Enabling Reliable Environmental Sensing with LoRa, Energy Harvesting, and Domain Adaptation

Aitian Ma[§], Jean C Tonday Rodriguez[§], Mo Sha
Knight Foundation School of Computing and Information Sciences
Florida International University
11200 SW 8th St, Miami, FL 33199
Email: {aima, jtond004, msha}@fiu.edu
Telephone: +1-305-348-1717

***Abstract*—Environmental sensing is essential for many applications. Many existing efforts rely on the readings provided by the weather stations maintained by federal, regional, or local government agencies. While the accuracy of the readings provided by those weather stations is high, the ability of such data to reflect the temperature variability experienced by urban populations is generally low. Therefore, recent studies have proposed to deploy new infrastructures with low-power communication and energy harvesting capabilities to provide fine-scale measurements. Recently, there has been an increasing interest in deploying environmental sensing systems with LoRa radios and solar panels. However, there have been very few studies looking into the reliability of solar power in the LoRa-based environmental sensing settings. In this paper, we present an empirical study that investigates how well solar energy powers an environmental sensing platform. Our study shows that solar energy generation forecasting plays an important role in the performance of the sensing platform. To address the challenges, we develop a novel solution that leverages LoRa, energy harvesting, and domain adaptation to enable reliable environmental sensing. Experimental results show that our solution outperforms the baselines and effectively supports end devices to perform environmental sensing operations without interruptions.**

***Index Terms*—Environmental sensing, LoRa, energy-harvesting, domain adaptation**

I. Introduction

Environmental sensing is essential for many applications. Many existing efforts rely on the readings, such as temperature and humidity, provided by the weather stations maintained by federal, regional, or local government agencies. For instance, Zhang et al. used the temperature readings provided by the weather stations to conduct research on predicting heat-related mortality in urban environments [1]. While the accuracy of the readings provided by those weather stations is high, the ability of such data to reflect the temperature variability experienced by urban populations is generally low because the measurements are collected at the mesoscale ($3{,}000-100{,}000m$). In addition, the weather stations are often located in open areas to ensure no interference from shading, therefore they do not reflect the distribution of populations, nor of built environments that can generate urban heat island effects. In reality, the temperature varies at the microscale ($<100m$) and the local scale ($100-3{,}000m$), and the health risks associated with extremely hot weather are assumed to vary with the exposure. To overcome such limitations, recent studies have proposed to deploy new infrastructures with low-power communication and energy harvesting capabilities to provide fine-scale measurements [2].

Recent years have witnessed rapid deployments of LoRa networks to support environmental sensing applications. As an emerging Low-Power Wide-Area Networks (LPWAN) technology, LoRa provides a low-cost wireless solution that supports long-range data collection for low data rate applications [3]. Over the past decade, LoRa networks have been deployed in 153 countries to support various applications, such as smart agriculture [2], smart city [4], and smart energy [5]. On the other hand, solar power harvesting is appealing for use in environmental sensing applications, because solar panels are relatively inexpensive and easy to deploy, and they provide a renewable power source to operate sensing platforms in locations that are remote, hard to reach, or simply difficult or expensive to run electrical wires or replace batteries. Therefore, there has been an increasing interest recently in deploying environmental sensing systems with LoRa radios and solar panels.

However, there have been very few studies looking into the reliability of solar power in the LoRa-based environmental sensing settings. In this paper, we present an empirical study that investigates how well solar energy powers a LoRa-based sensing platform. Our study shows that solar energy generation forecasting plays an important role in the performance of a sensing platform. However, accurately forecasting the amount of generated solar energy is challenging due to the location-specific gap. Weather variations, including cloud cover, temperature, and atmospheric conditions, significantly affect solar energy output. The models trained in one location may not work well in another because of different weather patterns. Deep learning techniques can improve forecasting accuracy but require extensive datasets for training and refinement. However, collecting sufficient labeled data for an accurate model is time-consuming and demands much human effort. Based on the insights gathered from our empirical study, we develop a solution that leverages a teacher-student neural network to train a precise solar energy generation forecasting model with a few labeled data and a new time-slot-based cycle

[§]The first two authors contributed equally.

assignment method to enable reliable environmental sensing. Specifically, we make the following contributions in this paper:

- We perform an empirical study that identifies the challenges of using solar energy to power a LoRa-based sensing platform;
- We formulate the solar power forecasting as a machine learning problem and reveal the location-specified gap, which prevents the model trained using publicly accessible data from providing good forecasting performance in real-world deployments;
- We develop a domain adaptation-based method to close the gap and train a good solar power forecasting model using publicly accessible data and a small number of local measurements;
- We develop a method that schedules the duty cycle of an end device to maximize the number of samples it collects in each time period without running out of battery;
- We implement our methods and test our solution in a real-world environment. Experimental results show that our solution outperforms the baselines and effectively supports end devices to perform environmental sensing operations without interruptions.

Our paper is organized into the following sections. Section II introduces our environmental sensing platform. Section III introduces our empirical study. Section IV and V present the designs of our solar power forecasting and time-slot-based cycle assignment methods. Section VI evaluates our methods. Section VII reviews the related work. Section VIII concludes this paper.

II. Environmental Sensing System

In this section, we introduce the hardware and software architecture of our environmental sensing platform.

A. Hardware

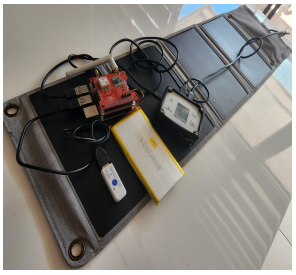


Fig. 1. Hardware Module

Module	Price
Raspberry Pi 4B	$55
TemperHum	$32
LoRa HAT	$32
PiJuice HAT	$69
PiJuice solar panel	$112
PiJuice battery	$35
Tektyte Log4USB	$171

Fig. 2. Retail prices

Figure 1 shows the hardware of our environmental sensing platform, which is built by integrating several commercial off-the-shelf hardware modules. Figure 2 lists all hardware modules and their retail prices. Our sensing platform uses a Raspberry Pi 4 Model B as its central processing unit, which controls all sensing and communication peripherals. The TemperHum hygrometer [6] has temperature and humidity sensors and forwards sensor readings to the Raspberry Pi through its USB port. The Dragino LoRa GPS Hardware

