function of the energy values in the past [9]. Machine learning (ML) models [12] have also been proposed to predict future EH. These models only provide a point prediction of energy without accounting for the uncertainty in energy. To overcome this challenge, prior work has investigated methods to obtain prediction intervals. For instance, the mean variance estimation method (MVEM) and its variants [13, 14] predict the mean and variance of EH to build prediction intervals. However, these methods typically assume a fixed distribution for EH, which is not applicable in most real-world scenarios. Therefore, there is a need to develop models that provide accurate prediction intervals without assuming a fixed distribution for the EH.

This paper presents a conformal prediction (CP) based method to predict EH in IoT devices. CP is an emerging ML technique that provides a set of most likely predictions instead of providing a single class label in a multi-class application. CP has been also adapted to time-series applications to provide prediction intervals [15]. However, prior CP methods for energy prediction suffer from *two limitations*: 1) they rely on a large training and calibration dataset. Specifically, for each new prediction instance, the IoT device must calculate conformity scores over the calibration dataset, leading to high overhead, 2) they consider a single-dimensional residual vector to construct the prediction intervals. That is, to predict EH at any interval of time, such as an hour, the CP methods use the errors from past hours only without considering the behavior of errors in the same interval in past days. This leads to lower coverage probability and higher prediction widths.

We propose to overcome the above challenges of CP in energy predictions using two *key* insights. First, we observe that maintaining a longer history of calibration data is not useful for EH prediction. Specifically, as user patterns or seasons change, residuals from past seasons have lower relevance. Based on this, we propose to use a sliding window of residuals to construct prediction intervals. Secondly, prior studies have shown that EH patterns follow a strong daily pattern with day-to-day variations [16]. Using this insight, we propose to include residuals from the same interval in past days in addition to past hours to construct prediction intervals. Including errors from past hours and days allows the proposed CP approach to improve coverage and prediction widths.

Once the energy is predicted, it must be used in an EM algorithm to optimally allocate energy to the system. Energy harvesting systems typically aim to achieve energy neutral operation (ENO), whereby the energy consumed in any given

horizon (e.g., a day) is equal to the energy harvested during the horizon [8]. Achieving ENO is challenging since we must make energy allocation decisions while accounting for uncertainty in future EH and maximizing application performance. To this end, we propose a rollout-based algorithm for obtaining energy allocations that account for uncertainty in future EH [17]. We start with an EM problem formulation that maximizes application utility in a given horizon. The proposed approach uses a 24-hour horizon since human activities and EH typically follow a daily pattern [16] with variations across users and time. The EM formulation first takes expected values of EH for 24 hours at the beginning of the horizon to obtain initial energy allocations. However, these allocations may change due to variations in EH. To this end, we propose to perform rollout using the EH bounds provided by CP. Specifically, the rollout evaluates multiple trajectories of possible EH values and allocations to calculate future application utility. The algorithm then provides a range of possible energy allocations that maximize application utility. We also propose a gradient descent-based approach to optimize the rollout phase to minimize the number of trajectory evaluations. Overall, the EM algorithm provides energy allocations that account for uncertainty in EH while maximizing application utility.

We evaluate the proposed CP and EM approaches on ARAS and Mannheim activity datasets [18, 19]. Experiments show that energy predictions with CP achieve greater than 90% coverage with less than 10 J width. Compared to MVEM [13], CP achieves higher coverage and lower widths when the magnitude of the EH is low. Achieving higher coverage and accuracy in low energy intervals is especially important because it can have a significant impact on the IoT device battery life. The CP model is implemented on the Texas Instruments (TI)-CC2652R microcontroller [20] to characterize the runtime overhead. Our measurements show that CP takes 217 ms to obtain the predictions with 3 mJ energy overhead, which is less than 0.14% of typical EH value of Mannheim data. Next, the EM algorithm provides energy allocations that are within 3 J of an optimal Oracle that uses actual values of EH. We also compare the rollout-based EM algorithm against a baseline approach that uses expected values of EH for energy allocations. The comparison shows that the proposed rollout-based approach achieves 71% and 46% higher utility than the baseline for ARAS and Mannheim datasets, respectively. Comparisons against a reinforcement learning approach show that the rollout achieves 15–100% better utility. Implementation on the TI-CC2652R device shows that the rollout algorithm consumes about 390 mJ, on average. In summary, this paper makes the following contributions:

- A conformal prediction-based method to predict and construct prediction intervals for future EH,
- A sliding window of two-dimensional residuals to adaptively construct prediction intervals,
- Rollout-based energy management algorithm that accounts for uncertainty in EH with minimal overhead,
- Experimental validation of the proposed CP and EM approaches using real-world energy data with diverse weather patterns and variable human activities.

Rest of this paper is organized as follows. Section II reviews prior related work. Section III provides background on the EM problem and preliminaries on CP. Section IV introduces the proposed CP approach, while Section V provides details on the rollout-based EM algorithm. We provide the experimental results in Section VI, discuss potential impact in Section VII, and conclude the paper in Section VIII.

II. Related Work

Prediction of future EH is an important component of EM algorithms. Several approaches have been proposed to predict EH [7–9, 12]. The work in [8] proposes one of the first EH prediction approaches that maintains an exponentially-weighted moving average filter to maintain predictions at each interval. However, moving average predictions have low accuracy when there are sudden changes in weather. Proenergy [9] is another approach that uses stored profiles of typical days to predict future EH. However, these approaches only provide point predictions without accounting for the uncertainty in EH.

Ambient energy sources are highly stochastic due to changes in weather or user activity patterns. EH prediction algorithms must account for the uncertainty to improve EM decisions. Recent research has proposed several approaches for uncertainty prediction in EH [11, 13, 21]. For instance, MVEM [13] directly predicts the mean and variance using two neural networks. However, MVEM implicitly assumes Gaussian distribution, which is not true for EH from wearable devices. The lower-upper bound estimation (LUBE) method [21] aims to overcome this limitation by directly predicting upper and lower bounds of energy. However, LUBE is challenging to train and relies on proper choice of cost function, which might vary based on user-specific activities and location. Overall, most prior methods for uncertainty estimation assume standard EH distribution, which is not feasible in real-world scenarios. In contrast to prior approaches, we propose a distribution-free CP algorithm for EH prediction.

EH predictions must be used to obtain effective EM decisions that maximize performance of the device while maintaining ENO. A number algorithms [1, 8, 22–24] have been proposed to perform EM while taking EH predictions into account. For instance, Kansal et al. utilize a exponentially-weighted moving average method to predict future EH and employ linear programming to optimize energy allocation [8]. The authors in [22] utilize the deviation between expected EH and actual EH in a dynamic optimization formulation to obtain EM decisions. While these approaches can be effective, they only use point predictions of EH and provide single point decisions for EM, thus not accounting for the uncertainty. Recently, ML and reinforcement learning (RL) approaches [25, 26] have been proposed to predict future EH and obtain EM decisions. RL methods can be useful to account for any deviation in EH during runtime, but it often requires a large number of trajectories to learn effective policies, leading to higher overhead. Moreover, stochasticity in future EH may lead to errors in the learned model. In contrast to the prior approaches, we propose a low-overhead gradient descent-based rollout algorithm that utilizes CP prediction intervals and constructs energy allocation bounds for EM.

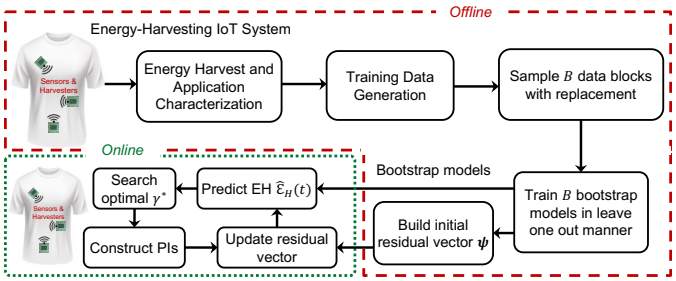

Fig. 1: Overview of the proposed conformal prediction algorithm.

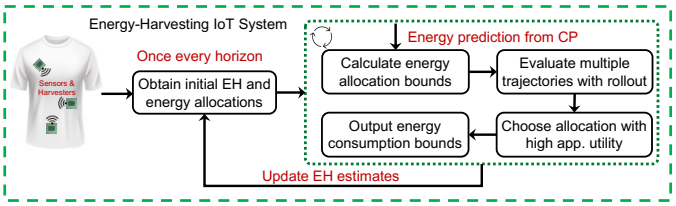

Fig. 2: Overview of the proposed rollout algorithm.

