Non-Intrusive Load Identification for Smart Outlets

Sean Barker, Mohamed Musthag, David Irwin, and Prashant Shenoy
University of Massachusetts Amherst
[sbarker,musthag,shenoy]@cs.umass.edu, irwin@ecs.umass.edu

***Abstract*—An increasing interest in energy-efficiency combined with the decreasing cost of embedded networked sensors is lowering the cost of outlet-level metering. If these trends continue, new buildings in the near future will be able to install "smart" outlets, which monitor and transmit an outlets power usage in real time, for nearly the same cost as conventional outlets. One problem with the pervasive deployment of smart outlets is that users must currently identify the specific device plugged into each meter, and then manually update the outlets meta-data in software whenever a new device is plugged into the outlet. Correct meta-data is important in both interpreting historical outlet energy data and using the data for building management. To address this problem, we propose *Non-Intrusive Load Identification* (NILI), which *automatically* identifies the device attached to a smart outlet without any human intervention. In particular, in our approach to NILI, we identify an intuitive and simple-to-compute set of features from time-series energy data and then employ well-known classifiers. Our results achieve accuracy of over 90% across 15 device types on outlet-level energy traces collected from multiple real homes.**

I. Introduction

Despite recent advances in energy-efficiency, residential and commercial buildings continue to be responsible for over 40% of the energy consumption in the United States, which is significantly more than the other two broad sectors of energy consumption—industry and transportation [1]. As a result, even small improvements in building energy-efficiency, if implemented across many buildings, have the potential to substantially improve energy-efficiency. This potential has led to an increased interest in the design of "smart buildings," which dynamically regulate energy usage to optimize their energy-efficiency, e.g., by responding to changes in building occupancy, electricity prices, renewable generation, etc.

The foundation of smart building design is access to timely, accurate, and fine-grained data about a building's operations, including its energy usage and occupancy information. Put simply, the more data a building management system has at its disposal, the more opportunities for optimization it can identify and act on. Ideally, a building management system would reliably receive accurate fine-grained power data in real-time from each of a building's individual electrical loads.[1] Such data would enable analyses that both infer characteristics of the building's environment, e.g., occupancy [2], [3], and inform fine-grained scheduling of loads to improve efficiency [4], [5]. For example, in the latter case, a building management system might identify high demand periods and then automatically defer elastic loads, such as a washing machine or dryer, to low demand periods. Utilities often provide price incentives for customers to perform this type of adaptive load scheduling.

Achieving the ideal above is challenging, since even a small home may contain 30 or more individual outlets [6], with larger commercial buildings containing hundreds. In the past, real-time metering of tens to hundreds of individual outlets required specialized equipment and was cost prohibitive. As a result, researchers focused largely on analysis techniques for disaggregating data from smart meters, which monitor an entire building's energy usage, to infer the energy usage of individual loads [7]. Unfortunately, disaggregation, which is also referred to as Non-Intrusive Load Monitoring (NILM), remains largely inaccurate when applied to even small buildings [7].

Recently, however, the increasing interest in energy-efficiency combined with the decreasing cost of embedded networked sensors has lowered the cost of outlet-level metering. If these trends continue, in the near future, we expect new buildings to be able to install "smart" outlets, which monitor and transmit an outlet's power usage in real time, for nearly the same cost as conventional "dumb" outlets. Examples of smart outlet-level meters that are now widely available commercially include the Belkin WeMo Insight Switch [8], the Insteon iMeter Solo [9], the Budderfly controllable outlet [10], and the Z-Wave Smart Energy Switch [11]. Typically, companies also provide dashboard software to collect and record outlet-level data, and allow users to view it. As a whole, the mainstream home automation sector, which includes both the low-cost smart sensors above, as well as outlets capable of remote load control, is expected to grow by 60% from 2012 to 2018 [12].

While the widespread deployment of low-cost outlet-level meters will provide new visibility into building energy consumption, it also raises new challenges related to managing a large and diverse sensor deployment. In particular, since the meters above are built into general-purpose outlets, rather than devices themselves, users must manually identify each specific device plugged into each meter and then update the outlet's meta-data in the dashboard software any time someone plugs a new device into an outlet. For example, if someone plugs a toaster into an outlet, a user must manually associate the outlet with the toaster. The correct association is important, since an particular device might also be associated with other useful attributes, such as its degree of scheduling flexibility or its peak power consumption. While some static outlets may power the same device for long periods of time, as with a refrigerator, many outlets are dynamic and frequently changing due to the use of transient devices, including laptops, vacuums, seasonal air conditioners, and niche kitchen appliances. Even for static outlets attached to the same device, users must still correctly enter the device's name into the dashboard software during setup, which often requires manually recording an obscure outlet identifier—often printed on the back of each outlet—with each device prior to installation.