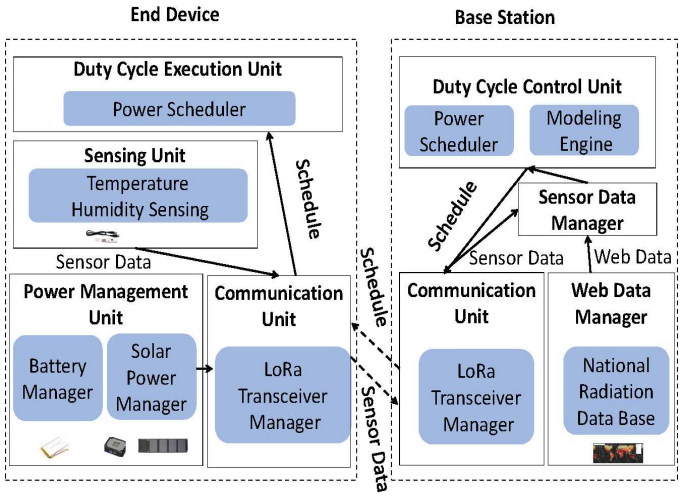


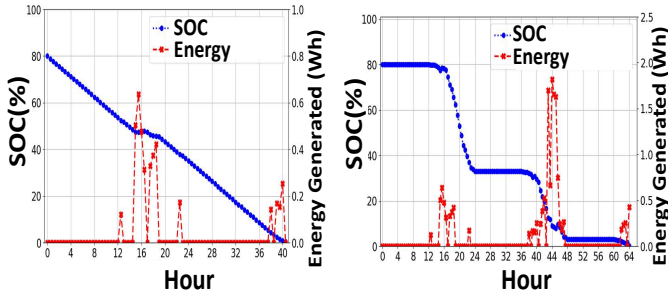
Fig. 3. Software architecture of our platform and base station.

Attached on Top (HAT) [7] with a Semtech SX1276/SX1278 LoRa transceiver [8] is integrated with the Raspberry Pi to support LoRa communication. The PiJuice HAT [9] and PiJuice solar panel [10] are integrated with the Raspberry Pi to harvest solar energy. The excess energy is stored in the PiJuice battery [11]. Tektyte Log4USB multi-meter connects the PiJuice HAT and the PiJuice solar panel to measure the power generated by the solar panel [12]. Using those PiJuice modules allows us to put the Raspberry Pi into sleep model, which reduces the energy consumption from seven $watts$ to almost zero. The cost of the sensing platform is around $506 in total.

B. Software Architecture

Figure 3 plots the software architecture of our environmental sensing system. The software that runs on the end device has four units: **Sensing Unit**, **Power Management Unit**, **Communication Unit**, and **Duty Cycle Execution Unit**. The Sensing Unit is responsible for managing the temperature and humidity sensors. The Power Management Unit is responsible for measuring the State of Charge (SOC), which represents the percentage of the remaining energy in the battery, and the electric current generated by the solar panel. The temperature, humidity, SOC, and electric current readings are collected by the Communication Unit and transmitted to the base station. Duty Cycle Execution Unit periodically puts the platform into sleep mode to reduce energy consumption. In each cycle, the end device wakes up from the sleep mode, performs a set of activities (e.g., generates and transmits sensor readings), and then goes back to sleep. Each end device follows the schedule (i.e., how many cycles each end device can have each day) generated by the base station to perform activities.

The software that runs on the base station consists of four units: **Sensor Data Manager**, **Web Data Manager**, **Communication Unit**, and **Duty Cycle Control Unit**. The Sensor Data Manager collects temperature and humidity readings from the Communication Unit and forwards them to the cloud. The Web Data Manager is responsible for gathering



(a) Algorithm 1 (b) Algorithm 2

Fig. 4. SOC and energy harvested from the solar panel

weather and solar information from the National Solar Radiation Database (NSRDB) provided by the National Renewable Energy Laboratory (NREL) [13] and forwarding it to the Duty Cycle Control Unit. The Duty Cycle Control Unit leverages the NSRDB data, temperature, and humidity readings to determine the schedule for each end device.

III. Empirical Study

In this section, we present our empirical study that investigates how well solar energy powers the LoRa-based sensing platform by examining the performance of two classic cycle assignment algorithms.

A. Using a Fixed Number of Cycles

Algorithm 1: Cycle Assignment Algorithm

Input: Power consumption of the device (P^c), battery capacity (C), and the total number of cycles in one day (N_{dc})

Output: Number of cycles (N_c)

1: **for** each day i in m days **do**
2: Measure SOC (S_i^b) at the beginning of the day and keep the end device operating without sleep;
3: Measure SOC (S_i^e) after 24 hours;
4: Compute the energy generated during the last 24 hours by calculating $E_i = 24 * P^c - C * (S_i^b - S_i^e)$;
5: **if** $i == 1$ or $S_i^e > S_{i-1}^b - S_{i-1}^e$ **then**
6: i++;
7: **else**
8: Break;
9: **end if**
10: **end for**
11: Output $N_c = N_{dc} * \sum_{j=1}^{m} E_j/(m * 24 * P^c)$;

Our empirical study starts with running Algorithm 1, which determines the number of cycles during a 24-hour window based on the end device's power consumption (P^c) and battery capacity (C). The key idea is to measure the average energy generated by the solar panel over a few days and then calculate the number of cycles by dividing the average energy by the total consumption if the end device keeps operating without sleep. We run Algorithm 1 on the end device and get $N_c = 240$. We then configure the end device to perform 240 cycles during each 24-hour window and continue our experiment. Unfortunately, the end device runs out of battery after experiencing a cloudy day. Figure 4(a) plots SOC and the amount of energy generated by the solar panel over the 40 hours before the end device ran out of battery. As Figure 4(a) shows, SOC keeps decreasing from 80% until the sun rises in the morning (around the $15th$ hour) as expected. However, the amount of energy generated by the solar panel fluctuates between 0 and $0.64Wh$ during a cloudy day. The solar panel only charged the battery to 48% before sunset, which is not enough to power the device for another day. The end device's battery runs out at the $40th$ hour. The results demonstrate that using a fixed number of duty cycles over the entire day cannot work well due to the fluctuations of the generated energy.