III. Problem Setup and Background

A. Problem Setup and Overview

We consider a system with multiple energy harvesters and sensors mounted on a user's body, as shown in Figure 1. The energy harvesters scavenge energy from the environment. The harvested energy is routed through an EM algorithm that decides the energy consumption in each interval. Optimal decision-making requires knowledge of future EH, which is stochastic in nature. Therefore, EM algorithms must be provided with accurate predictions of future EH.

Let us consider that a day is divided into T equal intervals for EM decisions. The stochastic EH in an interval t $(1 \leq t \leq T)$ is given by $\mathcal{E}_H(t)$. This energy must be accurately predicted to aid in the EM. Point predictions are insufficient due to inherent stochasticity in EH. Therefore, our goal is to obtain lower and upper bounds for future energy $[\hat{\mathcal{E}}_H^{ub}(t), \hat{\mathcal{E}}_H^{lb}(t)]$ given a set of features $X(t)$. We want to ensure that the true EH lies in the prediction interval with a probability of at least (1-α), where $\alpha \in (0,1)$ is a miscoverage level. Given this, one can either obtain *conditionally* valid intervals where the conditional probability $P\{\mathcal{E}_H(t) \in [\hat{\mathcal{E}}_H^{ub}(t), \hat{\mathcal{E}}_H^{lb}(t)] | X(t)\} \geq (1-\alpha)$ or *marginally* valid intervals where $P\{\mathcal{E}_H(t) \in [\hat{\mathcal{E}}_H^{ub}(t), \hat{\mathcal{E}}_H^{lb}(t)]\} \geq (1-\alpha)$. In this paper, we obtain marginally valid intervals for EH.

The EM algorithm starts by performing an initial energy allocation using expected values of EH at the beginning of each 24-hour horizon, as shown in Figure 2. The initial allocations are not optimal since future EH will deviate from the expected values. The energy allocations must be adjusted at the beginning of each interval t to account for uncertainty in EH. To this end, outputs of the energy prediction block are provided as inputs to the rollout-based EM algorithm.

The rollout algorithm evaluates multiple trajectories for the system as a function of predicted EH and allocation values. The rollout trajectories provide us with overall application utility for each EH and allocation combination that satisfies system energy constraints. Since our goal is to maximize utility with ENO, the algorithm chooses EH and allocation combinations that provide more than 90^{th} percentile of the utility as the energy allocation range. The rollout is repeated at the beginning of each interval t with new EH predictions.

B. Conformal Prediction Preliminaries

CP has emerged as an effective method to quantify uncertainty in ML models by providing prediction sets instead of single predictions [27]. For time series or regression problems, CP algorithms provide a prediction interval that contains the true value with a given probability $1-\alpha$. Existing CP methods can be classified into three categories: split CP, full CP, and Jackknife. We provide brief descriptions of each method next.

Split Conformal Prediction: Split CP divides data into two sets: training and calibration [27]. The training set is used to obtain a base prediction model. The calibration set is used to evaluate conformity of new test inputs to prior data. Split CP suffers from two key limitations for time-series data on IoT devices: 1) the calibration set must be large enough to provide useful intervals, thus increasing the memory overhead, 2) evaluating conformity scores can add computational overhead. Split CP can also lead to lower accuracy since the base model must be independent of the calibration set [28].

Full CP: Full CP aims to avoid accuracy loss by using the entire training dataset to build prediction models and intervals. However, full CP has high computational overhead since it trains multiple models for a range of output values [28].

Jackknife+: Jackknife+ [28] aims to balance split and full CP by employing leave-one-out training for base predictors. Jackknife+ trains a model corresponding to each data point while leaving it out of the training data. Due to the leave-one-out models, Jackknife+ has higher accuracy compared to split CP and lower complexity when compared to full CP. However, Jackknife+ needs retraining as new data points become available, thus making it computationally expensive.

Key Challenges: In addition to the above challenges, CP methods typically assume that training data are exchangeable and independent [15]. However, time-series energy data are generally not exchangeable since different sequences of EH can change the joint probability distribution. Therefore, CP methods for EH must provide prediction intervals in non-exchangeable settings to be effective for wearable devices.

IV. Proposed Conformal Prediction Algorithm

A. Overview of the Proposed Solution

We adapt the recently proposed Ensemble batch Prediction Interval (EnbPI) approach [15] to solve the above challenges and construct accurate prediction intervals. The *key* idea behind EnbPI is to train an ensemble of base predictions using bagging of training data and using a weighted average of predictions. Predictions from the ensemble of predictors are used with a sliding window of conformity scores for past data points. The sliding window is updated with new scores at the end of an interval when the EH is observed. Using an ensemble of models allows us to use entire training data to train the models while also avoiding overfitting models to any particular set of data. Similarly, sliding window of residuals

enables the approach to eliminate the need to re-train models by adapting prediction widths to recent data distributions.

EnbPI algorithm requires a large window of residuals to obtain accurate predictions. In case of EH prediction, not all residuals in the history are useful due to changes in EH potential throughout the day. For instance, residuals recorded for early hours of past days are not as relevant when making predictions for mid-day hours. Using this insight, we propose to use a two-dimensional residual vector that includes errors for the same interval in past days in addition to past intervals. This helps reduce the residual vector size or improve accuracy.

B. Bootstrap Models for Point Prediction

The first step of the proposed CP algorithm uses supervised bootstrap models as base predictors of EH. To capture the daily behavior of energy, we divide each day into T equal-length intervals and predict the future energy at the beginning of each interval t. We assume that energy observations for D_{train} days are available to train the bootstrap models. Using the training data, we obtain supervised feature and label pairs $(X(t), \mathcal{E}_H(t))$ for each observation in the training data. The feature set consists of energy observations for the past 24 hours at any given time t while the label is the actual energy.

After obtaining the training feature set, we train B bootstrap models with subsets of training data so that we do not overfit the models. We use the block bootstrap method with M non-overlapping blocks since it is known to work well for dependent data [15]. The block bootstrap method splits the training data into M non-overlapping blocks each of length $\lfloor (D_{train} \times T)/M \rfloor$. The training data for each model b_j is sampled randomly from M blocks with replacement.

Bootstrap Model Weights: At runtime, each bootstrap model provides a prediction of the EH in a given interval t. These individual predictions must be aggregated to obtain a single prediction. We can either use equal weights $1/B$ for each model or assign custom weights to each model based on data used to train each bootstrap model. In this paper, we assign custom weights to each model using the equation below:

$$w_j = \frac{\sum_{i=1}^{D_N} w_{d_i}}{D_N} \quad 1 \leq j \leq B \tag{1}$$

where, $D_N = D_{train} \times T$, total number of training samples, and w_{d_i} is the weight that we assign to each training sample. This weight is given by,

$$w_{d_i} = \begin{cases} \frac{1}{B-b}, & \text{if used in training } b < B \text{ models} \\ 0, & \text{if used in training all } B \text{ models} \end{cases} \tag{2}$$

The goal of using weights on each training sample is to assign higher weights to models that have been fitted using fewer training samples. Using model weights, the base energy prediction for an interval t is obtained as:

$$\hat{\mathcal{E}}_H(t) = \sum_{j=1}^{B} w_j \hat{\mathcal{E}}_{H,j}(t) \tag{3}$$

where $\hat{\mathcal{E}}_{H,j}(t)$ is prediction from j^{th} bootstrap model.

C. Prediction Interval Construction

CP methods keep a history of nonconformity scores to construct prediction intervals. Commonly used nonconformity scores include mean absolute deviation, absolute deviation, deviation, or maximum deviation [15]. Among these, we choose deviation as the nonconformity score. Since deviation captures prediction error, we also refer to it as a residual. For each prediction, we calculate the residual as follows:

$$\epsilon_t = \mathcal{E}_H(t) - \hat{\mathcal{E}}_H(t) \tag{4}$$

We maintain a two-dimensional residual vector that consists of errors from past hours and days. We can represent the residual vector at a given interval t on day d as:

$$\boldsymbol{\psi} = \{[\epsilon^d_{t-R}, \epsilon^d_{t-R+1}, \cdots, \epsilon^d_{t-1}], [\epsilon^{d-S}_t, \cdots, \epsilon^{d-1}_t]\} \tag{5}$$

where R is the number of residuals for past intervals immediately preceding the current interval. Similarly, S is the number of residuals from past days for the same interval t. The first part of the residual vector captures errors in the past hours without considering the interval, while the second part captures errors in past days for the same interval. Using the residual vector, we obtain the upper and lower bounds as:

$$\hat{\mathcal{E}}^{lb}_H(t) = \hat{\mathcal{E}}_H(t) + \alpha \text{ quantile of } \boldsymbol{\psi} \tag{6}$$

$$\hat{\mathcal{E}}^{ub}_H(t) = \hat{\mathcal{E}}_H(t) + (1-\alpha) \text{ quantile of } \boldsymbol{\psi} \tag{7}$$

