Adaptive, Correlation-Based Training Data Selection for IoT Device Management

Anders Eivind Braten, Frank Alexander Kraemer and David Palma
Department of Information Security and Communication Technology
Norwegian University of Science and Technology, NTNU
Trondheim, Norway
anders.e.braten@ntnu.no

***Abstract*—Device management can enhance large-scale deployments of IoT nodes in non-stationary environments by supporting prediction and planning of their energy budget. This increases their ability for perpetual operation and is a step towards maintenance-free IoT. In this paper we consider how to accelerate the collection of relevant training data for nodes that are introduced into an existing deployment to increase the accuracy of their predictions. In particular, we investigate how nodes powered by solar energy can learn their energy intake faster and more accurately by using data from selected nodes that are working in similar conditions. We explore an architecture that utilizes different training data selection policies to manage the learning processes. For validation, we perform a case study to explore how nodes with correlated data can contribute to the learning process of other nodes. The obtained results indicate that this approach improves the accuracy of the predictions of a new node by 14 %.**

***Index Terms*—Cognitive device management, autonomous operation, adaptive energy management, solar powered devices, energy harvesting, machine learning, training data selection.**

I. Introduction

One step towards maintenance-free IoT systems is to provide devices with solar panels or other energy-harvesting power-supplies to ensure perpetual operation. Since these power sources often are stochastic in nature [1], nodes can benefit from planning their energy budget ahead [2], and hence align their power consumption with the expected incoming energy for improved overall performance. The incoming energy depends on the specific node instance, for example the type of solar panel, orientation and location, and is often non-stationary, i.e., it changes its characteristics over time. Therefore, each node requires individual adaptation, which implies to configure each device separately and at runtime [3], taking current context and previous experience into account [4], [5]. As IoT nodes are typically constrained with regard to computation power, memory and scope of data, we examine how fine-grained individual energy planning can be part of the device management for an IoT system, for instance as part of a cloud service.

The scale of IoT systems makes it unfeasible to tune the required processes manually for each device [6], which is why autonomous operation and self-adaptation are required. In [7] we have shown how the energy for a node with solar panels can be predicted using publicly available weather forecasts and relatively simple machine learning models, in an autonomous and scalable way. The prediction can be directly used by planning algorithms like the ones presented in [8], [9]. A remaining challenge is that such approaches require training data. Hence, we argue that managing the acquisition of suitable training data is an important task of device management.

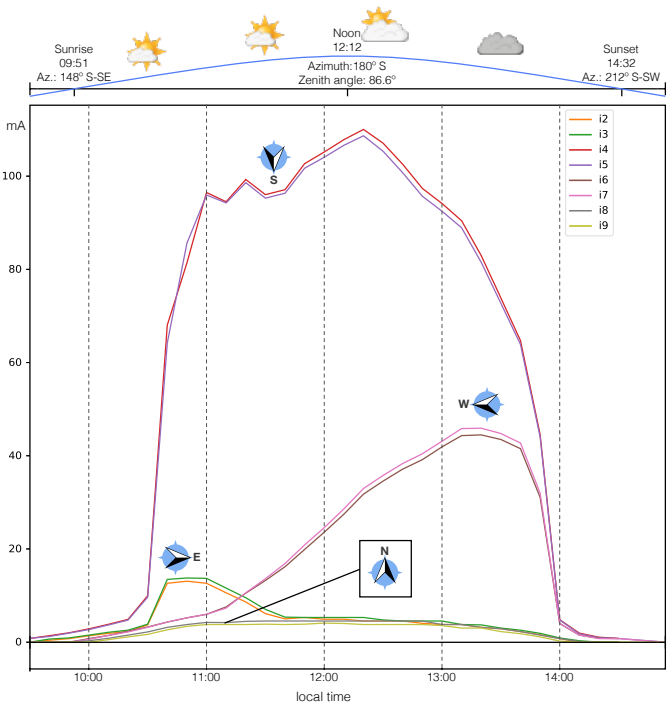

Fig. 1. Energy intake from eight solar panels located in Trondheim, Norway, on December 12th, 2018. The solar panels are pointing pairwise in four different directions.

In this paper, we investigate how solar-powered nodes introduced into an existing deployment can accelerate their learning. Our approach is to give them training data collected from selected nodes that have been working in similar situations, as illustrated in Fig. 1. This figure shows the energy intake of eight solar-powered IoT nodes facing towards four different directions. Nodes facing the same direction have similar intake profiles.

When preparing machine learning models for the energy intake prediction for a newly deployed node, taking training data from any other node indiscriminately leads to inaccurate

This is the authors' copy of the accepted article. The final version is accessible via https://doi.org/10.1109/IOTSMS48152.2019.8939220.

predictions. To temper this, we study how to identify suitable training data for a node by selecting data from nodes with correlated data. The results show that by selecting the most relevant training data based on the correlation between devices, we can train a model that decreases the error of the predictions by 14 %, compared to using data from all previously deployed nodes.

We continue with a discussion of related work in Sect. II and describe the device management architecture we designed as context for the training data selection mechanism in Sect. III. In Sect. IV, we present the experimental setup and elaborate the different selection policies in Sect. V. We then present the results and discuss them in Sect. VI and Sect. VII, respectively.

