Classification of Sparse and Imbalanced Time-series for IoT

Carl Ridnert, Hossein S. Ghadikolaei, *Member, IEEE*, Carlo Fischione, *Senior Member, IEEE*

***Abstract*—We address the problem of classifying sparse, imbalanced, and weakly labeled time-series. This is the typical case in Internet-of-Things (IoT) using cheap and massive number of sensors, where the datasets are sparse (e.g., due to packet loss) and imbalanced (e.g., due to heterogenous sensing rates). Unfortunately, there is no established method to perform classification in these situations. We address this problem by proposing a method that builds a new dataset consisting of shapelets extracted from the time-series, followed by a deep neural network training phase to predict the class of a shapelet. To classify a new time-series, we propose a data fusion approach based on a majority vote to substantially reduce misclassification rate of individual shapelets. Such data fusion approach is investigated analytically and numerically throughout the paper. We evaluate our approach on a highly sparse and imbalanced dataset of time-series from real-world smart-meters, and numerically show the superior performance of our approach compared to the state-of-the-art.**

***Index Terms*—IoT, Smart grids, time-series classification, probabilistic models, deep neural networks.**

I. Introduction

There is a consensus in the scientific communities that Internet of Things (IoT) and Artificial Intelligence (AI) are core technological components of future digital society [1]. IoT systems will connect devices, people, services and critical infrastructures (water, energy, transportation of people and goods) to the internet and thus IoT will potentially allow to collect massive amount of data [2]. In addition to the connectivity offered by IoT and its data collections, AI will use such data to automate and improve a wide range of municipal critical infrastructures, activities and operations, and to take autonomous controls [1], [2].

Realizing the goal of the digital society is technically very challenging due to, among others, low quality of the datasets in most practical scenarios. Indeed, IoT devices are mostly cheap (due to the economic sustainability of using a large number of devices) leading to datasets with many missing values (e.g., due to packet loss) and imbalanced classes (e.g., due to low quality or even inaccurate measurements, or variable data sizes from a multitude of technologically different devices). This is the case in various IoT scenarios [3] [4] [5] [6]. The characteristics of these data substantially complicate the tasks of AI to learn and properly react on time. Due to the dominance of time-series in monitoring use-cases, in this paper we address the problem of time-series classification with long, imbalanced and many missing data values. Currently there is no established method for classifying time-series with such properties. We propose a three-stage method of pre-processing through shapelet extraction, classifying of shapelets using deep learning, and finally a majority vote rule for fusing the classification decisions.

The authors are with KTH Royal Institute of Technology, Stockholm, Sweden (email: {ridnert, hshokri, carlofi}@kth.se).

A. Literature Review

The time-series classification task traditionally involves three main components: data preprocessing, training, and decision making. In particular, the first component includes extracting subsequences of time-series (hereafter called shapelets) for the training. We consider datasets that are sparse, imbalanced, with time-series sequences of unequal sizes. In the following, we review the existing approaches for classification of time-series where we face the problems of shapelet extraction, sparse datasets and imbalanced classes.

1) Shaplet Extraction: For time-series classification tasks, the current most accurate model that is tested on the UCR archive [7](commonly used benchmark dataset), is the hierarchical vote collective of transformation-based ensembles (HIVE-COTE) [8]. This is based on a combination of judgements of 38 different classifiers. To achieve a high classification performance, HIVE-COTE suffers a significant time-complexity for the preprocessing of the data and then training. In the preprocessing phase, HIVE-COTE uses a sequence of algorithms to extract the shapelets of every time-series, find their distances to the original time-series, and build a new training dataset. The computational complexity of the first algorithm is $\mathcal{O}(N^2T^4)$ [9], where N and T respectively are the number of training samples and the length of the time-series, making it unfavourable for applications with big datasets and long time-series.

The data preprocessing and shapelet extraction combined with deep learning classifiers have been addressed in several papers [10], [11]. The underlying assumption in these works is that the class-discriminative features of the time-series can be found in subsequences [9], [12], [13]. However, HIVE-COTE-based methods finds shapelets whose *distances* to time-series form the discriminative features. This is incompatible with the deep learning methods which, as a black-box, automatically find the discriminative features within the data. Surprisingly, feeding the raw time-series into a deep learning

classifiers may give a higher accuracy [14], eliminating the need for the tedious shapelet extraction and manual tuning of many parameters. In a recent overview of deep learning methods for time-series classification [15], it was concluded that the well known ResNet [16], fine-tuned to classify new datasets, can outperform most of the existing approaches on the UCR data. Reference [17] proposed a time-growing neural network architecture to classify time-series with cyclic patterns and improved the classification performance using spectral information. Unfortunately, these approaches are usually limited to non-sparse, fixed shapelet length, and balanced datasets.

2) Sparse Datasets: Deep learning achieves impressive performance on the time-series that contain no missing values and have balanced classes [16]. The classical approach to handle missing values is interpolation to fill values in the missing places [18]. However, this entails additional training (regression) pipeline to estimate the missing values, which unavoidably introduces some regression errors to an already low-quality dataset. Moreover, the quality of the regression can potentially be very poor for a long block of consecutive missing values, which we observe in many IoT use-cases. Reference [19] showed that many real-world time-series have weak label characteristics. That is, the class discriminatory patterns appear sparsely in a long sequences of data (say a sequence of shapelets) so that much of the data is redundant. Therefore, there is no need to reconstruct many of those missing shapelets, during which we may introduce artifact patterns. Given no missing values, reference [19] proposed a framework for extracting the relevant sequences of interest from the time-series to build a data dictionary of the relevant patterns.

A relatively recent alternative approach is classification without filling. Reference [20] addressed the sparse time-series classification task by modelling the shapelets with a Gaussian process whose parameterized kernels are the subject of learning. Reference [21] extended the classical dynamic time warping approach to measure the distance between two potentially sparse shapelets and use that for data preprocessing and classification. This approach requires storing a matrix of size $T \times T$, where remember T is the time-series length. This is potentially a memory bottleneck for very long time-series. Supervised matrix factorization is another alternative to handle sparse time-series classification [22]. This family of methods tries to find a linear mapping from the latent space to the observation space, and then use the latent representation to classify new time-series. In particular, they find two matrices whose product approximates the observed values in the matrix of collected time-series (observations). After finding the matrices, one of them can be called the "latent representation" and be used to train a neural network classifier.

3) Imbalanced Classes: The problem of class imbalance has been dealt with in several works [23] [24]. Fundamentally the problem can be attacked in a data-driven or an algorithmic-driven approach. In the data driven approach, one balances the dataset by either under-sampling classes with many samples or oversampling classes with low number of samples by some generative methods.

Reference [25] proposed the synthetic minority oversampling technique, which oversampled minority classes by generating synthetic samples rather than oversampling with replacement. Another approach is cost-sensitive classification where, by increasing the cost for misclassification of samples from minority classes, one increases the relative importance of those samples [26]. In [27], the authors showed that the more separable the data the less sensitive to class imbalance [28].

To the best of our knowledge, there is no prior work to handle low-latency time-series classification for sparse and imbalance classes with weak labels, which are prevalent in IoT scenarios [3] [4] [5] [6].