The prediction width is given by directly using the α and $1-\alpha$ quantiles of the residual vector. However, this may not be the smallest width for a given prediction. Obtaining tight prediction intervals is crucial since large prediction widths do not offer useful information to EM algorithms. Therefore, we perform a local search around quantiles following the approach outlined in the EnbPI algorithm. The width is minimized by defining a factor γ for local search as follows:

$$\gamma^* = \underset{\gamma \in [0,\alpha]}{\arg\min}((1-\alpha+\gamma) \text{ quant. of } \boldsymbol{\psi} - \gamma \text{ quant. of } \boldsymbol{\psi}) \tag{8}$$

where γ^* is the value for γ that gives the minimum width. Equation 8 searches through multiple options for prediction width and chooses the gamma that specifies minimum width. Using γ^*, we obtain the final upper and lower bounds as:

$$\hat{\mathcal{E}}^{lb}_H(t) = \hat{\mathcal{E}}_H(t) + \gamma^* \text{ quantile of } \boldsymbol{\psi} \tag{9}$$

$$\hat{\mathcal{E}}^{ub}_H(t) = \hat{\mathcal{E}}_H(t) + (1-\alpha+\gamma^*) \text{ quantile of } \boldsymbol{\psi} \tag{10}$$

Finally, at the end of each prediction interval, we obtain the actual EH value and update the residual vector by removing oldest entries. The sliding window enables the proposed CP algorithm to adapt to changes in the environment.

D. Conformal Prediction Summary

Algorithm 1 summarizes the overall CP algorithm for EH prediction at runtime. The inputs to the algorithm include trained bootstrap models, required confidence interval, and initial residual vector $\boldsymbol{\psi}$. Given these inputs, the algorithm predicts EH values using each bootstrap model at the beginning of each interval t. The predictions are then aggregated using bootstrap model weights in line 5. Next, we find the optimal value of γ by performing a local search over prediction widths using Equation 8. The optimal γ^* is used to obtain lower and upper bounds for energy in the current interval t. Finally, we update the residual vector by removing oldest entries ϵ^d_{t-R} and ϵ^{d-S}_t, while appending it with new residual $\mathcal{E}_H(t) - \hat{\mathcal{E}}_H(t)$. The upper and lower bounds are returned to the EM algorithm.

Algorithm 1: CP for Indoor-Outdoor EH Prediction

Input: Bootstrap models, Confidence interval α, Initial residual vector $\boldsymbol{\psi}$
for $d = 1, 2, \dots$ **do**
  for $t = 1, 2, \dots, T$ **do**
    Predict EH values $\hat{\mathcal{E}}_H(t)$ using bootstrap models
    Compute γ^* using Equation 8
    $\hat{\mathcal{E}}_H^{lb}(t) = \hat{\mathcal{E}}_H(t) + \gamma^*$ quantile of $\boldsymbol{\psi}$
    $\hat{\mathcal{E}}_H^{ub}(t) = \hat{\mathcal{E}}_H(t) + (1 - \alpha + \gamma^*)$ quantile of $\boldsymbol{\psi}$
    Update $\boldsymbol{\psi}$ by discarding oldest entries and appending $\mathcal{E}_H(t) - \hat{\mathcal{E}}_H(t)$
  end
end
return $\hat{\mathcal{E}}_H^{lb}(t)$ and $\hat{\mathcal{E}}_H^{ub}(t)$

V. Proposed Energy Management Algorithm

A. Problem Formulation

We utilize a commonly used problem formulation for ENO in IoT devices for energy management [8]. The optimization problem can be written as follows:

$$\max \quad U(\mathbf{E_c}) = \sum_{t=0}^{T-1} \beta^t \ln\left(\frac{E_c(t)}{M_E}\right) \quad \text{s. t.} \tag{11}$$

$$E_B(t+1) = E_B(t) + \eta \mathcal{E}_H(t) - E_c(t),\ 0 \le t \le T-1 \tag{12}$$

$$E_B(t+1) \ge E_{min} \quad 0 \le t \le T-1 \tag{13}$$

$$E_B(T) \ge E_{target} \tag{14}$$

where $\mathbf{E_c}$ denotes a vector of energy allocation $E_c(t)$ over all the intervals, β is a discount factor, M_E is the minimum energy allocation for non-negative utility, and $E_B(t)$ is the battery level at the beginning of interval t. $\mathcal{E}_H(t)$ is the stochastic EH in interval t, η is the EH efficiency, E_{min} is the minimum battery level, and E_{target} specifies the battery level at the end of a horizon. Next, we provide details on each component.

Objective Function: The EM objective is to maximize the total utility of the application over the horizon, as shown in Equation 11. We use a logarithmic utility function to capture application behavior with increasing energy allocation. Intuitively, the application can provide a higher quality of service with higher energy. For instance, IoT devices can operate sensors at higher sampling frequencies to improve data quality. At the same time, increase in quality of service diminishes after a maximum energy consumption for the device. Logarithmic functions capture this behavior well since they have higher slope at lower energy values and the slope reduces with increase in energy. We also scale the energy consumption in the utility function by M_E since a minimum level of energy is required for non-zero utility.

System Dynamics and Constraints: Equations 12–14 describe the battery energy dynamics and constraints for ENO. Specifically, Equation 12 specifies that battery energy at the beginning of an interval $t+1$ is given by battery in the previous interval $E_B(t)$, stochastic EH $\mathcal{E}_H(t)$, and energy consumption $E_c(t)$. Next, we would like to maintain a minimum level of energy in the battery to account for any unexpected events. The constraint in Equation 13 ensures that battery in the device at all intervals is greater than or equal to E_{min}. Finally, the device must maintain ENO target energy E_{target} at the end of each horizon to enable uninterrupted operation.

Optimization Variables and Challenges: The optimization variables in the EM problem are energy allocations for each interval. It is challenging to solve the EM problem since optimal solution requires knowledge of future EH, which is not feasible in practice. Moreover, any decisions made with expected values for future EH must be reevaluated at each interval to account for deviations from the expected EH.

Solution Approach: The EM problem in Equations 11–14 is a convex optimization problem. We can utilize convex optimization solvers to obtain solutions to the problem. At the same time, using convex solvers on IoT devices can lead to high overhead. Therefore, we propose to use the dual problem formulation and an iterative algorithm that allows us to trade-off solution precision and overhead [29]. We provide description of the dual problem and iterative algorithm next.

The Lagrangian of the EM problem is:

$$\begin{aligned}\mathcal{L} = U(\mathbf{E_c}) + \sum_{t=0}^{T-2} \lambda_t (E_B(t) + \eta \mathcal{E}_H(t) - E_c(t) \\ - E_{min}) + \lambda_T (E_B(T) - E_{target})\end{aligned} \tag{15}$$

where $\mathcal{L}$ is the Lagrangian and λ_t are the dual variables. Since the problem is convex in nature, we can obtain the optimal solution using the Karush-Kunh-Tucker (KKT) conditions [29]. The KKT conditions can be written as:

$$\frac{\partial L}{\partial E_c(t)} : \beta^t \left[\frac{M_E}{E_c(t)} - \lambda_t\right] = 0 \quad 0 \le t \le T-1 \tag{16}$$

$$\lambda_t (E_B(t) + \eta \mathcal{E}_H(t) - E_c(t) - E_{min}) = 0 \quad 0 \le t \le T-1 \tag{17}$$

$$\lambda_T (E^B(T) - E_{target}) = 0 \tag{18}$$

Since our goal is to enable an efficient runtime solution, we utilize the iterative gradient projection (IGP) algorithm proposed in [30] to solve the above system of equations.

Algorithm 2 shows the procedure for obtaining energy allocation with a given set of EH values. The algorithm takes EH values, required tolerance levels, step size, and maximum iterations as inputs. The algorithm starts by choosing random values for energy allocations and dual variables λ_t. Then, in each iteration of the algorithm, we perform the following steps: 1) assign new energy allocations using Equation 16, 2) set new values for lambda using the step size and Equations 17 and 18. The optimization continues until the maximum iterations have been reached or change in λ is less than the threshold.

B. Rollout Algorithm for Runtime Energy Allocation

The IGP algorithm is useful to obtain energy allocations when the EH values are deterministic. However, our goal in this paper is to account for uncertainty in future EH and provide energy consumption bounds for the device. We also aim to minimize the number of minimum and target energy constraint violations. To this end, we propose to use a rollout-based algorithm to obtain allocation bounds.

Rollout is a dynamic programming approach that explores multiple trajectories for a given decision making problem [17].