II. Related Work

Several papers focus on autonomous, adaptive device management within the IoT. Zhang et al. [10] describe the concept of *cognitive IoT*, which aims at improving performance in the network as a whole by monitoring network conditions, analyze the collected data, make intelligent decisions based on the incurred knowledge, and perform adaptive actions. Further, they propose a network architecture that makes use of cognitive nodes that have the ability to autonomously adjust their network performance to current conditions. Sheth [3] argues that future IoT management will need to handle a large variety of devices and applications, that are subject to unplanned and unexpected events and is using predictive processing to solve their problems at the edge of the network. In this light, situational awareness is important for IoT devices, to derive value from data and learn from experience. Afzal et al. [11] advocate that cognition must be extended to incorporate IoT specific design challenges, like energy harvesting, cognitive spectrum access and mobile cloud computing technologies. Vlacheas et al. [12] address the challenges of technological heterogeneity and propose a framework that shows not only how, but also why and when devices should be connected to a network. Their framework was later implemented by Sasidharan et al. [13], who added a learning- and reasoning engine that takes contextual and situational parameters in consideration in order to improve the decision process. With their architecture they investigate connectivity issues and run performance analyses, using a network of solar energy harvesting devices deployed with central coordinators to control the management. However, the main focus in these works is either on managing the network as a whole or investigating the connectivity between the devices, while we investigate the operational part of management of the individual devices with the aim of achieving self-management.

We can also find relevant research in the domain of autonomous operation for IoT devices located in non-stationary environments. Wu et al. [14] argue that cognitive mechanisms should be used for more than network management and connectivity. Their main argument is that in an IoT framework, objects need to have the capability to reason about their physical and social environment in an independent fashion. They provide a conceptual framework based on a perception-action cycle, data analytics, knowledge discovery, decision making and service provisioning. In their framework, physical and virtual things are represented as agents that enable smart resource allocation, automatic network management and intelligent service provisioning through interaction. Foteinos et al. [4] state that support for smart and self-adaptive applications and objects are factors that need to be in place before the process of connecting heterogeneous IoT devices can be managed in a dependable, scalable and autonomous manner. To solve this, they present a cognitive management framework that adapts the configuration and behavior of devices according to the current status and context. They also show that their framework is able to improve situation awareness, reliability, and energy efficiency of IoT applications. This is in alignment with our work. However, their focus is to overcome the technological heterogeneity and complexity of the underlying networks and IoT infrastructure, while we look at the heterogeneity and complexity that is found in the physical environments of the devices. Preden et al. [5] stress the importance of situation awareness and attention when monitoring overall system performance in a dynamically changing environment, since selecting the most proper action in a given situation is highly dependent on the context of the device in question. They propose a conceptual architecture that explores these aspects in a self-aware health monitoring prototype and show that both are critical to self-awareness.

We found few works addressing the process of selecting training data for machine learning or the mechanisms that are needed to achieve this. Han et al. [15] demonstrate the challenge of estimating the disturbance covariance matrix based on a secondary data set, when the number of secondary datasets is large and the data is heterogeneous due to non-stationary environments. This results in a combinatorial problem that is computationally expensive or even infeasible. To mitigate this, they present an algorithm based on the minimal covariance determinant that chooses training data with similar disturbance properties and discards vectors that contain possible outliers. Fraternali et al. [16] discuss the challenge of tuning individual IoT devices for perpetual performance, when there is a need to adapt to changing environmental conditions. Their approach focuses on autonomous configuration of learning algorithms on constrained devices, based on identifying the environmental context for solar-powered devices. They argue that it is unfeasible to train a different reinforcement learning policy for each individual node. Instead, they propose to use a single policy for nodes that share similar lighting conditions. In a case study, they conduct an indoor experiment that shows the performance of the devices dropping significantly when using a single policy across all devices, due to the differences in lightning conditions. This means that identify devices that experience similar conditions is an important task when managing a large number of devices. However, they only state that auto-configuration is an important aspect, but do not answer the problem of identifying devices that experience similar conditions.

III. An Architecture for Managing Learning and Planning Processes

To address the problem of large-scale, maintenance-free IoT systems, we explore an architecture that manages the learning and planning processes on behalf of connected devices and sensor nodes autonomously. With this approach we can empower constrained IoT devices with the ability to adapt to different situations occurring in their environment. This opens a path towards cloud-based, *cognitive* device management platforms [12], which in turn is a step towards the vision of self-managed computing systems [17] for IoT.

According to Vernon [18], autonomous operation in non-stationary environments requires that devices have the ability to see themselves in relation to their context, learn from experience, predict the outcome of future events, act to pursue goals and adapt to changes in the environment. Vernon refers to this as *artificial cognitive systems*. Fig. 2 shows an abstract model of the underlying cognitive process. It contains two cycles, a perception-action cycle and a cycle of learning, predicting and adaptation through planning. The planning activity is in the center of the perception-action cycle, binding the two cycles together. The architecture for device management presented in this paper is based on these principles. The main elements are autonomous agents hosted in the device management as part of a cloud or fog computing service. They provide constrained IoT nodes with the capability to create plans to handle future events, and thus adapt to different situations occurring in their environment [19].