B. Our Contributions

We investigate the problem of time-series classification in the context of IoT. We consider datasets that have several challenging characteristics such as sparsity, length-variation, class imbalance, and weak labels. To build a practical classification algorithm that addresses all these challenges, we construct a new dataset by extracting shapelets from our time-series, train a deep neural network on this new dataset, and use a collaborative decision making technique (voting) to significantly improve the classification accuracy. More precisely, our main contributions are as follows:

- We propose the first approach, to the best of our knowledge, that can handle the classification of time-series with sparsity, length-variation, class imbalance, and weak labels.
- We characterize the performance of our proposed voting procedure. We characterize the regime under which this procedure can improve the classification accuracy.
- We apply our algorithm on a real-world dataset of smart meters as well as some publicly available datasets and make a comparison to previous methods dealing with spare time-series classification.

The results of this paper shows that our approach is applicable way beyond the considered datasets and can have important contributions to field of machine learning in IoT.

C. Paper Organization and Notation

The rest of the paper is organized as follows. In Section II, we describe the general problem statement and make some key definitions. In Section III, we describe our methodology and some backgrounds to the models. Section IV-B presents experimental results of our approach on various datasets. We conclude the paper in Section V.

Notations: Normal font a or A, bold font small case $\boldsymbol{a}$, bold font capital letter $\boldsymbol{A}$, and calligraphic font $\mathcal{A}$

denote scalar, vector or a time-series, matrix, and set, respectively. $\mathbb{E}[X]$ denotes the expected value of a random variable X, whose cumulative distribution function (CDF) is denoted by $F_X(x)$. For any $n \in \mathbb{N}$, we define $[n] := \{1, 2, \ldots, n\}$.

II. System Model

We assume that time-series are generated from C distinct stochastic processes; see Appendix A for formal definitions. Each process i forms a unique class with label y_i. A process models the outputs of a class in a particular situations. We denote by $\mathcal{S} = \{\mathcal{S}_1, \ldots, \mathcal{S}_C\}$ a set of time-series where each $\mathcal{S}_i = \{\boldsymbol{x}_{ij}\}_j$ contains some samples/time-series coming from class i. In the classification setting, we are interested in creating a model that is able to infer the correct class for a new time-series after a training phase on $\mathcal{S}$.

We will furthermore assume that the time-series contain missing values. We say that a time-series is sparse if for some data points $x \in \boldsymbol{x}_{ij}$ we have information only about i but the value is unknown.

Definition 1 (Sparsity)**.** *Consider a time-series $\boldsymbol{x}$ of finite length n, containing k disjoint sets of consecutive missing values with length $\{l_i\}_{i\in[k]}$, where $1 \leq l_i \leq n$ for all $i \in [k]$. For this time-series, we define the average sparsity level, denoted by $s_a(\boldsymbol{x})$, as the number of missing values normalized to n:*

$$\textit{Average sparsity level}: \quad s_a(\boldsymbol{x}) = \frac{\sum_{i\in[k]} l_i}{n}. \tag{1}$$

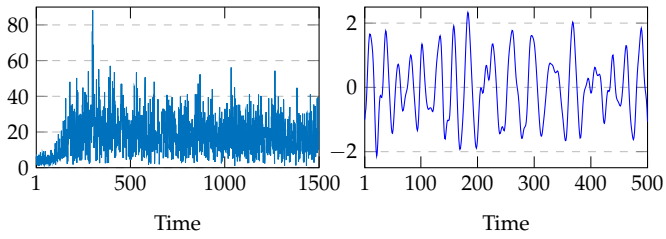
We also define the burst sparsity level (BSL), denoted by $s_b(\boldsymbol{x})$, as the quotient of consecutive missing values and the number of missing values normalized by the number of disjoint sets of consecutive missing values:

$$\textit{Burst sparsity level}: \quad s_b(\boldsymbol{x}) = \frac{\sum_{i\in[k]} l_i}{k} \tag{2}$$

For a small enough $s_b(\boldsymbol{x})$, we may approximate the missed values (using interpolation or regression [18]) and fill in with a tolerable error. The interpolation of large gaps (high BSL values) may lead to the introduction of a high degree of approximation error to the data, which could introduce artifacts and possibly spoil the possibility of properly training a classifier. Notice that many real-world applications of cyber-physical systems involving wireless communications may suffer from a high BSL where one of the reasons is the error burstiness nature of wireless transmissions [29] and blockage in modern multi-antenna systems [30]. We now formally define shapelets.

Definition 2 (Shapelet)**.** *A shapelet of some time-series $\boldsymbol{x}$ of length n is a sequence $(x_i, x_{i+1}, \ldots, x_j)$, $1 \leq i \leq j \leq n$ with a shapelet length $m := j - i + 1$.*

To illustrate the use cases of the shapelet extraction technique, we have depicted two samples from two different datasets of the UCR-archive in Fig. 1. It can be seen that the left time-series exhibits a noisy character. Moreover, its first 100 time-points have a different pattern compared to the rest of the time-series. Given that this changed distribution of the underlying stochastic process is the class-discriminatory feature, a partition of the series into shapelets could possibly result in a poor carry-over of information to the shapelets such that they contain only noise. For the FordA dataset, coming from sensors measuring the vibration of engines, it is possible to obtain more shapelets with class-discriminatory information. For this kind of data, classification-methods based on shapelet extraction would work more efficiently; indeed it was shown on the sensor datasets of the UCR archive (such as FordA) that the shapelet-extracting deep learning model proposed in [10] performed better than the other state-of-the-art models [15].



(a) SEMG. (b) FordA.

Fig. 1: Example time-series in the UCR archive. Y-axis show the readings of the meters (y values). The SEMG time-series is significantly more noisy compared to the FordA time-series.

Similar to [19], we will assume that the data is weakly labeled in the sense that each time-series has a unique label but that there exists shapelets which are not class-discriminatory. Consequently, there are many redundant shapelets, which may be detected and removed from the training pipeline both to address the challenges of our dataset and to expedite the training process [31].

III. Methodology

Our methodology consists of three stages:

- *Preprocessing:* build new dataset based on extracting shapelets from time-series and recuding the redundant due to weak labels;
- *Training:* train neural network classifiers to predict class belongings of the shapelets; and
- *Decision making:* predict new test time-series by a voting method on its shapelets.

In the following subsections, we outline these three stages, as illustrated in Fig. 2.

A. Preprocessing

Ideally we want to perform the shapelet extraction such that 1) each shapelet contains good class-discriminatory information, 2) alleviate BSL, 3) enforce fix length to shapelets, which substantially simplifies the training phase since classifiers assume a constant input size.

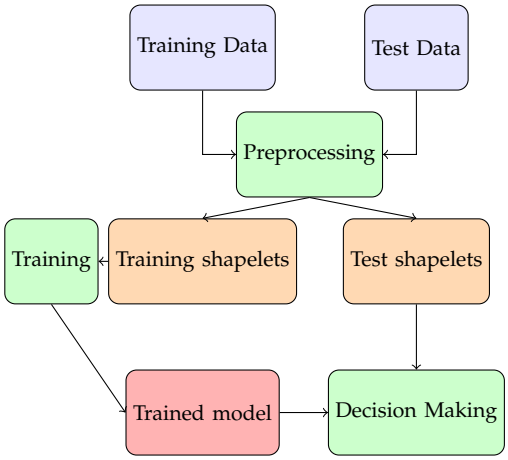


Fig. 2: Methodology flowchart. Green nodes represent the three different methodology steps, orange nodes represents processed data, grey nodes the original data and the red node the predictive model.

Algorithm 1 Shapelet extraction for each time-series in the dataset.

Require: A time-series $\boldsymbol{x}$ with length n, label y, shapelet size m, dictionary size K, variance threshold σ.