Algorithm 2: Iterative gradient projection algorithm

Input: $\mathcal{E}_H$, tolerance, Step Size δ, and $\text{Iter}_{\max}$
$E_c(t) \leftarrow rand()$ $\quad 0 \le t \le T-1$
$\lambda_t, \lambda_t^* \leftarrow rand()$ $\quad 0 \le t \le T$
while $\sum_{t=0}^{T}(\lambda_t^* - \lambda_t)^2 >$ tolerance *and* iter $\le \text{Iter}_{\max}$ **do**
  $\lambda_t \leftarrow \lambda_t^*$ $\quad 0 \le t \le T-1$
  $E_c(t) \leftarrow \frac{\beta^t \alpha M_E}{\lambda_t}$ $\quad 0 \le t \le T-1$ (follows from Equation 16)
  $\lambda_t^* \leftarrow \max$
    $\{\lambda_t + \delta(E_c(t) - E_B(t) - \eta\mathcal{E}_H(t) + E_{min}), 0\}$ $\quad 0 \le t \le T-1$
  $\lambda_T^* \leftarrow \max\{(E_{target} - E^B(T)), 0\}$
end
return $E_c(t)$

Rollout algorithms use a base policy to perform trajectory evaluations with low overhead. At the end of rollout, decisions that maximize the objective are chosen for implementation. Key idea behind rollout is that it allows us to obtain improved results compared to the underlying base policy [17]. Following this insight, we use rollout to account for uncertainty in EH and utilize IGP as the base policy.

Overview of Rollout Algorithm: Figure 2 shows an overview of the proposed rollout algorithm for EM. We start with an initial energy allocation with expected values of EH at the beginning of each horizon. The initial energy allocations provide us with a baseline for performing trajectory evaluations with rollout. Moreover, the expected EH values aid in evaluating rollout trajectories using Algorithm 2 since CP provides EH bounds for the immediate future interval.

After obtaining the initial allocation, we perform runtime rollout at the beginning of each interval. The first step is to obtain a finite number of EH bins from the upper and lower bounds provided by CP. We divide the EH into finite bins to avoid a large number of trajectory evaluations. Then, for each EH bin we obtain a possible range of energy allocations. Next, we rollout the complete trajectory for the horizon using the IGP algorithm to obtain application utility with the current allocation. We discard any trajectory that violates battery energy constraints. At the end of rollout, we obtain the potential utility for each pair of EH and energy allocation. Finally, EH and energy allocation pairs that provide highest 10 percentile of utility are used to obtain energy allocation bounds for the current interval. We provide details of each step in the following sections.

1) Initial Energy Allocation: The first step in the proposed rollout-based EM algorithm is to obtain initial energy allocations at the beginning of each horizon. The initial energy allocations provide a baseline for trajectory evaluations in each interval. Obtaining initial energy allocations requires expected values of EH in the entire horizon for use in the IGP algorithm. To this end, we utilize the mean EH over the past three days in each interval as the expected EH $\hat{\mathcal{E}}_H^{init}$. Using the mean of three days allows us to obtain a low-overhead estimation of future energy and then use CP for fine-grained predictions. The expected EH is used in Algorithm 2 to obtain the initial energy allocations as $\hat{E}_c(t)$ $(0 \le t \le T-1)$.

2) Runtime Rollout Algorithm: Actual EH will deviate from the expected EH at runtime. To this end, we leverage the proposed CP algorithm to obtain EH bounds for the next interval before making energy allocation decisions. Each possible EH value in the predicted bounds leads to different energy allocations and utilities. We must also explore multiple values of energy allocations at each interval to account for uncertainty in application behavior. To this end, we calculate a range of possible energy allocations using rollout.

Energy Allocation Bounds for Rollout: First, let us consider that the EH bounds are divided into multiple bins with a width of τ_{EH}. Now, consider that the current EH value under evaluation is represented with $\tilde{\mathcal{E}}_H(t)$ $(\hat{\mathcal{E}}_H^{lb}(t) \le \tilde{\mathcal{E}}_H(t) \le \hat{\mathcal{E}}_H^{ub}(t))$. This EH may differ from the initial EH estimation that was used to obtain initial allocations. Consequently, the difference in energy must be accounted for in future intervals. We can represent the difference in EH as follows:

$$\Delta_{\mathcal{E}_H}(t) = \tilde{\mathcal{E}}_H(t) - \hat{\mathcal{E}}_H^{init}(t) \tag{19}$$

where $\Delta_{\mathcal{E}_H}(t)$ is the difference in EH values. If $\Delta_{\mathcal{E}_H}(t) > 0$, it means that there is a surplus in energy, which can be used in future intervals. Next, the EH value used in the previous interval $t-1$ for deciding the allocation may also deviate from the actual energy harvest. The difference in EH values is:

$$\Delta_{\mathcal{E}_H}(t-1) = \hat{\mathcal{E}}_H(t-1) - \mathcal{E}_H(t-1) \tag{20}$$

where $\hat{\mathcal{E}}_H(t-1)$ is the EH used in the previous interval and $\mathcal{E}_H(t-1)$ is the actual EH. The energy allocated to the device in prior intervals may deviate from the initial energy allocations, leading to changes in allocation for future intervals to maintain ENO. The difference in energy allocations is:

$$\Delta_{E_c}(t-1) = E_c(t-1) - E_c^{init}(t-1) \tag{21}$$

where $E_c(t-1)$ is the energy allocation chosen for the previous interval and E_c^{init} is the initial energy allocation. $\Delta_{E_c}(t-1) > 0$ means that the system consumed higher levels of energy in the past interval that need to be accounted for in the future. Combining, we can obtain the total difference in energy as:

$$\Delta(t) = \Delta_{\mathcal{E}_H}(t) + \Delta_{\mathcal{E}_H}(t-1) - \Delta_{E_c}(t-1) \tag{22}$$

The energy difference is used to obtain the energy allocation bounds for the rollout as:

$$E_c^{lb}(t), E_c^{ub}(t) = \frac{E_c(t-1) + E_c^{init}(t)}{2} \pm \Delta(t) \tag{23}$$

where $E_c^{lb}(t)$ and $E_c^{ub}(t)$ are the lower and upper bounds, respectively. The bounds use an average of initial energy allocation and actual consumption in the prior interval to account for deviations in the application, while noting that using initial energy allocations also give comparable results.

After obtaining the energy allocation bounds, we divide it into a finite number of bins for rollout. Specifically, we set the energy allocation for the current interval to the candidate energy allocation $\hat{E}_c(t)$. Then, we use the IGP algorithm to obtain energy allocation for future intervals. The optimization uses the current EH prediction value $\tilde{\mathcal{E}}_H(t)$ for the current interval, while the initial energy allocation is used for future intervals. The optimization algorithm is executed for $T-t$ intervals since all intervals before t have already been executed, as illustrated in Figure 3. The cumulative utility obtained from the rollout is stored in a table with candidate EH as rows and allocations as columns. If any of the ENO constraints are violated, we set the utility to negative infinity.

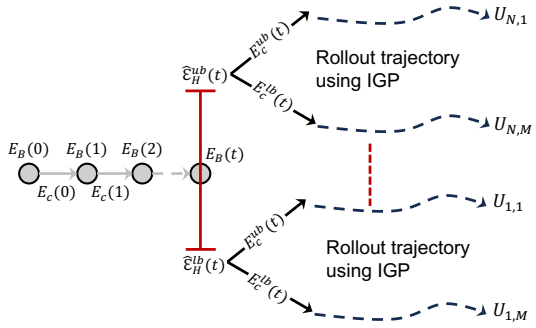

Fig. 3: Illustration of the rollout process.

The utility table provides the potential quality of decisions taken with each EH and allocation combination. Our goal is to obtain energy allocation bounds for the device such that overall application utility is maximized. To this end, we choose EH and allocation pairs that provide utility in the top 10 percentile to obtain the final energy allocation bounds. The energy allocation bounds are provided to the device and it consumes energy $E_c(t)$ that lies within the bounds. In summary, the rollout algorithm accounts for uncertainty in EH and device behavior by providing energy consumption bounds that satisfy battery energy constraints.

3) Rollout Optimization to Minimize Overhead: The rollout algorithm evaluates all possible pairs of EH prediction and energy allocation bins to future utility. This can result in a large number of evaluations if we have large widths for EH or allocations. Therefore, we use structure of the EM optimization problem and utility function to minimize the number of evaluations. Specifically, for a given EH prediction, we can make following inferences about energy allocations:

- If the lowest energy allocation in the range violates any constraints, then all other allocations will violate the constraints. This is because total energy available for remaining hours is not sufficient to maintain ENO and any increase in the allocation will further reduce energy available for future intervals. We can use this insight to discard all higher energy allocation values.
- Cumulative utility from the rollout is a convex function for valid energy allocation candidate values. This is because the lowest value of allocation will obtain lower utility for the current interval while increasing the utility for future intervals. As we increase the allocation for the current interval, the utility increases for the current interval with small decreases for future intervals, leading to increase in the total utility. This trend continues until utility of the current intervals becomes the dominant factor. At this point, the cumulative utility starts to fall. Based on this insight, we employ gradient descent to obtain the energy allocations with the highest utility.