To illustrate the concept, we designed an architecture that models the behavior of a cognitive device manager responsible for energy planning for solar-powered, constrained devices. Figure 3 shows a diagram of the architecture. It is built around two components:

- The *planning manager* represents the perception-action cycle seen in Fig. 2, centered around the planning component in the associated learn-predict-plan cycle. Its main responsibility is to keep track of the status of the devices, observe events occurring in their environment and act if there is a need to adapt by sending new configurations, for instance by adjusting the power consumption to the predicted energy intake. The actual configurations are handled by an internal configuration manager. The need to adapt may be caused by a change in predictions, or by sudden or planned events detected by the planning manager.
- The *learning manager* is responsible for handling the data, policies and actions needed for managing the learning and prediction processes. It represents the learn-predict-plan cycle in Fig. 2 and have two sub-managers. The prediction manager is charged with the task of training the machine learning models and produces the actual predictions. It may contain a number of different models, depending on the purpose of the system. In our case, it has access to models for predicting the solar energy intake, the energy consumption and the energy buffer. The training data manager analyses training data and evaluates the accuracy of different subsets of training data, with the aim of feeding the most relevant data for training a model to the prediction model manager. This task is important to support devices operating in a non-stationary environment.

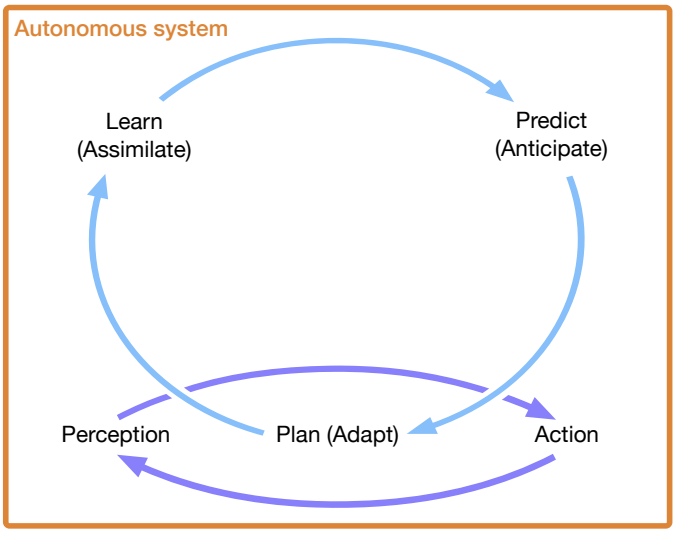

Fig. 2. Abstract model of an autonomous system, based on a perception-action cycle, which in turn is contributing to and maintained by a second cycle of learning, predicting and planning. Adapted from [18].

To ensure adaptation, all managers have mechanisms to trigger different decisions, shown as $T_1 ... T_5$ in Fig. 3. For the planning manager, the trigger is a decision to send new configurations to a device when there is a need to adjust energy consumption to the anticipated energy harvest. The trigger for a prediction model manager fires if the performance of its model decreases significantly, which may indicate a change in the environment. For the training data manager, the trigger is an assessment that a different subset of training data will produce more accurate predictions for a given model. This change will cause the prediction model manager to select that training data subset next time it trains that particular model.

In the following section, we focus on the internal mechanisms of the training data manager. To this end we have performed a case study to explore how to identify suitable training data for a node by selecting data from nodes with correlated data.

IV. Experimental Setup

A. Data Collection

We collect data from a testbed consisting of eight solar panels, with two panels facing east, south, west and north, respectively. A ninth panel is mounted horizontally, i.e., in plane with the ground, for reference. The setup of these panels is shown in Fig. 4. From the panels we collect data about the actual energy that is produced. Previously, we have identified that the position of the sun and the amount of clouds that is blocking and scattering the direct sunlight are the two

