Adaptation for Automated Drift Detection in Electromechanical Machine Monitoring

Daisy H. Green, Aaron W. Langham, Rebecca A. Agustin, Devin W. Quinn, Steven B. Leeb

***Abstract*—Practical machine learning applications for streaming data can involve concept drift (the change in statistical properties of data over time), one-shot or few-shot learning (starting with only one or a few examples for each class), a scarcity of representative training data, and extreme verification latency (only the initial dataset has ground-truth labels). This work presents a framework for organizing signal processing and machine learning techniques to provide adaptive classification and drift detection. Nonintrusive load monitoring serves as an ideal case study, as modern sensing solutions provide a wellspring of electromechanical data sources. There is a lack of training datasets that generalize across different load and fault scenarios. Accordingly, training must be accomplished with a limited set of data when deploying a nonintrusive load monitor to a new power system. Also, loads can exhibit concept drift over time either due to faults or normal variation. Nonintrusive load monitoring field data is used as an illustrative case study to demonstrate the proposed framework for adaptation and drift tracking.**

***Index Terms*—concept drift, few-shot learning, stream learning, power monitoring**

I. Introduction

Many pattern classifiers fail to account for the time-evolution or dynamic behavior of observed data [1]–[3]. Concept drift can obliterate the effectiveness of a classifier trained on a static dataset [4], [5]. The order of observations and the relationship between the timing and the evolution of trends contain valuable information. A lack of representative training data in an otherwise tractable domain exacerbates these problems, as data often cannot be assumed to be independent and identically distributed (i.i.d.) or stationary. Decision boundaries in a feature space are frequently not readily explainable or based in physical understanding.

Power monitoring of electromechanical loads is an illustrative stream learning problem with practical industrial applications. Applications for energy efficiency, energy management, condition-based maintenance, and fault detection and diagnostics all rely on power data. Here, a nonintrusive load monitor (NILM) provides a convenient means for power monitoring which uses a single set of current and voltage sensors at the feeder of an electrical sub-panel to monitor a collection of loads. Effective nonintrusive load monitoring requires accurate load signatures and load identification, i.e., the identification of individual loads from the aggregate power stream [6].

Nonintrusive load monitoring is fundamentally a non-stationary problem exhibiting concept drift. Changes in load behavior can be related to various forms of concept drift. A load's electrical behavior may change over time, potentially due to load aging and degradation. Variability is expected even for a healthy load, arising from normal mechanical variation, changing operating conditions, and environmental factors. A sudden change in machine health is analogous to sudden concept drift. A slow change over time is a form of incremental concept drift. A return to a previous health condition or operating state is similar to recurring concept drift. The challenge in load identification is ensuring correct results even amidst changing operating conditions and fault scenarios. Most nonintrusive load monitoring research assumes training data is forever representative of new data, without regard to changing load behavior [7].

Many machine learning applications, such as image recognition, rely on large, generalizable datasets. Although datasets exist for nonintrusive load monitoring [8]–[10], they are generally restricted to healthy residential appliances, and cannot generalize to the avalanche of loads that exist in practical industrial and commercial sites. Thus, the training data for a practical nonintrusive load monitoring classifier will likely need to be collected by a NILM on the system of interest. Nonintrusive load monitoring is therefore an example of few-shot learning [11]. At its extreme, it becomes a one-shot learning problem, in which the model must train starting with only a single example for each load [12]. The few-shot nature of the problem means that the limited training data is not likely to be representative of the load's long-term operation. As such, deep learning models are prone to overfitting [13].

Many concept drift detection methods use error rates to detect drift using labelled data [14], [15]. This is an unrealistic constraint for real-time nonintrusive load monitoring, as it would require periodic manual labelling of data. Instead, extreme verification latency (i.e., ground-truth labels are never received after the initial dataset) must be assumed. Relatively little work has addressed semi-supervised or unsupervised drift detection and adaptation [5]. An automated solution is desired.

Nonintrusive power monitoring is an illustrative one-shot or few-shot learning problem that must handle concept drift even without supervision after the initial dataset. However, in literature it has largely been regarded as a stationary problem [16]. This paper introduces a multi-level framework of classification techniques applicable to machine learning problems facing the following challenges: 1) concept drift, 2) one-shot or few-shot learning, and 3) extreme verification latency. In order to be robust to outliers, our proposed method is designed to handle incremental and recurring concept drift, whereas sudden concept drift is out of this work's scope. "Coarse" and "fine" classification levels enhance existing pattern classifiers' abilities. Given the scarcity of training data and few-shot nature of the problem, coarse classifiers use extracted features and physically realistic boundaries to avoid overfitting and

remain robust to load concept drift. Fine classifiers use higher-dimensional data to resolve any ambiguities in the extracted course classifier feature space. An online clustering algorithm provides drift metrics to enable continual load identification and diagnostics even in the presence of concept drift. These steps work together to ensure confidence in classification, given that ground-truth labels are not available after initialization. Experimental results are presented on real-world, non-stationary data. Power data was collected over four years aboard US Coast Guard Cutter (USCGC) SPENCER and used for framework evaluation.

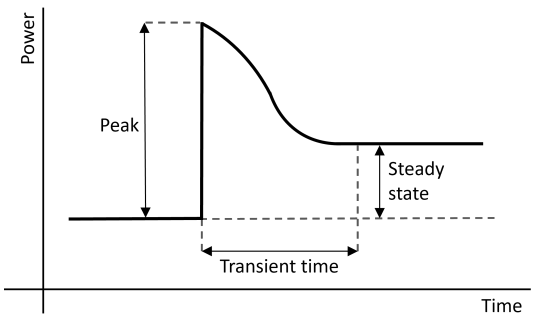


Fig. 1. Steady-state (ss), peak, and transient time features for an example load turn-on transient.

II. Literature Review

Drift detection algorithms can generally be divided into error rate-based and data distribution-based [5], [17]. Ensemble methods incorporate multiple classifiers. Error rate-based approaches focus on tracking some error-related metric and form the largest category of algorithms [5]. Common methods include the Drift Detection Method (DDM) [18] and Early Drift Detection Method (EDDM) [4], [19], which signal that drift has occurred when there is a statistically significant change in the error rate or distance between classification errors, respectively. The drift detection method for online class imbalance (DDM-OCI) [20] uses the reduction in minority-class recall to detect drift, thus staying more robust to class imbalance. Data distribution-based methods use a distance metric to quantify the dissimilarity between the distribution of historical data and new data [5], [17]. A drift is signaled when the dissimilarity is proven to be statistically significant.

All error rate-based methods and most data distribution-based methods are supervised approaches which assume that an instance's ground-truth class label is available immediately after prediction. However, this is an unrealistic constraint in many real-world applications. Labelling data is often a costly process that involves manual labelling by domain experts. With this motivation, some semi-supervised and unsupervised drift detection algorithms have been proposed [14]. One semi-supervised approach is the Semi-Supervised Adaptive Novel Class Detection and Classification (SAND) algorithm [21]. This method bases drift detection on classifier confidence. When drift is detected, a new model is created, using predicted labels for instances with high confidence. True labels are requested for instances with low classifier confidence. Another approach is the Dynamic Selection Drift Detector algorithm which bases drift detection on a pseudo-error rate [22]. It requests true labels for instances that are near a set warning level. Both of these methods request labels for a subset of data after concept drift is detected.

For nonintrusive load monitoring, it is not realistic to assume that even a subset of labelled data will be available when requested. Extreme verification latency must be assumed. A semi-supervised framework that handles drifting environments with extreme verification latency is proposed in [23], referred to as Compacted Object Sample Extraction (COMPOSE). It uses a base classifier trained on the labelled data at the initial step, then extracts core supports from the classified data to retrain the classifier. The core supports represent the geometric center of each class distribution to serve as labelled instances. Different core support shapes have been proposed such as an α-shape [23] and Gaussian mixture models [24], while in [25], all the classified samples are used instead of core support extraction. For affinity-based COMPOSE [26], an affinity matrix is formed between the labelled and unlabelled samples and those with high similarity scores are classified and used to retrain the classifier. While these methods do address extreme verification latency, they are not applicable for one-shot or few-shot learning. The methods assume a good base classifier without mentioning how to choose such a classifier. The choice of base classifier and its resulting decision boundaries has a large affect on any subsequent classification, especially if it overfits the limited initial data. To the best of our knowledge, there is no work in literature that simultaneously addresses incremental and recurring concept drift, one-shot or few-shot learning, and extreme verification latency.

III. Nonintrusive Load Monitoring

The utility of nonintrusive load monitoring for energy management and condition-based maintenance relies on accurate load identification, i.e., the identification of individual loads from the aggregate power stream. Any data stream that captures the electromechanical nature of a load and reflects how the load consumes energy can potentially serve as a feature for recognizing the load. Relevant data streams can, for example, include voltages and currents, real (P), reactive (Q) and apparent (S) power, higher order harmonic content (e.g., third, fifth, and seventh order harmonics), and impedances.

The NILM captures voltage and current waveforms sampled at 8 kHz and processes this raw data into real and reactive power at an output frequency equal to the line frequency (60 Hz) [27]. When a load energizes or changes state it manifests in the power streams as transient behavior. This transient behavior can be detected with a change-of-mean detector and features can be extracted. For demonstration purposes, we present a feature space consisting of the steady-state real power (P_{ss}), steady-state reactive power (Q_{ss}), the maximum apparent power at inrush (S_{peak}), and the transient time. These features are shown in Fig. 1 for an example load turn-on transient, representative of any power stream (e.g., P, Q, or S).

While useful for load identification, the electrical characteristics of a load can change over time. As operating conditions

change, power varies with reasonable variation of mechanical load operation [28]. Changes or deviations in power could also indicate the beginning of a "soft fault," or the gradual degradation of equipment performance. These changes create variability within the feature space that may make it difficult for a NILM to recognize a load. At the same time, this variability may be a useful prognostic indicator. Our goal is twofold: 1) accurate load identification even in the presence of load power variation and drift, and 2) tracking changes in load power characteristics to use for fault detection and diagnostics.