Such manual identification is both cumbersome and error-prone: users often do not enter any per-outlet meta-data, and whatever meta-data they do enter is either too general to be useful, e.g., "living room outlets," or is never updated and

[1] We use the term electrical load, or simply *load*, to refer to any distinct appliance or device that consumes electricity.

quickly becomes stale. Ultimately, meta-data errors reduce the usefulness of the data to automated management systems and operators. Thus, rather than require users to manually enter device meta-data, we propose a technique for *Non-Intrusive Load Identification* (NILI) that *automatically* identifies new devices plugged into smart outlets without any user intervention. NILI is inspired by the well-studied NILM problem. However, rather than analyze building energy data to disaggregate it into data for individual loads, as with NILM, NILI analyzes outlet energy data to identify the type of load plugged into the outlet.

While there has been substantial work on NILM [7], [13], [14], the NILI problem, which is becoming increasingly relevant, has received little attention. Furthermore, we expect NILI to continue to remain relevant in the future, since embedding sensing into general-purpose outlets is more cost-effective than relying on sensors being embedded into devices themselves (i.e., self-reporting devices). While the sensor meta-data does not change in the latter case, since each sensor is tightly coupled to the device, this approach requires a sensor for each device rather than a sensor for each outlet. In addition, outlets are easily standardized during building construction and management, while reliance on third-party manufacturers for device-level support is likely to introduce additional complexities, e.g., differing sensor hardware capabilities, data formats, and network protocols.

Our approach to NILI is based on the categorization of outlet-level (or for large appliances, circuit-level) timeseries power data using standard classification techniques. We first perform training on a set of input devices representing basic device types to construct the classifier. The input classes may either be models for specific device types, e.g., a specific type of GE refrigerator, or general models of broad device classes, e.g., a generic refrigerator of any type. During runtime, we periodically ingest recent data from each smart outlet to the classifier and then update the mapping in the outlet meta-data table based on the classifier's output, which specifies the type of device plugged into the outlet. In this paper, we describe the set of features we use for each type of device, as well as our choice of classifiers. We then evaluate each classifier using a dataset of labeled device energy data collected from several homes, and consider identification of both previously seen and unseen devices. We find that our classifier can achieve accuracy of over 90% on our sample dataset, even with a relatively small and straightforward set of classification features.

II. Problem Statement and Approach

Formally, we define the NILI problem for a smart outlet O_i as inferring the name of the device d_j plugged into O_i at time t, given the outlet's average power usage $p_i(t)$ each $(t-\tau, t]$. Equivalently, NILI computes the function $O_i(t) \in \{name_j\}$ $\forall j$ and $t > 0$, given $p_i(t)$. NILI assumes a table ("database") of known devices and the key energy characteristics (e.g., distinguishing features) of each device. The table may be either general, including only coarse features that distinguish one type of device from another, or highly specific, including detailed features that distinguish two different models of the same device. In addition to the features, the table may also include other meta-data associated with the device useful for a building management system, such as a device's peak power (which may also be a feature) or its degree of elasticity, i.e., how far in the future a scheduler may shift its power usage without violating its operating constraints, such as maintaining temperature within a specified guardband.

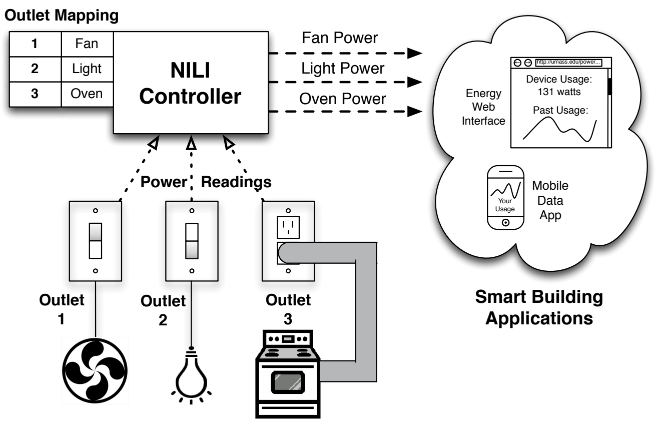

Fig. 1. Software and hardware architecture for NILI-enabled smart building.