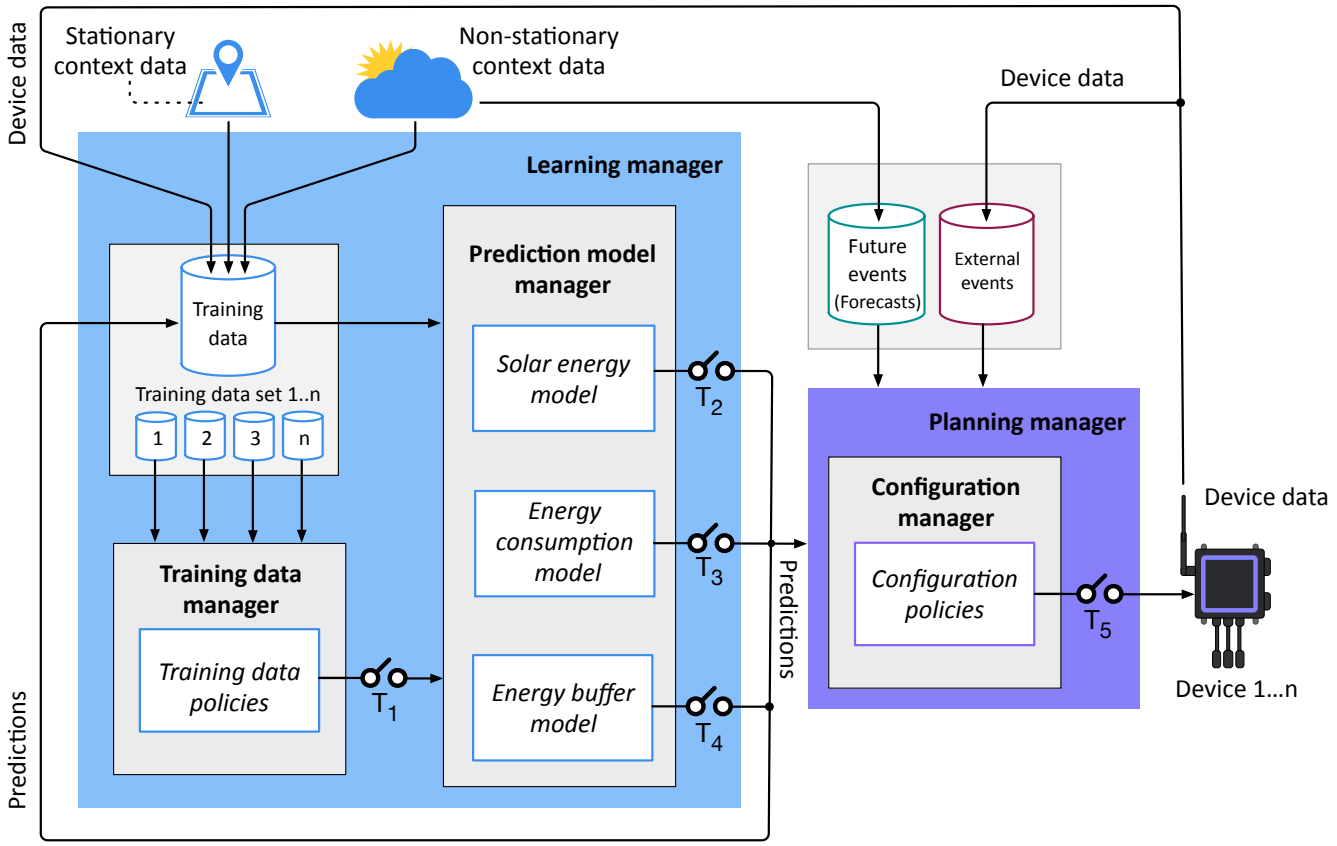

Fig. 3. Architecture of a cognitive device manager responsible for energy planning for constrained IoT devices.

most important features for predicting the energy produced by a solar panel [7], [20]. The sun position is represented by the features *zenith* and *azimuth*. Cloudiness at three different altitudes is obtained from a public weather forecast service and added to the dataset. Finally, we resample the data to ten-minute intervals.

Our observations are based on data collected between October 12th, 2018 and May 9th, 2019 [21]. To ensure that the data is useful and relevant to the experiment, we do some data cleaning. Data points where none of the panels register any energy intake are removed, since only periods with actual energy production are relevant. Snow covering the solar panels causes noise in the training data, so we also remove those days from the dataset. In the period when the sun is lowest, cloudy weather may cause very few data points to be registered during a day. This makes it hard to find correlations. We therefore take away days where the number of collected data points is below a 150, which is a manually selected threshold. This results in a dataset that consists of a total of 168 days.

The data is then organized in nine overlapping subsets. This increases the number of observations and make it possible to find patterns and generalize the results. Each set is made up of 28 days with initial training data and 28 days where we train the model and test the algorithm. We define the start of period P_{n+1} to be 14 days after period P_n.

The training data is collected from nodes I_2, I_4, I_6 and I_8, pointing east, south, west and north, respectively. In addition, we add data from the horizontal panel I_0.

At day 29, we deploy four additional devices, named I_3, I_5, I_7 and I_9, which are oriented pairwise in the same direction as I_2, I_4, I_6 and I_8, respectively. These are the nodes we want to predict the energy intake for. We then monitor the energy intake for all nodes for 28 days. This makes up the test period. Thus, each set is made up of data from 56 days.

The result can be seen in Table I. Columns *1st-* and *2nd batch deployment* show the dates for the first and second deployment, respectively, while column $Zenith_{noon}$ shows the zenith on the date of the second deployment. Note that high zenith values mean that the sun elevation is low.

B. Machine Learning Models

For the prediction models, we use a random forest regressor from scikit-learn [22]. We choose this model since earlier experiments has proved it to be suitable [7], [20]. In addition, the model is relatively fast to train, which is an important factor when we need to train several models for each device, possibly on an agent located close to the edge of the network.

One of the limitations of statistical learning algorithms is that they are unable to extrapolate beyond the range of data that has been used to train the models [23]. To temper this, we train the models regularly. This way, previously unseen weather conditions and seasonal changes in sun position are

Fig. 4. Setup of the nine solar panels.

TABLE I
OVERVIEW OF DATES USED TO DEFINE PERIODS

Period	Weeks	1st batch deployment $I_0, I_2 ... I_8$	2nd batch deployment $I_3, I_5 ... I_9$	$\text{Zenith}_{\text{noon}}$ (2nd depl. date)
P_1	Week 1-4	2018-10-12	2018-11-09	80.65°
P_2	Week 3-6	2018-10-26	2018-11-23	84.05°
P_3	Week 5-8	2018-11-09	2018-12-12	86.61°
P_4	Week 7-10	2018-11-23	2019-01-02	86.37°
P_5	Week 9-12	2018-12-12	2019-01-26	82.20°
P_6	Week 11-14	2019-01-02	2019-02-15	76.17°
P_7	Week 13-16	2019-01-26	2019-03-04	69.94°
P_8	Week 15-18	2019-02-15	2019-03-28	60.53°
P_9	Week 17-20	2019-03-04	2019-04-12	54.85°

added to and reflected in the training data. Thus, the learning process is handled autonomously and continuously.