Rollout Algorithm Complexity: The primary complexity in the rollout algorithm consists of obtaining the solution to the EM problem for each candidate EH value and corresponding allocation bounds. The EM problem solution consists of obtaining the inverse gradient of each time interval. This results in T steps at most. Similarly, updating the values of λ^* takes T steps. Considering a maximum of $\mathrm{Iter}_{\max}$, each EM solution results in worst case complexity of $O(\mathrm{Iter}_{\max}\mathrm{T})$. Next, assuming N_{EH} and N_{E_c} bins for candidate EH and allocation values, the overall worse-case complexity for the rollout is $O(\mathrm{Iter}_{\max}N_{E_c}N_{EH}T)$. We note that this is the worst case complexity and the complexity is significantly lower in practice due to the optimizations in Algorithm 3.

Algorithm 3: Optimized rollout algorithm

Input: $\hat{\mathcal{E}}_H^{lb}(t)$, $\hat{\mathcal{E}}_H^{ub}(t)$, $E_c(t-1)$, $\hat{\mathcal{E}}_H^{init}(t)$
for $\tilde{\mathcal{E}}_H(t)$ *in* $[\hat{\mathcal{E}}_H^{lb}(t), \hat{\mathcal{E}}_H^{ub}(t)]$ **do**
  $\Delta(t) \leftarrow$ Calculate $\Delta(t)$ using Equations 19–21
  $E_c^{lb}(t), E_c^{ub}(t) \leftarrow \frac{E_c(t-1)+E_c^{init}(t)}{2} \pm \Delta(t)$
  for $\hat{E}_c(t)$ *in* $[E_c^{lb}(t), E_c^{ub}(t)]$ **do**
    $U \leftarrow$ Solve EM problem with $\hat{E}_c(t)$ and initial EH for remaining intervals
    if *Constraint violation* **then**
      Discard $\hat{E}_c(t)$
    end
    Choose next $\hat{E}_c(t)$ using gradient descent
  end
end
return: Allocation bounds with highest 10 percentile utilities

4) Rollout Algorithm Summary: Algorithm 3 summarizes the overall rollout algorithm at each interval t. The inputs to the algorithm include current EH predictions from CP, initial EH prediction, and energy consumption in the previous interval $E_c(t-1)$. With these inputs, we first obtain the EH bins for rollout. For each possible value of EH, the algorithm first calculates $\Delta(t)$ using Equations 19–21. Then, using $\Delta(t)$ the algorithm obtains bounds for possible energy allocation values. The allocation bounds are then used to evaluate the potential utility of each energy allocation trajectory. The algorithm utilizes optimized rollout to minimize the number of evaluations at runtime. Finally, EH and allocation combinations that provide highest 10 percentile utilities are returned as possible energy allocation bounds.

VI. Experimental Evaluation

A. Experimental Setup

Wearable IoT Device Model: We consider a wearable system with five energy harvesters. Specifically, we include piezoelectric sensors on the knees and elbows (a total of four locations) to harvest energy from user motion. The system also includes an SP3-37 [31] flexible PV-cell to obtain energy from light.

Datasets and Data Pre-processing: Our goal is to provide uncertainty-aware predictions of EH for both indoor and outdoor scenarios. Accurate evaluations of the proposed CP approach require datasets that provide actual energy values in both scenarios. However, to the best of our knowledge, there are no datasets that provide user-specific energy measurements in both indoor and outdoor scenarios. Moreover, energy from harvesters mounted on the body is a strong function of the user's activity. Therefore, we use activity traces from two publicly available datasets: ARAS [18] and Mannheim [19] for our evaluation. The ARAS dataset contains activity data collected from two houses with four users over 30 days, while the Mannheim dataset contains data from seven users recorded

for two weeks. We utilize data from six users in the Mannheim dataset since data for one of the users is incomplete.

Data Processing Steps: The data provided by ARAS and Mannheim datasets includes fine-grained activity information with multiple activity labels having similar EH potential. For instance, eating breakfast and lunch in the ARAS dataset have similar activity levels. We merge labels with similar EH potential in both datasets to obtain a simplified set of activities. We obtain a total of nine activity labels for the ARAS dataset: {Active, cook, eat, clean, sleep, leisure, work, care, and other} and four activities for the Mannheim dataset: {shopping, transport, cycling, and movement}.

Next, we use models for light and motion energy harvesters to obtain ground truth EH values by setting intensities to each activity. The potential EH available from the piezoelectric harvesters is a function of the user motion during each activity. Consequently, we set intensities for legs and arms of users for each activity. Starting with a baseline intensity of 1 for exercise, we set the intensity for the 'active' activity as 0.5, while the intensity for sleeping is zero. The activity intensities are then combined with a baseline power of 13 μW [6] available from piezoelectric harvesters.

Energy available from light is a function of the location and activity. The locations are classified as either bedroom, office, kitchen, or bathroom when the user is indoors and outside or transport when a user is outdoors. Following this setup, we obtain typical irradiance in each of these rooms from the literature [5]. When the user is outdoors, we utilize the solar irradiation measured by National Renewable Energy Laboratory (NREL) [32] in Golden, CO. The irradiance is combined with time of the day to calculate the light energy using the area and I-V characteristics of the PV-cell.

ARAS and Mannheim datasets provide user activities for one month or two weeks, which is not sufficient for evaluation since our goal is to obtain accuracy evaluations over different seasons and time of year. Therefore, we extend both datasets for six years from 2015–2020 by shuffling and augmenting activity data to the original datasets.

Data Variations: The proposed approach uses 24-hour horizon for energy decisions since prior research has shown that user activities and energy follow 24-hour routines with variation [16]. At the same time, this does not mean that the evaluation setup is static. Indeed, the expanded dataset with the ARAS and Mannheim datasets provides a high degree of diversity and dynamic environment for evaluations. Each user experiences wide variations in environmental settings due to changes in weather and location. For instance, ARAS user one has higher variation of indoor activities while other users have more outdoor activities. These changes in location, coupled with changes in outdoor energy patterns due to weather provide environmental variations for the proposed approach. Moreover, ARAS dataset provides activities over 30 days with considerable variation in activity and EH patterns. The Mannheim dataset also shows variance in activities and EH patterns across users and days. Finally, the solar irradiance data across six years also has dynamic variations due to changes in seasons and months. These dynamic variations in the expanded dataset provide a rich set of EH conditions for evaluation.

Training, Validation, and Test Data: We split the six-year energy data into three sets: training, validation, and test. Data from 2015 is used for training, while 2016 is used for validation. Finally, data from 2017–2020 is used as for testing.

Proposed Conformal Prediction Parameters: The parameters for CP include choice of base predictor, interval length, number of bootstrap models, miscoverage level α and length of residual vector. We use a neural network with four neurons as the base model for energy prediction. The interval length is set to one hour since EM algorithms typically make decisions at hourly or 30 minute intervals [8]. The value of α for prediction interval miscoverage level is set to 0.90. The residual vector lengths R and S are set to 96, respectively.

Proposed Rollout Algorithm Parameters: The parameters for the proposed rollout algorithm include optimization parameters and partitioning width to perform rollout. We set the battery energy target to 100 J and minimum energy to 10 J, respectively. The discount factor β is set to 0.99 and ME is set to 8 J. Furthermore, we set the EM horizon to one day and divide each day into 24 intervals for making EM decisions. Each day starts with an initial battery level of 100 J to provide sufficient energy at the beginning of a day. During the rollout phase, we partition energy harvest prediction intervals into five bins and explore possible energy allocation values with a bin width of 0.2 J. These bin sizes are chosen to balance the complexity of rollout with accuracy of EM decisions.

Conformal Prediction Evaluation Metrics: We use coverage and width of the prediction interval as metrics. Coverage indicates the percentage of instances when the actual EH falls within the predicted bounds. The prediction width is the range of EH values given by the predicted bounds. Our goal is to maximize coverage while minimizing the width.

Energy Management Evaluation Metrics: We utilize total utility to the application over a day as the primary evaluation metric since utility of remaining intervals guides the energy decisions. Specifically, we compare the utility obtained by the rollout algorithm against the baselines. We also use the quality of individual EM decisions and compare it to the optimal Oracle that uses actual EH values to make decisions.