As might be expected, selecting the important features for each device is critical; we discuss feature selection in the next section. A smart outlet's sampling interval τ also affects NILI accuracy. In general, a longer sampling interval "averages out" features in $p_t(t)$, revealing fewer identifiable features and decreasing NILI's accuracy, while a shorter interval reveals more identifiable features and increases accuracy. Our work specifically targets the consumer-grade smart outlets mentioned in the previous section, which commonly provide a sampling resolution on the order of seconds, e.g., $\tau \sim$ seconds.

Figure 1 depicts the software and hardware architecture for a building management system that includes a NILI controller, which dynamically updates the meta-data for each of the building's outlets. Specifically, for each of $1 \ldots k$ smart outlets, the controller continuously receives energy data transmitted by the outlet and analyzes it to determine the attached device and update the device name associated with the outlet in the meta-data table. The table represents only the current mapping of outlets to devices; our NILI controller also stores all prior mappings by annotating each outlet's timeseries of power data to record each time t a certain device attaches or detaches from an outlet. These annotated data streams can then be used by higher-level data-driven applications, e.g., such as schedulers [4] that use each load's power usage data to determine which loads to defer and when. These data-driven applications require sensors attached to each device, and often implicitly assume a static mapping (or tight coupling) between the sensor and the device. However, as we discuss in Section 1, such a static mapping is usually not feasible in practice.

Our general approach to determining the device attached to each outlet is to train a timeseries classifier on historical power data for each device. The classifier uses the training data to learn an association between the device and the high-level features of its timeseries power usage. Once trained, the classifier simply outputs a device name for a fixed set of consecutive energy readings from each outlet. We represent a set of consecutive energy readings as a series of three-tuples that specify a timestamp, outlet, and average power in watts over τ. Since the device name may change, the controller periodically re-executes the classifier on new outlet data. The interval at which the controller updates each outlet's mapping may be either static, e.g., once every 10 minutes, or dynamic, e.g., based on sudden changes in an outlet's power usage.

III. Algorithm

Before classifying an outlet's timeseries of energy readings, our NILI algorithm first converts them into a small set of features that serve as inputs to the classifiers. As discussed below, this process requires first preprocessing the raw timeseries data, then extracting the necessary features for classification, and finally applying various classifiers to the feature set.

Preprocessing. The raw input data consists of average power readings every τ seconds from a smart outlet. We store these power readings, since the classifier operates over a rolling window of historical data. The length of the window necessary to classify an outlet is device-dependent: some device behavior is distinctive enough to classify within seconds of being turned on, e.g., a microwave, while other devices may require multiple duty cycles to discern a distinctive pattern of usage, e.g., a refrigerator. To aid in feature extraction, we preprocess the raw timeseries by computing a timeseries of *energy deltas*, or the difference between two consecutive power readings. Analyzing energy deltas is common in NILM algorithms, since the size of a delta is device dependent, e.g., a 60W power increase due to a light bulb being turned on, and not affected by a building's aggregate absolute energy usage.

Thus, storing and operating on energy deltas is useful for filtering background noise due to the energy usage of a power strip or the smart outlet, itself. Additionally, since consecutive deltas of the same direction, e.g., +40W followed immediately by +20W) often result from changes in power usage occurring across a measurement boundary, we collapse them in a single delta, e.g., +60W. Preprocessing the data to consider such steps as single energy deltas provides a more accurate representation of changes in device power usage, especially given that most energy deltas are zero, i.e., there is no change in power.

Feature extraction. Given a recent window of raw timeseries power data and energy deltas from preprocessing, we next compute a feature vector that captures the behavior of the device. While many features are possible given the input data, we choose a compact set of features that are both intuitive and easily derived directly from the input data.

- **Statistical Metrics**. The simplest set of features consist of simple statistical metrics of the timeseries power data, including the average power, variance, maximum power, and minimum power over the input time interval. Since infrequently used devices often consume no power, thereby skewing the average power towards zero, we exclude measurements under a threshold (typically slightly more than the minimum recorded value) to ensure that we only considering periods when the device is operating.
- **Duty Cycle**. The duty cycle feature is useful for distinguishing continuously operating devices, e.g., an air conditioner, from devices that typically operate only for short periods, e.g., a toaster. We capture a device's duty cycle as the proportion of time is operates over the input interval, called the 'on ratio,' calculated as the number of average power readings over the threshold wattage above divided by the total number of readings.