B. Assigning Different Number of Cycles at Different Time

Algorithm 2: Time-Slot based Cycle Assignment Algorithm

Input: Power consumption of the device (P^c), battery capacity (C), number of time slots during 24 hours (N_t), and the total number of cycles in one time slot (N_{tc})

Output: Number of cycles (N_i) during time slot i

1: **for** each day j in m days **do**
2: **for** each time slot i in day j **do**
3: Measure SOC (S_{ji}^b) at the beginning of the time slot i on day j and keep the end device operating without sleep;
4: Measure SOC (S_{ji}^e) at the end of the time slot i on day j;
5: Compute the energy generated during time slot i on day j by $E_{ji} = 24 * P^c/N_t - C * (S_{ji}^b - S_{ji}^e)$;
6: **end for**
7: **if** $i == 1$ or $S_i^e > S_{i-1}^b - S_{i-1}^e$ **then**
8: j++;
9: **else**
10: Break;
11: **end if**
12: **end for**
13: Output $N_i = N_{tc} * \sum_{j=1}^{m} E_{ji} * N_t/(m * 24 * P^c)$;

Our empirical study then examines the effectiveness of assigning different numbers of cycles at different times throughout the day. The key idea is to divide 24 hours into a set of time slots, calculate the average generated energy during each time slot, and then assign a different number of cycles (N_i) to the different time slots. Algorithm 2 shows the method we used to achieve this. We divide 24 hours into 48 time slots, run Algorithm 2 on the end device, and obtain the 48 cycle assignments. Figure 4(b) plots SOC and the amount of energy generated by the solar panel over the 64 hours before the end device runs out of battery. As Figure 4(b) shows, SOC stays the same as the device sleeps before the sun rises at the $15th$ hour and when the solar panel produced enough energy between the

$43rd$ and $45th$ hour. SOC decreases to 34% when the solar panel fails to produce enough energy during a cloudy day. SOC remains at 33% until the next day and then gradually reduces to 3%. The end device runs out of battery at the $64th$ hour. As Figure 4(a) and 4(b) show, assigning different numbers of cycles at different times allows the end device to operate more reliably compared to the solution that uses a fixed cycle for the entire day. However, none of them can effectively prevent the end device from running out of battery. Therefore, it is crucial to consider the energy produced during the next day when assigning the number of cycles to the end device. This motivates us to develop a new method for accurate solar power forecasting.

IV. Solar Power Generation Forecasting

In this section, we formulate the solar power generation forecasting problem, present the location-specified gap, and introduce our solution to close the gap.

A. Solar Power Generation Prediction

The primary task in solar power generation prediction is to predict the power generated by the solar panel during the next day based on historical data and real-time environmental measurements. We consider three metrics: temperature measurements $Temp$, humidity readings Hum, and the power generation data $Power$. The input during each time period i is $\mathbf{x_i} = concatenation(Temp, Hum, Power, Timestamp)$. We formulate the solar power generation forecasting task as a multivariate time series forecasting problem. Let $\mathbf{X} = (\mathbf{x_1}, \mathbf{x_2}, ..., \mathbf{x_m})$ denote the sequence of historical feature vectors, and y_j denote the solar power generation in the future time period j. Our goal is to predict future solar power generation for n time periods, represented as $\mathbf{Y} = (\mathbf{y_1}, \mathbf{y_2}, ..., \mathbf{y_n})$. We aim to establish a nonlinear mapping $f_\theta(\cdot) : \mathbf{X} \rightarrow \mathbf{Y}$, which translates our input sequence $\mathbf{X}$ to the output sequence $\mathbf{Y}$, where θ symbolizes the parameters of our model. Those parameters are optimized in a data-driven fashion. Given the continuous nature of solar power generation predictions, f_θ serves as a predictive model. It leverages the concatenated features from the input sequence to forecast the solar power generation of future days.

B. Location-Specified Gap

Our primary objective is to train a model tailored for time series data to predict solar power generation on each end device. We can train the model using either local measurements or publicly accessible data such as NSRDB. We perform an empirical study to investigate the efficiency and effectiveness of both solutions. We create the dataset $\mathcal{D}^l$ with seven days of temperature, humidity, and power generation measurements collected by our end device, and the dataset $\mathcal{D}^w$ by collecting 720 days of temperature, humidity, and power generation readings from NSRDB. We create the two testing datasets, $\mathcal{D}^{tl}$ and $\mathcal{D}^{tw}$, with another two days of measurements: one collected by our end device and the other from NSRDB.

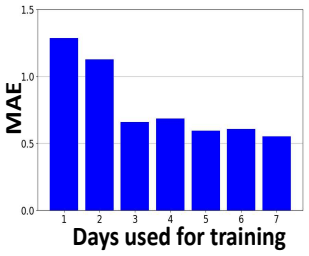


Fig. 5. MAE when using different amounts of data in D^l for training.

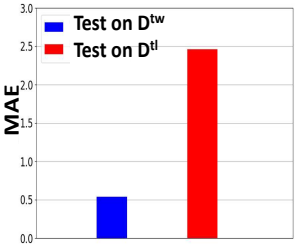


Fig. 6. MAE on web testing data ($\mathcal{D}^{tw}$) and local testing data ($\mathcal{D}^{tl}$) using web training data ($\mathcal{D}^w$).

Using only local measurements ($\mathcal{D}^l$) for training: We first leverage $\mathcal{D}^l$ and Multi-layer Perceptron (MLP) [14] to train the forecasting model. The input to the models is temperature, humidity, and generated power readings with timestamps. The output is a sequence of the power generated by the solar panel in the next 48 time slots (24 hours). We normalize the training data ($\mathcal{D}^l$) into the $[0, 1]$ range. Figure 5 shows the Mean Absolute Error (MAE) when using one to seven days of local measurements to train the model and evaluating the testing data $\mathcal{D}^{tl}$. As Figure 5 shows, MAE decreases from 1.28 to 0.55 as the training data increases from using one to seven days. The results show that a week of measurements can effectively train a good solar power generation forecasting model. However, there exists a significant challenge to leverage the end device to collect enough training data. As Figure 4(a) and 4(b) show, the end device may run out of battery when experiencing a cloudy day without good cycle assignments. Our end device can only collect data for two days without energy harvesting. However, MAE is high (1.13) when using two days of data from training. In our experiments, we had to install power cables to power our end device to create the training data ($\mathcal{D}^l$) with seven days of measurements. However, deploying dedicated power supplies for environmental sensors incurs significant costs, even infeasible in many cases.