IV. Classification and Drift Detection

The framework consists of four steps for classifying and detecting drift. This multi-level framework ensures a high level of confidence in labelled events since ground-truth labels are assumed to not be available. Events that do not pass the multi-level check are determined unclassifiable. A one-vs-all check, referred to as the "preliminary" check, is used first as a "negative" classifier that eliminates the classification of physically implausible events. This first step appeals to the physical expectations for load behavior to ensure physically realistic boundaries. Next, classification operates with two levels of granularity. The two levels are designed to improve classification while gradually increasing dimensionality as needed. The first, "coarse" level examines high-level features extracted from the waveform (e.g., features such as peak and steady-state power) to avoid overfitting due to the few-shot nature of the problem. The second, "fine" classification step uses a more in-depth examination of sampled data (e.g., time-domain shape recognition). That is, the coarse classifier uses extracted features from the power waveform, whereas the fine classifier uses windows of the time-domain power waveform directly. Finally, load drift is detected and tracked using "drift clusters" to characterize evolving load behavior and concept drift. The concept of an "exemplar" is defined as a load event that is representative of a load's short-term behavior. Each load has an "initial exemplar," i.e., the initial load event. Each load also has an "active exemplar," which is updated to represent the most recent operation state based on the drift clusters.

Algorithm 1 Algorithm for organizing classifiers.

```
L ← list all loads
for each l ∈ L do
    Set initial exemplar
    Set active exemplar
    Set up preliminary and coarse boundaries
end for

while True do
    x ← incoming feature vector

    M ← PreliminaryCheck(x, L)
    if M is empty then
        Determine x as unclassifiable
        continue
    end if

    N ← CoarseClassifier(x, M)
    if N contains only one load then
        Classify x as the load in N
        TrackDrift(x)
        continue
    end if

    P ← FineClassifier(x, N)
    if P contains only one load then
        Classify x as the load in P
        TrackDrift(x)
        continue
    end if
    Determine x as unclassifiable
end while

function TrackDrift(x)
    Update drift clusters with x
    Update active exemplar
    Update preliminary and coarse boundaries
end function
```

A. Framework Overview

The framework process is presented in Algorithm 1. To initialize the framework, a list of all load classes is obtained and denoted as L. Initial data is collected and the initial exemplar is set for each load to be the load's first event. During the initialization step, if there is only a single event for each load, the active exemplar is set to be the initial exemplar. Otherwise, the drift clustering algorithm is run on the load's initial data to determine the active exemplar. Then, the preliminary and coarse boundaries are drawn in the feature space. When features that do not have common units are used in this framework, min-max normalization is performed so that Euclidean distances in the feature space are well-defined. For each feature axis, i, the range of the data is transformed into $[0, 1]$ through the transformation,

$$x_s^i = \frac{x^i - x_{min}^i}{x_{max}^i - x_{min}^i}. \tag{1}$$

Min-max parameters are obtained using the initialization data. As a result, it is possible that incoming data may be scaled outside $[0, 1]$.

Once initialization is complete, every incoming feature vector is initially examined via the preliminary check. A list of loads that pass the check is generated and denoted as M. There are two high-level possibilities that could occur: 1) M is empty, indicating the feature vector falls outside any known load preliminary boundaries, and 2) M contains at least one load. If M is empty, the event is considered unclassifiable. That is, the event does not go through the remaining checks, and the procedure moves on to the next incoming feature vector. This is represented in Algorithm 1 with the continue statement. If M is not empty, the loads that passed the preliminary check are passed to the coarse classifier.

With a power sampling rate of 60 Hz, a transient can easily have a dimensionality on the order of several hundreds. The few-shot nature of the problem implies that working with such

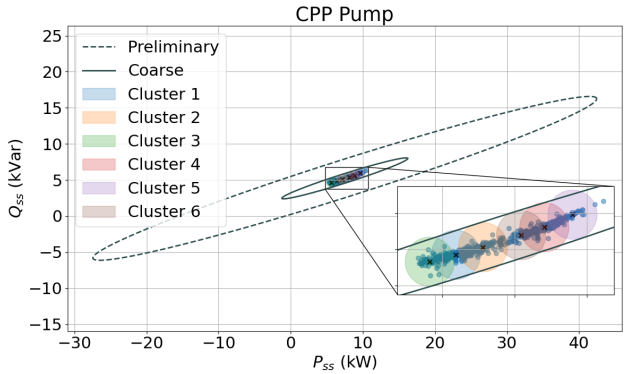


Fig. 2. CPP pump preliminary boundary, coarse boundary, and drift clusters for the Q_{ss} versus P_{ss} feature space.

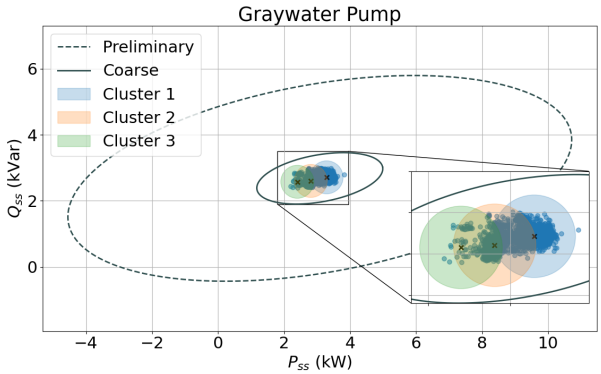


Fig. 3. Graywater pump preliminary boundary, coarse boundary, and drift clusters for the Q_{ss} versus P_{ss} feature space.

a high-dimensionality feature space may lead to overfitting, also known as the curse of dimensionality [29]. Thus, the coarse classifier operates on the lower-dimensionality feature space of extracted features. The coarse level performs classification using one-vs-all classifiers for each load. This permits overlapping load decision boundaries and allows for the possibility that a load's decision boundaries can change over time independent of other loads. The coarse classifier returns a list of loads denoted as N. There are three possible outcomes for the coarse classifier: 1) N is empty, indicating that the feature vector falls outside any known load coarse boundaries (but still within a preliminary boundary), 2) N contains exactly one load, and 3) N contains more than one load, indicating overlap of the coarse boundaries. When a feature vector is inside a single load coarse boundary, it is classified as that load without running the fine classifier. If N contains zero loads, the fine classifier is run on the M loads that passed the preliminary check. If N contains more than one load, the fine classifier is run on those N loads.

The fine classifier uses the higher-dimensionality transients if the coarse classifier is unable to confidently identify one load. The fine classifier returns a list P, which either contains a single load or is empty. If P contains a single load, the event can be classified as that load. Otherwise, the event is considered unclassifiable. When an event is classified, a clustering algorithm is run using geometric distances in an easily-conceivable extracted feature space. This makes possible the tracking of drifting power signatures and designation of the active exemplar. The preliminary boundaries, coarse classifiers, and fine classifiers are adapted as necessary to track diagnostic changes and ensure accurate load recognition. Example preliminary boundaries, coarse boundaries, and drift clusters are shown in a two-dimensional feature space for two shipboard loads, the controllable pitch propeller (CPP) pump and graywater pump, in Fig. 2 and Fig. 3, respectively.

B. Preliminary Check

Decision boundaries created by classifiers are not guaranteed to be compact or physically realistic. Thus, it is necessary to establish which loads an incoming feature vector could plausibly belong to before attempting classification. For the domain of power monitoring and incremental concept drift, it is reasonable to assume that features that are physically relevant to the load's drift will change gradually. As a result, an extracted feature space-based "preliminary check" rules out loads that are physically implausible candidates for a given load event. An N-dimensional hyperellipsoid boundary is created for each load in the feature space, where N is the number of dimensions of the feature space. The hyperellipsoid can represent a spread of points with a small number of parameters [30]. A hyperellipsoidal region can be represented by

$$(\mathbf{x}-\mathbf{m})^T\mathbf{R}(\mathbf{x}-\mathbf{m}) \leq 1, \tag{2}$$

where $\mathbf{m}$ is an $N \times 1$ vector representing the centroid of the hyperellipsoid and $\mathbf{R}$ is a real symmetric positive-definite $N \times N$ matrix representing the shape and orientation of the hyperellipsoid. For the preliminary check, any point $\mathbf{x}$, satisfying the inequality in Eq. (2), is either inside or on the surface of the hyperellipsoid [31]. Estimates of variance in observed load behavior can define a loose hyperellipsoidal boundary where load observations can be expected to be located with high confidence. Using principal component analysis (PCA) [32], the variance of the data can be obtained in each of the principal component directions, as well as the mean in each feature dimension. PCA also yields a "transformation matrix" whose rows are each principal component axis. The standard deviations of the data in each principal component axis are obtained as the positive square root of the variances. A tunable number of standard deviations, denoted here as A, is chosen by the user to generate the radii for the hyperellipsoid. For the USCG vessel, $A = 28$ standard deviations has proven effective for construction of hyperellipsoidal boundaries for the preliminary check.

Since PCA cannot be reliably used with few data points, the implementation does not set up a given load's PCA hyperellipsoid until 10 events have been recorded for the load. Before then, a provisional preliminary boundary is used, consisting of an unrotated N-dimensional hyperellipsoid whose radius in axis i is equal to $\max(0.2, x_i)$, where x_i is the initial feature vector's value in that axis (using min-max normalization). By setting a minimum axis value of 0.2, events close to a feature axis (e.g., heaters that consume no reactive power) are not assigned impractically small preliminary boundaries.

Only loads whose preliminary boundaries contain the incoming event's feature vector are considered as candidate loads for the following coarse classifier. If no loads pass the preliminary check, then the load event is considered unclassifiable. Every time a new event is classified to the load, the load's preliminary boundary fits a new PCA hyperellipsoid to the load data.

C. Coarse Classifier

Coarse boundaries drawn in the feature space can further reduce the number of candidate loads and potentially perform final classification. Since drift in the feature space can result in load feature vectors occupying the same general space, the coarse classifier is chosen to be a one-vs-all rather than multiclass classifier. Once a list of candidate loads has been obtained from the preliminary check, each candidate load's coarse boundary is checked as to whether it contains the incoming feature vector. If only one candidate load's boundary contains this feature vector, the incoming event is classified to that load. If multiple loads' coarse boundaries contain this feature vector, then there is not yet sufficient information to determine which of these loads to classify the event to. These containing loads are retained as candidate loads for the fine classifier. If no loads' coarse boundaries contain the feature vector, then no classification is made, and the same list of candidate loads are used as the input for the fine classifier.

The coarse classifier should be chosen such that it yields a binary result. Classifiers such as deep neural networks (DNN) and random forests (RF) provide nonlinear boundaries that can take on complicated shapes. Another option is to use the N-dimensional standard deviation hyperellipsoid described earlier, but with a much smaller number of standard deviations. This is the method demonstrated in this work, using 7 standard deviations. Eq. (2) is used to determine which hyperellipsoids contain a given feature vector. This method has the advantage that the hyperellipsoid can likely be obtained faster than a DNN or RF can be trained, and only uses data from the load in question (as opposed to the DNN and RF, which require other load data to train for binary classification).