B. Baseline Methods for Comparison

1) Baseline Mean variance Estimation Method: We use MVEM [13] as a baseline approach for comparing the effectiveness of the proposed CP method. MVEM uses two neural networks to predict mean and variance of EH. We use a factor of 1.65 to construct prediction intervals using the mean and variance which is equivalent to prediction interval confidence of 90% given by [LB, UB] = Mean $\pm$ 1.65 $\cdot$ Variance

2) Baseline Energy Management Approaches: We use two baseline approaches for EM.

EM with Expected EH: Prior EM methods typically use expected values of EH and make decisions at the beginning of each horizon. The baseline algorithm utilizes EH estimates to solve the optimization problem in Equations 11-14 using IGP. The average EH of previous three days is used as the estimate.

tinyMAN [26]: tinyMAN is a RL-based approach for runtime EM. The tinyMAN approach uses proximal policy optimiza-

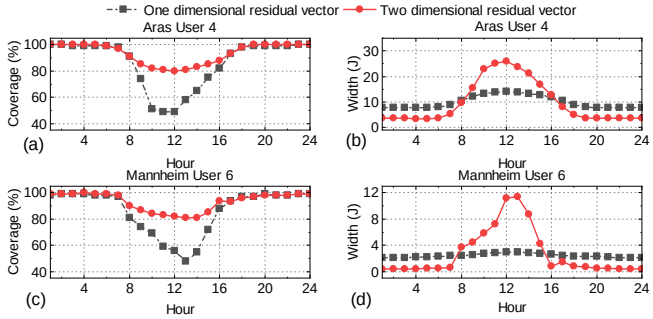

Fig. 4: Comparison of average hourly coverage and width of (a)-(b) ARAS user 4, and (c)-(d) Mannheim user 6 using one-dimensional residual vector and two-dimensional residual vector, respectively.

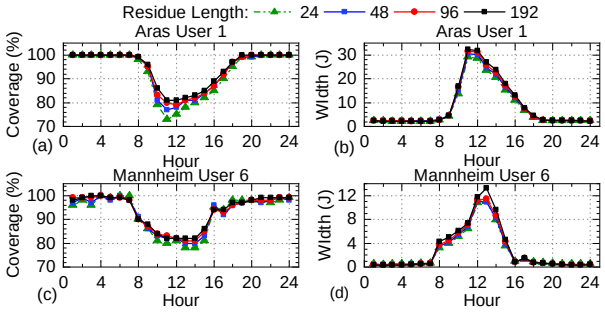

Fig. 5: Comparison of the hourly coverage and width of the prediction interval by proposed approach using different residual lengths for (a)-(b) ARAS user 1 and (c)-(d) Mannheim user 6.

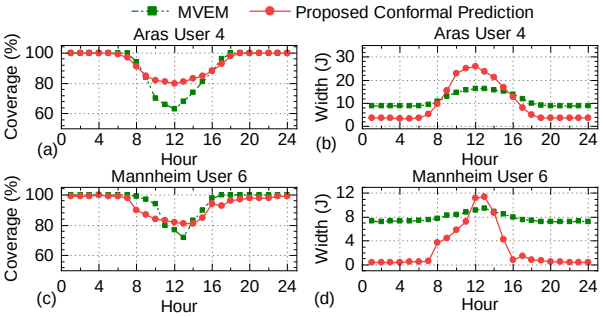

Fig. 6: Comparison of the hourly coverage and width of the prediction interval using proposed approach and MVEM for (a)-(b) ARAS user 4 and (c)-(d) Mannheim user 6.

tion to train an RL agent that maximizes utility of an application under energy constraints. Actual EH data are used to train an agent that is used at runtime without explicitly needing EH estimates. While it is useful, tinyMAN suffers from two major limitations: 1) It does not account for EH prediction intervals and provides a single energy allocation for each interval, thus it does not account for the uncertainty. 2) RL approaches require large number of episodes to achieve good performance and design of appropriate rewards functions can be challenging [33]. In contrast, the proposed rollout for EM provides decision intervals and does not require any training.

C. Evaluation of the Proposed EH Prediction Approach

1) Validation of Using 2-D Residual Vector: We start with validation of the proposed CP approach by analyzing benefits provided by two-dimensional residual vector when compared to one-dimensional residual used in prior approaches. To this end, we first set the one-dimensional residual vector length to 192 hours. The proposed two-dimensional residual vector contains 192 entries by ensuring $R = 96$ and $S = 96$.

Figure 4 shows a comparison of average coverage and width for one user from ARAS and Mannheim datasets at each hour of the day. The coverage for both users improves significantly using two-dimensional residual vector, especially when the EH is high during mid-day hours. The prediction width is also higher for mid-day hours to account for the higher coverage and magnitude of EH in mid-day hours. Conversely, during early hours of the day two-dimensional residual vector achieves significantly tighter prediction width while maintaining high coverage. This shows that the two-dimensional vector is able to adapt to varying EH magnitudes throughout the day while maintaining high coverage, thus providing useful information for EM algorithms.

2) Selection of Bootstrap Models and Residual Vector Length: The number of bootstrap models and residual vector length directly impact accuracy and overhead. We perform a design space exploration to select the number of bootstrap models and length of the sliding residual vector. Specifically, we compare the coverage and width of CP with 5, 10, and 15 bootstrap models. Our analysis shows that 15 bootstrap models offer the highest accuracy without significant overhead.

To choose the appropriate residual vector length, we vary it from 24 to 192, as shown in Figure 5. Residual values for 24, 48, 96, and 192 are shown, while noting that residual length of 120 provides results similar to 96. The residual lengths represent values of R and S, resulting in total length that is twice the values shown in the figure. Higher residual lengths provide the higher coverage while maintaining similar prediction width. In particular, residual lengths of 96 and 192 provide higher coverage for early hours of the day when the EH is lowest. At the same time, memory and computational overhead increase with increase in residual length. Consequently, we choose 96 as the residual length because it provides a good balance between accuracy and overhead.

3) Comparison of Proposed CP with MVEM: Next, we evaluate the accuracy of CP and compare it with MVEM. We start with a comparison of average hourly coverage and prediction width for one user in each dataset in Figure 6. We see that both CP and MVEM have close to 100% coverage during early and late hours. However, MVEM has a significantly higher width when compared to CP, showing that it is unable to adapt to lower energy values. In fact, for Mannheim user 6, MVEM has a constant width of about 8 J throughout the day, as shown in Figure 6(d). The coverage for CP decreases during mid-day hours due to higher variance in energy. At the same time, coverage achieved by CP is higher than MVEM while having widths that are comparable to MVEM.

Next, we compare the performance of CP and MVEM for ARAS user 4 on a specific day in June 2017. Figure 7 shows that MVEM prediction interval does not cover the actual EH during hours 10–12 and 15–16. In contrast, CP adapts prediction widths throughout the day as per user activity.

Finally, Figure 8 shows the average coverage and width

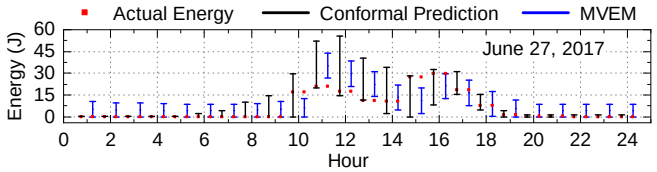

Fig. 7: Comparison of (a) coverage, and (b) width of the prediction interval achieved for ARAS user 4 on June 27, 2017.

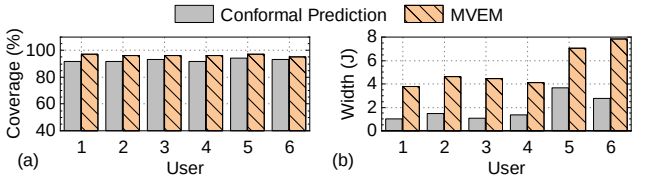

Fig. 8: Comparison of (a) Average coverage, and (b) average width of the prediction interval achieved by proposed approach and MVEM for 6 users of Mannheim data for five years (2016-2020).

for all users in the Mannheim dataset. While the coverage for both approaches is comparable, CP achieves, on average, 64% lower prediction width for all users in the Mannheim dataset. We also note that CP achieves 18% lower average width for ARAS dataset users.

In summary, results in this section show that CP is able to accurately capture differences in user activity and location patterns to provide tailored prediction intervals.

D. Evaluation of the Proposed EM using Rollout

1) Validation of Energy Allocation Bounds from CP Prediction Intervals: Recall that the proposed rollout algorithm provides energy allocation bounds for each interval in a day. Specifically, we provide energy allocation bounds that provide utility in the top 10% percentile at the end of rollout. The device then consumes energy within the bounds. It is crucial to ensure that the energy allocation bounds contain the optimal energy allocation. This section evaluates the allocation bounds provided by the proposed approach against the optimal value.