Using only web data ($\mathcal{D}^w$) for training: We then leverage $\mathcal{D}^w$ and MLP to train the forecasting model. The NSRDB dataset contains environmental measurements and solar irradiance data for our region. We divide the data based on 30-minute time intervals and convert the Global Horizontal Irradiance (GHI) in our region to the amount of generated power in terms of Watts using the linear equation $P = GHI * A * \alpha$, where α is the efficiency factor specified by the solar panel manufacturer and A is the surface area of the solar panel. Figure 6 plots MAE of the model trained with $\mathcal{D}^w$ and tested on two different testing datasets $\mathcal{D}^{tw}$ and $\mathcal{D}^{tl}$. As Figure 6 shows, the MLP model trained using the web data ($\mathcal{D}^w$) provides high modeling accuracy when we test the model on the web testing data $\mathcal{D}^{tw}$ ($MAE = 0.54$), as the blue bar shows. However, the modeling accuracy drops significantly when we test the models on the local testing data $\mathcal{D}^{tl}$, as shown in the red bar ($MAE = 2.46$). The differences in the modeling accuracy clearly show the effect of the location-specified gap, a subtle but important discrepancy between the measurements collected by NREL and the ones gathered by our end device,

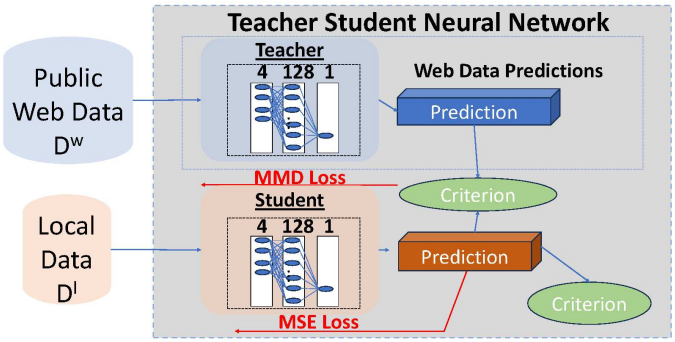


Fig. 7. Teacher-student neural network.

that prevents the forecasting knowledge learned from the web data from enabling effective performance in the area where our end device locates. The location-specified gap exists because the web data cannot provide fine-scale solar information that can be applied to our local area. Because of the location-specific gap, the machine learning models trained using the web data for solar forecasting, no matter how large the data volume is, may not generalize well to a local dataset.

C. Close the Gap by Domain Adaptation

The location-specific gap found between $\mathcal{D}^w$ and $\mathcal{D}^l$ motivates us to explore the feasibility of using the extensive web data together with a small number of local measurements to train a good model for future solar power forecasting. To this end, our objective narrows down from solving a regression problem to using domain adaptation to address the domain discrepancy issue. Specifically, we first gather N^w data tuples from the publicly accessible web data (source domain) and then acquire N^l data tuples by collecting data samples from the local device (target domain). We assume $N^w \gg N^l$ due to the significant data collection overhead on local data collection. We assume that the source and target domains are characterized by different probability distributions q_1 and q_2, respectively. Our goal is to construct a deep learning model that can learn transferable features that bridge the cross-domain discrepancy and build a regression $\mathbf{y} = f_\theta(\mathbf{x})$, which can maximize the target domain accuracy ($f_w \rightarrow f_l$) by using a small amount of local data ($\mathcal{D}^l$).

Figure 7 shows our teacher-student neural network for domain adaptation. We first train our teacher neural network with web data ($\mathcal{D}^w$). To keep our model lightweight, we employ MLP with three layers: 4 and 128 neurons in the first two hidden layers and one neuron in the output layer to forecast the solar power generation in each time slot. Rectified Linear Unit (ReLU) and softmax activate the hidden and output layers, respectively. The teacher's parameters (θ_1) are learned by minimizing the Mean Squared Error (MSE) loss:

$$\mathcal{L}_{MSE}(\theta_1) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2, \tag{1}$$

where n is the number of the training data samples, y_i is the power generation ground truth at the time slot i, and $\hat{y}_i$ is the power generation prediction at the time slot i.

Similarly, we can train our student neural network with our local dataset ($\mathcal{D}^l$). To bridge the gap between the teacher and the student, we use the Maximum Mean Discrepancy (MMD) loss criterion. MMD loss measures the distances in probability using a Hilbert Matrix. To do so we first must calculate the kernel base matrix as an approximation to the Hilbert Matrix given the two inputs y^l, and $\hat{y}^l$ to obtain our matrices $\mathbf{K}$, $K_l = k(y^l, \hat{y}^l)$, $K_w = k(y^w, \hat{y}^w)$, $K_{lw} = k(y^l, y^w)$, where $\mathbf{K}$ can be defined by equation 2:

$$\mathbf{K}(\mathbf{X}, \mathbf{Y}) = e^{-\frac{||(\mathbf{X}-\mathbf{Y})||^2}{2\cdot\sigma^2}} \tag{2}$$

where $\mathbf{X}$ and $\mathbf{Y}$ serve as inputs, $\mathbf{K}$ is the output kernel matrix, and σ is the kernel bandwidth factor. The kernel bandwidth σ represents the width of the Gaussian kernel used for matrix estimation. By varying the size of σ, we can impact the smoothness in the kernel of our data. In particular, by increasing σ, the kernel becomes wider and smoother. We then use the kernels K_{lw}, K_l, and K_w calculated from the previous equation to find the MMD loss described by equation 3:

$$\begin{aligned}\mathcal{L}_{MMD} = &\frac{1}{|K_w|^2}(\sum diag(K_w) + \sum K_w) \\ &-\frac{2}{|K_w| * |K_l|}(\sum K_{LW}) \\ &+\frac{1}{|K_l|^2}(\sum diag(K_l) + \sum K_l)\end{aligned} \tag{3}$$

We use the respective lengths of our Kernel matrix and our diagonals to calculate the MMD loss required for our solution according to existing literature [15]. After obtaining both the model-specific MSE loss and the MMD loss, we can calculate our combined loss $\mathcal{L}(\theta)$, which we use to optimize the parameters of our teacher-student neural network.

$$\mathcal{L}(\theta) = \mathcal{L}_{MSE} + \mathcal{L}_{MMD} \tag{4}$$

$\mathcal{L}_{MSE}$ represents our individual model loss, and $\mathcal{L}_{MMD}$ represents our combined model loss calculated using MMD, and θ denotes the parameters learned by our model. The teacher-student neural network allows our solar power generation forecasting method to produce a forecasting model with extensive web data and a smaller amount of local measurements.

