On Practical Aspects of Using RNNs for Fault Detection in Sparsely-Labeled Multi-sensor Time Series

Narendhar Gugulothu[1], Vishnu TV[2], Priyanka Gupta[3], Pankaj Malhotra[4], Lovekesh Vig[5], Puneet Agarwal[6], and Gautam Shroff[7]

[1,2,3,4,5,6,7]*TCS Research, Noida, Uttar Pradesh, 201309, India*
narendhar.g@tcs.com vishnu.tv@tcs.com priyanka.g35@tcs.com malhotra.pankaj@tcs.com
lovekesh.vig@tcs.com puneet.a@tcs.com gautam.shroff@tcs.com

ABSTRACT

In this work, we attempt to address two practical limitations when using Recurrent Neural Networks (RNNs) as classifiers for fault detection using multi-sensor time series data: Firstly, there is a need to understand the classification decisions of RNNs. It is difficult for engineers to diagnose the faults when multiple sensors are being monitored at once. The faults detected by RNNs can be better understood if the sensors carrying the faulty signature are known. To achieve this, we propose a sensor relevance scoring (SRS) approach that scores each sensor based on its contribution to the classification decision by leveraging the hidden layer activations of RNNs. Secondly, lack of labeled training data due to infrequent faults (or otherwise) makes it difficult to train RNNs in a supervised manner. We pre-train an RNN on large unlabeled data via an autoencoder in an unsupervised manner, and then fine-tune the RNN for the fault detection task using small amount of labeled training data. Through experiments on a public gasoil heating loop dataset and a proprietary pump dataset, we demonstrate the efficacy of the proposed solutions, and show that i) SRS can help point to the sensors relevant for a fault, ii) large unlabeled data can be used to pre-train an RNN-based fault detector in an unsupervised manner in sparsely-labeled scenarios, and iii) a purely unsupervised approach for fault detection (e.g. based on RNN-autoencoders) may not suffice when the number of sensors being monitored is large while the signature for fault is present in only a small subset of sensors.

1. INTRODUCTION

With the advent of Industrial Internet of Things (IIOT) (Xu et al., 2014), there is an increasing interest in remote monitoring of equipment as large amounts of temporal sensor data is available. Complex systems typically have hundreds of sensors installed across various components and sub-systems, making manual monitoring infeasible. Machine learning (ML) models can aid domain experts and engineers in monitoring data streams for: i) detecting events of interest such as anomalies, faults or novel events from time series of sensor readings as in Malhotra et al. (2015), and Malhotra, Ramakrishnan, et al. (2016), ii) forecasting machine health degradation trends for estimating remaining useful life as shown in Malhotra, TV, et al. (2016), and Gugulothu et al. (2017), and iii) diagnosing the faults or finding faulty signatures from data streams to aid root cause analysis as in Vishnu et al. (2017).

Building ML models for fault detection can help in real-time monitoring of equipment as well as help explore historical data effectively to help take key engineering decisions, e.g. to improve future manufacturing processes. Recently, deep recurrent neural networks (RNNs) based on gated units such as Long Short Term Memory (LSTM) Networks (Hochreiter & Schmidhuber, 1997) have been successfully used for modeling behavior of machines based on multi-sensor time series with applications to anomaly and fault detection (Malhotra et al., 2015; Malhotra, Ramakrishnan, et al., 2016; Yadav et al., 2016; Filonov et al., 2016).

In this work we highlight few practical challenges in using RNNs for building fault detection models, and then propose ways to address those challenges:

1.1. Limitations of RNNs-based Fault Detection Models

In this work, we note and attempt to address the following challenges while using RNNs for multivariate (multi-sensor) time series classification for fault detection with two classes of interest: Normal and Faulty.

Limitation-I: RNN Classifiers cannot provide actionable insights or explanations for their decisions. *In fact, the reason why a classifier method assigns a label to a data point can be found in the mathematical analysis of the classifier, however, it is not straightforward to identify which input (sensor) contributes more to the label estimated by the classifier*. There is

a *need for explaining classification decisions* (Lipton, 2016) of an RNN to provide actionable insights to domain experts and engineers.