$D = []$
for k in $[0, 1, \ldots, n-1]$ **do**
  $q \leftarrow \{x_k, \ldots, x_{m(k+1))}\}$ ▷ get candidate shapelet
  if # missing values in q is less than $m/10$ and shapelet variance is bigger than σ **then**
    Interpolate/Extrapolate(q) ▷ fill in missing values
    Append D by q ▷ accept the candidate
  else
    Reject the candidate q.
  end if
end for
Run Alternating Data Selection and Function Approximation in [33] on D with K.

The shapelet extraction works by first considering all possible non-overlapping sequences of some fixed length of a time-series, followed by measuring the quality of the candidate-shapelets by the number of missing values and variance of its values. In particular, we consider a shapelet of poor quality if the percentage of missing values in that candidate is above some threshold, which is a parameter in our model. Otherwise, we accept the candidate and interpolate all potential missing values using a quadratic interpolation [32]. Since we wish to obtain some discriminatory quality of shapelets, we omit the ones that has a variance lower than some threshold. By iterating over the candidates and noting that each candidate inherits the label of the original time-series, we build a new dataset of shapelets. We finally apply the recently proposed approach to find K most representative shapelets of this dictionary and use them for the training. Algorithm 1 summarizes the shapelet extraction procedure.

With slice extraction performed on each time-series in the dataset, we obtain a new dataset. Then, we perform a feature scaling across the features of the shapelets, we use the min-max normalization defined in (3). Here $\boldsymbol{x}$ denotes the vector of a single feature across all the shapelets in the dataset.

$$\frac{\boldsymbol{x} - \min(\boldsymbol{x})}{\max(\boldsymbol{x}) - \min(\boldsymbol{x})}\,. \tag{3}$$

B. Training

Our classification task with C classes involves finding the best mapping between the observations of the sequence $\boldsymbol{x}$ or the *input* and the corresponding class labels. Assume a data set $D = \{\boldsymbol{x}, \boldsymbol{y}\} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N, \boldsymbol{y}_1, \ldots, \boldsymbol{y}_N\}$ is given where each sample $\boldsymbol{x}_n \in \mathbb{R}^d$ has a corresponding one-hot encoded label $\boldsymbol{y}_n \in \{0,1\}^C$. The model function $f(\boldsymbol{x}; \boldsymbol{\theta}) : \mathbb{R}^d \to \mathbb{R}^C$, parameterized by $\boldsymbol{\theta}$, takes as input a sample from the input space and returns a probability distribution or *likelihood* of the labels $p(\boldsymbol{y}_n | \boldsymbol{x}_n)$. The goal in classification task is to select $\boldsymbol{W}$ that maximizes the likelihood function for the observed data.

Deep neural networks (DNNs) is a family of non-linear functions $f(\boldsymbol{x}; \theta)$ equipped with a set of parameters or weights $\boldsymbol{\theta} = \{\boldsymbol{W}, \boldsymbol{b}\}$. The network with L layers equipped with weights $\boldsymbol{W} = [\boldsymbol{W}_1, \boldsymbol{W}_2, \ldots, \boldsymbol{W}_L]$ and biases $\boldsymbol{b} = [\boldsymbol{b}_1, \ldots \boldsymbol{b}_L]$ can be written as a composition of L functions $\{f_i := g_i(\boldsymbol{W}_i f_{i-1} + \boldsymbol{b}_i)\}_{i=1}^{L}$, namely $f(\boldsymbol{x}; \theta) = f_L \circ \cdots \circ f_1 = f_L(f_{L-1}(\ldots f_1(\boldsymbol{x}))$, where $\boldsymbol{W}_i$ is the set of weights connecting the outputs of the previous layer f_{i-1} to the next layer and $\boldsymbol{b}_i$ is a bias vector. The function g_i is an element-wise non-linear activation function and is applied at each element in the vector of layer i. We use the rectified linear unit activation function (ReLu) for layers $\{1, 2, \ldots, L-1\}$, defined as

$$g_i(x) = \begin{cases} x, & \text{if } x \geq 0 \\ 0, & \text{otherwise.} \end{cases}, \quad i = 1, 2, \ldots, L-1\,. \tag{4}$$

The output layer in a classification setting has typically the same number of nodes as the number of classes. We use a softmax activation function for the last layer (g_L) to obtain a probability distribution over the outcomes. Formally, let $\boldsymbol{z} = \boldsymbol{W}_L f_{L-1}(\boldsymbol{x}_n) + \boldsymbol{b}_L$ be the input to the softmax function for sample n, $[y_n^1, \ldots, y_n^C]$ denote the labels where $y_n^c = 1$ iff $\boldsymbol{x}_n$ belongs to class n, and $[\boldsymbol{z}]_k$ denote element k of vector $\boldsymbol{z}$. The probability distribution is the defined as

$$\Pr(y_n^c | \boldsymbol{x}_n, \boldsymbol{\theta}) = \frac{\exp\{[\boldsymbol{z}]_c\}}{\sum_{j=1}^{C} \exp\{[\boldsymbol{z}]_j\}}\,, \quad \text{for } c \in [C]\,. \tag{5}$$

Then classification is done by selecting the outcome with the highest likelihood.

We use the cross-entropy loss for the objective function, defined as

$$F(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{c=1}^{C} -y_n^c \log \Pr(y_n^c | f(\boldsymbol{x}_n))\,. \tag{6}$$

This function quantifies how well the data is mapped to the corresponding labels, the goal is to minimize the

Algorithm 2 Collaborative Inference.

Require: Sufficiently long time-series $\boldsymbol{x}$, trained f
  Extract M shapelets $\{\boldsymbol{x}_i\}_{i=1}^{M}$ from $\boldsymbol{x}$
  for i in $[0, 1, \ldots, M-1]$ **do**
    $y_i^* \leftarrow f(\boldsymbol{x}_i; \theta)$ ▷ obtain prediction for shapelet i
  end for
  for j in number of classes **do**
    $N_j \leftarrow \sum_{i=1}^{M} \mathbb{1}\{y_i^* = j\}$ ▷ # predictions of class j
  end for
  Prediction for $\boldsymbol{x}$ is random draw from $\{\arg\max_j N_j\}$

loss function for the training data. Gradient descent is a family of algorithms to find stationary points (local minima and saddle points) of functions with respect to their parameters $\theta = \{\boldsymbol{W}, \boldsymbol{b}\}$ [34]. The basic parameters update at iteration t is as follows:

$$\theta_{t+1} = \theta_t - \eta_t \nabla_\theta F(\boldsymbol{x}, \boldsymbol{y}, \theta_t)\,, \tag{7}$$

where η_t is the step size or learning rate parameter, and $\nabla_\theta F$ is the gradient with respect to θ. In order to compute the gradient of the loss function we can use the backpropagation algorithm [35].

The choice of learning rate is important for the convergence characteristics, if the learning rate is too large the optimal solution might be overshot and the parameter updates may not converge to a stationary point; if it is too low the updates will take longer to converge [36]. We use the ADAM-optimizer to automatically tune the learning rate for every parameter, which has been shown to work well in practice [37]. This optimizer gives a noticeable performance boost in our scenario as well. Finally, we use random perturbation to efficiently escape non-degenerate saddle points [38].