Just as for the preliminary boundaries, some amount of data must be collected until a coarse classifier can be trained. Until 10 events have been recorded for a load, the same technique used for the provisional preliminary boundaries is used, but here the radii of the N-dimensional hyperellipsoids are set to $\max(0.05, 0.25 \cdot x_i)$ (using min-max normalization). Fig. 4 shows an example three-dimensional feature space with three shipboard loads and hyperellipsoidal coarse boundaries for demonstration. If an event is within only a single load coarse boundary, such as a point inside the graywater pump coarse boundary, it can be classified as that load. If an event is within multiple load coarse boundaries, such as a point that is inside both the CPP pump and air compressor coarse boundaries, that event goes to the fine classifier. If an event is not within any coarse boundaries, but is still within a preliminary boundary, it also goes to the next and final classifier. Using hyperellipsoids as coarse classifiers involves fitting a new PCA hyperellipsoid to the load data every new event for that load.

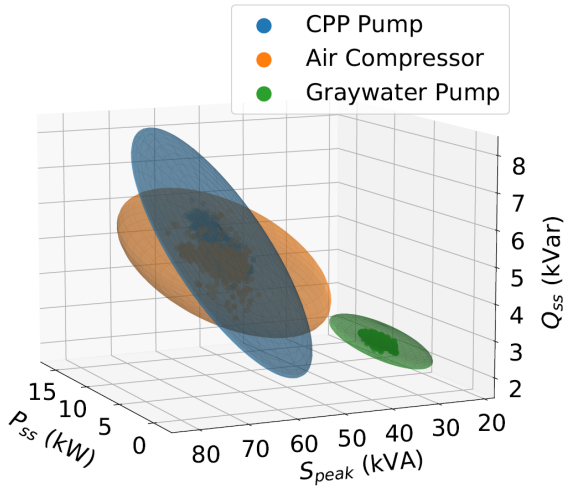


Fig. 4. Coarse boundaries drawn in the P_{ss}, S_{peak} and Q_{ss} feature space for the CPP pump, air compressor, and graywater pump.

D. Fine Classifier

For feature vectors that were in multiple coarse load boundaries or were not in any coarse load boundaries, but were within at least one preliminary load boundary, a multiclass fine classifier is run as the final classification step. The fine classifier operates on higher-dimensional data than the coarse classifier in order to resolve overlap in the extracted feature space. The fine classifier relies on having an accurate representation of recent load operation through the active exemplar and drift clusters. This work uses six seconds of the real and reactive power time-domain transients associated with the incoming event as the fine classifier feature space. For fine, time-series classifiers, this work demonstrates both a correlation matching algorithm [6] and gated recurrent units (GRUs) [33].

Using a correlation matching algorithm requires only a single exemplar transient for each load, i.e., that of the active exemplar. This method can be used for one-shot learning, in which only a single event is available for each load. The initial exemplar is used as the active exemplar until the drift clustering algorithm identifies a new active exemplar. To resolve a set of loads using the correlation matching algorithm, the incoming load event's transient is matched to each of the candidate loads' active exemplars' transients, and a correlation score is generated for each. Consider two sampled waveforms f and g, where f is an observed waveform and g is a load exemplar's transient. The correlation score, C, is:

$$C = \left| 1 - \frac{(f - \bar{f}) \cdot (g - \bar{g})}{|g - \bar{g}|^2} \right| \tag{3}$$

where $\bar{f}$ and $\bar{g}$ are the mean of f and g, respectively, and are subtracted from the original signals in order to remove the dc offsets. When C approaches zero, this indicates that the exemplar and observation transients match in both shape and amplitude [6]. The minimum correlation score for the candidate loads is found. If it is less than a tunable threshold, the event is classified to the corresponding load. For events within multiple coarse boundaries, the threshold is 0.25. If

an event is not within any coarse boundaries and the fine classifier uses candidate loads from the preliminary check, a lower maximum correlation score of 0.10 is used, since there was originally less confidence in the candidates.

When using a GRU, the initial training cannot be done on a single event. For these classifiers, it is assumed that there is sufficient data to train an initial classifier. As shown in Section V, an adaptive GRU approach is demonstrated for this work. The initial multiclass model is implemented with a GRU layer with 50 ReLU-activated neurons, then a densely connected layer with 30 ReLU-activated neurons, and finally a softmax-activated layer. The training approach used Adam-optimized backpropagation with a learning rate of 0.001 and categorical crossentropy as the loss function. The initial data was split into 80% training and 20% validation with data stratification. A batch size of 64 was used for mini-batch gradient descent. Validation loss was used for early stopping, such that after fifteen epochs of no significant improvement, training was stopped.

After an initial multiclass GRU model is trained, the model must be incrementally retrained in order to learn the detected load drift. However, models retrained using only recent data often suffer from "catastrophic forgetting," that is, the degradation of performance on previous tasks when learning new tasks [1], [34]. One approach to prevent catastrophic forgetting is the replay-based or rehearsal approach, in which some of the previous samples are stored and repeatedly reused when the model is retraining on new data [1], [35], [36]. Another approach is regularization-based, in which a penalty consolidates the important weights, and selectively slows down learning on those weights [37]. The third approach is dynamic architecture-based, which iteratively updates the network architecture by network masking or network pruning [38]. In this work, the first method is applied, in which a memory buffer is used to store a limited number of transients. For each load, the thirty most recent transients are stored in a temporary memory buffer. Every time a new drift cluster is formed for a given load, the twenty most recent transients are stored in a permanent memory buffer. This is analogous to storing the active exemplar for the correlation matching technique, assuming that the load drift is incremental. Every time a new drift cluster is formed, backpropagation is run on the model using every load's temporary and permanent memory buffer of transients. A learning rate of 0.0001 is used. This is smaller than the learning rate of the initial classifier to ensure only incremental improvements are being made. The same data stratification split is used as for the initial classifier. Training is stopped after five epochs of no improvement in the validation loss.

For each load the number of transients stored is at most $30 + 20 \cdot x$, where x is the number of drift clusters for that load. Using the same size temporary memory buffer for each load helps prevent biasing the classifier towards the class with the must abundant number of recent events. Classification with a GRU classifier uses the output score of the softmax output layer associated with each candidate load to perform classification. Specifically, for the list of candidate loads in multiple coarse boundaries, the event is classified as the load with the maximum softmax output score. If an event is not within any coarse boundaries, the event is classified with the load with the maximum softmax output score, as long as the score is greater than 0.5.

Algorithm 2 Algorithm for drift clustering.

Input: r_0: micro-cluster radius
Input: γ: density threshold
Output: ST: short-term drift metric
Output: LT: long-term drift metric

$E \leftarrow$ incoming event
$d_{min} \leftarrow$ distance of E to closest micro-cluster center
if $d_{min} \leq r_0$ **then**
    Add E to nearest micro-cluster
    Set active exemplar as nearest micro-cluster exemplar
    if $d_{min} \leq \frac{r_0}{2}$ **then**
        Update micro-cluster center
    end if
else
    Add E to outliers
    $y \leftarrow$ outlier with most outliers within distance r_0
    $Z \leftarrow$ outliers within distance r_0 of y
    if Z contains at least γ points **then**
        New micro-cluster $\leftarrow Z$
        New micro-cluster center $\leftarrow$ mean of Z
        New micro-cluster exemplar $\leftarrow E$
        Set E as active exemplar
    end if
end if
$ST \leftarrow$ distance from E to active exemplar
$LT \leftarrow$ distance from E to initial exemplar
return ST, LT

E. Drift Clusters

Over time, as a load ages or operating conditions change, load observations may drift away from the initial exemplar, for both the coarse classifier and the fine classifier. Keeping track of load concept drift and distribution changes serves an important role for classification. It is especially important due to the few-shot nature of the problem, as the initial dataset likely does not capture a load's possible drift. In the framework, drift is tracked by designating certain load events as representative exemplars. Furthermore, the distance in the feature space between an observation and an exemplar serves as an important metric for diagnostics [39].

A density-based online clustering algorithm is used in order to determine the representative exemplars. Example density-based clustering algorithms include clustering online data in arbitrary shapes clusters (CODAS) and clustering evolving data streams into arbitrary shapes (CEDAS) [40], [41]. These algorithms both use the concept of a micro-cluster, which is formed based on local density. In this context, each micro-cluster represents a load drift cluster. When a new micro-cluster is formed, a new exemplar is designated to represent it.

The density-based clustering can be focused on the features that are most significant for recognizing load drift. As an

illustrative example, steady-state power level changes (P_{ss} and Q_{ss}) often indicate an underlying gradual shift in load performance. Since these features both use power units, min-max normalization is not used for this clustering process. The clustering and drift tracking process is summarized in Algorithm 2. Each load's initial exemplar is set as its first event. Until a load's first drift cluster is formed, the load's active exemplar is its initial exemplar. After every new data sample labelled by the preliminary check, coarse classifier, and fine classifier, as outlined above, the clustering algorithm is run. A micro-cluster is formed when an area in the feature space reaches a fixed density threshold, γ, in this work fixed to five points for all loads. The density threshold should be chosen based on an estimate for when events can be deemed a repeatable occurrence. For example, in power monitoring, a single event would not be enough to declare a new drift state. If a load has energized five times in the same area in the feature space, it is more likely to be a repeatable event. The micro-cluster radius, r_0 is calculated for each load as $\max(0.5 \text{ kW}, P_{ss}/10)$, where P_{ss} is the steady-state real power of the first load event. The micro-cluster radius should be chosen based on expected normal variation. For power monitoring, we set a minimum radius of 0.5 kW so that loads with small steady-state values are not assigned impractically small micro-cluster radii. For larger loads, we select 10% of the steady-state value to represent incremental drift. When the criteria for creating a micro-cluster are met, the most recent event is selected as the exemplar to represent the load state for events that belong to the micro-cluster.

In the original CEDAS algorithm, a decay rate is used to remove defunct micro-clusters. However, in our modified approach, each of the previously active micro-clusters is maintained. This is necessary for continued load diagnosis and to take into account recurring concept drift. That is, the micro-clusters are never removed, merged, or split once identified. The micro-cluster center is updated when a new event is added that is within half the radius of the micro-cluster. In addition to keeping all previous active exemplars, a short-term and long-term distance metric are calculated, where the distance is defined as the Euclidean distance within the steady-state feature space. The short-term metric is the distance from an incoming event's feature vector to the active exemplar's feature vector and should remain small for incremental drift. The long-term metric is the distance from the incoming event's feature vector to the initial exemplar's feature vector. This metric is useful for load diagnostics, as an increasing long-term drift metric provides indication of possible load degradation. These drift metrics are demonstrated in Section VI.