Figure 9 shows energy allocation bounds and optimal energy allocation for the fourth user in ARAS dataset on a specific day. The black vertical line represents initial energy allocation range for rollout while the blue lines represent final energy allocation range. We can see from the figure that on both days, we start with a wider range of initial energy allocations for rollout. The bounds are made tighter using rollout, as shown with the blue lines. The energy allocation bounds provided by the rollout algorithm contain the optimal energy allocation for most of the hours in the day. Even when the bounds do not contain the optimal, the range is close to the optimal. This shows that the proposed approach effectively provides energy allocation bounds while accounting for uncertainty in EH.

2) Comparison of EM with Baseline Approaches: Next, we show a comparison of the energy allocation and battery levels for two different days for fourth user in the ARAS dataset and sixth user in the Mannheim dataset, respectively. Specifically, Figure 10(a) and (c) shows comparison of the energy allocation of the proposed algorithm along with the optimal allocations and the two baseline approaches, while Figure 10(b) and (d) compare the corresponding battery energy levels for the same day. We can see from the figure that energy allocation by

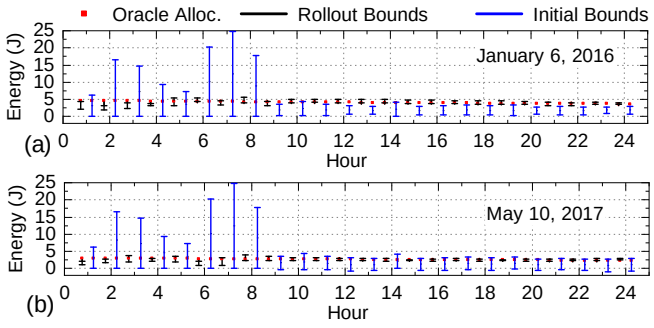

Fig. 9: Validation of obtaining energy allocation bounds w.r.t optimal allocation for (a) ARAS User 4 and (b) Mannheim User 6.

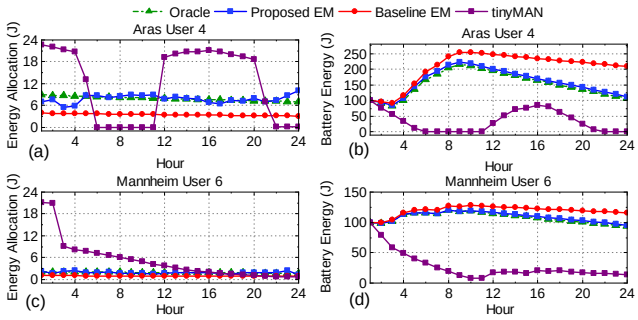

Fig. 10: Comparison of energy allocations and battery levels obtained using proposed approach to the optimal and baseline approaches over a finite horizon for (a)-(b) ARAS user 4 and (c)-(d) Mannheim user 6.

proposed EM closely matches the optimal while the iterative approach under-allocates throughout the day. This results in additional battery energy accumulation for the baseline while the battery energy levels of the proposed approach closely match the Oracle. Similarly, tinyMAN allocates higher energy in the initial hours and then goes into a low power state due to battery exhaustion. The energy allocations recover during middle hours, before going to a low power state at end of the day. Moreover, tinyMAN is unable to maintain the target energy constraint for both users. In contrast, the proposed approach closely follows the optimal value and maintains ENO. It also quickly overcomes any under-allocations due to accounting for deviations in the following hours. Better performance of the proposed EM algorithm is due to higher quality of EH predictions from CP and trajectory evaluations using rollout. These results show that the proposed approach is able to closely follow the optimal allocations while maintaining ENO.

3) Utility Improvement with Rollout: Overall application utility is an important metric since it captures the quality of service to the users. To this end, we measure the percentage utility improvement of the proposed EM algorithm compared to the baseline approaches, as shown in Table I. Outliers with low actual EH are removed from the comparison since it is challenging to provide high utility for any approach on such days. After removing the outlier days with low EH, we observe that the proposed algorithm provides significant utility improvements compared to both the iterative EM and tinyMAN. In summary, our experimental results show that accounting for uncertainty in future EH in energy prediction and management leads to significantly better performance.

TABLE I: Utility Improvement with rollout. ARAS user one has NA for tinyMAN since it has high degree of zero allocations and violations.

	User	Utility Improv. (%) Iterative	tinyMAN	User	Utility Improv. (%) Iterative	tinyMAN
ARAS	1	55	NA	3	77	100
	2	75	100	4	78	100
Mannheim	1	44	43	4	45	22
	2	51	15	5	43	21
	3	53	40	6	42	25

E. Implementation Overhead

We use the TI-CC2652R device to measure the energy consumption of proposed CP and rollout, while simulations are used to evaluate the EM performance by streaming data into Python [34] functions for CP and rollout. We first implement the CP and rollout methods in C for integration into the TI board. The C code is instrumented with timing calls from standard C libraries to measure the execution time. Compiled binaries are then flashed on the TI device to measure execution time and energy consumption. The energy consumption is measured using TI's EnergyTrace [35] technology that provides accurate current and power consumption on TI devices. Power consumption from EnergyTrace is combined with the execution time to obtain the energy consumption.

The overhead measurements capture the execution time and energy for executing the proposed algorithms. The measurements capture raw energy and latency values without including the application. This is because we perform EM in an application-agnostic manner using an abstract utility function. Moreover, the proposed methods are not dependent on any particular application and are applicable any IoT application.

Overhead for the CP consists of running bootstrap models and prediction interval construction. Our measurements on the TI-CC2652R device show that each bootstrap model takes about 11 ms to execute, resulting in 165 ms execution time for 15 bootstrap models. The energy consumption for each bootstrap model is 160 μJ, resulting in 2.4 mJ of energy for 15 models. Next, prediction interval construction using CP takes about 52 ms to execute with 0.6 mJ energy. The memory required for storing features, residual vector, and neural network weights is about 7 KB. In summary, the proposed EH prediction approach takes about 217 ms execution time and 3 mJ of energy overhead, which is negligible when compared to the interval length of one hour and EH in each interval.

We also implement the EM algorithm on the TI-CC2652R processor to measure its overhead. Measurements show that it takes about 300 ms for each rollout. On average, we perform about 45 rollouts in each interval, leading to 13 s of overhead each hour. This results in about 390 mJ energy. The total overhead is less than 15% of the mean EH values. We note that tinyMAN has lower overhead [26], however, it suffers from lower utilities and higher ENO violations. If the potential EH and available energy are less than the overhead, the system goes into a low power state to conserve energy. Furthermore, we can optimize EM by limiting the rollouts as per the available energy budget since rollout iterations are the major component of overhead. Overall, the overhead is reasonable compared to the utility gains provided by rollout.

VII. Discussion and Potential Impact

Potential Societal Benefits: Widespread adoption of wearable devices can have several societal benefits by enabling continuous monitoring of health parameters [36]. Distribution-free conformal prediction and EM approaches proposed in this paper form an important component of this vision because they can lead to recharge-free operation for wearable devices. Individuals can have a better user experience since the device will adapt to various energy setups, leading to more widespread adoption of wearable technology.

Using EH instead of manual recharging for wearable devices will lead to significant environmental benefits by lowering the carbon footprint. To further understand these benefits, we evaluate the total EH in each month of the year for both datasets. Figure 11 shows the average energy harvest for each user in the two datasets. The black triangles in the figure denote the mean of EH along with one standard deviation on the top and bottom. If we conservatively estimate that a million users will use wearable devices, this results in 10 MWh energy savings per year, leading to about 7 metric tons reduction in CO_2 emissions as per estimates by the US EPA [37].

Limited Energy Scenarios: We also observe that the ambient energy levels are low for several hours or days. These energy levels are not enough to power the sensors or processors on the device. Therefore, the proposed approach uses a backup battery to store harvested energy. The battery provides required power levels when the harvested power is not sufficient. At the same time, if the EH is low throughout the day, it may not be sufficient to maintain ENO. In such cases, the device either has to go into a lower power state or violates ENO.

In terms of applications, low energy scenarios can be useful when continuous monitoring of users is not required. The device can store energy during periods with no activity so that sufficient reserves are available during active periods. However, further optimizations in processors and sensors may be needed if the available EH levels are insufficient.

Synergy with Intermittent Computing: Intermittent computing technologies, such as checkpointing, are being used in energy-constrained wearable devices [38]. The key idea is to perform computations *only* when sufficient energy is available from ambient sources. Co-design of applications with intermittent computing and CP for energy predictions can potentially improve EM in wearable devices. We believe that this will be an interesting problem for the community.