C. Decision Making

So far, we have obtained our shapelet dataset (Algorithm 1) and an algorithmic approach, given by (7), to train our classifier on this dataset. For any given test time-series $\boldsymbol{x}$, we first extract the shapelets of the proper size. We can then classify these shapelets individually. Knowing that all these shapelets are coming from the same time-series $\boldsymbol{x}$, we use a collaborative inference technique that takes as input the sequence of class predictions on individual test shapelets and returns a final decision as the class that receives the highest number of predicted shapelets. Algorithm 2 summarizes our collaborative inference. The probability of correct classification for a given shapelet can be estimated in a frequentist manner by looking at the performance of the model on training data, the probability for a shapelet being classified into each class is given as the average accuracy of the respective class. Assuming that these accuracies are high enough, as shown in the following, the intuition tells us that as we include more shapelets, the majority vote should have a higher probability of correctly classifying the time-series.

To formalize this intuition, we assume independent classification of the shapelets of a time-series and therefore model the voting procedure through a multinomial distribution. That is, we view the process of classifying a given set of shapelets as the outcome of a set of independent trials with some success probabilities.[1] In the multinomial distribution, we let $N_i \sim \text{Bin}(M, p_i)$ denote the random variables that represents the number of times an outcome of M experiments ends up in class i. Then, $\{N_i\}_{i=1}^{C}$ jointly forms a multinomial distribution and are dependent random variables since we require $\sum_{i=1}^{C} N_i = M$. The distribution function of the N_i's can be written as:

$$\Pr(N_1 = n_1, \ldots, N_c = n_C; M, p_1, \ldots, p_C) = \begin{cases} \frac{M!}{n_1! \cdots n_C!} \prod_{i=1}^{C} p_i^{n_i}, & \text{if} \quad \sum_{i=1}^{C} n_i = M \\ 0, & \text{otherwise}\ , \end{cases} \tag{8}$$

where $n_1, \ldots, n_C \in \mathbb{N}^+$, and p_i is the probability that class i is selected for a shapelet such that $\sum_{i=1}^{C} p_i = 1$. For the rest of the section, we assume without loss of generality that the correct class is 1. We start with the probability that the correct class gets the most number of votes.

$$\begin{aligned} \Pr(N_1 > \max_{j \neq 1} N_j) &= \mathbb{E}[\mathbb{1}_{\{N_1 > \max_{j \neq 1} N_j\}}] \\ &= \sum_{n_1=1}^{M} \sum_{n_2=0}^{n_1-1} \cdots \sum_{n_c=0}^{n_1-1} \Pr(n_1, \ldots, n_C; K, p_1, \ldots, p_C). \end{aligned} \tag{9}$$

We have the following lemma, proved in Appendix C:

Lemma 1. *Let the true class of a test time-series be c,* $p_c > 0.5$, *and* $\{N_1, \ldots, N_C\} \sim \text{Multinomial}(p_1, \ldots, p_C; M)$. *Then,* $\lim_{M \to \infty} \Pr(N_c > \max_{j \neq c} N_j) = 1$.

From this lemma, the collaborative decision making can improve the classification accuracy and make it asymptotically error-free, given enough shapelets M. In order to analyze the rate of convergence to any arbitrary accuracy, we start by the following example.

Illustrative Example: Recall that the correct class is 1, and assume $C = 4$. Consider three example misclassification probability vectors $\boldsymbol{q}_1 = [p, 1-p, 0, 0]$, $\boldsymbol{q}_2 = [p, 1-p-\epsilon_1, \epsilon_1, 0]$, and $\boldsymbol{q}_3 = [p, 1-p-\epsilon_1-\epsilon_2, \epsilon_2, \epsilon_2]$ for some p and small positive ϵ_1 and ϵ_2. Considering definition of majorization (Definition 8 in Appendix A), $\boldsymbol{q}_1 \succ \boldsymbol{q}_2 \succ \boldsymbol{q}_3$. Fig. 3 shows that the voting performance on these three example distributions. We observe that the performance of the majority vote on scenario $\boldsymbol{q}_i$ outperforms that of $\boldsymbol{q}_j$ when $\boldsymbol{q}_i$ majorizes $\boldsymbol{q}_j$ ($\boldsymbol{q}_i \succ \boldsymbol{q}_j$), possibly due to the Schur-convexity of (9). In particular, as the probability vector get more spread out, the probability of correct classification by the voting procedure increases, and that the probability converges to one.

These observations suggest the following conjecture.

[1]Such assumption make the mathematical analysis tractable. We do not have such assumption our experiments and numerical studies.

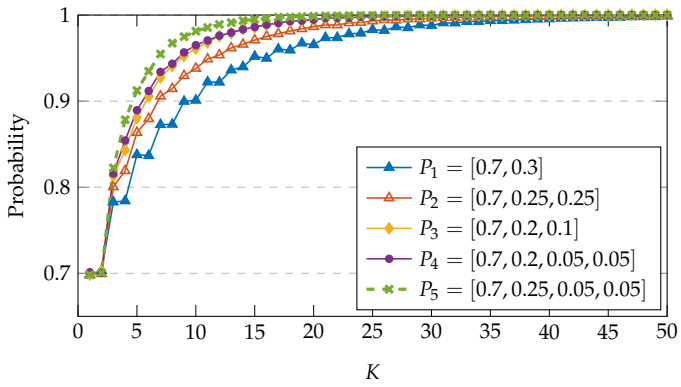


Fig. 3: The probability of correct classification using the voting procedure. Here, K is the number of votes and P_1, P_2 are probability distributions of the classes sorted in order of majorization.

Conjecture 1. *Let time-series $\boldsymbol{x}$ be of class 1 with correct classification probability of a shapelet $p_1 > 0.5$. Consider $\{N_1, N_2, \ldots, N_C\} \sim Multinomial(\{p_i\}_{i=1}^C; M)$. Consider a binary classification instance by predicting class 1 with probability p_1 and not class one with probability $1 - p_1$. The voting procedure on the a binary classification instance gives a lower bound for the performance of the voting procedure on the original problem.*

This conjecture uses the fact that the probability vectors for the binary classification instance majorizes all other probability vectors $[p_1, p_2, \ldots, p_C]$. Now, we can analyze the binomial distribution associated to $[p_1, 1 - p_1]$ lower bound. We further use the tail bounds for the Binomial distribution. In particular, by the Hoeffding's inequality, the probability that there are at least k successes out of M trials is lower bounded by [39]

$$1 - \exp\left\{-2\frac{(Mp_1 - k)^2}{M}\right\},$$

leading to the following chain of arguments:

$$\Pr(N_1 \geq \max_{i \neq 1} N_i; [p_1, \ldots, p_C]) \geq \Pr(N_1 > N_2; [p_1, 1 - p_1]) \geq 1 - e^{\frac{-2(Mp_1 - M/2)^2}{M}}. \quad (10)$$

To ensure a collaborative decision performance arbitrary close to 1, we set $\Pr(N_1 \geq \max_{i \neq 1} N_i) \geq 1 - \delta$ and obtain the following sample complexity result

$$M \geq \frac{1}{2(p_1 - 0.5)^2} \log\left(\frac{1}{\delta}\right) \quad (11)$$

for $p_1 > 0.5$.

IV. Experimental Results

In this section, we first introduce our real-world datasets and illustrate their characteristics and then report the performance of our approach on these datasets.

TABLE I: Class-distribution of time-series in our smart meter dataset.

Parameter	Cooling	Electricity	Heating	Water
# time-series (N)	385	14166	2946	1250
mean s_b	0.007	0.04	0.007	0.009
mean s_a	0.233	0.209	0.189	0.150

A. Datasets

1) Smart Meter Dataset: The data consists of several thousands of time series containing hourly measures of the consumption of utility in various buildings distributed on five classes. Table I shows some key statistics of the data. It is clear that there is a significant degree of missing values. Fig. 4 shows some examples of time-series from different classes.