Limitation-II: *Lack of labeled training data*. Despite having access to large volumes of unlabeled data, rare occurrence of faults implies not enough faulty data to train supervised models. An assumption that a machine exhibits normal behavior during initial operational life is often used to circumvent this issue by building models for normal behavior, for example in (Malhotra, TV, et al., 2016; Gugulothu et al., 2017), and then using any deviation from the modeled normal behavior to detect faults. However, sensor data may not be available (collected) for the initial life of a machine, for example, owing to late adoption of IIOT Technology. Therefore, even though large volumes of sensor data may be available, it is difficult to extract data from regions where the machine exhibited (almost) perfect behavior.

We also note that RNN autoencoder-based unsupervised approaches (Malhotra, Ramakrishnan, et al., 2016; Filonov et al., 2016) are less effective when multiple sensors are being monitored - especially when fault signature may be present in only a small subset of sensors, and domain knowledge of relevant sensors is not available.

1.2. Key contributions

If the number of sensors being monitored is large, a relevance score for each sensor can guide engineers to look at the subsystems corresponding to most relevant sensors closely - rather than going through readings of each sensor one-by-one to find the signature for the fault. *We propose an approach for sensor relevance scoring that provides useful insights to understand a given classification decision*. Our SRS approach represents a multivariate time series by the final hidden layer activations of RNNs similar to Malhotra et al. (2017) and Gugulothu et al. (2017). These representations are used to understand which input sensors discriminate the 'similar' instances belonging to different classes around a test time series of interest (refer Section 5.2 for details). This can help a domain expert to quickly validate and interpret the results of the RNN Classifier to address **Limitation-I**. The output of SRS is a relevance score for each input sensor so that higher score for a sensor implies higher likelihood of presence of fault signature in that sensor.

Our approach for building RNN classifiers first *leverages large amounts of unlabeled sensor data* via unsupervised pre-training of the RNN (Dai & Le, 2015) to overcome **Limitation-II** as described in Sections 4 and 5.1. We observe that a *semi-supervised approach* can improve the performance of unsupervised fault detection by fine-tuning the RNN through a small amount of labeled seed data. Such an approach also saves labeling costs and aids domain experts in preparing labeled datasets for further analysis.

1.3. Organization of paper

The rest of the paper is organized as follows: In Section 2, we introduce notation and describe the tasks of RNN Classification via semi-supervised learning, and formulate the problem of sensor relevance scoring. We provide a summary of the related literature in Section 3. Section 4 provides a brief overview of RNN-based sequence-to-sequence learning framework which we use for unsupervised pre-training of RNN Classifiers as described in Section 5.1, followed by the description of SRS in Section 5.2. We provide empirical evaluation and observations of the proposed approach on a pump dataset and a gasoil heating loop dataset in Section 6, and conclude in Section 7.

2. PROBLEM FORMULATION

Consider a learning set $\mathcal{D} = \{\mathbf{x}^{(i)}, c^{(i)}\}_{i=1}^{n}$ of n time series instances, where each $\mathbf{x}^{(i)} = \mathbf{x}_1^{(i)} \dots \mathbf{x}_T^{(i)}$ is a multivariate time series with $\mathbf{x}_t^{(i)} \in R^p$ for $t = 1 \dots T$, with p being the number of sensors, and $c^{(i)} \in \{c_1, \dots, c_K\}$ (e.g., c_1=Normal, c_2=Faulty for $K = 2$) is one of the K labels/classes such that the corresponding target for the classification task is a one-hot vector of length K given by $\mathbf{y}^{(i)} = [y_1^{(i)}, y_2^{(i)}, \dots, y_K^{(i)}] \in \{0, 1\}^K$. We denote the set of sensors by $\mathcal{S} = \{s_1, s_2, \dots, s_p\}$.

We consider the scenario of sparsely-labeled data, i.e. labels are available only for a few instances in $\mathcal{D}$. Let $\mathcal{L} \subset \mathcal{D}$ denote the set of labeled instances, i.e. instances for which $\mathbf{y}^{(i)}$ is known, and $\mathcal{U} \subset \mathcal{D}$ denote the set of unlabeled instances, i.e. instances for which $\mathbf{y}^{(i)}$ is unknown. Note that $\mathcal{D}$ can contain time series from multiple instances/installations belonging to multiple models (e.g. based on manufacturing year, OEM, etc.) of an equipment.