V. Time-Slot Based Cycle Assignment

In this section, we present our method that leverages our solar power forecasting model to assign cycles on each end device to maximize the number of samples it collects in each time slot without running out of battery.

A. Overview

Our time slot-based cycle assignment method takes the solar power forecasting model $f_\theta(x)$, the temperature, humidity, and SOC measurements collected during the last 24 hours as input and performs four processing steps including **Solar Power Prediction**, **Power Distribution Calculation**, **Power**

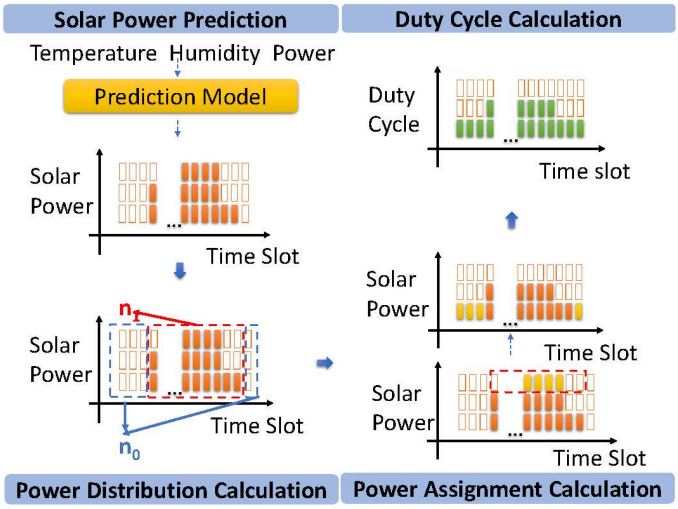


Fig. 8. Four-step time slot-based cycle assignment.

Assignment Calculation, and **Duty Cycle Calculation**, as Figure 8 shows. Solar Power Prediction takes the temperature, humidity, and power with a timestamp in the previous day as input to predict the solar power generation in the upcoming day. It forwards the prediction to Power Distribution Calculation, which computes the number of time slots with zero and non-zero power generation. Leveraging the solar power prediction and the distribution of solar power in different time slots provided by Power Distribution Calculation, Power Assignment Calculation rebalances the energy distribution for zero and non-zero power time slots. Duty Cycle Calculation then uses the rebalanced energy distribution to generate the assignments on the number of cycles for each time slot in the next 24 hours. In this section, we present those four steps in detail.

B. Solar Power Prediction

Algorithm 3: Solar Power Generation Prediction Algorithm

Input: Power consumption of the device (P^c), battery capacity (C), length of time slot (T_c), solar power generation prediction model (f_θ)

Output: Solar power generation prediction ($\mathbf{P}$)

1: **for** each time slot i over previous day **do**
2: Collect temperature ($Temp_i$), humidity (Hum_i), and timestamp ($Timestamp_i$) at time slot i;
3: Measure SOC (S_i^b) at the beginning of time slot i;
4: Measure SOC (S_i^e) at the end of time slot i;
5: Measure uptime of the end device (t_i^u) in time slot i;
6: Compute the power generated during time slot i by $\hat{P}_i = (t_i^u * P^c - C * (S_i^b - S_i^e))/T_c$;
7: **end for**
8: $\mathbf{P} \leftarrow f_\theta(\mathbf{Temp}, \mathbf{Hum}, \hat{\mathbf{P}}, \mathbf{Timestamp})$
9: Output $\mathbf{P}$

Algorithm 3 takes the end device's power consumption (P^c) and battery capacity (C) as input and predicts the solar energy generation in the next 24 hours ($\mathbf{P}$) by harnessing the temperature, humidity, and SOC readings and the forecasting model (f_θ) presented in Section IV. Specifically, Algorithm 3 first gathers the temperature ($Temp_i$) and humidity (Hum_i) readings with timestamps ($Timestamp_i$) for each time slot from the preceding day (line 2). It then measures SOC (S_i^b) at the beginning of time slot i (line 3) and records SOC (S_i^e) at the end of that time slot (line 4). Algorithm 3 measures the active operational duration, or uptime (t^u), within the time slot (line 5). The power generated by the solar panel ($\hat{P}_i$) is calculated by dividing the difference between the energy flux from (or to) the battery ($C * (S_i^b - S_i^e)$) and the cumulative energy expended over uptime during that time slot ($t^u * P^c$) by the time slot cycle (T_c) during time slot i (line 6). Iterations continue until reaching the end of that day (line 7). Finally, leveraging the collated data consisting of temperature (**Temp**), humidity (**Hum**), solar power output ($\hat{P}$), and respective timestamps (**Timestamp**) from every time slot of the preceding day, Algorithm 3 determines the anticipated solar power generation ($\mathbf{P}$) based on the solar power generation prediction model (f_θ) (line 8).

C. Power Distribution Calculation

Algorithm 4: Power Distribution Calculation Algorithm

Input: The solar power prediction ($\mathbf{P}$), number of time slots during 24 hours (N_t)

Output: Number of time slots with zero power generation (n_0),
Number of time slots with non-zero power generation (n_1)

1: Initialize $n_0 = 0$;
2: **for** time slot i in N_t time slots **do**
3: **if** $P_i == 0$ **then**
4: $n_0 + +$;
5: **end if**
6: i++;
7: **end for**
8: $n_1 = N_t - n_0$;
9: output n_0, n_1

Power Distribution Calculation takes the predicted solar power generation ($\mathbf{P}$) as input and then discerns the number of time slots characterized by zero solar energy generation (n_0) versus those exhibiting non-zero solar energy generation (n_1). Algorithm 4 shows the algorithm. The key idea is to perform a methodical examination of each time slot within the solar power forecast ($\mathbf{P}$) to tally time slots with either the absence or presence of solar power generation. Specifically, Algorithm 4 initializes the zero time slot counter (n_0) at 0 (line 1) and then sequentially inspects each power prediction at time slot i (P_i) within the set of N_t time slots (line 2). For each time slot,

if P_i is zero, the counter n_0 is incremented (lines 3-5). The algorithm repeats this process until it reaches the last time slot (N_t) (lines 6-7). After this, Algorithm 4 ascertains the count of time slots exhibiting non-zero solar power (n_1) (line 8).