C. Metric for Prediction Performance

To assess the accuracy of the predictions, we need a metric suitable to compare the prediction performance in various seasons. With $a_{i,d,n}$ we denote the n-th value measured for solar energy of sensor i on day d, and with $p_{i,d,n}$ a prediction for the corresponding value. Since measurement points are equally spaced in time, we calculate the total energy collected during a day by summing over the individual measurements:

$$E_{total}(i,d) = \sum_n a_{i,d,n}$$

and likewise, for the prediction:

$$\hat{E}_{total}(i,d) = \sum_n p_{i,d,n}$$

We further calculate the exponentially weighted moving average (EWMA) of the daily measured energy:

$$s(i,d) = \begin{cases} E_{total}(i,1), & d = 1 \\ \alpha \cdot E_{total}(i,d) + (1-\alpha) \cdot s(i,d-1), & d > 1 \end{cases}$$

With $\alpha = 0.095$ we consider the average of the last 20 days. As an error metric for each day, we consider the scaled absolute difference between the total energy predicted and observed:

$$STAPE(i,d) = \frac{100}{s(i,d)} \left| E_{total}(i,d) - \hat{E}_{total}(i,d) \right|$$

We call this the scaled total absolute percentage error, STAPE. It describes the error of the total daily energy in percent, relative to the energy one expects on average at that day. For example, a STAPE of 20 % means that the prediction was 20 % off the actual value, relative to the average daily energy $s(i,d)$ that corresponds to 100 %. This scaling allows to compare prediction performances from different seasons, where the total energy varies considerably. At the same time, by using the EWMA, it prevents outliers on days with exceptionally low solar energy.

V. TRAINING DATA SELECTION POLICIES

In Figures 5 and 6 we can see the weather conditions, the energy intake of all nine nodes, and a scatter diagram showing the correlation between the energy intake of the nodes at the day when we deploy the second batch of nodes, for periods 8 and 9, respectively. From the weather symbols and energy intake pattern, we can see that on March 28th the weather was volatile with both cloudy and sunny periods, while on April 4th the conditions were stable with sunny weather.

Looking at the two correlation matrices, we see that for the day with volatile weather the correlation between the nodes has a high variance (the data-points are spread), while it has less variance on the day with stable weather (the data-points form a curved line). However, in both diagrams we see that for the nodes that are pointed pairwise in the same direction, the correlation graph can be seen as a straight line with a constant slope close to 1. This is a sign that the energy intake between these nodes has a high positive correlation both in stable and unstable conditions. Using Pearson's correlation, we can express this as a single number between 1 and -1, where 1 means the data are fully positively correlated, while -1 means the data are completely negatively correlated:

$$r(i_1,i_2,d) = \frac{\sum_n (a_{i_1,d,n} - \bar{a}_{i_1,d})(a_{i_2,d,n} - \bar{a}_{i_2,d})}{\sqrt{\sum_n (a_{i_1,d,n} - \bar{a}_{i_1,d})^2 \sum_n (a_{i_2,d,n} - \bar{a}_{i_2,d})^2}}$$

where $\bar{a}_{i,d}$ and $\bar{p}_{i,d}$ are the daily averages of the actual measured values and the predictions, respectively.

Using the correlation, we define the selection policy CORR-MOD. To provide comparison, we also define three other policies SELF-MOD, REF-MOD and ALL-MOD, that obtain training data using other methods for training data selection. In addition, we define a control algorithm CONTROL-MOD as a baseline. The total set of selection policies is then:

- The CORR-MOD policy collects data from a single, previously deployed device that displays the highest correlation with the newly deployed device on the date of deployment, starting from the date of the first deployment until the date of the second deployment. The data produced by the device itself is then collected for the second half of the period.

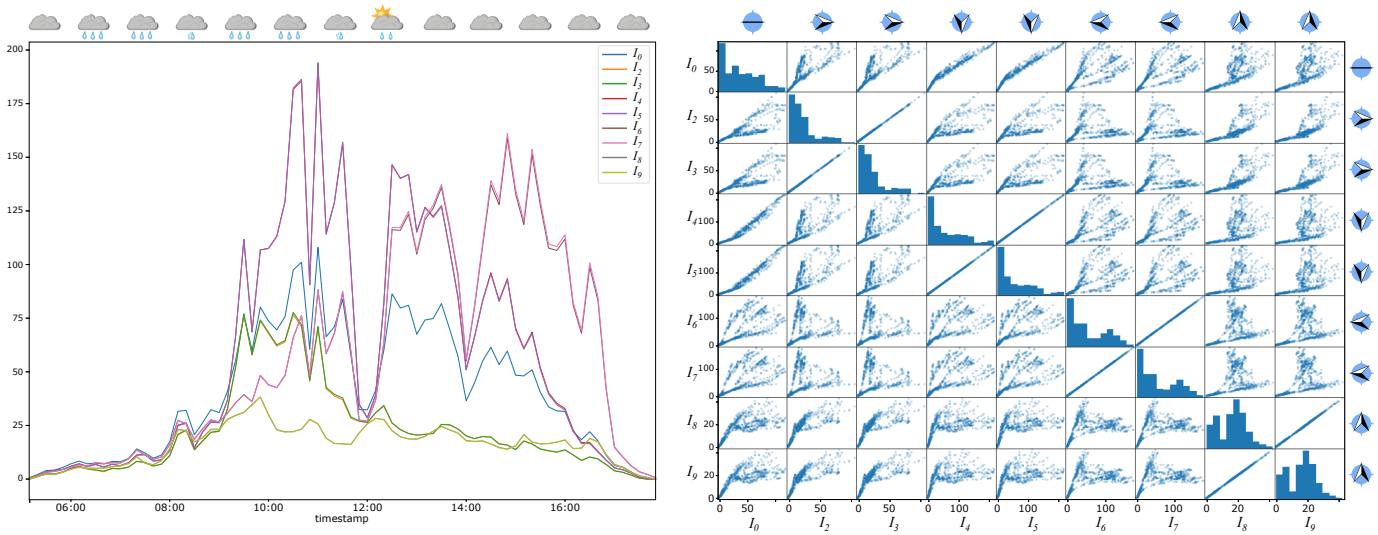

Fig. 5. Weather conditions, energy intake and correlation graphs of all nodes for March 28th, 2019.