Our goals are two-fold:

- *Semi-supervised RNN classifier learning*: The goal is to leverage the unlabeled instances in $\mathcal{U}$ and the labeled instances in $\mathcal{L}$ to learn an RNN classifier f_C.
- *Computing Sensor Relevance Scores*: Given a learned classifier f_C and an estimate $\hat{c}^{(i)}$ for actual class $c^{(i)}$, the goal of SRS is to provide relevance scores for the sensors in $\mathcal{S}$, such that, the sensor that has the most discriminative information to predict $\hat{c}^{(i)}$ via f_C gets the highest relevance score.

3. RELATED WORK

Fault detection using RNNs: RNNs are used for fault detection in de Bruin et al. (2017), Ping Zhao & Khorasani (2007), and Wang et al. (2009). Approaches such as LSTM-AD (Malhotra et al., 2015) and EncDec-AD (Malhotra, Ramakrish-

nan, et al., 2016; Filonov et al., 2016) can be used for fault detection using RNNs but rely on the knowledge of normal operating regions making them less effective when such an information is not available. In this work, we extend such unsupervised approaches to the scenario where a small amount of labeled data is available to improve fault classification.

Semi-supervised time series classification using unsupervised pre-training: Semi-supervised approaches for sequence classification in text domain using RNN autoencoders has been explored in Dai & Le (2015). Ergen et al. (2017) proposed semi-supervised anomaly detection where fixed length sequences are obtained for variable length sequences by passing through an RNN and then support vector machines is used to classify the sequences. Semi-supervised approaches are used for fault detection in Zhao et al. (2015), Isermann (2005), Monroy et al. (2010), and Jiang et al. (2013), but none of these approaches are for time series. Other semi-supervised approaches using non-temporal models such as Deep Belief Nets (Wulsin et al., 2010) have been used for EEG time series anomaly detection. We have explored similar approach of unsupervised pre-training followed by supervised training on smaller data. To the best of our knowledge, this is the first attempt to adopt such an approach for fault detection from sensor data using RNN autoencoders.

Using hidden layer activations for time series representation: Recent approaches such as Malhotra et al. (2017), and Gugulothu et al. (2017) suggest that hidden layer activations of an RNN encoder can be used for learning robust time series representations. For example, Malhotra et al. (2017) use hidden layer activations of RNNs to represent time series for classification task. In this work, we leverage the representations based on hidden layer activations for explaining the classification decisions of the RNN classifier, making it different from such approaches.

Interpretable machine learning models: Explainability and interpretability of complex machine learning models, especially deep learning models, is an open research problem (Lipton, 2016). The idea of using local gradients to predict the classifier label for a data point was explored in Baehrens et al. (2010). LIME (Ribeiro et al., 2016) learns simpler models around an instance of interest in the representation space that is human-interpretable. Recently, Ribeiro et al. (2018) proposed a model that explains the behavior of complex models with high-precision rules called anchors, representing local, sufficient conditions for predictions. As detailed in Section 5.2, our work is similar to LIME (Ribeiro et al., 2016) and Vishnu et al. (2017) in the sense that it attempts to build locally interpretable simplified models. Our approach can be seen as an extension of Vishnu et al. (2017), which was proposed to understand the anomaly scores. In our approach, we explain RNN Classifiers by using the idea of representing time series via hidden state activations instead of relying only on the RNN predictions to find neighborhood. A neighborhood defined using hidden state activations is likely to better capture the relevant discriminating properties of time series compared to a neighborhood defined using occurrence in time, and provide more robust explanations. Further, it is not obvious how a LIME-like approach to explore neighborhood of a test instance can be used to explain multivariate time series classification models. Our approach tries to bridge this gap by using hidden state activations to define neighborhood and then use it to obtain sensor relevance scores via a Bayesian Network (BN).