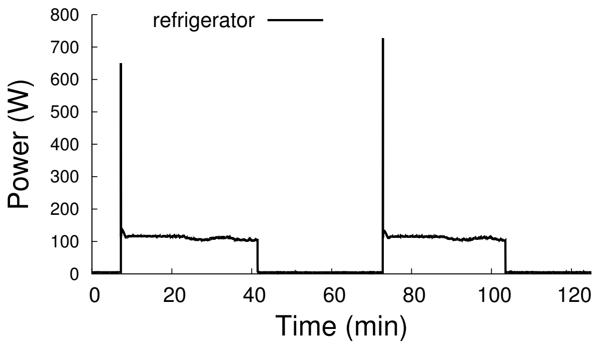

Fig. 2. An example of a spike feature in a refrigerator compressor.

- **Waveform**. The most distinguishing feature of a device's power usage is its complete waveform, which represents a long sequence of specific changes in power specific to the device. For example, the refrigerator power usage in Figure 2 includes a large spike followed by a decrease in power to a steady state. We indirectly capture the waveform as a feature by separating energy deltas into different bins, where the size of each bin represents a distinct feature. We bin the energy deltas as follows: for ten distinct bin sizes ranging from 5W to 500W (with most bin sizes in the $<$ 100W range), we filter the energy deltas to include only changes in average power ranging from the bin size to 5 times the bin size (e.g., 25W to 125W).

  For each bin size, we calculate three features, resulting in 30 features total, as follows: (a) the number of changes in average power in the filtered set of energy deltas, (b) the average time interval between steps in the filtered energy deltas, and (c) the number of 'spikes' in the filtered energy deltas, where a 'spike' is defined as a positive step of at least 10 times the bin size, followed immediately by a negative step of at least 30% of the magnitude of the positive step. Informally, a spike is simply a large but very brief period of energy use caused by the inrush current when a device turns on. Spike features appear prominently in many kinds of motor-driven devices, such as the refrigerator shown in Figure 2.

Classification. Finally, we pass the vector of computed device features to a classifier, which returns the inferred device name. The classifier output may either be a general device type, e.g., refrigerator, or a specific device model, e.g., a particular refrigerator manufacturer and model. We evaluate the three different well-known classification algorithms below, ordered by complexity:

- **Naive Bayes**. We first consider the naïve Bayes algorithm for classification due to its simplicity and efficiency. The key assumption made in naïve Bayes classifiers is the independence of all features, i.e., each feature is conditionally independent of every other feature given the class. Through the application of Bayes' theorem, the conditional distribution over the classes C, i.e., the device types, given the features $f_1, f_2, \ldots, f_n$ is defined by:

$$P(C|f_1, \ldots, f_n) \propto p(C) \prod_{i=1}^{n} P(f_i|C)$$

We use the standard implementation of the naïve Bayes'

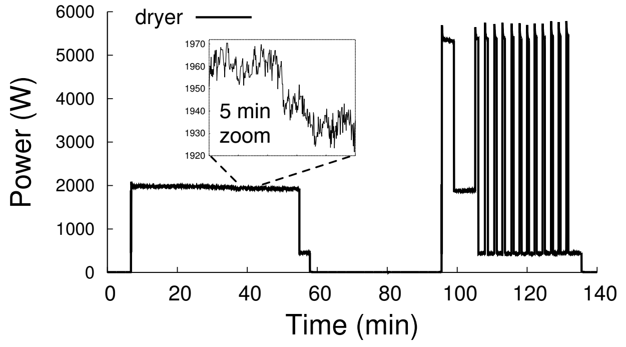

Fig. 3. Energy trace of a dryer with one-second granularity.

classifier [15] used in the Weka toolkit [16].