- The SELF-MOD policy collects data from the device itself, starting from the date of the second deployment.
- The REF-MOD policy collects data from the previously deployed reference node I_0, from the date of the first deployment until the date of the second deployment. The data produced by the device itself is then collected for the second half of the period.
- The ALL-MOD policy collects data from all previously deployed devices (I_0, I_2, I_4, I_6 and I_8), starting from the date of the first deployment. From the date of the second deployment it adds the data collected by the device itself.
- The CONTROL-MOD policy resembles SELF-MOD, except it is given data collected from the entire period, starting at the date of the first deployment. Thus, it uses future knowledge not available in a real setting.

All five policies are applied once for each of the four devices deployed in the second batch (I_3, I_5, I_7 and I_9), before we use the models for predicting future energy intake. This results in five series of predictions, for each device and each day. Each series of predictions is then assessed by the STAPE metric. This results in one measure for each selection policy, for each device, for each day. Since we want to compare the overall accuracy of the models that we train, the next step is to calculate the arithmetic mean of the STAPE for each policy, for all devices during the whole test period. Lastly, we calculate the arithmetic mean for all periods. Thus, we end up with five measures for each period, plus five measures representing all nine periods, as shown in Table II.

VI. Results

Table I shows the zenith at noon, on the day we deploy the second batch of devices. Figures 5 and 6 show the weather condition, energy intake and the correlation graphs of the deployed nodes, for two of those days. From this we can identify some of the challenges related to selecting training data for IoT devices working in a non-stationary environment. Firstly, seasonality and volatile weather conditions have a large influence on the number of data points that is collected on a given day. A high value for the zenith means that we have fewer hours of daylight, which again means that there is less data collected. Also, heavy clouds might block the sun completely, and thus further decrease the number of data points. Secondly, we see that the correlation of the data is more spread on days with volatile weather than on days with stable conditions. Even so, on days with stable weather and plenty of sun, the orientation of the solar panels has a great influence on how well the data correlates. These observations indicate that transferring training data indiscriminately between devices operating in a non-stationary environment should be avoided.

Table II shows the mean STAPE for each policy for each 4-week period and for all periods overall. We see that the prediction accuracy given by this metric is highly dependent on the training data that is fed to the machine learning algorithm. If we look at all periods as a whole, the mean STAPE of the predictions is lowest for CORR-MOD, i.e., the model that is fed training data from a node that displays high correlation with a newly deployed node (36.32 %). This means it has the highest overall accuracy. It also closely resembles CONTROL-MOD, which is as expected since the panels used to collect training data for these two policies are pointing pairwise in the same direction. For the three other models the performance is less accurate, but on about the same level, with a STAPE of 42.45 %, 42.44 %, and 43.83 %, respectively. When we calculate the percentage decrease in STAPE of CORR-MOD (36.32 %) compared to ALL-MOD (42.44 %), the second-best overall selection policy, we find that the CORR-MOD policy improves prediction accuracy by around 14 %.

For all four models, the accuracy is worst in the 4th period (P_4). This is when the sun elevation at noon is lowest, that is, when we have the fewest hours of daylight and when the beams from the sun hit the solar panels from the lowest angle.

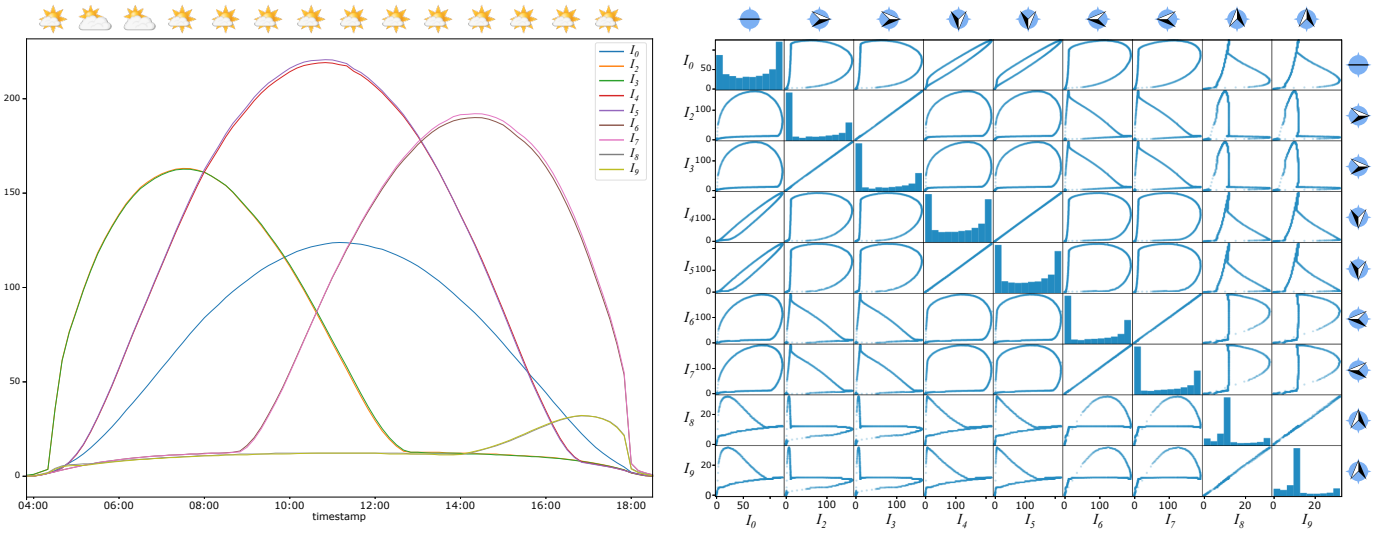

Fig. 6. Weather conditions, energy intake and correlation graphs of all nodes for April 12th, 2019