In addition to our new smart meter dataset, we evaluate the performance of our approach on other datasets where the assumptions in the system model is somewhat fulfilled. In particular, the time-series should be reasonably cyclic (or cyclostationary), long with local class-discriminatory patterns. We then check the sparsity levels (resp. and class imbalance) of the time-series and make them sparse (resp. and imbalance) by randomly dropping the values (resp. time-series).

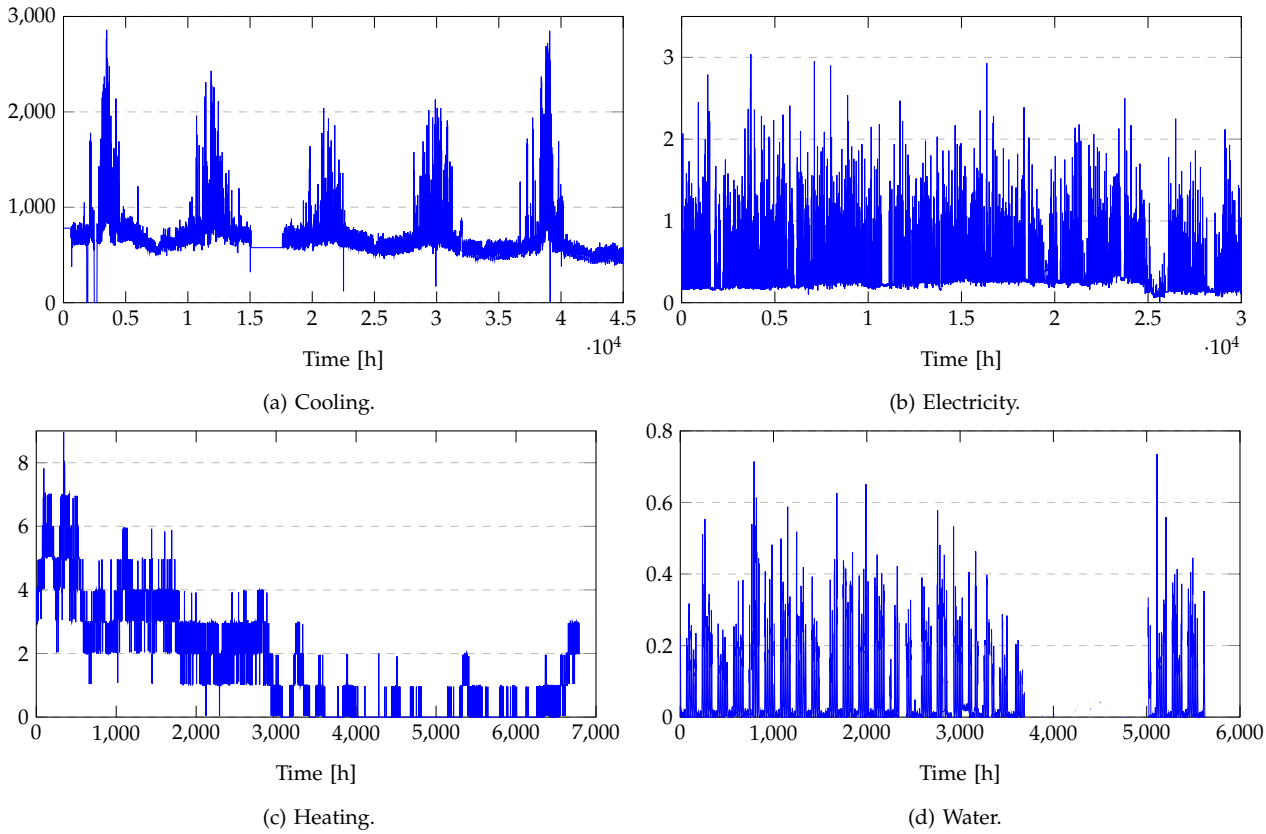
2) FordA: The FordA dataset from the UCR archive contains 3601 training and 1320 test samples of time-series of length 500 coming from sensors measuring engine noise from cars. There are two classes, representing the presence or absence of an engine problem.

3) SemgHandSubjectCh2: This dataset from the UCR archive consists of surface electromygraphy power spectrum signals from five different classes. Each time-series is of length 1500 and there are 450 samples in the training and testing sets. In addition we test our method on a dataset called SemgHandGenderCh2, which is created from the same data but where there are only two classes.

In order to assess the performance of our model, we sparsify the datasets from the UCR dataset using a burst sparsity level of $s_b(\boldsymbol{x}) = 0.3$.

B. Results

In addition to our deep neural network model, we implement a Bayesian neural network (BNN) of the same size, whose details can be found in Appendix D. For some classes of our smart metering dataset, we have observed a few meters (time-series) with many good shapelets, implying that most of the shapelets after the extraction procedure would come from those few high-quality meters. To reduce the potential bias to those meters, we set the maximum number of shapelets taken from a meter to 200. The maximum percentage of interpolated missing values per slice was 20%. Table II shows the main parameters setting of our neural networks, optimized by cross-validations. The models incorporated the Adam-optimizer using a cross-entropy loss function for the DNN and the ELBO for the BNN (see Appendix D), both using dropout [40] and batch-normalization after



(a) Cooling. (b) Electricity.

(c) Heating. (d) Water.

Fig. 4: A sample time-series from the smart meter dataset. Y-axis show the readings of the meters. Some classes show a high number of missing values. The time-series of the classes have diverse lengths. The dataset suffers class imbalance, with many time-series in some classes and much fewer in others.

each hidden layer [41]. The networks were trained on a Nvidia 970GTX GPU. During the training process, we monitored the training and test accuracies together with the loss functions.[2]

TABLE II: Model hyperparameters.

Hyper-parameter	DNN and BNN
Hidden Layers	3
Neurons Per Layer	100
Dropout rate	0.7
Learning Rate	0.001

Fig. 5 shows the accuracies of the models on smart meter dataset when the shapelet size parameter is varied and all other parameters are kept constant. From the figure, shapelet size of 24/48 samples provide a good tradeoff between performance and complexity of the underlying neural network. Interestingly, the dataset time-series include one value per hours, so shapelets of size 24 correspond to 1 day recording. The figure suggest that there are cyclostationary pattern of the utility usage in every 24/48 hours, which is meaningful. Moreover, for most experiments, the performance of BNN and DNN are very close, though the DNN training was slightly

[2]All the codes are available in https://github.com/hshokrig/Papers-Time-series-classification-IoT.

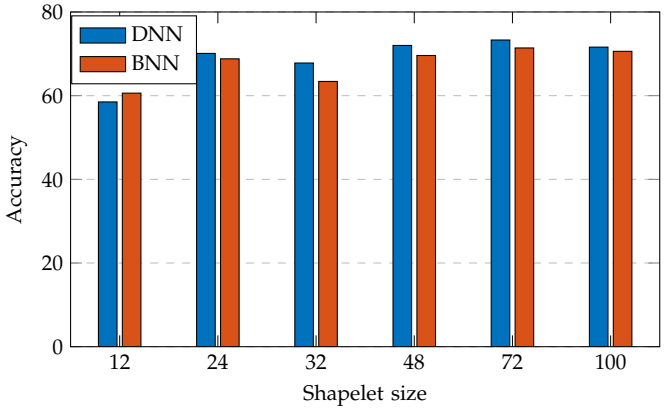


Fig. 5: Performance as a function of the shapelet size.

faster. In our observations, the BNN outperforms DNN only when we have very few shapelets in the training dataset. In the rest of the section and for the clarity of the figures, we only present the results of DNN.