4. BACKGROUND: RNN BASED ENCODER-DECODER

We provide a brief introduction to sequence-to-sequence (seq2seq) learning framework consisting of RNN-based encoder and decoder pair (Sutskever et al., 2014; Bahdanau et al., 2014). In general, a seq2seq model consists of a pair of multilayered RNNs – an encoder RNN and a decoder RNN – trained together. Let $\mathbf{x}_{1...T}$ denote a sequence $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_T$ of length T where each $\mathbf{x}_t \in R^p$. A seq2seq model is trained to learn a mapping from input sequences of the form $\mathbf{i} = \mathbf{x}_{1...T}$ to output sequences of the form $\mathbf{o} = \mathbf{y}_{1...T'}$. The encoder ingests the input sequence $\mathbf{i}$ and maps it to a fixed dimensional representation $\mathbf{z}_T$ through a function f_e. The decoder uses $\mathbf{z}_T$ to generate an estimate for the output sequence $\mathbf{o}$ through a function f_d. A multilayered encoder with L hidden layers, having recurrent units, iterates through the points in $\mathbf{i}$. At time t, the encoder uses the current input $\mathbf{x}_t$ and the previous hidden state $\mathbf{z}_{t-1}$ to compute the current hidden state $\mathbf{z}_t$ (through a sequence of operations as described in Appendix A.1). The final hidden state $\mathbf{z}_T$ is given by the concatenation of the hidden state vectors from all the layers in the encoder, s.t. $\mathbf{z}_T = [\mathbf{z}_{T,1}, \mathbf{z}_{T,2}, \ldots, \mathbf{z}_{T,L}]$, where $\mathbf{z}_{T,l}$ is the hidden state vector for the l^{th} layer of encoder. The total number of recurrent units in the encoder is given by $c = \sum_{l=1}^{L} c^l$, s.t. $\mathbf{z}_t \in R^c$, where c^l is the number of units in l^{th} layer. The decoder has the same network structure as the encoder, and uses the hidden state $\mathbf{z}_T$ as its initial hidden state, and iteratively goes through a set of transformations to generate $\hat{\mathbf{o}}$ as an estimate for $\mathbf{o}$. We provide details of how we use seq2seq for unsupervised pre-training of RNN Classifier in our context in Section 5.1.

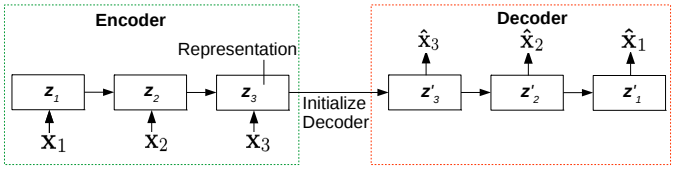


Figure 1. Inference using RNN Encoder-Decoder pair for a toy time series $\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$.

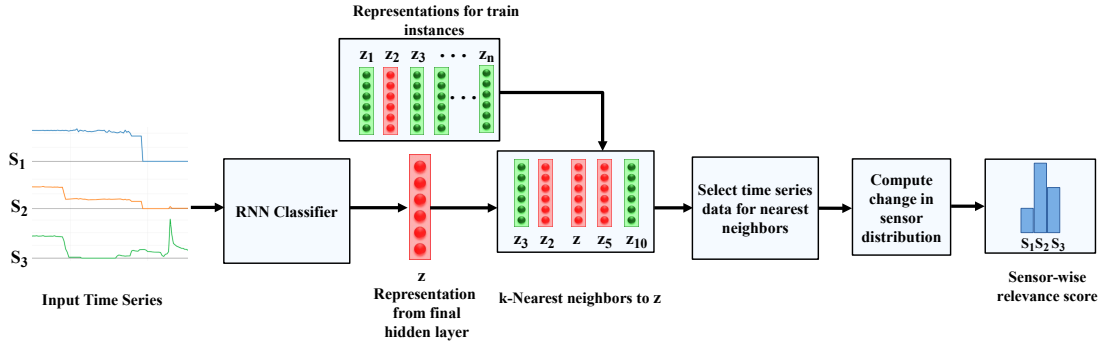


Figure 2. Sensor Relevance Scoring Approach. Here, Red: 'Faulty' predictions, Green: 'Normal' predictions

5. FAULT DETECTION AND SRS

We consider an RNN classifier trained in a semi-supervised manner through a small labeled ($\mathcal{L}$) and a large unlabeled ($\mathcal{U}$) set for building the fault detection model. We use reconstruction task for RNN autoencoder to incorporate unlabeled data in the standard supervised learning framework. We then provide details of how to get further insights into the decisions of the RNN classifier through SRS approach.