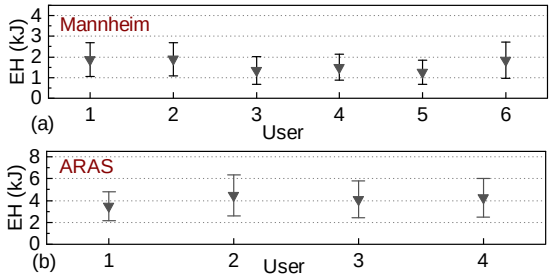

Fig. 11: Mean and Standard deviation of monthly energy harvest during 2017-2020 in Colorado for (a) Mannheim dataset (b) ARAS datasets.

VIII. Conclusion

Wearable and IoT devices offer promising applications in health monitoring and digital agriculture. However, their widespread usage is constrained by their limited battery capacities. Ambient EH is an effective solution to augment the battery of wearable devices. However, stochastic nature of EH makes EM in wearable devices challenging. Effective EM requires knowledge of future EH and associated uncertainty. This paper proposed a distribution-free conformal prediction approach for prediction intervals of future EH in IoT devices. We also proposed a rollout-based EM approach that provided significant utility improvements compared to a baseline approach. Our immediate future work will focus on co-design of wearable applications with intermittent computing and proposed CP-based energy predictions.

References

[1] M. Odema *et al.*, "Energy-Aware Design Methodology for Myocardial Infarction Detection on Low-Power Wearable Devices," in *ASPDAC*, 2021, pp. 621–626.

[2] L. Atzori, A. Iera, and G. Morabito, "The Internet of Things: A Survey," *Computer Networks*, vol. 54, no. 15, pp. 2787–2805, 2010.

[3] D. Vasisht *et al.*, "Farmbeats: An Iot Platform for Data-Driven Agriculture," in *USENIX NSDI*, 2017, pp. 515–529.

[4] L. Ye *et al.*, "The Challenges and Emerging Technologies for Low-power Artificial Intelligence IoT Systems," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 12, pp. 4821–4834, 2021.

[5] W. S. Wang *et al.*, "Design Considerations of Sub-mW Indoor Light Energy Harvesting for Wireless Sensor Systems," *J. Emerg. Technol. Comput. Syst.*, vol. 6, no. 2, pp. 1–26, 2008.

[6] Y. Tuncel *et al.*, "Towards Wearable Piezoelectric Energy harvesting: Modeling and Experimental Validation," in *Proc. ISLPED*, 2020, pp. 55–60.

[7] J. R. Piorno *et al.*, "Prediction and Management in Energy Harvested Wireless Sensor Nodes," in *Int. Conf. Wireless Comm., Vehicular Tech., Info. Theory and Aerospace & Electron. Syst. Tech.*, 2009, pp. 6–10.

[8] A. Kansal *et al.*, "Power Management in Energy Harvesting Sensor Networks," *ACM Trans. Embedd. Comput. Syst.*, vol. 6, no. 4, p. 32, 2007.

[9] A. Cammarano *et al.*, "Pro-Energy: A Novel Energy Prediction Model for Solar and Wind Energy-Harvesting Wireless Sensor Networks," in *Int. Conf. on Mobile Ad-Hoc and Sensor Syst.*, 2012, pp. 75–83.

[10] N. Yamin and G. Bhat, "Online solar energy prediction for energy-harvesting internet of things devices," in *Int. Symp. on Low Power Electon. Des. (ISLPED)*, 2021, pp. 1–6.

[11] ——, "Uncertainty-aware energy harvest prediction and management for iot devices," *ACM Transactions on Design Automation of Electronic Systems*, vol. 28, no. 5, pp. 1–33, 2023.

[12] A. K. Yadav and S. Chandel, "Solar Radiation Prediction using Artificial Neural Network Techniques: A Review," *Renewable and Sust. Energy Rev.*, vol. 33, pp. 772–781, 2014.

[13] D. A. Nix and A. S. Weigend, "Estimating the Mean and Variance of the Target Probability Distribution," in *Proc. Int. Conf. Neural Netw.*, vol. 1, 1994, pp. 55–60.

[14] L. Sluijterman, E. Cator, and T. Heskes, "Optimal Training of Mean Variance Estimation Neural Networks," *arXiv preprint arXiv:2302.08875*, 2023.

[15] C. Xu and Y. Xie, "Conformal Prediction for Dynamic Time-Series," *arXiv preprint arXiv:2010.09107*, 2020.

[16] T. Huynh, M. Fritz, and B. Schiele, "Discovery of Activity Patterns using Topic Models," in *Proc. Conf. Ubiquitous Comput.*, 2008, pp. 10–19.

[17] D. Bertsekas, "Rollout algorithms for discrete optimization: A survey," *Handbook of Combinatorial Optimization*, 2010.

[18] H. Alemdar *et al.*, "ARAS Human Activity Datasets in Multiple Homes with Multiple Residents," in *Int. Conf. Perv. Comput. Tech. Health. and Work.*, 2013, pp. 232–235.

[19] T. Sztyler *et al.*, "Self-tracking Reloaded: Applying Process Mining to Personalized Health Care from Labeled Sensor Data," in *Trans. Petri Nets and other Models Concur.* Springer, 2016, pp. 160–180.

[20] Texas Instruments Inc., "CC2652R Microcontroller," [Online] https://www.ti.com/product/CC2652R, accessed August 12, 2022, 2018.

[21] A. Khosravi *et al.*, "Lower Upper Bound Estimation Method for construction of Neural Network-based Prediction Intervals," *IEEE Trans. Neural Netw.*, vol. 22, no. 3, pp. 337–346, 2010.

[22] G. Bhat, J. Park, and U. Y. Ogras, "Near-Optimal Energy Allocation for Self-Powered Wearable Systems," in *Proc. Int. Conf. on Comput.-Aided Design*, 2017, pp. 368–375.

[23] N. Yamin and G. Bhat, "Near-Optimal Energy Management for Energy Harvesting IoT Devices Using Imitation Learning," *IEEE Trans. Comput.-Aided Des. of Integr. Circuits Syst.*, vol. 41, no. 11, pp. 4551–4562, 2022.

[24] N. Yamin, G. Bhat, and J. R. Doppa, "DIET: a dynamic energy management approach for wearable health monitoring devices," in *Des., Autom. & Test in Europe Conf. & Exhibition (DATE)*, 2022, pp. 1365–1370.

[25] F. A. Aoudia *et al.*, "RLMan: An energy manager based on reinforcement learning for energy harvesting wireless sensor networks," *IEEE Trans. Green Commun. Netw.*, vol. 2, no. 2, pp. 408–417, 2018.

[26] T. Basaklar, Y. Tuncel, and U. Y. Ogras, "tinyMAN: Lightweight Energy Manager using Reinforcement Learning for Energy Harvesting Wearable IoT Devices," in *tinyML Research Symposium*, 2022, pp. 1–7.

[27] G. Shafer and V. Vovk, "A Tutorial on Conformal Prediction," *J. Machine Learn. Research*, vol. 9, no. 3, 2008.

[28] R. F. Barber *et al.*, "Conformal Prediction beyond Exchangeability," *arXiv preprint arXiv:2202.13415*, 2022.

[29] S. P. Boyd and L. Vandenberghe, *Convex optimization*. Cambridge university press, 2004.

[30] N. Karakoç *et al.*, "Multi-layer decomposition of network utility maximization problems," *IEEE/ACM Transactions on Networking*, vol. 28, no. 5, pp. 2077–2091, 2020.

[31] FlexSolarCells, "SP3-37 Datasheet," 2013, https://bit.ly/3dcJ0lK, accessed 3/20/2023.

[32] A. Andreas and T. Stoffel, "NREL Solar Radiation Research Laboratory (SRRL): Baseline Measurement System (BMS); Golden, Colorado (Data); NREL Report No. DA-5500-56488," 1981, accessed 7/28/2023.

[33] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of Real-World Reinforcement Learning," *arXiv preprint arXiv:1904.12901*, 2019.

[34] G. Van Rossum and F. L. Drake, "Python Library Reference," 1995.

[35] J. Lindh *et al.*, "Measuring CC13xx and CC26xx Current Consumption," *Texas Instrument, Application Report*, 2019.

[36] A. J. Espay *et al.*, "Technology in Parkinson's Disease: Challenges and Opportunities," *Movt. Disorders*, vol. 31, no. 9, pp. 1272–1282, 2016.

[37] EPA, "U.S. annual national emission factor, year 2019 data. U.S. Environmental Protection Agency, Washington, DC." 2019, https://www.epa.gov/egrid, accessed 8/11/2023.

[38] B. Lucia *et al.*, "Intermittent Computing: Challenges and Opportunities," *Summit on Advances in Prof. Lang.*, 2017.