Fig. 6 presents the results of the collaborative inference on different meters (time-series) in the test dataset. To obtain these results, we considered all the available shapelets for one meter, partitioned them into groups of M shapelets in each group, and ran collaborative inference on each group to find the average voting accuracy for that meter. The results match out theoret-

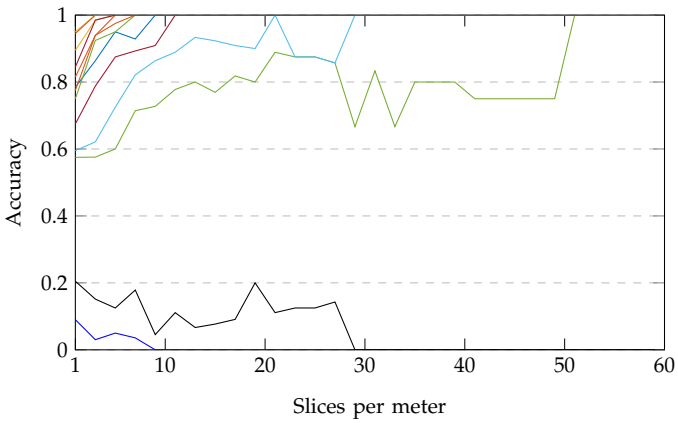


Fig. 6: Collaborative decision making on test time-series.

TABLE III: Comparison of model accuracies with reference on various datasets. The values inside parentheses for FordA, sEMG, and sEMGgender datasets show the performance on sparsified data with $s_b = 0.3$. The Smart-meter dataset has $s_b = 0.3$. "LB" stands for lower bound and shows the accuracy of random guess.

Dataset	BNN	DNN	AWARP	DD	LB
FordA	86.5 (81.5)	91.0 (84.9)	61,0 (58,6)	68.3 (67.2)	0.5
Smart-meter	77.4 (77.4)	71.8 (71.8)	N/A	35.4 (35.4)	0.25
sEMG	41.5 (21.5)	43.6 (22.7)	61.8 (62.9)	32.2 (32.4)	0.2
sEMGgender	58.3 (63.0)	61.1 (62.2)	81.1 (72.2)	66.8 (65.0)	0.5

ical conclusions that given $p_c > 0.5$, the collaborative inference finds the true class C asymptotically in M. Moreover, the higher p_1 the faster the convergence in a quadratic form, see (11). For the time-series with a very low classification accuracy on individual shapelet, say $p_c < 0.5$ (correspond to the classification accuracy for $K = 1$ in this figure), collaborative inference may be actually harmful and lead to a vanishing classification accuracy.

Next, we have compared the performance of our approach to the state-of-the-art algorithms on our three datasets. In particular, we have simulated AWARP [21] and Data Dictionary (DD) [19]. For the comparisons of our method with the data dictionary method we use the same slice size parameter. For the sake of completeness, we have reviewed DD method in Appendix B. We have also reported the lower bound, which obtained by random guess of the target class. Table III shows the performance comparisons.

First, notice that the computational complexity of the AWARP method prohibits its application to the Smart-meter dataset, where some time-series could be as long as 100k datapoints. Second, the DNN and BNN show similar performance on the datasets and vastly outperforms the DD on the Smart-meter and FordA dataset. However, the results show that our method fails to outperform the other methods on the EMG-data. This is possibly due to that EMG-data does not show a cyclostationary pattern for which or proposed algorithm works the best. Moreover, the class-discriminatory information of EMG-data exist only in long sub-sequences, similar to the example in Fig. 1(a). These results suggest that our approach is suitable for cases where the features of interest lies within relatively small sub-sequences.

V. Concluding Remarks

This paper presented a comprehensive framework to handle the classification of time-series that exhibit sparsity, class-imbalance, and large variance in lengths, which are prevalent in many IoT scenarios. The method consists of a data preprocessing by the extraction of shapelets from the time-series, training neural network classifiers on this data, and a collaborating inference procedure on shapelets when classifying new time-series. Theoretical and extensive numerical studies showed superior performance of our method to the state-of-the-art algorithms on multiple datasets, including our new dataset of smart meters.

In the future our method could be improved by using other kinds of classifiers, experimenting with various voting rules, and by finding more clever ways of optimizing the hyper parameters involved in the shapelet extraction process.

References

[1] A. Zanella, N. Bui, A. Castellani, L. Vangelista, and M. Zorzi, "Internet of things for smart cities," *IEEE Internet of Things journal*, vol. 1, no. 1, pp. 22–32, Feb. 2014.

[2] R. Du, P. Santi, M. Xiao, A. V. Vasilakos, and C. Fischione, "The sensable city: A survey on the deployment and management for smart city monitoring," *IEEE Communications Surveys & Tutorials*, vol. 21, pp. 1533–1560, Sept. 2019.

[3] D. Niyato, P. Wang, Z. Han, and E. Hossain, "Impact of packet loss on power demand estimation and power supply cost in smart grid," in *2011 IEEE Wireless Communications and Networking Conference*. Citeseer, 2011, pp. 2024–2029.

[4] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for internet of things data analysis: A survey," *Digital Communications and Networks*, vol. 4, no. 3, pp. 161–175, 2018.

[5] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, 2019.

[6] A. H. Ngu, M. Gutierrez, V. Metsis, S. Nepal, and Q. Z. Sheng, "Iot middleware: A survey on issues and enabling technologies," *IEEE Internet of Things Journal*, vol. 4, no. 1, pp. 1–20, 2016.

[7] Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, "The ucr time series classification archive," July 2015, www.cs.ucr.edu/~eamonn/time_series_data/.

[8] J. Lines, S. Taylor, and A. Bagnall, "Hive-cote: The hierarchical vote collective of transformation-based ensembles for time series classification," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 1041–1046.

[9] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Mining and Knowledge Discovery*, vol. 28, no. 4, pp. 851–881, Jul 2014. [Online]. Available: https://doi.org/10.1007/s10618-013-0322-1

[10] Z. Cui, W. Chen, and Y. Chen, "Multi-scale convolutional neural networks for time series classification," *CoRR*, vol. abs/1603.06995, 2016.

[11] A. Le Guennec, S. Malinowski, and R. Tavenard, "Data Augmentation for Time Series Classification using Convolutional Neural Networks," in *ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, Riva Del Garda, Italy, Sep. 2016. [Online]. Available: https://halshs.archives-ouvertes.fr/halshs-01357973

[12] J. Grabocka, N. Schilling, M. Wistuba, and L. Schmidt-Thieme, "Learning time-series shapelets," in *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2014, pp. 392–401.

[13] J. Lines, L. M. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 289–297.

[14] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," *2017 International Joint Conference on Neural Networks (IJCNN)*, May 2017. [Online]. Available: http://dx.doi.org/10.1109/IJCNN.2017.7966039

[15] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," 2018.

[16] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[17] A. Gharehbaghi and M. Lindén, "A deep machine learning method for classifying cyclic time series of biological signals using time-growing neural network," *IEEE transactions on neural networks and learning systems*, vol. 29, no. 9, pp. 4102–4115, 2018.

[18] M. Lepot, J.-B. Aubin, and F. Clemens, "Interpolation in time series: An introductive overview of existing methods, their performance criteria and uncertainty assessment," *Water*, vol. 9, no. 10, p. 796, 2017.