5.1. Semi-supervised RNN classifier learning

We first use instances in $\mathcal{D} = \mathcal{L} \cup \mathcal{U}$ (without the label information for instances in $\mathcal{L}$) to **train RNN Encoder-Decoder (RNN-ED) in an unsupervised manner** using reconstruction error as loss (refer Equation 1) to obtain the parameters $\mathbf{W}_e$ of the encoder function f_e. This unsupervised approach to train an RNN encoder using large amount of unlabeled data is used to obtain a robust model capturing the statistical properties of data. We train the RNN-ED in a manner that output time series $\mathbf{o} = \mathbf{x}_{T...1}$ is in reverse order to the input time series $\mathbf{i} = \mathbf{x}_{1...T}$ (similar to Gugulothu et al. (2017)). Figure 1 provides inference flow of RNN-ED for a sample time series $\mathbf{x}_1\mathbf{x}_2\mathbf{x}_3$. The overall process can be thought of as a non-linear mapping of the input multivariate time series to a fixed-dimensional vector representation (embedding) via an encoder function f_e, followed by another non-linear mapping of the fixed-dimensional vector to a multivariate time series via a decoder function f_d:

$$\begin{aligned} \mathbf{z}_T^{(i)} &= f_e(\mathbf{x}^{(i)}; \mathbf{W}_e) \\ \hat{\mathbf{x}}^{(i)} &= f_d(\mathbf{z}_T^{(i)}; \mathbf{W}_d) \\ \mathbf{e}_t^{(i)} &= \mathbf{x}_t^{(i)} - \hat{\mathbf{x}}_t^{(i)},\ t = 1, \ldots, T \\ C_1(\hat{\mathbf{x}}^{(i)}, \mathbf{x}^{(i)}) &= \frac{1}{T} \sum_{t=1}^{T} \| \mathbf{e}_t^{(i)} \|_2^2 \end{aligned} \tag{1}$$

where, $\mathbf{W}_e$ and $\mathbf{W}_d$ represent the parameters of the encoder and decoder, respectively, and $\| \ . \ \|_2$ denotes the L_2-norm. The RNN-ED is trained to minimize the loss function given by the squared reconstruction error $\sum_{i=1}^{n} C_1(\hat{\mathbf{x}}^{(i)}, \mathbf{x}^{(i)})$. (Note: As mentioned in Equation 1, the decoder takes $\mathbf{z}_T^{(i)}$ as the only input. This ensures that all the relevant information in the time series is captured by the encoder as shown in Malhotra et al. (2017) and Gugulothu et al. (2017).

Initialize RNN Classifier using $\mathbf{W}_e$ and tune classifier using labeled set $\mathcal{L}$: Since the RNN-ED is already trained to reconstruct the time series, and therefore, capture the relevant information in the final hidden state of the RNN Encoder, it is a reasonable choice for initialization of the weights of the RNN Classifier as demonstrated in our experiments in Section 6.

The weights $\mathbf{W}_e$ of the encoder are used to initialize a supervised RNN Classifier. The RNN Classifier with parameters $\mathbf{W}_e$ from the encoder and parameters $\mathbf{W}_c$ connecting the encoder state $\mathbf{z}_{T,L}^{(i)}$ at last time-step T with a softmax layer with K units, are then trained together by minimizing the cross-entropy loss given by C_2:

$$\begin{aligned} \hat{\mathbf{y}}^{(i)} &= \texttt{softmax}(\mathbf{W}_c\, \mathbf{z}_{T,L}^{(i)} + \mathbf{b}) \\ C_2(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)}) &= -\sum_{k=1}^{K} y_k^{(i)} \cdot log(\hat{y}_k^{(i)}) \end{aligned} \tag{2}$$

The final model is obtained via stochastic gradient descent based training of the neural network to minimize $\sum_{i=1}^{m} C_2(\hat{\mathbf{y}}^{(i)}, \mathbf{y}^{(i)})$, where m is the number of labeled instances in $\mathcal{L}$ ($m << n$ in our experiments). For the binary classification task, we have $K = 2$ corresponding to "Normal" and "Faulty" classes. The same approach can be extended to fault diagnostics where K would refer to the number of fault types with $K > 2$ (with one Normal class and $K - 1$ fault modes).