TABLE II
PERFORMANCE OF FOUR RFR-MODELS, TRAINED ON DIFFERENT SETS OF TRAINING DATA

Selection Policy	Mean STAPE for each 4 week period									Mean STAPE for all periods
	Week 1-4	3-6	5-8	7-10	9-12	11-14	13-16	15-18	17-20	
SELF-MOD	35.67 %	51.04 %	45.94 %	*73.98 %*	**46.65 %**	**44.70 %**	**37.51 %**	29.11 %	17.49 %	42.45 %
REF-MOD	*24.49 %*	36.03 %	50.91 %	79.61 %	56.90 %	52.36 %	48.21 %	29.79 %	16.13 %	43.83 %
CORR-MOD	**23.82 %**	**25.01 %**	**30.06 %**	**65.96 %**	*50.11 %*	*45.19 %*	*46.19 %*	**24.74 %**	*15.73 %*	**36.32 %**
ALL-MOD	28.85 %	*30.53 %*	*39.83 %*	77.81 %	61.83 %	52.97 %	50.17 %	25.95 %	**14.00 %**	*42.44 %*
CONTROL-MOD	25.44 %	24.91 %	32.96 %	66.54 %	54.11 %	48.56 %	47.75 %	26.33 %	16.06 %	38.07 %

Surprisingly, the results indicate that in periods where training data is collected while the sun is at the lowest (P_5, P_6 and P_7), the overall accuracy is best for SELF-MOD, i.e., the model that is *not* given any extra training data. However, CORR-MOD still has the next best overall accuracy in these three periods.

VII. DISCUSSION

In some periods the best training data selection policy is the one where the model is trained from scratch, without feeding the machine learning model any extra training data. There can be several reasons for this behavior:

1) For these specific periods, there are relatively few data points per day in the transferred training data. Thus, predictions are based on fewer observations, which in turn might lower the accuracy.
2) As seen in Table I, if we look at the two periods preceding the deployment of P_4, P_5 and P_6, respectively, we see that the zenith at noon is near 90°, that is, the sun is low. These are the periods used to collect extra training data. Meanwhile, for the days being predicted, the zenith at noon is rapidly decreasing, that is, the sun height is increasing. Since the zenith is the most important feature for predicting the energy intake [7], this means that the training data has little relevance for the predictions made in these periods.
3) The weather conditions in November and December in Trondheim are often volatile and can change from one hour to the next. This might introduce noise in the training sets.
4) Both the seasonality and the weather conditions cause more light to be scattered, and thus less energy hits the solar panels directly. This has a big influence on how much energy that can be harvested.

Since all these explanations are closely connected to the training data used to produce the predictions, it supports the hypothesis that selecting training data for transfer learning is an important task for an architecture that handles non-stationary environmental data.

The results are especially interesting when we look at them from an architectural point of view. Self-configuration, self-optimization, self-healing and self-protection, are important aspects of autonomic computing [17]. The proposed training data selection can support these aspects in several ways:

- By adding relevant learning data when training a model, it is possible to improve the prediction accuracy in the first period after deployment. This will improve the systems

ability to perform self-optimization.

- If for some reason a node is unable to report the data that is collected, data collected from a correlated node can be used to substitute the missing data. This will improve the self-healing of the system.
- By applying methods for comparing data from two or more correlated devices, we can detect nodes that suddenly deviate from expected behavior, which might indicate that the device in question is faulty or hi-jacked. This will improve the overall ability for self-protection.

Thus, the case study also illustrates how training data selection and continuous learning is a possible method for improving the ability of constrained IoT devices to adapt to changes and act more autonomously.

VIII. Conclusion

We discussed a step towards maintenance-free IoT device management for large deployments of constrained IoT nodes working in non-stationary environments. In particular, we have investigated how solar-powered nodes that are introduced into an existing deployment can accelerate their learning by giving them training data collected from selected nodes that are working in similar situations. To illustrate the problem, we designed an architecture that models the behavior of a cognitive device manager that is responsible for energy planning for solar-powered constrained devices. For validation, we performed a case study where we studied how to identify suitable training data for a node by selecting data from nodes with correlated data. The experiment and discussion were based on real data collected under realistic conditions. Our results indicate that by using our data training selection algorithm we can train a model that decreases the error of the predictions by 14 %, compared to using data from all previously deployed nodes. This shows that managing the acquisition of suitable training data is an important task of device management when new devices are deployed and introduced into an existing system.

References

[1] C. Moser, L. Thiele, D. Brunelli, and L. Benini, "Adaptive Power Management for Environmentally Powered Systems," *IEEE Transactions on Computers*, vol. 59, no. 4, pp. 478–491, 2010.