[19] B. Hu, Y. Chen, and E. Keogh, "Time series classification under more realistic assumptions," in *Proceedings of the 2013 SIAM International Conference on Data Mining*. SIAM, 2013, pp. 578–586.

[20] S. C.-X. Li and B. M. Marlin, "Classification of sparse and irregularly sampled time series with mixtures of expected gaussian kernels and random features."

[21] A. Mueen, N. Chavoshi, N. Abu-El-Rub, H. Hamooni, and A. Minnich, "Awarp: fast warping distance for sparse time series," in *2016 IEEE 16th International Conference on Data Mining (ICDM)*. IEEE, 2016, pp. 350–359.

[22] J. Grabocka, A. Nanopoulos, and L. Schmidt-Thieme, "Classification of sparse time series via supervised matrix factorization." 2012.

[23] H. He and E. A. Garcia, "Learning from imbalanced data," *IEEE Transactions on Knowledge & Data Engineering*, no. 9, pp. 1263–1284, 2008.

[24] Y. Sun, A. K. Wong, and M. S. Kamel, "Classification of imbalanced data: A review," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 23, no. 04, pp. 687–719, 2009.

[25] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "Smote: synthetic minority over-sampling technique," *Journal of artificial intelligence research*, vol. 16, pp. 321–357, 2002.

[26] Y. Sun, M. S. Kamel, and Y. Wang, "Boosting for learning multiple classes with imbalanced class distribution," in *Sixth International Conference on Data Mining (ICDM'06)*. IEEE, 2006, pp. 592–602.

[27] X. Guo, Y. Yin, C. Dong, G. Yang, and G. Zhou, "On the class imbalance problem," in *2008 Fourth international conference on natural computation*, vol. 4. IEEE, 2008, pp. 192–201.

[28] R. C. Prati, G. E. Batista, and M. C. Monard, "Class imbalances versus class overlapping: an analysis of a learning system behavior," in *Mexican international conference on artificial intelligence*. Springer, 2004, pp. 312–321.

[29] A. Goldsmith, *Wireless communications*. Cambridge university press, 2005.

[30] H. Shokri-Ghadikolaei, C. Fischione, G. Fodor, P. Popovski, and M. Zorzi, "Millimeter wave cellular networks: A MAC layer perspective," *IEEE Trans. Commun.*, vol. 63, no. 10, pp. 3437–3458, Oct. 2015.

[31] Y. Gal, R. Islam, and Z. Ghahramani, "Deep bayesian active learning with image data," *arXiv preprint arXiv:1703.02910*, 2017.

[32] I. J. Schoenberg, "Contributions to the problem of approximation of equidistant data by analytic functions," in *IJ Schoenberg Selected Papers*. Springer, 1988, pp. 3–57.

[33] H. S. Ghadikolaei, H. Ghauch, C. Fischione, and M. Skoglund, "Learning and data selection in big datasets," in *International Conference on Machine Learning (ICML)*, 2019, pp. 312–321.

[34] S. Bubeck *et al.*, "Convex optimization: Algorithms and complexity," *Foundations and Trends in Machine Learning*, vol. 8, no. 3-4, pp. 231–357, 2015.

[35] D. E. Rumelhart, R. Durbin, R. Golden, and Y. Chauvin, "Backpropagation: The basic theory," *Backpropagation: Theory, architectures and applications*, pp. 1–34, 1995.

[36] Y. Nesterov, "Introductory lectures on convex programming volume i: Basic course," 1998.

[37] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014.

[38] C. Jin, R. Ge, P. Netrapalli, S. M. Kakade, and M. I. Jordan, "How to escape saddle points efficiently," in *International Conference on Machine Learning (ICML)*, 2017, pp. 1724–1732.

[39] W. Hoeffding, "Probability inequalities for sums of bounded random variables," in *The Collected Works of Wassily Hoeffding*. Springer, 1994, pp. 409–426.

[40] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[41] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," 2015.

[42] R. H. Shumway, "Time series analysis and its applications."

[43] W. A. Gardner, "The spectral correlation theory of cyclostationary time-series," *Signal processing*, vol. 11, no. 1, pp. 13–36, 1986.

[44] L. Seymour, "An overview of periodic time series with examples," *IFAC Proceedings Volumes*, vol. 34, no. 12, pp. 61–66, 2001.

[45] A. W. Marshall, I. Olkin, and B. C. Arnold, *Inequalities: theory of majorization and its applications*. Springer, 1979, vol. 143.

[46] D. J. Berndt and J. Clifford, "Using dynamic time warping to find patterns in time series," in *KDD workshop*, vol. 10, no. 16. Seattle, WA, 1994, pp. 359–370.

[47] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: A review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

Appendix A
Preliminaries

Definition 3 ([42]). *A time-series is defined as a sequence of real-valued data points $\boldsymbol{x} = (x_i)_{i\in[n]}$ ordered in time.*

Definition 4 ([42]). *A stochastic process is defined as a set of random variables $\{X_t\}_{t=1}^n$ indexed by some deterministic variable.*

Thus we can view a time-series as a realization of some stochastic process. We can move on and define some properties of the stochastic processes.

Definition 5 ([42]). *A stochastic process $X = \{X_t\}$ equipped with some cumulative distribution function is said to be stationary if $F_X(x_{t_1+\tau}, \dots, x_{t_n+\tau}) = F_X(x_{t_1}, \dots, x_{t_n}), \quad \forall \tau, t_1, \dots, t_n \in \mathbb{R}, n \in \mathbb{N}$. This means that the joint distribution shift-invariant.*

We can relax the assumptions of the stationarity and define the following

Definition 6 ([42]). *A stochastic process is weakly stationary if it holds that*

$$\mathbb{E}[X_t] = \mathbb{E}[X_s] \tag{12a}$$

$$\operatorname{cov}(X_t, X_s) = \operatorname{cov}(X_t - X_s, 0) \tag{12b}$$

$$\mathbb{E}[|X_t|^2] < \infty, \quad \forall t, s \in \mathbb{N}. \tag{12c}$$

This means that the process has a constant mean-value and that the auto-covariance between any two time points only depends only on the distance between them.

Definition 7 ([43]). *A stochastic process is said to be periodically stationary or cyclostationary with period T if it holds that*

$$\mathbb{E}[X_{t+T}] = \mathbb{E}[X_t] \tag{13a}$$
$$\operatorname{cov}(X_{s+T}, X_{t+T}) = \operatorname{cov}(X_s, X_t) \quad \forall s, t \in \mathbb{N}. \tag{13b}$$

Note that a cyclostationary processes is not necessarily weakly stationary since we only require Definition 13 to hold for one T. If it holds for all periods, then the process is also weakly stationary. Moreover it holds that if we write the cyclostationary process as a T-variate process, then it is T-variate stationary [44].

Definition 8 ([45]). *Given $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$ and where $\boldsymbol{a}^{\downarrow}, \boldsymbol{b}^{\downarrow} \in \mathbb{R}^d$ denotes vectors with the same components but sorted in descending order. We say that $\boldsymbol{a}$ majorizes $\boldsymbol{b}$, denoted by $\boldsymbol{a} \succ \boldsymbol{b}$ iff*

$$\sum_{i=1}^{k} a_i^{\downarrow} \geq \sum_{i=1}^{k} b_i^{\downarrow}, \quad \text{for} \quad k = 1, \ldots, d, \quad \sum_i a_i = \sum_i b_i. \tag{14}$$

Appendix B
Benchmark Model

We implement a modified version of the method proposed by Hu et al [19]. The method iteratively builds a dataset of shapelets, called a *Data Dictionary*. The proposed algorithm does not handle missing values in the time-series or collaborative inference. So, we augment their algorithm to support these two functionalities. Due to the computational complexity of running their algorithm on our smart meter dataset, we will consider non-overlapping candidate shapelets. The rejection distance of a shapelet (denoted by r in [19]), is set sufficiently low in the voting procedure so that only the characteristic shapelets of a time-series will be compared to the data dictionary and thus used in the voting.