D. Power Assignment Calculation

Algorithm 5: Power Assignment Algorithm

Input: The solar power generation prediction (P), the number of time slots with zero power generation (n_0), the number of time slots with non-zero power generation (n_1), number of time slots during 24 hours (N_t), length of time slot (T_c), the night-time ratio parameter used for assigning the power distribution (σ_0)

Output: Energy distribution after reassignment ($\hat{E}$)

1: Calculate the Energy distribution ($E = P * T_c$) according to the power forecasting;
2: Calculate the Energy assigned to zero solar power generation time slot by $E_{as} = \sum_{i=1}^{N_t} P_i * T_c * \sigma_0 / n_0$;
3: Calculate the Energy taken from non-zero solar power generation time slot by $E_{am} = \sum_{i=1}^{N_t} P_i * T_c * \sigma_0 / n_1$;
4: **for** time slot i in N_t time slots for E **do**
5: **if** $E_i < E_{am} + E_{as}$ **then**
6: $\hat{E}_i = E_{as}$;
7: **else**
8: $\hat{E}_i = E_i - E_{am}$;
9: **end if**
10: i++;
11: **end for**
12: return $\hat{E}$

Power Assignment Calculation takes the solar power generation prediction (P), the number of time slots with zero (n_0), and non-zero (n_1) power generation to rebalance the power distribution. Algorithm 5 shows our power rebalance policy, which distributes energy based on the predicted power generation (P) across the time slots characterized by zero (n_0) and nonzero (n_1) energy generation while considering the night time ratio (σ_0). The ratio σ_0 represents the proportion of the total solar energy designated for time slots with zero power generation, typically occurring at night. The key idea is to reallocate a fraction of energy from the time slots with nonzero power generation to those with zero energy generation and produce a new energy prediction ($\hat{E}$).

Specifically, Algorithm 5 first finds the predicted energy generated (E) by evaluating the predicted power generations (P) through the entire time slot (N_t) (line 1). Algorithm 5 then finds the energy assigned to close to zero energy time slots (E_{as}) by aggregating the values of E, factoring in the ratio σ_0, and then dividing by n_0 (line 2). It then calculates the energy derived from the time slots with nonzero solar energy generation (E_{am}), achieved by aggregating the values of P, considering the σ_0 ratio, and then dividing by n_1 (line 3). Algorithm 5 then examines each time slot i within the total set of N_t time slots from the forecast (P) (line 4). If the solar energy generated during the current time slot falls below the threshold ($E_{am} + E_{as}$) (line 5), the adjusted power distribution ($\hat{E}_i$) is set to E_{as} (line 6). Otherwise, E_{am} is subtracted from the current forecast value (E_i) (lines 7-8). This process is repeated for every time slot in E (lines 9-11). Finally, Algorithm 1 generates a rebalanced energy distribution ($\hat{E}$) (line 12).

E. Duty Cycle Calculation

Duty Cycle Calculation is the last step in our method. It determines the duty cycle (N_i) for a specific time slot. Distinct from prior steps, Duty Cycle Calculation operates at the runtime of each time slot, which enables dynamic optimization tailored to each interval. With the distributed energy ($\hat{E}_i$) during time slot i, Duty Cycle Calculation calculates the number of cycles (N_i) for each time slot i as follows:

$$N_i = N_{tc} * N_t * \hat{E}_i / (P^c * 24) \quad (5)$$

Where N_t denotes the total number of time slots within a 24-hour window, $\hat{E}_i$ is the energy distribution post-rebalancing for time slot i, N_{tc} is the total number of cycles in one time slot, and P^c denotes the device's power consumption. Consequently, we can ascertain the number of duty cycles (N_i) for each time slot dynamically.

VI. Evaluation

We perform a series of experiments to validate the efficiency of our solution to provide continuous operations to the end device. We first evaluate the capability of our domain adaptation-based method to produce a good solar power generation forecasting model. We then apply the forecasting model trained by our method in our time slot-based cycle assignment method to examine the performance of our solution.

A. Performance of Our Solar Power Generation Forecasting Method

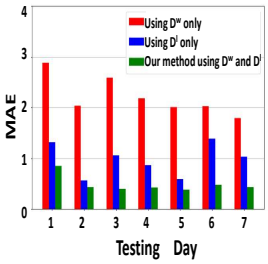


Fig. 9. MAE of our model and models trained on $\mathcal{D}^w$ and $\mathcal{D}^l$ over a 7 day period.

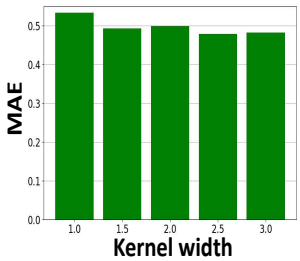
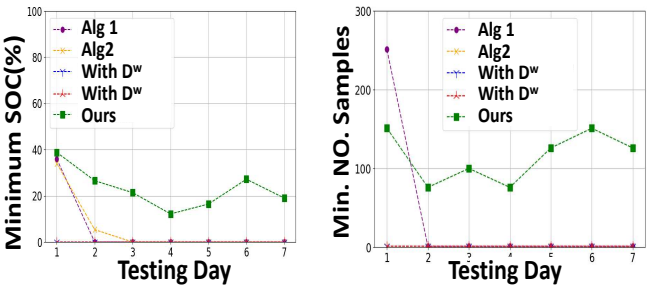


Fig. 10. MAE of our model given different kernel space widths.