- **Decision Tree**. We also consider a decision tree classifier [17], which is trained by recursively partitioning the input space, then defining a local model in each resulting region of the input space according to feature values. Although finding the optimal data partitioning is NP-complete, greedy approximation algorithms perform well and benefit from very low training overhead. In our experiments, we use J48, an implementation of the C4.5 decision tree algorithm [18] used in Weka.
- **Support Vector Machines**. Finally, we consider a classifier using support vector machines (SVMs), a more complex algorithm based around mapping the input feature space into a second, linearly separable feature space using a kernel function. We use the `libSVM` implementation [19] of SVMs supported through Weka. Our reported results in Section IV use a polynomial kernel of degree 2, which was chosen after experimentation with several different kernels.

IV. Evaluation

We evaluate our algorithm using a dataset collected from a sensing deployment in three homes. Each home is instrumented with a wide array of energy sensors collecting outlet-level energy usage at approximately one second granularity. This granularity of data is readily available using off-the-shelf meters [9] and reveals many interesting properties of device energy usage that may be lost in lower resolution sampling. As an example, a trace of average energy usage each second collected from a dryer is shown in Figure 3 and reveals multiple modes of operation, including highly variable usage in the first phase and cyclic, decaying usage in the second.

We train our classifiers using a dataset gathered from several dozen devices collected over a three month period. For each device over the three month period, we first split the data into 24-hour blocks, then compute a feature vector over each day-long period as described in Section III. Thus, each device results in roughly 90 instances used in training, though we exclude days in which devices went completely unused.

We consider two scenarios – identifying specific devices models, e.g., a specific refrigerator, or identifying general device types, e.g., any refrigerator. The primary advantage of the latter approach is the ability to generalize to previously unseen devices; while the classifier can only output the specific models that it has observed in training, returning device types allows classification of devices not represented in the training dataset. In practice, we envision training a classifier on a very large dataset of devices collected from many homes, then using that classifier on both existing and new devices not present at the time of training. Although our evaluation dataset is relatively small, i.e., 3 instances of most major appliance types, we consider both approaches to illustrate the potential of unseen device identification.

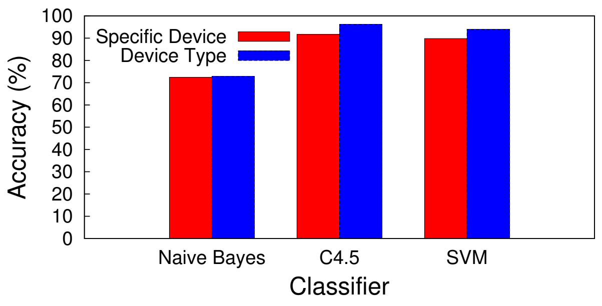

Fig. 4. Classification performance on the entire dataset, i.e., identifying previously seen devices, using 10-fold cross-validation. Both performance on specific device instances and general device classes is shown.

Identification of previously seen devices. We first consider classification performance when training on the complete dataset – i.e., identification of previously seen devices. For each of the three classifiers—Naive Bayes, C4.5, and `libSVM`—we perform 10-fold cross-validation on the dataset to quantify identification accuracy both for specific devices and for device types. The results are shown in Figure 4. We see that accuracy is quite high in all scenarios – both C4.5 and SVM demonstrate accuracy of over 90% for device types, with naïve Bayes somewhat lower at roughly 70%. Performance on specific device identification is modestly lower than for general device types. The difference stems from the ability of the classifier to generalize the properties of the device types, e.g., a refrigerator, given a broader training dataset, as well as the smaller number of possible classes. However, this different only amounts to less than 10% in all cases.

Breakdown by device and type. The results in Figure 4 demonstrate the overall performance of the classifiers, but classification accuracy may vary significantly from device to device, due to the presence of unique characteristics, or lack thereof, reflected in the feature vector. For example, refrigerators have a regular cyclic power usage pattern, which typically results in a high 'on ratio', while most electronic loads have highly erratic power consumption, due to the variable behavior of switch-mode power supplies, which typically results in higher power variance than other types of devices.

Figure 5 shows the individual, device-level classification accuracy for a subset of our devices. As in Figure 4, the best performance for nearly all devices is observed with the C4.5 or SVM classifiers, with accuracy of over 95% for many devices. The performance of the naïve Bayes classifier, on the other hand, is inconsistent, with some devices showing quite poor performance (many less than 50%) – in these cases, the naïve Bayes classifier has difficulty distinguishing between multiple instances of the same device type, e.g., multiple dishwashers or multiple dryers. As a result, accuracy on one such instance of a given type remains high, while performance on other instances of that type is low (as these instances are identified as the 'dominant' first instance).