F. Unclassifiable Loads

The framework is designed with the assumption that ground-truth labels will never be available. The four-step process ensures high confidence in those that are classified, in order to prevent the cascading result of an incorrect label. There are several underlying reasons for why an event could be unclassifiable. First, it could simply be the result of an imperfect event detector, and the detected event may not correspond to an actual load event. It is important that the model not use these events, so as to not be impacted by noisy data. Second, if a feature vector is in multiple coarse load boundaries and is further unidentifiable by the fine classifier, it likely means both the extracted feature space and transient features need more dimensions in order to increase separability. Third, the event could be an instance of a new or unknown load class. Although out of the scope of this work to automatically identify a new load class, a human operator can identify and label a new load that was added to the system, or that was not accounted for during the NILM installation. The new load would then be added to the load list, L, and initialized as described in Algorithm 1 by setting the initial and active exemplars and preliminary and coarse boundaries for the load. When using adaptive correlation matching as the fine classifier, drift clusters can be used to identify the exemplar for correlation matching. When using an adaptive GRU, the GRU can be incrementally trained with the new labelled data. Finally, it is possible that the unclassifiable load is the result of a load experiencing a sudden fault leading to a sudden concept drift. If the fault causes its events to be outside the load's feature space boundaries and also unrecognizable by the fine classifier, it is an instance of sudden concept drift, and operator supervision will again be necessary to identity the faulty load.

TABLE I
MONITORED LOADS

System	Equipment	Power Rating	# of Events
Port-side MPDE keep-warm system	Jacket water heater	9.0 kW	117
	Lube oil heater	12.0 kW	278
	Prelube pump	2.2 kW	242
Port-side SSDG keep-warm system	Jacket water heater	7.5 kW	1178
	Lube oil heater	1.3 kW	2732
Fuel oil purifier system	Centrifugal motor	9.5 kW	1131
	Feed pump	2.6 kW	341
Additional engine room loads	Controllable pitch propeller pump	7.5 kW	561
	Graywater pump	3.7 kW	2000
	Bilge and ballast pump	5.6 kW	213
Auxiliary room loads	Air compressor	7.5 kW	1390
	Air conditioner	17.0 kW	74

V. EXPERIMENTAL RESULTS

The framework was tested on a nonintrusive load monitoring dataset containing observations from three sub-panels on USCGC SPENCER: the port-side engine room sub-panel, starboard-side engine room sub-panel, and auxiliary room sub-panel. Data was collected over the course of four years, with gaps in the data during some of the in-port periods. Many of the loads are part of larger controlled systems, including the port-side main propulsion diesel engine (MPDE) keep-warm system, the port-side ship service diesel generator (SSDG) keep-warm system, and the fuel oil purifier (FOP) system. The remaining loads are additional engine room loads and auxiliary room loads. Table I lists the individual loads, their rated powers, and the number of events in the dataset. For the graywater pump, a random subset of 2000 events from the dataset was used to keep the event count on the same order as the remaining loads. Fig. 5 shows the initial time-domain

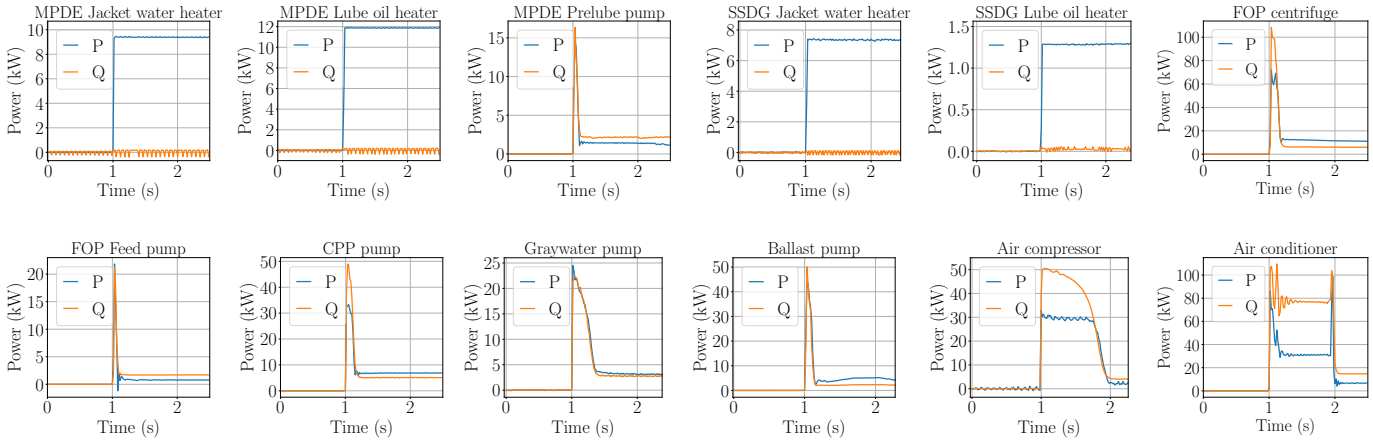


Fig. 5. Zoomed-in view of the initial time-domain transients for each of the loads in the dataset.

transients for each of the loads. Since the data collection for the auxiliary room panel was started later than the port and starboard sub-panels, the data was aligned such that the start of data for each sub-panel was set to $t = 0$.

The extracted features for the preliminary check and coarse classifier include the steady-state real power (P_{ss}), steady-state reactive power (Q_{ss}), the maximum apparent power at inrush (S_{peak}), and transient time. Changes in steady-state levels are calculated as the difference between the median values over Δt_M length windows before and after an identified event, where $t_M = 0.5$ seconds. The maximum power at inrush is defined as the difference between the maximum value of the transient and the median value of a Δt_M length window before the transient. For transient time, first a ten-point rolling mean is applied to the first difference stream of apparent power. If there is a first difference of -500 W or less, indicating a large negative slope, the steady state is determined to be after this value. Then, if the rolling mean of the first difference stream is less than 5 W, it indicates that the apparent power stream has reached steady state. The feature axes are normalized with min-max normalization. The fine classifier features are six seconds of the P and Q waveforms, centered at the detected transient.

A. Two-Dimensional Static Classifier

First, a subset of these loads are used in a two-dimensional feature space of P_{ss} and Q_{ss} as illustration of the reduced performance of a static classifier due to changing load behavior over time. A deep neural network (DNN) is trained and used as a classifier for six loads from USCGC SPENCER. A limited dataset is used for training consisting of the first month of data after installation of the NILM. This is a practical scenario for nonintrusive load monitoring classifiers, as algorithms may need to be trained on data as it arrives in time. Ideally, the end user will not need to wait for years of data collection before the utility of NILM can be realized for energy scorekeeping and fault detection. The DNN was implemented with two hidden layers of 50 and then 30 ReLU-activated neurons, followed by a softmax-activated layer. The DNN was trained with Adam-optimized backpropagation and categorical crossentropy as the loss function. The data was split into 80% training and 20% validation with data stratification. Mini-batch gradient descent was used with a batch size of 64. The validation loss was used as a stopping criterion, such that training was stopped after fifteen epochs in which the validation loss did not significantly improve.

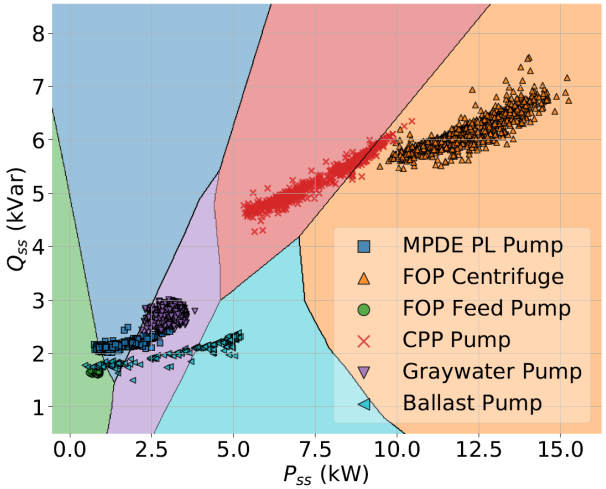


Fig. 6. Q_{ss} versus P_{ss} decision boundaries, trained on a limited dataset. The full dataset is plotted.

For these six loads, the main propulsion diesel engine (MPDE) prelube (PL) pump, fuel oil purifier (FOP) centrifugal motor, FOP feed pump, controllable pitch propeller (CPP) pump, graywater pump, and bilge and ballast pump, the trained model had perfect classification for all load events in the validation dataset. When the model was tested on the rest of the data collected on the USCG vessel (from October 2016 through September 2020) there was significantly reduced performance due to the drifting in the feature space of several loads. Fig. 6 shows the decision boundaries from the model trained on the training data with the data points from October 2016 through September 2020 plotted. The CPP pump has drifted into the FOP classification region, the MPDE PL pump has drifted into the FOP feed pump and graywater pump classification regions, and the bilge and ballast pump

has drifted into the graywater pump, MPDE prelube pump, and FOP feed pump classification regions. This performance degradation is due to the fact that with limited data, there is usually more than one large-margin low-density separator that can accurately classify the load points. There could be many separators that are consistent with the limited labelled data, but are very diverse with respect to the feature space [42]. It is difficult to determine the optimal separator based on the limited data alone.

B. Framework Verification

Two main scenarios are tested with the adaptive framework. In the first, classification with the framework starts with only a single event for each load, also known as one-shot learning [12]. In the second scenario, classification with the framework begins after one month of data has been collected by the NILM and all events have been hand-labelled. To use the framework in the first scenario, with only a single exemplar for each load, provisional preliminary and coarse boundaries are implemented in the four-dimensional feature space, as described in Section IV. Each load's initial exemplar is set to be that load's active exemplar for the correlation matching fine classifier, to start with. For each load, after 10 events have been classified, the preliminary and coarse boundaries are updated by fitting PCA hyperellipsoids to the classified load data. Algorithm 2 is used to identify and update the active exemplar. For the second scenario, the preliminary and coarse boundaries are implemented as PCA hyperellipsoids and the two fine classifiers described in Section IV-D are tested: correlation matching and a gated recurrent unit (GRU) classifier. The correlation matching algorithm is deterministic, so it is only run once. Since the GRU training is stochastic, the test using the GRU fine classifier is run ten times and averaged.