We first evaluate the capability of our domain adaptation method to produce good solar power generation forecasting models. We use two days of temperature, humidity, and power generation measurements collected by our end device ($\mathcal{D}^l$) and the 720 days of readings from the NSRDB dataset ($\mathcal{D}^w$) to train the forecasting model. Figure 9 plots the MAE values of



(a) Minimum SOC (b) Minimum number of samples

Fig. 11. Performance of our solution over seven days

our forecasting model over the seven testing days. As comparisons, Figure 9 also plots the MAE values of two baseline models: one trained on only $\mathcal{D}^w$ and the other trained on only $\mathcal{D}^l$. As Figure 9 shows, the forecasting model produced by our method consistently provides the best performance. For example, the MAE value offered by our method is 0.85 on Day 1, while the MAE values provided by our baselines are 1.31 (using $\mathcal{D}^l$) and 2.88 (using $\mathcal{D}^w$), respectively. Similarly, the MAE value offered by our method is 0.38 on Day 5, while the MAE values provided by our baselines are 0.59 and 2.01, respectively. The results show that our domain adaptation method effectively closes the location-specified gap and produces good solar power generation prediction with the web data and only two days of local measurements.

We then evaluate the effects of the Gaussian kernel bandwidth σ, on the performance of our method. The selection of σ value affects the width of our Gaussian distribution kernel. A smaller bandwidth can lead to kernel estimation under smoothing. Figure 10 plots the MAE values of the forecasting model trained with our method when it uses different σ values. As Figure 10 shows, the forecasting performance slightly changes (ranging from 0.48 to 0.53) when using different σ values. The forecasting model offers the best performance ($MAE = 0.48$) when our method sets σ to 2.5. The results show that the solar power generation forecasting model produced by our method is resilient to changes in the bandwidth factor.

B. Performance of Our Solution

We then evaluate the performance of our time slot-based cycle assignment method combined with our solar forecasting method, comparing it against four baselines: Algorithm 1, Algorithm 2, our method with the forecasting model trained with $\mathcal{D}^w$, and our method with the forecasting model trained with $\mathcal{D}^l$. We deploy the base station and four end devices in an outdoor environment and measure their performance when they use four different solutions. Figure 11(a) plots the minimum SOC achieved during each of the seven testing days. As Figure 11(a) shows, only our solution can prevent the end device from running out of battery. The SOC values under our solution vary from 17% to 39%. As comparisons, the end devices that employ the forecasting models trained with $\mathcal{D}^w$ or $\mathcal{D}^l$ run out of battery during its first day. The end devices that use Algorithm 1 or Algorithm 2 to assign cycles run out of battery during the second or third day.

TABLE I
NUMBER OF TIME SLOTS WITHOUT ANY SAMPLES IN EVERY TESTING DAY BECAUSE OF THE DEAD BATTERY. HERE, W $\mathcal{D}^w$ MEANS MODEL WITH WEB DATA, W $\mathcal{D}^l$ MEANS MODEL WITH LOCAL DATA.

Day	Alg 1	Alg 2	w $\mathcal{D}^w$	w $\mathcal{D}^l$	Ours
1	0	26	9	19	0
2	0	26	31	31	0
3	26	31	32	33	0
4	27	29	30	30	0
5	15	27	28	29	0
6	2	28	28	28	0
7	14	28	27	27	0
Overall	84	195	185	197	0

Figure 11(b) plots the minimum number of samples collected by the end devices in any time slot during each day. By running our solution, the end device can collect at least 75 samples during each 30-minute window. In comparison, the end devices that run all the baselines cannot perform environmental sensing cycles without interrupts.

Table I lists the number of time slots without any samples collected. As Table I lists, our solution is the only one that allows the end device to carry out data collection without interruptions during the entire seven days.

VII. Related Works

Solar forecasting has been studied in the literature of Wireless Sensor Networks (WSNs) [16], [17]. For example, numerical weather prediction is a physics-based prediction model widely used with machine learning post-processing [18]. Selvi et al. demonstrated the benefits of using regressive and Deep Neural Network (DNN) models for solar forecasting [19]. Solar energy generation is highly dependent on weather conditions, including cloud cover, temperature, and atmospheric conditions. The inherent variability and unpredictability of weather make it challenging to accurately forecast the amount of generated solar energy, especially for short-term or intra-day forecasts [20]. While machine learning models offer promising improvements in solar forecasting accuracy, they require large datasets for training and validation. Developing and refining these models to improve their predictive accuracy and generalizability across different geographical locations and conditions is an ongoing challenge. In contrast to the existing efforts, our work focuses on closing the location-specified gap by leveraging the correlation between temperature, humidity, and solar power generation to train a solar power generation forecasting model with publicly accessible data and a small number of local measurements.

Many approaches have been developed in the literature to balance the power consumption of an end device by scheduling its workloads. For instance, Caruso et al. proposed a method that schedules a list of tasks by solving an optimization problem with the current allotted energy available on the device as input [21]. Audet et al. developed an algorithm that schedules recurring tasks based on the harvested energy and the cost of each task [22]. Compared to the existing work, this paper focuses on investigating how to leverage solar power forecasting to schedule the duty cycle of an end device to

maximize the number of samples it collects in each time period without running out of battery. Our work is therefore orthogonal and complementary.

Domain Adaptation aims to learn from one or multiple source domains and produce a model that performs well on a related target domain and has been successfully used in computer vision [23], natural language processing [24], industrial network configuration [25], and building occupancy estimation [26]. Recent work has focused on transferring DNN representations from a labeled source dataset to a target domain where labeled data is sparse or non-existent. The main strategy is to guide feature learning by minimizing the difference between the source and target feature distributions. MMD has been successfully used for domain adaptation, which computes the norm of the difference between two domain means (the expectations of the source and target domain) [27]. For example, Tzeng et al. used MMD in addition to the regular classification loss on the source to learn a representation that is both discriminative and domain invariant [28]. Despite the extensive literature on domain adaptation, little work has been done to investigate whether it can be applied to close the location-specified gap in solar energy generation forecasting with the teacher-student neural network. To our knowledge, our work is the first that leverages the teacher-student neural network to adapt the solar energy generation forecasting model trained from one location to another using publicly accessible data and a small number of local measurements.