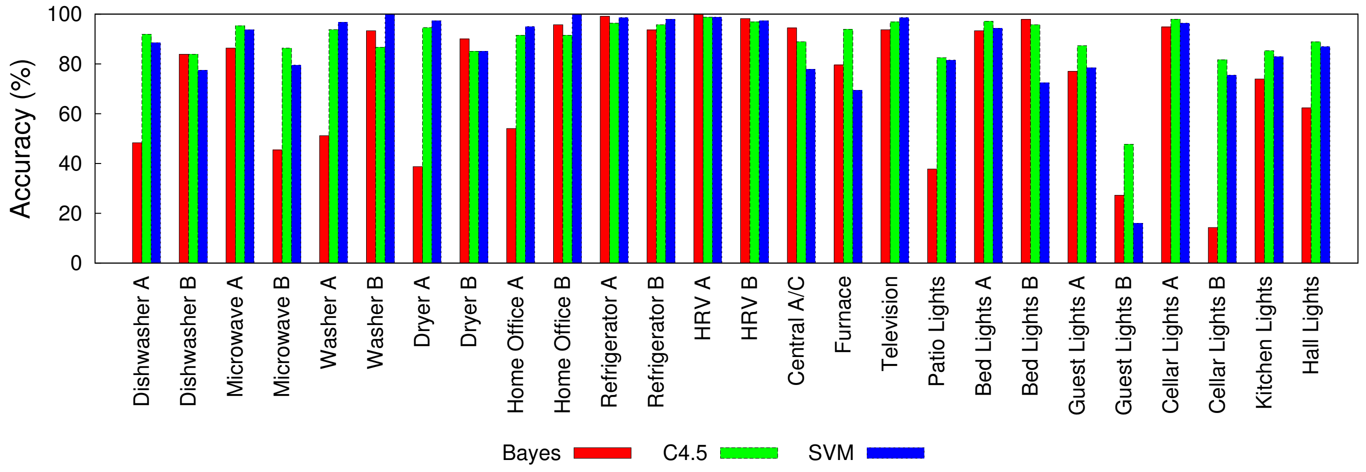

Fig. 5. Individual device identification accuracy per device. Devices A and B represent two instances of the same device type (with different specific models).

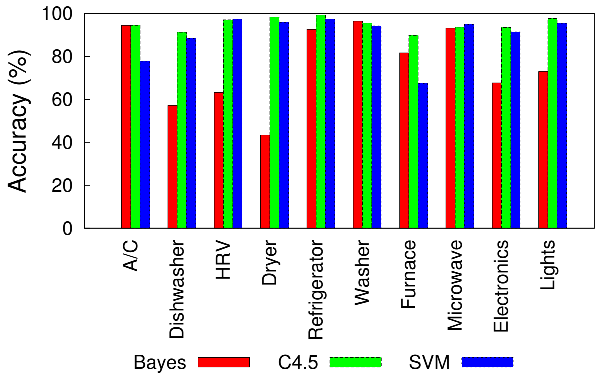

Fig. 6. General device type identification accuracy, broken down by type.

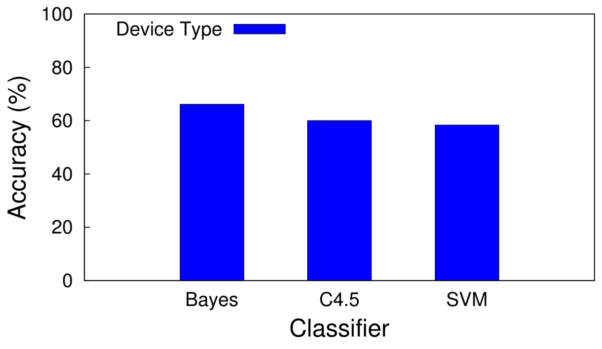

Fig. 7. Overall accuracy of device type identification on previously 'unseen' devices.

Figure 6 shows the corresponding results broken down by device types, rather than individual devices. Performance is more consistent in this case, although naïve Bayes continues to show significantly lower performance for certain device classes, such as dishwashers and clothes dryers. These types of devices exhibit more complex behavior than most of the other device types, e.g., as indicated by the dryer's average power trace in Figure 3, which implies that the simplistic naïve Bayes classifier is not able to identify them as accurately as the more sophisticated C4.5 and SVM classifiers.