Several static techniques are also trained and tested. Since the static techniques cannot be trained well using only a single data point for each load, these classifiers are only used for the second scenario, i.e., after one month of data has been collected. A DNN classifier and support vector machine (SVM) classifier are trained using the same four-dimensional feature space that the preliminary checks and coarse classifiers use: P_{ss}, Q_{ss}, S_{peak}, and transient time. A GRU classifier is trained using the same fine feature space as the fine classifier, six seconds of P and Q. For all three models, the data was split into 80% training and 20% validation, with data stratification to allocate samples evenly based on sample class. For the SVM classifier, a radial basis function kernel is used with a regularization parameter of one. The kernel coefficient is one over the number of features times the variance of the training data. Both the DNN and GRU were trained with Adam-optimized backpropagation with a learning rate of 0.001 and categorical crossentropy as the loss function. The validation loss was used as a stopping criterion, such that training was stopped after fifteen epochs of no significant improvement. The DNN was implemented with two hidden layers of 50 and then 30 ReLU-activated neurons, followed by a softmax-activated layer. The GRU was implemented with a GRU layer with 50 ReLU-activated neurons, then two densely connected layers with 30 ReLU-activated neurons each, and finally a softmax-activated layer. For the DNN and GRU, ten different models were trained and the results were averaged. The tests are summarized below:

- Starting with a single event for each load (one-shot)
  - Adaptive correlation matching
- Starting with one month of data
  - Adaptive correlation matching
  - Adaptive GRU
  - Static SVM
  - Static DNN
  - Static GRU

The precision, recall, and F_1-score are calculated for each load,

$$\begin{gathered} precision = \frac{TP}{TP+FP}, \qquad recall = \frac{TP}{TP+FN}, \\ F_1\text{-score} = 2 \cdot \frac{precision \cdot recall}{precision + recall}, \end{gathered} \tag{4}$$

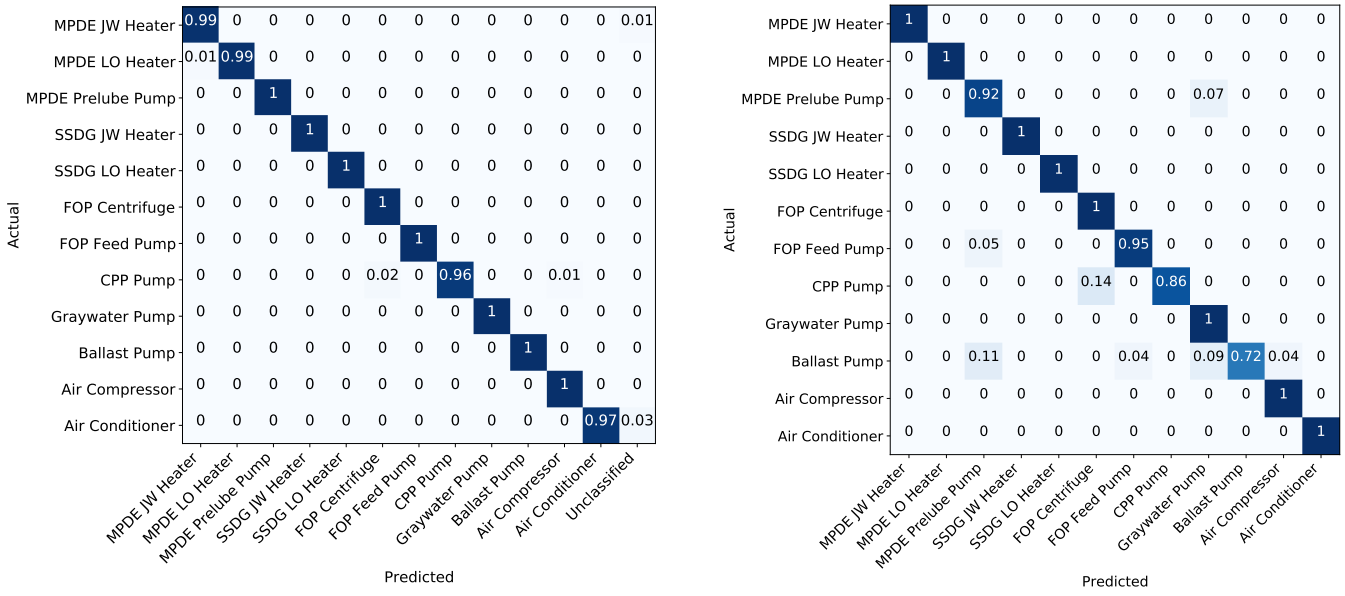
where TP is the number of true positives, FP is the number of false positives, and FN is the number of false negatives [43]. Precision is the proportion of reported events that are correct. Recall is the proportion of load events that are reported. The F_1 score is the the harmonic mean of precision and recall. Precision and recall values of one indicate perfect performance in identifying a specific class, leading to "perfect" F_1 scores equal to one. Lesser values indicate imperfect classification. The results are presented in Table II. The adaptive classifiers show improved performance over the static classifiers for several loads, including the CPP pump, MPDE prelube pump, and bilge and ballast pump. The one-shot scenario using adaptive correlation matching performs suboptimally on only the FOP centrifugal motor, due to an abnormal load operating pattern described in the following section. Fig. 7 shows two normalized confusion matrices to compare the adaptive versus static classifiers starting with one month of data. Fig. 7a shows the results of adaptive correlation matching and Fig. 7b shows the results of the static SVM. For the static SVM, the CPP pump is often misclassified as the FOP centrifugal motor, the MPDE prelube pump is often misclassified as the graywater pump, and the bilge and ballast pump is often misclassified as the MPDE prelube pump and graywater pump. These results are consistent with the demonstration in the previous two-dimensional demonstration of Fig. 6. The separability issues persist even in higher dimensions.

VI. Physical Interpretation of Results

The improved performance of the adaptive framework over the static classifiers can be explained by analyzing the physical operation of these loads over time. Load electrical characteristics are sometimes dependent on changing operating conditions. These conditions can manifest in the feature space as concept drift, which the adaptive methods seek to cope with. The drift metrics tracked by the framework are also useful prognostic indicators.

TABLE II
ACCURACY OF CLASSIFYING ON-EVENTS

	One-Shot		Trained with One Month of Data					
		Adaptive Correlation Matching		Adaptive		Static		
				Correlation Matching	GRU	SVM	DNN	GRU
Equipment	**# of Events**	F_1 **score**	**# of Events**	F_1 **score**				
MPDE Jacket water (JW) heater	116	0.991	91	0.978	0.997	0.995	0.991	0.830
MPDE Lube oil (LO) heater	277	0.993	268	0.992	0.996	0.998	0.995	0.980
MPDE Prelube pump	241	0.966	235	0.998	0.998	0.884	0.903	0.675
SSDG Jacket water (JW) heater	1177	1.0	956	1.0	1.0	1.0	1.0	0.993
SSDG Lube oil (LO) heater	2731	0.999	2705	0.999	0.999	0.999	0.999	0.992
FOP Centrifugal motor	1130	0.714	1103	0.994	0.996	0.968	0.972	0.934
FOP Feed pump	340	0.974	329	1.0	1.0	0.953	0.910	0.814
CPP pump	560	0.991	530	0.982	0.971	0.925	0.934	0.830
Graywater pump	1999	0.999	1965	0.999	0.999	0.990	0.990	0.944
Bilge and ballast pump	212	0.990	208	0.998	0.972	0.838	0.791	0.540
Air compressor	1389	0.996	229	0.983	0.954	0.977	0.986	0.874
Air conditioner	73	0.993	39	0.987	0.997	1.0	0.996	0.920



(a) (b)

Fig. 7. Normalized confusion matrices rounded to two decimal points for a) adaptive correlation matching and b) static SVM, both with one month of training data.

A. Bilge and Ballast Pump

The bilge and ballast pump is used for emptying machinery space bilges of excess water and for taking on ballast water for stability purposes [44], [45]. The bilge and ballast pump electrical signature is highly variable, likely due to air pockets within the bilge and ballast pumping system. When pumping bilges and ballast tanks, operators try to get the tanks and bilges to the lowest level possible. As a result, the pump takes in a mixture of air and water. After the pump is turned off and suction is shifted to a new tank, the air remains in the system, resulting in a prolonged start sequence in which the pump draws a variable amount of power. Over time, six micro-clusters are formed in the drift tracking process, as shown in Fig. 8.

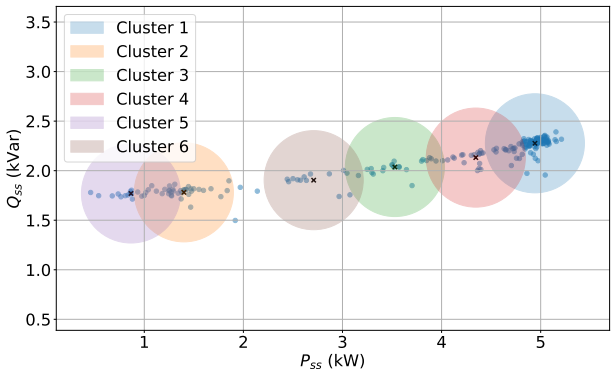


Fig. 8. Drift clusters for the bilge and ballast pump in the Q_{ss} versus P_{ss} feature space.

Fig. 9 shows the first five transients of the bilge and ballast pump, corresponding to the first month of data, showing that in each of these transients, the pump quickly reaches steady state. Each of these transients are within the first drift cluster. Also shown in Fig. 9 are five subsequent transients for the most extreme case, in which the pump draws approximately

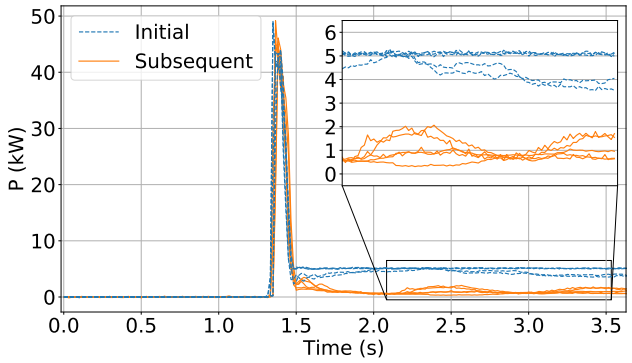


Fig. 9. The first five transients of the bilge and ballast pump compared with five subsequent transients.