5.2. Sensor Relevance Scoring Approach

We propose Sensor Relevance Scoring (SRS) algorithm to provide insights into the predictions of the fault detection model (summarized in Algorithm 1 and illustrated in Figure

2). SRS provides relevance scores for the input sensors in $\mathcal{S}$ by using hidden layer activation $\mathbf{z}_{T,L}^{(i)}$ to represent the input $\mathbf{x}^{(i)}$. We define the neighborhood of test instance i based on similarity of its representation $\mathbf{z}_{T,L}^{(i)}$ to the representations of other instances in $\mathcal{L}$. A Bayesian Network (BN) is then used to obtain the sensor relevance scores as we describe next.

Algorithm 1: Sensor Relevance Scoring

Require: $\mathbf{x}^{(i)}$, $\hat{c}^{(i)}$, $\mathbf{z}_{T,L}^{(i)}$, $\mathcal{Z}$, $\mathcal{L}$;

1: Get $\mathcal{L}_{+}^{(i)} \subset \mathcal{Z}$ as N nearest neighbors of $\mathbf{z}_{T,L}^{(i)}$ with estimated class $\hat{c}^{(i)}$;
2: Get $\mathcal{L}_{-}^{(i)} \subset \mathcal{Z}$ as N nearest neighbors of $\mathbf{z}_{T,L}^{(i)}$ with estimated class $\neq \hat{c}^{(i)}$;
3: Estimate $P(S, C)$ using $\mathcal{L}_{+}^{(i)}$, $\mathcal{L}_{-}^{(i)}$, and $\mathbf{x}^{(i)}$;
4: For s_j in $\mathcal{S}$;
Compute $P(S_j \mid C = \hat{c}^{(i)})$ and $P(S_j \mid C \neq \hat{c}^{(i)})$;
Compute $R(s_j)$ using Equation 3;

5.2.1. Finding neighborhood based on time series representation

Let $\mathcal{Z}$ denote the set of representations for the time series in $\mathcal{L}$, and let $d(z_1, z_2)$ denote the metric computing the distance between two representations (e.g. Euclidean distance). Consider $\mathcal{L}_{+}^{(i)} \subset \mathcal{L}$ to be the set of N time series whose representations are closest to $\mathbf{z}_{T,L}^{(i)}$ according to the metric d, and for which the estimated class is same as $\hat{c}^{(i)}$. Also, let $\mathcal{L}_{-}^{(i)} \subset \mathcal{L}$ denote the set of N time series whose representations are closest to $\mathbf{z}_{T,L}^{(i)}$ based on the metric d but for which the estimated class is different from $\hat{c}^{(i)}$.

5.2.2. Computing Sensor Relevance Scores using Bayesian Network

Consider a categorical random variable C corresponding to the estimated class, and a set of p discrete random variables $S = \{S_1, S_2, ..., S_p\}$ corresponding to the p sensors. We model the dependence between S and C via a BN with $p + 1$ nodes. The network models the joint distribution $P(S_1, S_2, ..., S_p, C)$ for the set of random variables $X = \{S_1, S_2, ..., S_p, C\}$. (Since we are only interested in modeling the dependence between each sensor and the predicted class, a naive Bayes model with C being the parent node and S_js being children can be assumed as shown in Figure 3. Wherever possible, a network structure can be assumed based on domain knowledge of the sensors dependencies.) The parameters of the BN, i.e. the conditional probability tables, are learned using the values the sensors take from all the time series instances in $\mathcal{L}_{+}^{(i)} \cup \mathcal{L}_{-}^{(i)} \cup \{\mathbf{x}^{(i)}\}$.

A random variable $S_j \in S$ is considered to have k possible outcomes $[b_i^1, b_i^2, ..., b_i^k]$ corresponding to k discretized bins for the range of values the variable can take. A p-dimensional vector of sensors $\mathbf{x}'^{(i)}_t$ at time t in time series instance $\mathbf{x}'^{(i)} \in \mathcal{L}_{+}^{(i)} \cup \mathcal{L}_{-}^{(i)} \cup \{\mathbf{x}^{(i)}\}$ yields one observation for the set of random variables $S = \{S_1, S_2, ..., S_p\}$. The marginal probability distribution for S_j is given by $P(S_j)$ and the conditional probability distribution for S_j, given an estimated class $C = \hat{c}^{(i)}$ is given by $P(S_j | C = \hat{c}^{(i)})$. The change in the distribution of the random variable S_j conditioned on C is used to quantify the effect of the jth sensor on the estimated class, and to obtain its relevance score. We quantify this change in terms of a relevance score R given by:

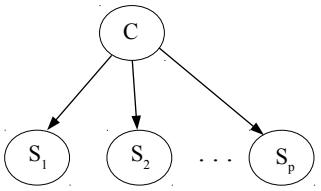


Figure 3. Bayesian Network considered to compute Sensor Relevance Scores.

$$R(s_j) = D_H(P(S_j \mid C = \hat{c}^{(i)}), P(S_j \mid C \neq \hat{c}^{(i)})) \quad (3)$$

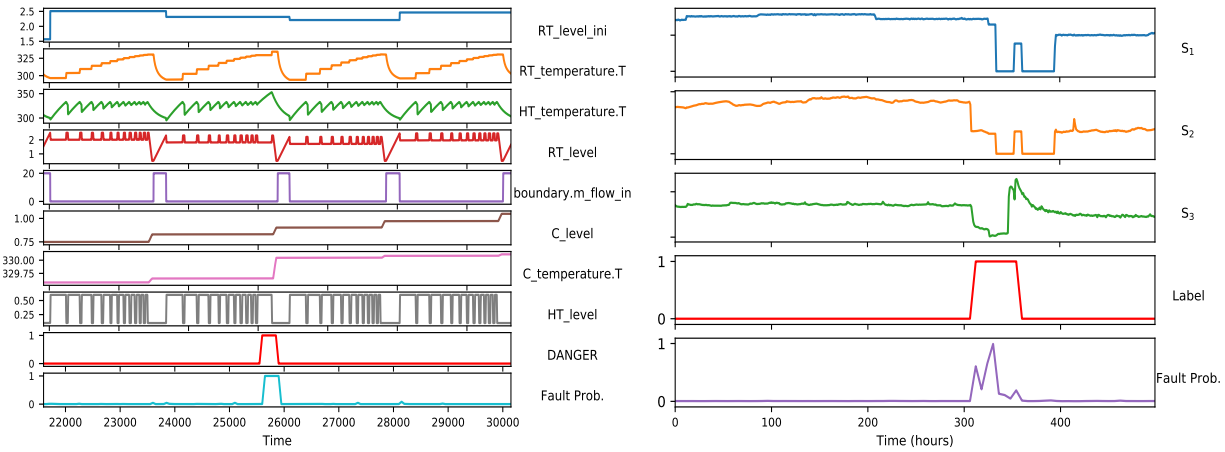
where $D_H(P, Q)$ is a metric that quantifies the difference between two probability distributions P and Q. We compute $D_H(P, Q)$ based on Euclidean Distance, Hellinger Distance (Equation 7) and Earth Mover's Distance (Equation 8). Higher the value for $R(s_j)$, higher is the effect of sensor s_j on the estimated class.

6. EXPERIMENTAL EVALUATION

We use two multi-sensor time series datasets (Table 1) for our experiments: i) GHL: a publicly available Gasoil Heating Loop dataset (Filonov et al., 2016), ii) Pump: a proprietary real-world pump data.

GHL dataset contains data from 14 sensors[1] capturing operational behavior of a Gasoil Heating Loop (GHL) System. There are two types of faults; one of the sensors (RT_level) is relevant for detecting 24 faults while another sensor (HT_temperature.T) is relevant for detecting remaining 24 faults (refer Fig. 4(a)). Pump dataset contains data from 5 sensors where the signature for fault can be captured by considering at least two of the most relevant sensors (referred to as S_1 and S_2 here) as the temporal correlation between them goes off during faulty operation (refer Fig. 4(b)). We provide more details on the datasets in Appendix A.3.

[1] Based on electronic communication with authors of (Filonov et al., 2016), GHL dataset contains data from 19 sensors, but 5 of those sensors are auxiliary sensors used to add random noise to the values of remaining sensors.



(a) GHL Fault: Increase in HT_temperature.T (DANGER is used as target label)

(b) Pump Fault: Correlation b/w S_1 and S_2 goes off.