Identification of previously unseen devices. Finally, we consider the case where we wish to identify devices that have *not* been previously observed during training. This approach limits us to identifying device types rather than specific device models, as it is impossible to generate a class label that was not seen during training. For this experiment, we trained our classifier on devices within two of the three houses, then attempted to classify devices in the third house, which are not represented in the training data. As before, we report the 10-fold cross-validation accuracy of identification.

Overall identification accuracy is shown in Figure 7. Unsurprisingly, identification accuracy falls substantially, as we are relying strictly on the ability of the classifier to distill the essential properties of the device *type* rather than any specific device instance. Accuracy of both the C4.5 and SVM classifier fall to below 60%. Interestingly, the naïve Bayes classifier degrades substantially less (to 65%) and actually outperforms the other classifiers, reversing the trend seen when identifying previously seen devices. This result suggests that the simple naïve Bayes classifier more effectively generalizes the device type, but at the expense of distinguishing specific instances of device types (as seen previously in Figure 5). Furthermore, while the absolute result of 65% is not particularly high, we stress that we are attempting to generalize the device type given a very limited set of training instances (just two in most cases), so we view these results as encouraging and would improve with more training data.

Finally, Table I shows the confusion matrix for the classification of unseen devices using the C4.5 classifier (i.e., the third bar of Figure 7). We see that there is a wide variation in the accuracy of identification of the various device types. For example, every light is correctly identified as a light (i.e., perfect recall), which is understandable given the flat energy profile of nearly all lights. The same is true of the microwaves, which as short-lived but high-power devices are easily identified. The washing machines, on the other hand, are frequently misidentified as dryers – both types are large, sporadically active devices with complex and highly variable power signatures, and as such it is difficult for the classifier to distinguish the two. A significant portion of the overall classifier error comes from the poor performance of the Electronics device type (nearly all of which are identified as lights), likely due to the fact that there are many different types of electronics displaying differing power signatures. Regardless, we see that the classifier is generally able to accurately distill device types with unique energy characteristics and use those characteristics to identify unseen devices.

	Dryer	Fridge	Washer	MWave	Electronics	Light
Dryer	9	0	0	0	0	0
Fridge	1	99	0	34	21	45
Washer	44	0	5	5	0	0
MWave	3	1	0	53	0	1
Electronics	0	0	0	0	2	14
Light	0	0	0	0	0	86

TABLE I. CONFUSION MATRIX FOR THE DECISION TREE CLASSIFIER ON UNSEEN DEVICES (ROWS ARE ACTUAL, COLUMNS ARE PREDICTED).

V. RELATED WORK

Prior work on device identification has largely been in the context of whole-house metering, and thus falls under the umbrella of NILM. The goal of NILM is to extract all individual devices from a single *aggregate* trace as provided by a house-level smart meter. Existing work on NILM is substantial [7], [13], [14], including techniques such as Hidden Markov Models [20] and Viterbi's algorithm [21] to infer disaggregated usage. However, due to the number of loads in homes and the complexity of their operation, applying NILM to real-world environments remains an active research area. Device identification from outlet-level meters, i.e., NILI, however, has received substantially less attention.

Accurate load identification has been achieved using high-frequency smart meters [22], but this granularity of data is not typically available from off-the-shelf smart outlets. For lower frequency data, e.g., 1 Hz, the use of classifiers for learning device labels has been proposed, but not extensively evaluated [23], [24], particularly for previously unseen devices. Other approaches to device classification have focused on explicit per-device training to generate 'signatures' that can be used to detect devices in the future [25]. These techniques, however, rely on user-guided training periods to recognize specific devices, as opposed to transparent recognition of devices classes.

VI. CONCLUSIONS

In this paper, we considered the problem of Non-Intrusive Load Identification (NILI), in which devices connected to outlet-level energy meters, i.e., smart outlets, are automatically identified, alleviating the user from the cumbersome and error-prone task of manually maintaining meta-data on specific devices and outlets. We propose an approach to performing NILI that transforms energy time-series data into a compact set of intuitive features, then uses an off-the-shelf classifier to identify unknown devices. Using a dataset of device energy traces collected from three homes, our experiments demonstrate that we can achieve greater than 90% accuracy on devices represented in training data. Furthermore, even with a small sample of devices of a given type, e.g., refrigerators, we are often able to identify previously *unseen* devices as particular types of devices, demonstrating the ability of the classifier to generalize the properties of device types. As future work, we plan to consider other features and a larger set of training devices to further evaluate NILI's potential.