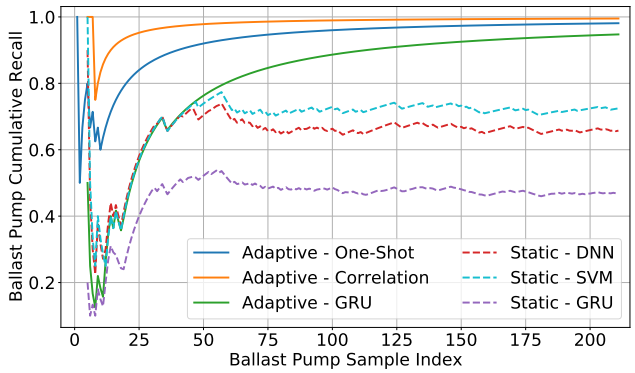


Fig. 10. Cumulative recall for six classifiers run on the full dataset for the bilge and ballast pump.

one-fifth of the power of the initial transients. These transients are all within the leftmost drift cluster. Because there are no instances of load drift in the first month of data, it would not be possible to predict this variable nature of the pump, even if synthetic data was added to the first month's training and validation data. All of the static classifiers have poor performance on the bilge and ballast pump, due to misclassification of the bilge and ballast pump as other loads. The large number of false negatives can be demonstrated by viewing a graph of the bilge and ballast pump recall over time, as shown in Fig. 10. The recall value is calculated at every load event for all the data up to that time index. Although the recall is initially poor for this load, it improves over time for the adaptive classifiers.

B. Controllable Pitch Propeller Pump

On ships, a controllable pitch propeller system provides the vessel greater maneuverability and prevents the underloading of diesel engines while operating at slow speeds [44], [46]. On USCGC SPENCER, the CPP system consists of three primary hydraulic pumps that provide pressurized hydraulic oil to the CPP system in order to maintain hydraulic control pressure at the propeller. The 'A' pump is a gear driven pump that is powered by the propulsion system's reduction gear. The pressure and flow provided by the 'A' pump is dependent on propeller shaft speed. The 'B' and 'C' pumps are electric hydraulic pumps that supplement the pressure and flow

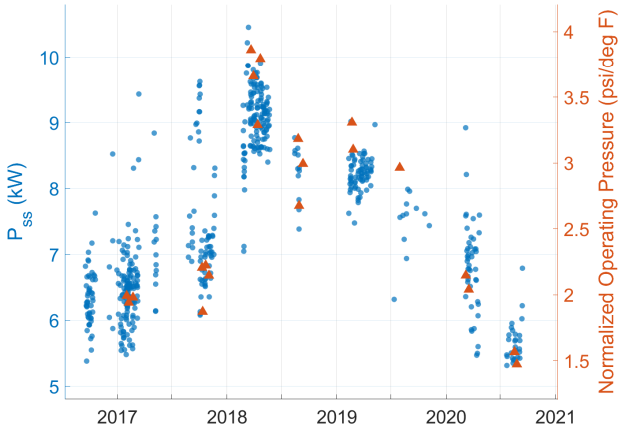


Fig. 11. CPP pump steady-state real power (P_{ss}) over time plotted with the operating fluid pressure normalized by temperature.

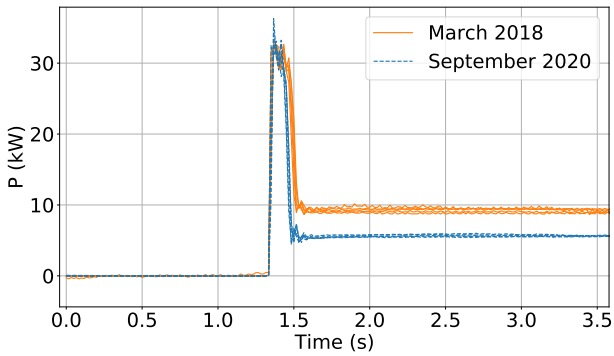


Fig. 12. Example of five turn-on transients from March 2018 and five turn-on transients from September 2020 of the CPP pump, demonstrating the significant difference in steady-state values.

provided by the gear driven pump. Hydraulic control valves maintain a constant system operating pressure.

The NILM detected changes in the monitored CPP 'C' pump's steady-state power consumption (P_{ss}), as shown in Fig. 11 for a four-year period of the SPENCER port-side CPP. This change in steady-state power correlates with the operating fluid pressure normalized by temperature, as shown on the right axis of Fig. 11. The operating pressure and temperature were obtained from the ship's logs. In February 2018, after replacement of the hydraulic control valves, there was a large increase in both the normalized operating pressure and the power draw. Then, as the normalized operating pressure slowly decreased over time, the power draw also decreased. The change was slow; however, at its worst the difference in real-power steady state between two different turn-on events is more than 4 kW. This is significant in comparison to the power drawn by the load. Fig. 12 shows an example of five transients in the time domain from March 2018, when the CPP pump was drawing significantly more than its rated power, compared with five transients from September 2020, after the CPP pump has drifted back to its original state. These time periods correspond to the maximum and minimum recorded normalized operating pressure, respectively.

As the power of the CPP pump increases, the radii of the

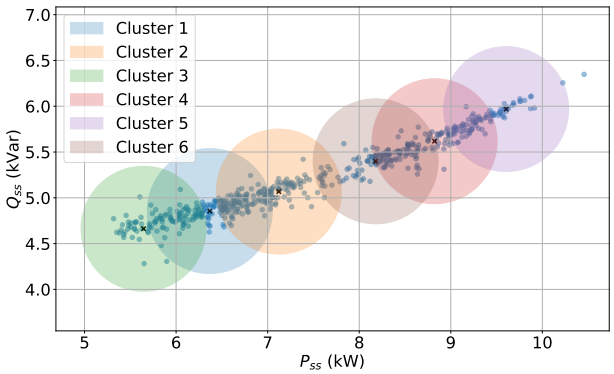


Fig. 13. Drift clusters for the CPP pump in the Q_{ss} versus P_{ss} feature space.

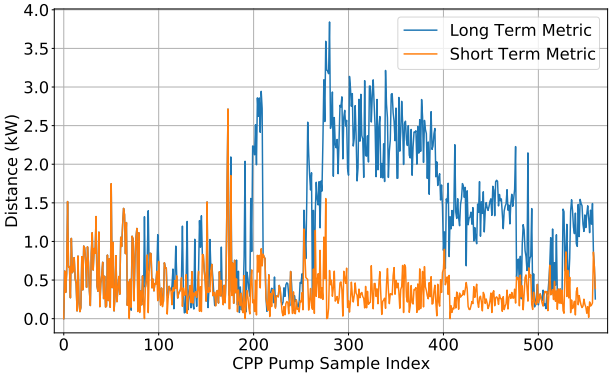


Fig. 15. CPP pump long-term and short-term distance metrics.

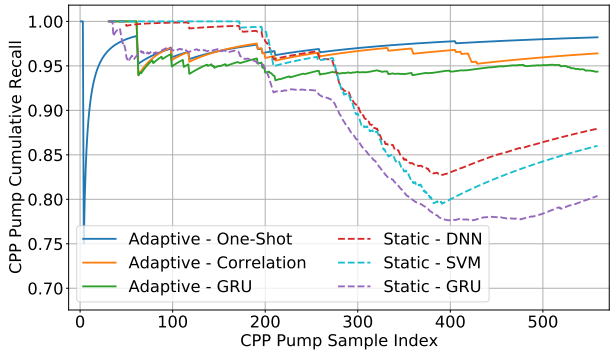


Fig. 14. Cumulative recall values for six classifiers run on the full dataset for the CPP pump.

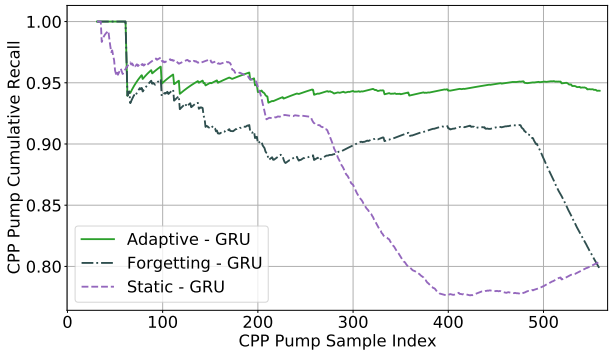


Fig. 16. Cumulative CPP recall for three GRU implementations run on the full dataset.

preliminary and coarse boundaries for this load also increase. Over time, six micro-clusters are formed in the drift tracking process, which are shown in Fig. 13. Fig. 14 shows the cumulative recall for the CPP pump to analyze the performance of both the static and adaptive classifiers over time as the load drifts, in particular the false negatives of the static classifiers. As noticed by the large negative slope in the recall, the static classifiers show a reduction in performance starting at around the 259^{th} CPP pump sample. This corresponds to February 2018, when the CPP pump showed an increase in power after replacement of the hydraulic control valves. When the CPP pump's electrical characteristics begin to drift back to its initial state, the performance of the static classifiers begin to improve. This improvement begins at around the 395^{th} CPP pump sample (August 2018) for the static classifiers.

In contrast, the adaptive classifiers do not exhibit this trend of its performance being correlated with the load drift. Potential errors of the static classifier are revealed by examining the long-term and short-term drift metrics. As shown in Fig. 15, the long-term metric increases at around the 259^{th} CPP pump sample point, the same time that the static classifiers show a reduction in performance. However, the short-term drift metric remains relatively small, corresponding to small changes between load events, indicating an incremental concept drift. Thus, the adaptive framework can still accurately identify and track the CPP pump. Then, the long-term metric can be used as an indicator of changing machine behavior.

The behavior of the CPP pump is used to demonstrate the proposed framework's ability to avoid catastrophic forgetting in the presence of recurring concept drift. For demonstration, an alternative GRU adaption scenario is tested, such that the framework was kept identical to before except for the implementation of the GRU retraining. In this alternative adaptive implementation, the GRU was retrained after the formation of every drift cluster, as before, except this time the memory for each load is the twenty most recent transients. Thus, over time, the GRU will not have access to the older data when retraining. This test was run ten times and averaged, with the recall values labelled "Forgetting" shown in Fig. 16. Although this alternative adaptive implementation is able to produce a high recall score until around the 479^{th} sample (July 2019), after this point the performance decreases drastically, presumably because it has no memory of this previously known state anymore. For the static GRU classifier, in the plot labelled "Static," the score improves when the load drifts back to the initial state, since this is the state the model was trained on. Both the adaptive GRU classifier with only recent memory ("Forgetting") and the static GRU classifier ("Static") exhibit many false negatives. In contrast, the proposed adaptive GRU implementation, labelled "Adaptive," has high recall values for the entire duration, even during recurring drift when the load drifts back to the initial state.