VIII. Conclusions

In this paper, we show that solar energy generation forecasting plays an important role in the performance of an environmental sensing platform, formulate solar power generation forecasting as a machine learning problem, and develop a novel domain adaptation-based method to train the solar power generation forecasting model using publicly accessible data and a small number of local measurements. In addition, we develop a new method that schedules the duty cycles of an end device to maximize the number of samples it collects in each time period without running out of battery. Experimental results show that our solution outperforms the baselines and effectively supports end devices to perform environmental sensing operations without interruptions.

Acknowledgment

This work was supported in part by the National Science Foundation under grant CNS-2150010.

References

[1] K. Zhang, Y. Li, J. D. Schwartz, and M. S. O'Neill, "What Weather Variables Are Important in Predicting Heat-related Mortality? A New Application of Statistical Learning Methods," *Environmental Research*, vol. 132, pp. 350–359, 2014.

[2] A. Pagano, D. Croce, I. Tinnirello, and G. Vitale, "A Survey on LoRa for Smart Agriculture: Current Trends and Future Perspectives," *IEEE Internet of Things Journal*, vol. 10, no. 4, pp. 3664–3679, 2023.

[3] wikipedia, "LoRa," 2024-02-12, [Online; accessed 12-February-2024]. [Online]. Available: https://en.wikipedia.org/wiki/LoRa

[4] D. Magrin, M. Centenaro, and L. Vangelista, "Performance Evaluation of LoRa Networks in a Smart City Scenario," in *ICC*, 2017.

[5] M. A. Jamshed, K. Ali, Q. H. Abbasi, M. A. Imran, and M. Ur-Rehman, "IoT-Enabled Smart Energy Grid: Applications and Challenges," *IEEE Access*, vol. 9, pp. 50 961–50 981, 2021.

[6] PCsensor, "TemperHum," 2024-02-12, [Online; accessed 12-February-2024]. [Online]. Available: https://pcsensor.com/product-details?product_id=791

[7] Dragino, "Dragino Lora/GPS Hat." [Online]. Available: https://wiki1.dragino.com/index.php/Lora/GPS_HAT

[8] Semetech, "Semetech sx1276/sx1278 lora transceiver," 2024-02-12, [Online; accessed 12-February-2024]. [Online]. Available: https://www.semtech.com/products/wireless-rf/lora-connect/sx1276

[9] Pi-Supply, "PiJuice Hat," 2024-02-12, [Online; accessed 12-February-2024]. [Online]. Available: https://uk.pi-supply.com/products/pijuice-standard

[10] P. Supply, "PiJuice Solar Panel - 22 Watt," 2024-02-12, [Online; accessed 12-February-2024]. [Online]. Available: https://uk.pi-supply.com/products/pijuice-solar-panel-22-watt?_pos=3\&_sid=020388625\&_ss=r

[11] Pi-Supply, "PiJuice 12000 mAh Battery," 2024-02-12, [Online; accessed 12-February-2024]. [Online]. Available: https://uk.pi-supply.com/products/pijuice-12000mah-battery

[12] Tektyte, "Tektyte Log4Usb," 2024-02-12, [Online; accessed 12-February-2024]. [Online]. Available: https://www.tektyte.com/log4usb.html

[13] N. R. E. Laboratory, "National Solar Radiation Database," 2024-02-12, [Online; accessed 12-February-2024]. [Online]. Available: https://nsrdb.nrel.gov

[14] M. Gardner and S. Dorling, "Artificial Neural Networks (the Multilayer Perceptron)—a Review of Applications in the Atmospheric Sciences," *Atmospheric Environment*, vol. 32, no. 14, pp. 2627–2636, 1998.

[15] C.-L. Li, W.-C. Chang, Y. Cheng, Y. Yang, and B. Póczos, "MMD GAN: Towards Deeper Understanding of Moment Matching Network," *arXiv preprint arXiv:1705.08584*, 2017.

[16] A. Kansal, J. Hsu, S. Zahedi, and M. B. Srivastava, "Power management in energy harvesting sensor networks," *TECS*, vol. 6, no. 4, pp. 32–es, 2007.

[17] H. K. Qureshi, U. Saleem, M. Saleem, A. Pitsillides, M. Lestas *et al.*, "Harvested energy prediction schemes for wireless sensor networks: Performance evaluation and enhancements," *Wireless Communications and Mobile Computing*, vol. 2017, 2017.

[18] T. Gneiting, S. Lerch, and B. Schulz, "Probabilistic Solar Forecasting: Benchmarks, Post-processing, Verification," *Solar Energy*, vol. 252, pp. 72–80, 2023.

[19] K. Tamil Selvi, S. Mohana Saranya, and R. Thamilselvan, *Solar Energy Forecasting for Devices in IoT Smart Grid*. John Wiley & Sons, Ltd, 2023, ch. 13, pp. 365–394.

[20] M. Al-Omary, K. Hassini, A. Fakhfakh, and O. Kanoun, "Prediction of energy in solar powered wireless sensors using artificial neural network," in *SSD*. IEEE, 2019, pp. 288–293.

[21] A. Caruso, S. Chessa, S. Escolar, F. Rincón, and J. C. López, "Task Scheduling Stabilization for Solar Energy Harvesting Internet of Things Devices," in *ISCC*, 2022.

[22] D. Audet, L. C. de Oliveira, N. MacMillan, D. Marinakis, and K. Wu, "Scheduling Recurring Tasks in Energy Harvesting Sensors," in *INFOCOM*, 2011.

[23] M. Wang and W. Deng, "Deep Visual Domain Adaptation: A Survey," *Neurocomputing*, vol. 312, no. 27, pp. 135–153, 2018.

[24] D. W. Otter, J. R. Medina, and J. K. Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 2, pp. 604–624, 2021.

[25] J. Shi, A. Ma, X. Cheng, M. Sha, and X. Peng, "Adapting wireless network configuration from simulation to reality via deep learning-based domain adaptation," *IEEE/ACM Transactions on Networking*, 2023.

[26] T. Zhang and O. Ardakanian, "A Domain Adaptation Technique for Fine-Grained Occupancy Estimation in Commercial Buildings," in *IoTDI*, 2019.

[27] A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf, *Covariate Shift and Local Learning by Distribution Matching*. MIT Press, 2009, pp. 131–160.

[28] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, and T. Darrell, "Deep domain confusion: Maximizing for domain invariance," *arXiv preprint arXiv:1412.3474*, 2014.