Acknowledgements. This work is supported by NSF grants CNS-1253063, CNS-1143655, CNS-0916577, CNS-0855128, CNS-0834243, CNS-0845349, and a grant from the Massachusetts Department of Energy Resources (DOER).

REFERENCES

[1] J. Kelso, Ed., *2011 Buildings Energy Data Book*. Department of Energy, March 2012.

[2] D. Chen, S. Barker, A. Subbaswamy, D. Irwin, and P. Shenoy, "Non-Intrusive Occupancy Monitoring using Smart Meters," in *BuildSys*, November 2013.

[3] W. Kleiminger, C. Beckel, T. Staake, and S. Santini, "Occupancy Detection from Electricity Consumption Data," in *BuildSys*, November 2013.

[4] S. Barker, A. Mishra, D. Irwin, P. Shenoy, and J. Albrecht, "SmartCap: Flattening Peak Electricity Demand in Smart Homes," in *PerCom*, March 2012.

[5] J. Taneja, D. Culler, and P. Dutta, "Towards Cooperative Grids: Sensor/Actuator Networks for Renewables Integration," in *SmartGridComm*, 2010.

[6] S. Barker, A. Mishra, D. Irwin, E. Cecchet, P. Shenoy, and J. Albrecht, "Smart*: An Open Data Set and Tools for Enabling Research in Sustainable Homes," in *SustKDD*, August 2012.

[7] K. Armel, A. Gupta, G. Shrimali, and A. Albert, "Is Disaggregation the Holy Grail of Energy Efficiency? the Case of Electricity," *Energy Policy*, vol. 52, no. 1, January 2013.

[8] "Wemo insight switch," http://www.belkin.com/us/F7C029-Belkin/p/P-F7C029/.

[9] "imeter solo," http://www.insteon.net/2423A1-iMeter-Solo.html.

[10] http://www.budderfly.com/, May 2014.

[11] http://www.aeon-labs.com/site/products/view/5/, May 2014.

[12] M. Brennan, "House of the future: How automation tech is transforming the home," in *Forbes*, October 2013.

[13] G. Hart, "Nonintrusive Appliance Load Monitoring," *IEEE*, vol. 80, no. 12, December 1992.

[14] M. Zeifman and K. Roth, "Nonintrusive Appliance Load Monitoring: Review and Outlook," *IEEE Transactions on Consumer Electronics*, vol. 57, no. 1, February 2011.

[15] G. H. John and P. Langley, "Estimating continuous distributions in bayesian classifiers," in *Eleventh Conference on Uncertainty in Artificial Intelligence*. San Mateo: Morgan Kaufmann, 1995, pp. 338–345.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: An update," *SIGKDD Explor. Newsl.*, Nov 2009.

[17] K. P. Murphy, *Machine learning: a probabilistic perspective*. The MIT Press, 2012.

[18] J. R. Quinlan, *C4. 5: programs for machine learning*. Morgan kaufmann, 1993, vol. 1.

[19] C.-C. Chang and C.-J. Lin, "Libsvm: a library for support vector machines," *TIST*, vol. 2, no. 3, p. 27, 2011.

[20] J. Kolter and M. Johnson, "REDD: A Public Data Set for Energy Disaggregation Research," in *SustKDD*, August 2011.

[21] H. Kim, M. Marwah, M. Arlitt, G. Lyon, and J. Han, "Unsupervised Disaggregation of Low Frequency Power Measurements," in *SDM*, April 2011.

[22] A. Reinhardt, P. Baumann, D. Burgstahler, M. Hollick, H. Chonov, M. Werner, and R. Steinmetz, "On the accuracy of appliance identification based on distributed load metering data," in *SustainIT*, 2012.

[23] D. Zufferey, C. Gisler, O. A. Khaled, and J. Hennebert, "Machine learning approaches for electric appliance classification," in *ISSPA*, July 2012.

[24] A. Ridi, C. Gisler, and J. Hennebert, "Automatic identification of electrical appliances using smart plugs," in *WoSSPA*, May 2013.

[25] A. Ruzzelli, C. Nicolas, A. Schoofs, and G. M. P. O'Hare, "Real-time recognition and profiling of appliances through a single electricity sensor," in *SECON*, 2010.