[2] B. Buchli, F. Sutton, J. Beutel, and L. Thiele, "Dynamic power management for long-term energy neutral operation of solar energy harvesting systems," in *Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems*, 2014, pp. 31–45.

[3] A. Sheth, "Internet of things to smart iot through semantic, cognitive, and perceptual computing," *IEEE Intelligent Systems*, vol. 31, no. 2, pp. 108–112, 2016.

[4] V. Foteinos, D. Kelaidonis, G. Poulios, P. Vlacheas, P. Demestichas, and V. Stavroulaki, "Cognitive management for the internet of things: A framework for enabling autonomous applications," *IEEE Vehicular Technology Magazine*, vol. 8, no. 4, pp. 90–99, 2013.

[5] J. S. Preden, K. Tammemae, A. Jantsch, M. Leier, A. Riid, and E. Calis, "The Benefits of Self-Awareness and Attention in Fog and Mist Computing," *Computer*, vol. 48, no. 7, pp. 37–45, 2015.

[6] I. Chatzigiannakis, H. Hasemann, M. Karnstedt, O. Kleine, A. Kroller, M. Leggieri, D. Pfisterer, K. Romer, and C. Truong, "True self-configuration for the IoT," in *2012 3rd International Conference on the Internet of Things (IOT)*. IEEE, 2013, pp. 9–15.

[7] F. A. Kraemer, D. Ammar, A. E. Braten, N. Tamkittikhun, and D. Palma, "Solar energy prediction for constrained IoT nodes based on public weather forecasts," in *Proceedings of the Seventh International Conference on the Internet of Things*. ACM Press, 2017, pp. 1–8.

[8] S. Shresthamali, M. Kondo, and H. Nakamura, "Adaptive Power Management in Solar Energy Harvesting Sensor Node Using Reinforcement Learning," *ACM Transactions on Embedded Computing Systems*, vol. 16, no. 5s, pp. 1–21, Oct. 2017.

[9] A. Murad, F. A. Kraemer, K. Bach, and G. Taylor, "Management of energy-harvesting iot nodes using deep reinforcement learning," *13th IEEE International Conference on Self-Adaptive and Self-Organizing Systems (SASO 2019)*, 2019.

[10] M. Zhang, H. Zhao, and R. Zheng, "Cognitive internet of things: concepts and application example," *IJCSI International Journal of Computer Science Issues*, vol. 9, no. 3, 2012.

[11] A. Afzal, S. A. R. Zaidi, M. Z. Shakir, M. A. Imran, M. Ghogho, A. V. Vasilakos, D. C. McLernon, and K. Qaraqe, "The Cognitive Internet of Things: A Unified Perspective," *Mobile Networks and Applications*, vol. 20, no. 1, pp. 72–85, Feb. 2015.

[12] P. Vlacheas, R. Giaffreda, V. Stavroulaki, D. Kelaidonis, V. Foteinos, G. Poulios, P. Demestichas, A. Somov, A. Biswas, and K. Moessner, "Enabling smart cities through a cognitive management framework for the internet of things," *IEEE Communications Magazine*, vol. 51, no. 6, pp. 102–111, 2013.

[13] S. Sasidharan, A. Somov, A. Biswas, and R. Giaffreda, "Cognitive management framework for Internet of Things:—A prototype implementation," in *2014 IEEE World Forum on Internet of Things (WF-IoT)*. IEEE, 2014, pp. 538–543.

[14] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, and K. Long, "Cognitive Internet of Things: A New Paradigm Beyond Connection," *IEEE Internet of Things Journal*, vol. 1, no. 2, pp. 129–143, 2014.

[15] S. Han, A. De Maio, V. Carotenuto, and X. Huang, "A novel radar training data selection method based on the minimal covariance determinant criterion," in *International Conference on Radar Systems (Radar 2017)*, 2017.

[16] F. Fraternali, B. Balaji, and R. Gupta, "Scaling configuration of energy harvesting sensors with reinforcement learning," in *Proceedings of the 6th International Workshop on Energy Harvesting & Energy-Neutral Sensing Systems*, ser. ENSsys '18. New York, NY, USA: ACM, 2018, pp. 7–13.

[17] J. O. Kephart and D. M. Chess, "The vision of autonomic computing," *Computer*, vol. 36, no. 1, pp. 41–50, Jan. 2003.

[18] D. Vernon, *Artificial cognitive systems: A primer*. MIT Press, 2014.

[19] Z. Sheng, H. Wang, C. Yin, X. Hu, S. Yang, and V. C. M. Leung, "Lightweight Management of Resource-Constrained Sensor Devices in Internet of Things," *IEEE Internet of Things Journal*, vol. 2, no. 5, pp. 402–411, 2015.

[20] A. E. Braten and F. A. Kraemer, "Towards Cognitive IoT: Autonomous Prediction Model Selection for Solar-Powered Nodes," in *2018 IEEE International Congress on Internet of Things (ICIOT)*. IEEE, 2018, pp. 118–125.

[21] A. E. Braten, "Dataset for solar energy intake correlated with weather forecast data (Version v1.0.0) [Data set]," Jun. 2019. [Online]. Available: http://doi.org/10.5281/zenodo.3386297

[22] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[23] G. J. Bowden, H. R. Maier, and G. C. Dandy, "Real-time deployment of artificial neural network forecasting models: Understanding the range of applicability," *Water Resources Research*, vol. 48, no. 10, p. 480, Oct. 2012.