C. Main Propulsion Diesel Engine Prelube Pump

The main propulsion diesel engine (MPDE) prelube pump ensures adequate lubricating oil distribution during startup and

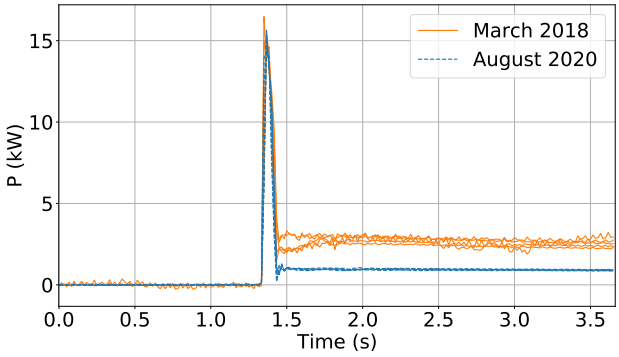


Fig. 17. Example of five turn-on transients from March 2018 and five turn-on transients from August 2020 of the MPDE prelube pump, demonstrating the significant difference in steady-state values.

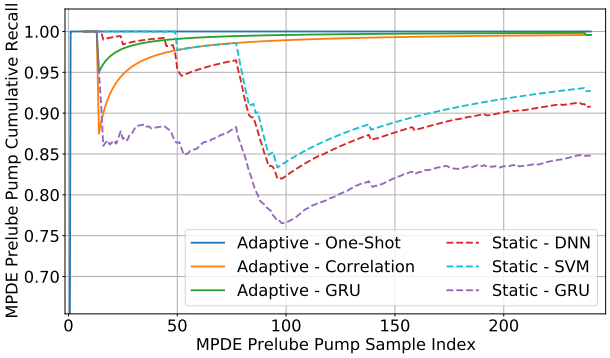


Fig. 19. Cumulative recall values for six classifiers run on the full dataset for the MPDE prelube pump.

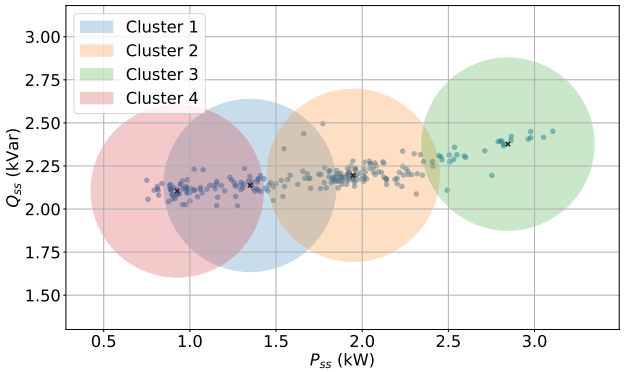


Fig. 18. Drift clusters for the MPDE prelube pump in the Q_{ss} versus P_{ss} feature space.

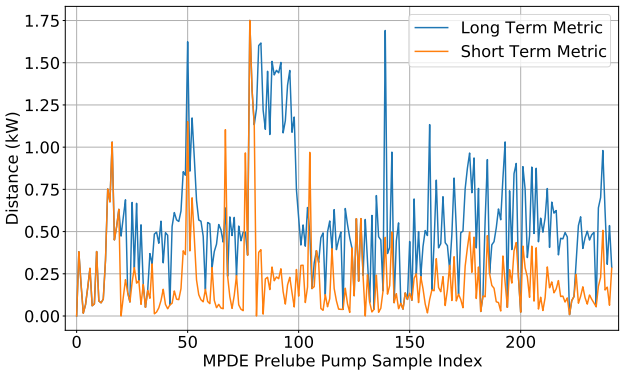


Fig. 20. MPDE prelube pump long-term and short-term distance metrics.

shutdown of the engine. The pump is manually energized prior to engine startup and operates automatically during engine shutdown [47]. The MPDE prelube pump steady-state power showed similar trends to the CPP pump. The control valves of the MPDE prelube pump were replaced at the same time as for the CPP pump, and there was subsequently a similar increase in the power of the MPDE prelube pump in February 2018, and a similar decrease in power over time afterwards. Fig. 17 shows an example of five transients in the time domain from March 2018 and five transients from August 2020. Over time, four micro-clusters are formed in the drift tracking process, as shown in Fig. 18. As the load drifts away from its initial state, the MPDE prelube pump is misclassified by the static classifiers, leading to an increasing number of false negatives. This is shown in the plot of recall over time in Fig. 19. The steep decrease in recall at about the 80^{th} MPDE prelube sample correlates exactly to February 2018. Meanwhile, the adaptive classifiers all show almost perfect performance. The short-term and long-term drift metrics are shown in Fig. 20, showing an increase in long-term metric also at the 80^{th} sample point. The short-term metric briefly increases at the time, but quickly returns to a relatively small distance.

D. Fuel Oil Purifier

A shipboard fuel oil purifier (FOP) presents an example of a sudden change in operating condition. The FOP is run frequently while underway to clean the diesel oil before use in the MPDEs or SSDGs. The FOP consists of a feed pump, which draws fuel from various storage tanks around the ship and a motor-driven centrifugal separator system [44], [48]. Observations showed two distinct operating conditions for the FOP centrifugal motor, a cold-start and a warm-start. Under normal operating conditions, the feed pump will energize, followed shortly by the centrifugal motor. The centrifugal motor energizes in the cold-start condition; that is, the motor is starting up after being off for a sufficiently long period. Large current is needed to begin rotation of the motor shaft. Often, while the feed pump is still energized, the centrifugal motor cycles. This means the centrifugal motor turns off, then re-energizes shortly after in a warm-start condition, often only a few seconds after turning off. It is likely that the motor shaft is still spinning, so the inrush current is significantly smaller. This faulty scenario puts unnecessary wear on the centrifugal motor, as these warm-starts do not have any function for the system. Example cold-start and warm-start transients are shown in Fig. 21.

Fig. 22 shows the recall and precision scores for the FOP centrifugal motor. The decreasing recall for the one-shot adaptive classifier is because the warm-start events were deemed unclassifiable. The first instance of the FOP centrifugal motor was in the cold-start condition. The warm-start condition acts as a sudden concept drift. Although the warm-start instances lead to an increase in false negatives for the FOP centrifugal motor, it is significant that they did not become false positives for another load. By identifying the instances as unclassifiable,

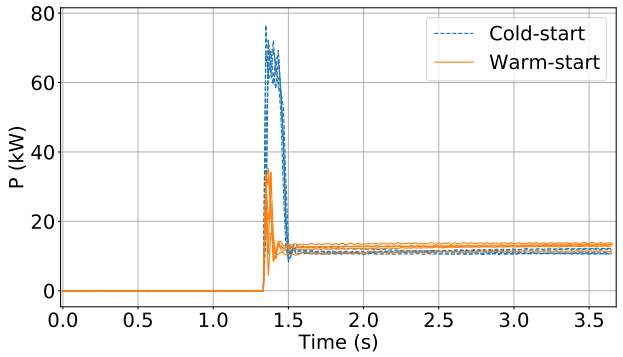
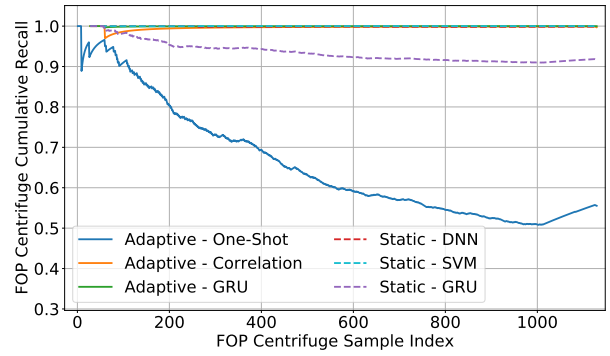
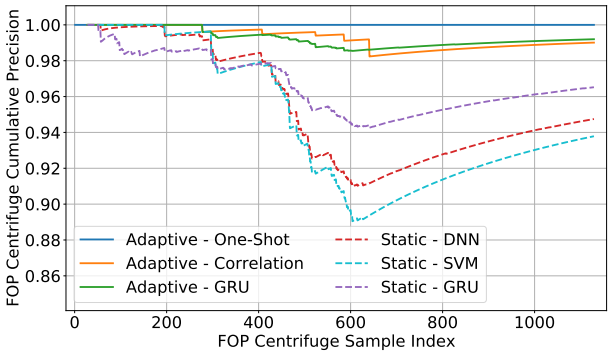


Fig. 21. Five examples each of the cold-start and warm-start turn-on transients of the fuel oil purifier centrifugal motor.



(a)



(b)

Fig. 22. Cumulative a) recall and b) precision values for six classifiers run on the full dataset for the fuel oil purifier centrifugal motor.

an operator could then identify the unclassified events as the centrifugal motor operating in a faulty warm-start condition. For the adaptive correlation matching which started with one month of data, because there was an example of the warm-start condition in the training data, the classifier was able to correctly identify both the cold-start and warm-start conditions.

Unlike the adaptive framework, which is robust to outlier load events, the static classifiers will label every incoming load event. This means every false negative of one load results in a false positive in another load. This is illustrated by examining the precision score over time of the FOP centrifugal motor. The steep decrease in precision starting at about the 408^{th} FOP centrifugal motor sample is in February 2018. As expected, this is exactly when the CPP pump recall starts to decrease, because the CPP pump is being misclassified as the FOP centrifugal motor. The precision begins to improve at around the 630^{th} FOP centrifugal motor sample (August 2018), the same time that the CPP pump recall begins to improve.

VII. Conclusion

The results of running the framework on a collection of shipboard loads demonstrate the ability of the framework to accurately detect loads even as they drift in the feature space over time. Because the classifiers are physically informed, they are effective in both one-shot and few-shot scenarios, while allowing for adaptation as data is collected in real time. For loads with time-dependent changes in steady state, the drift metrics present an indicator of possible load degradation. This information can be utilized by a watchstander as a condition-based maintenance aid, so equipment and systems can be repaired or replaced before complete failure occurs. Using the proposed semi-supervised framework, a dataset that includes variable load behavior can be built that will make supervised training more effective.

Acknowledgment

The authors gratefully acknowledge the U.S. Coast Guard and in particular the crew of USCGC SPENCER for granting access to their ship. This work was supported by the Office of Naval Research NEPTUNE program and The Grainger Foundation.