The first step in their proposed algorithm is to consider all possible shapelets of some fixed length. The subsequences are given a score based on their distances to a large number of queries, essentially judging them based on their predictive abilities. For each query q, a random subsequence from the training data, we split the data into two categories, one containing the shapelets with the same class as q and the other one containing the rest of the shapelets in the dataset. For a set of queries the scoring function is

$$\text{Rank}(S) = \sum_i \begin{cases} 1, & \text{if} \quad q_i \quad \text{has the same class as} \quad S \\ -2/(C-1), & \text{otherwise}, \end{cases}$$

where C is the number of classes. Once the training data of shapelets is scored, the highest scoring candidate is added to the data dictionary. We repeat this process for some number of times until the data dictionary contains a sufficient number of shapelets. For more details in the algorithm see [19].

To handle missing values in the shapelets, we have changed the original Euclidian-based distance to the well-known dynamic time warping [46] to find the distance of candidate shapelets to the ones in the data dictionary. Finally, the voting procedure is used to get a global prediction for a time-series based on the predictions of its shapelets.

Appendix C
Proof of Lemma 1

Assume without loss of generality that the true class is 1. We have that

$$\begin{aligned} \Pr(N_1 > \max_{j \neq 1} N_j) &\geq \Pr(N_1 > \sum_{j \neq 1} N_j) \geq \Pr(N_1 > \frac{M}{2}) \\ &= \Pr\left(\frac{N_1}{M} > \frac{1}{2}\right). \end{aligned} \tag{15}$$

Now since the marginal distribution of N_1 is Binomial, N_1 can be written as a sum of M i.i.d. Bernoulli random variables with mean p_1. The law of large numbers implies

$$\lim_{M \to \infty} \frac{N_1}{M} \to p_1, \quad \text{almost surely},$$

yielding

$$\lim_{M \to \infty} \Pr\left(\frac{N_1}{M} > \frac{1}{2}\right) \to 1 \quad \text{iff} \quad p_1 > 0.5. \tag{16}$$

This completes the proof.

Appendix D
Bayesian Neural Network

In this appendix, we outline the classification algorithm that is trained to classify the data of extracted shapelets. Consider a Bayesian neural network (BNN), which can be defined as follows:

$$\boldsymbol{W}_i \sim N(0, \boldsymbol{I}), \quad i = \{1, \ldots, L\}, \tag{17a}$$

$$\boldsymbol{y} = f(\boldsymbol{x}; \theta), \quad \boldsymbol{y} = \text{Categorical}(\boldsymbol{y}). \tag{17b}$$

The *Categorical* distribution has the following function

$$\Pr(y) = p_1^{\mathbb{1}\{y_1 = y\}} \cdots p_C^{\mathbb{1}\{y_C = y\}}. \tag{18}$$

This defines through the model a conditional likelihood $\Pr(y|x, \boldsymbol{W})$ for the data. In the following, we will take $\boldsymbol{W} = [\boldsymbol{W}_1, \ldots, \boldsymbol{W}_L]$ for notational simplicity. Lastly, $f(\boldsymbol{x}; \theta)$ is the function representing a feed-forward neural network with L layers, defined in the same way as the notation introduced for the DNN.

In the Bayesian setting, with the addition of a prior on the network parameters $\boldsymbol{W}$, it is possible to evaluate their posterior distribution. The joint posterior of the weights given the data is given by

$$\Pr(\boldsymbol{W}|\boldsymbol{x}, \boldsymbol{y}) = \Pr(\boldsymbol{W}) \frac{\Pr(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W})}{\Pr(\boldsymbol{y}|\boldsymbol{x})}.$$

The last equality holds because of the assumption that the inputs have no dependence on the parameters of the network. The normalizing constant $\Pr(y|x)$ is the true

likelihood of the data and is hard to estimate. Since the model only gives the categorical distribution conditioned on observing the weights one would have to compute the normalizing constant by integration with respect to the weights

$$\Pr(\boldsymbol{y}|\boldsymbol{x}) = \int_{\boldsymbol{W}} \Pr(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}) \Pr(\boldsymbol{W}) \, \mathrm{d}\boldsymbol{W} \, .$$

In general, this integration is computationally intractable and one has to turn to other methods for computing the posterior $\Pr(\boldsymbol{W}|\boldsymbol{x}, \boldsymbol{y})$. There exists methods such as Importance Sampling, Markov Chain Monte Carlo, and Variational Inference that can sample from such intractable distributions. Here, the choice of sampling method is Variational Inference, mainly because of its computational speed advantage over the other methods [47].

The idea in the variational inference is to choose a family of parameterized distributions $q_\phi(\boldsymbol{W}) = q_\phi(\boldsymbol{W}_1, \ldots, \boldsymbol{W}_n)$ and then find the parameters ϕ that makes the variational distribution $\boldsymbol{q}$ as similar as possible to the posterior distribution $\boldsymbol{p}$. The similarity between the posterior and the variational distribution can be computed using the Kullback-Leibner (KL), which is given by

$$\mathrm{KL}(q||p) = E_q \left[\log \frac{q}{p} \right] \, .$$

In order to estimate the posterior distribution of the weights in the network a simple normal parametric family for the variational distribution is selected with the assumption that the weights are independent. This is called the *mean-field approximation*, and it implies that the joint distribution factorizes and that the variational distribution can be written as

$$q_\phi(\boldsymbol{W}) = \prod_i^L q_{\phi_i}(\boldsymbol{W}_i) \, .$$

We use a simple normal distribution $N(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$ for the mean field approximation so that the distribution function $q_\phi(\boldsymbol{W})$ is parameterized by $\phi = \{\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2\}_{i=1}^L$. The goal is maximizing the log likelihood of the class labels given the model and data; $\log \Pr(\boldsymbol{y}|\boldsymbol{x})$ which we saw was computationally intractable. A lower bound of the likelihood can be constructed using the variational posterior distribution as follows

$$\begin{aligned}
\log \Pr(\boldsymbol{y}|\boldsymbol{x}) &= \log \int_{\boldsymbol{W}} \Pr(\boldsymbol{y}, \boldsymbol{W}|\boldsymbol{x}) \, \mathrm{d}W && \text{(19a)} \\
&= \log \int_{\boldsymbol{W}} \Pr(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}) \Pr(\boldsymbol{W}) \frac{q_\phi(\boldsymbol{W})}{q_\phi(\boldsymbol{W})} \, \mathrm{d}\boldsymbol{W} && \text{(19b)} \\
&= \log E_{q_\phi}[\frac{\Pr(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}) \Pr(\boldsymbol{W})}{q_\phi(\boldsymbol{W})}] && \text{(19c)} \\
&\geq E_{q_\phi}[\log \Pr(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{W}) \Pr(\boldsymbol{W}) - \log q_\phi(\boldsymbol{W})] && \text{(19d)} \\
&:= F(\boldsymbol{x}, \boldsymbol{y}, \phi, \boldsymbol{W}). && \text{(19e)}
\end{aligned}$$

This lower bound is simpler to evaluate and we will maximize this instead of the log-likelihood to train our BNN.