References

[1] B. Zhang, Y. Guo, Y. Li, Y. He, H. Wang, and Q. Dai, "Memory recall: A simple neural network training framework against catastrophic forgetting," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–13, 2021.

[2] S. Liu, S. Xue, J. Wu, C. Zhou, J. Yang, Z. Li, and J. Cao, "Online active learning for drifting data streams," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–15, 2021.

[3] M. Delange, R. Aljundi, M. Masana, S. Parisot, X. Jia, A. Leonardis, G. Slabaugh, and T. Tuytelaars, "A continual learning survey: Defying forgetting in classification tasks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[4] B. Celik and J. Vanschoren, "Adaptation strategies for automated machine learning on evolving data," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 9, pp. 3067–3078, 2021.

[5] J. Lu, A. Liu, F. Dong, F. Gu, J. Gama, and G. Zhang, "Learning under concept drift: A review," *IEEE Transactions on Knowledge and Data Engineering*, vol. 31, no. 12, pp. 2346–2363, 2019.

[6] J. Paris, J. S. Donnal, and S. B. Leeb, "NilmDB: The non-intrusive load monitor database," *IEEE Trans. Smart Grid*, vol. 5, no. 5, pp. 2459–2467, 2014.

[7] M. Kaselimi, N. Doulamis, A. Voulodimos, E. Protopapadakis, and A. Doulamis, "Context aware energy disaggregation using adaptive bidirectional LSTM models," *IEEE Transactions on Smart Grid*, vol. 11, no. 4, pp. 3054–3067, 2020.

[8] J. Z. Kolter and M. J. Johnson, "REDD: A public data set for energy disaggregation research," in *Proceedings of the SustKDD workshop on Data Mining Applications in Sustainability*, 2011, pp. 1–6.

[9] K. Anderson, A. Ocneanu, D. Benitez, D. Carlson, A. Rowe, and M. Berges, "Blued: A fully labeled public dataset for event-based nonintrusive load monitoring research," in *Proceedings of the 2nd KDD Workshop on Data Mining Applications in Sustainability*, 2012, pp. 12–16.

[10] J. Kelly and W. Knottenbelt, "The UK-DALE dataset, domestic appliance-level electricity demand and whole-house demand from five UK homes," *Scientific Data*, vol. 2, no. 150007, pp. 1–14, 2015.

[11] Y. Wang, L. Zhang, Y. Yao, and Y. Fu, "How to trust unlabeled data instance credibility inference for few-shot learning," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 1–1, 2021.

[12] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006.

[13] X. Pu and C. Li, "Online semisupervised broad learning system for industrial fault diagnosis," *IEEE Transactions on Industrial Informatics*, vol. 17, no. 10, pp. 6644–6654, 2021.

[14] R. N. Gemaque, A. F. J. Costa, R. Giusti, and E. Santos, "An overview of unsupervised drift detection methods," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 10, pp. 1–18, 2020.

[15] H. M. Gomes, J. Read, A. Bifet, J. P. Barddal, and J. a. Gama, "Machine learning for streaming data: State of the art, challenges, and opportunities," *SIGKDD Explor. Newsl.*, vol. 21, no. 2, p. 6–22, Nov. 2019.

[16] G. Fenza, M. Gallo, and V. Loia, "Drift-aware methodology for anomaly detection in smart grid," *IEEE Access*, vol. 7, pp. 9645–9657, 2019.

[17] H. Hu, M. Kantardzic, and T. S. Sethi, "No free lunch theorem for concept drift detection in streaming data classification: A review," *WIREs Data Mining and Knowledge Discovery*, vol. 10, no. 2, pp. 1–25, 2020.

[18] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

[19] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, and R. Morales-Bueno, "Early drift detection method," in *Fourth international workshop on knowledge discovery from data streams*, vol. 6, 2006, pp. 77–86.

[20] S. Wang, L. L. Minku, and X. Yao, "A systematic study of online class imbalance learning with concept drift," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 29, no. 10, pp. 4802–4821, 2018.

[21] A. Haque, L. Khan, and M. Baron, "Sand: Semi-supervised adaptive novel class detection and classification over data stream," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 1652–1658.

[22] F. A. Pinage, E. M. dos Santos, and J. Gama, "A drift detection method based on dynamic classifier selection," *Data Mining and Knowledge Discovery*, vol. 34, pp. 50–74, 2019.

[23] K. B. Dyer, R. Capo, and R. Polikar, "Compose: A semisupervised learning framework for initially labeled nonstationary streaming data," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 1, pp. 12–26, 2014.

[24] R. Capo, A. Sanchez, and R. Polikar, "Core support extraction for learning from initially labeled nonstationary environments using compose," in *2014 International Joint Conference on Neural Networks (IJCNN)*, 2014, pp. 602–608.

[25] M. Umer, C. Frederickson, and R. Polikar, "Learning under extreme verification latency quickly: Fast compose," in *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2016, pp. 1–8.

[26] R. Razavi-Far, E. Hallaji, M. Saif, and G. Ditzler, "A novelty detector and extreme verification latency model for nonstationary environments," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 1, pp. 561–570, 2019.

[27] J. Paris, J. S. Donnal, Z. Remscrim, S. B. Leeb, and S. R. Shaw, "The sinefit spectral envelope preprocessor," *IEEE Sensors Journal*, vol. 14, no. 12, pp. 4385–4394, 2014.

[28] E. K. Saathoff, D. H. Green, R. A. Agustin, J. W. O'Connell, and S. B. Leeb, "Inrush current measurement for transient space characterization and fault detection," *IEEE Transactions on Instrumentation and Measurement*, vol. 70, pp. 1–10, 2021.

[29] A. Jain, R. Duin, and J. Mao, "Statistical pattern recognition: a review," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 1, pp. 4–37, 2000.

[30] B. Wang, W. Shi, and Z. Miao, "Confidence analysis of standard deviational ellipse and its extension into higher dimensional Euclidean space," *PLOS ONE*, vol. 10, no. 3, pp. 1–17, 03 2015.

[31] J. M. Shapiro, G. B. Lamont, and G. L. Peterson, "An evolutionary algorithm to generate hyper-ellipsoid detectors for negative selection," in *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '05. New York, NY, USA: Association for Computing Machinery, 2005, p. 337–344.

[32] I. T. Jolliffe, *Principal Component Analysis and Factor Analysis*. New York, NY: Springer New York, 1986, pp. 115–128.

[33] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," in *NIPS 2014 Workshop on Deep Learning*, 2014, pp. 1–9.

[34] J. Peng, B. Tang, H. Jiang, Z. Li, Y. Lei, T. Lin, and H. Li, "Overcoming long-term catastrophic forgetting through adversarial neural pruning and synaptic consolidation," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–14, 2021.

[35] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, and C. H. Lampert, "icarl: Incremental classifier and representation learning," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5533–5542, 2017.

[36] H. Zhao, H. Wang, Y. Fu, F. Wu, and X. Li, "Memory efficient class-incremental learning for image classification," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1–12, 2021.

[37] J. Kirkpatrick, R. Pascanu, N. C. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, D. Hassabis, C. Clopath, D. Kumaran, and R. Hadsell, "Overcoming catastrophic forgetting in neural networks," *Proc. Nat. Acad. Sci. USA*, vol. 144, no. 13, pp. 3521–3526, 2017.

[38] A. Mallya and S. Lazebnik, "Packnet: Adding multiple tasks to a single network by iterative pruning," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2018, pp. 7765–7773.

[39] R. A. Agustin, "A load identification and diagnostic framework for aggregate power monitoring," Master's thesis, Massachusetts Institute of Technology, Feb 2021.

[40] R. Hyde, P. Angelov, and A. MacKenzie, "Fully online clustering of evolving data streams into arbitrarily shaped clusters," *Information Sciences*, vol. 382-383, pp. 96 – 114, 2017.

[41] M. Tareq, E. A. Sundararajan, M. Mohd, and N. S. Sani, "Online clustering of evolving data streams using a density grid-based method," *IEEE Access*, vol. 8, pp. 166 472–166 490, 2020.

[42] Y.-F. Li and Z.-H. Zhou, "Towards making unlabeled data never hurt," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 1, pp. 175–188, 2015.

[43] S. Makonin and F. Popowich, "Nonintrusive Load Monitoring Performance Evaluation," *Energy Efficiency*, vol. 8, pp. 809–814, 12 2014.

[44] T. J. Kane, "The NILM Dashboard: Shipboard automatic watchstanding and real-time fault detection using non-intrusive load monitoring," Master's thesis, Massachusetts Institute of Technology, June 2019.

[45] Technical Drawing 905-WMEC-529-1-1-K: Ballasting and Emergency Bilge Drainage System Diagram, accessed: 2021.

[46] Technical Manual 2817-245-A: Controllable Pitch Propeller System, accessed: 2021.

[47] Technical Manual 4647-233-A: Main Diesel Engine, accessed: 2021.

[48] Technical Manual 4921-261-A: Fuel Oil Purifier, accessed: 2021.

Daisy H. Green received the B.S. degree in electrical engineering from the University of Hawai'i at Mānoa, Honolulu, HI, USA, in 2015, and the M.S. degree and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2018 and 2022, respectively. Her research interests include the development of signal processing algorithms for energy management and condition monitoring.

Aaron W. Langham received the B.E.E. degree in electrical engineering from Auburn University, Auburn, AL, USA in 2018, and the M.S. degree in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA in 2022. He is currently pursuing the Ph.D. degree in electrical engineering and computer science at the Massachusetts Institute of Technology, Cambridge, MA, USA. His research interests include signal processing, machine learning, and computer systems for energy management.

Rebecca A. Agustin received the B.S. and M.Eng. degrees in electrical engineering and computer science from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 2019 and 2021, respectively.

Devin W. Quinn received the M.S. degree in mechanical engineering from the Massachusetts Institute of Technology in 2022. He was previously stationed as a Damage Control Officer aboard USCGC DILIGENCE and Engineer Officer onboard USCGC ESCANABA. He is currently a Lieutenant Commander with the United States Coast Guard, stationed in California.

Steven B. Leeb received the Ph.D. degree from the Massachusetts Institute of Technology, in 1993. Since 1993, he has been a member on the MIT Faculty with the Department of Electrical Engineering and Computer Science. He also holds a joint appointment with the Department of Mechanical Engineering, MIT. He is concerned with the development of signal processing algorithms for energy and real-time control applications.