Figure 4. Sample results using EDC on subsequences with faults.

Table 1. Datasets description: Instance level statistics. Here **T**: window length, **p**: number of sensors, **n**: number of windows.

Dataset	**T**	**p**	**#Faults**	**n**
GHL	100	19	48	56,750
Pump	48	5	28	21,742

Table 2. Datasets description: Window-level statistics (T: Train Set, V: Validation Set, Te: Test Set)

Dataset	**#instances**			**#windows (n)**					
	T	V	Te	T_N	T_F	V_N	V_F	Te_N	Te_F
GHL	22	10	16	26331	161	13809	89	16278	82
Pump	13	5	10	11130	102	3100	29	7312	69

6.1. Semi-supervised Classification

We compare following three models for the binary classification task of fault detection (Normal vs Faulty classification):

- *unsupervised* RNN Encoder-Decoder (**ED**) where reconstruction error is used as score for faulty behavior (similar to Malhotra, Ramakrishnan, et al. (2016), and Filonov et al. (2016))
- a purely *supervised* approach based on RNN Classifier (**C**)
- a *semi-supervised* RNN classifier (**EDC**) as described in Section 5.1.

To compare the significance of domain knowledge for unsupervised and supervised models, we consider variants S-ED and S-EDC for ED and EDC, respectively, where only the relevant sensors are used for learning the models.

We split the data into train, validation and test sets (Table 2). To evaluate the effectiveness of pre-training to help deal with small labeled set, we consider two scenarios: i) with large labeled train dataset (22 and 13 fault instances for GHL and Pump datasets, respectively, as shown in Table 2), ii) with only 25% of the large labeled train dataset (5 and 4 fault instances for GHL and Pump datasets, respectively). We consider the subset of large labeled train set to arrive at the small labeled set, while the test set remain same for both the scenarios. We consider fixed-length overlapping windows as time-series instances from one large sequence.

We use early stopping and dropout (Zaremba et al., 2014) with a value of 0.2 over the feedforward connections for regularization, and use Adam optimizer (Kingma & Ba, 2014) for optimizing the weights of the networks with initial learning rate of 0.0005 for all our experiments. Since the datasets are highly imbalanced with large number of normal windows and very few faulty windows (Table 2), we use minority class over-sampling in each mini-batch of size 128 (ensuring at least 4 instances from faulty class in each mini-batch). We chose the best architecture (via grid search on number of layers (L) and number of hidden units per layer) as the one with maximum AUROC for supervised and semi-supervised models, and the one with least reconstruction error for unsupervised models on the validation set.

We make the following ***key observations*** from the results in Table 3:

- C_7 *vs* C_6: Fine-tuning an unsupervised model for the classification task using small labeled data outperforms a purely supervised model trained using small labeled data.
- C_6 *vs* C_1: Little labeling effort to generate small labeled set to train supervised or semi-supervised models can significantly improve classification performance compared to purely unsupervised approaches.

Table 3. Comparison of Fault Detection Models: Supervised (C), Unsupervised (ED), Semi-supervised (EDC), approaches in terms of AUROC. Here, R.S.: Relevant sensor(s), A.S.: All Sensors.

	Unlabeled data		**All labeled data**			**Small labeled data**	
	C_1	C_2	C_3	C_4	C_5	C_6	C_7
Dataset	**ED** (A.S.)	**S-ED** (R.S.)	**S-EDC** (R.S.)	**C** (A.S.)	**EDC** (A.S.)	**C** (A.S.)	**EDC** (A.S.)
GHL	0.582	0.779	0.991	0.979	0.981	0.776	0.958
Pump	0.805	0.868	0.959	0.964	0.969	0.937	0.965

- C_7 *vs* C_2: Fine-tuning an unsupervised model for the classification task using small amount of labeled data can yield significant improvement in classification performance over an unsupervised model, even when the knowledge of sensors containing the faulty signatures is not available.
- C_7 *vs* C_4: Unsupervised pre-training using large unlabeled data can lead to robust time series encoders that need small amount of supervision via a small labeled set to achieve performance comparable to purely supervised models trained on large labeled datasets. This can be very useful in practical settings to save labeling effort.
- C_3 *vs* C_5: Semi-supervised models trained using raw sensors without any knowledge of relevant sensors perform comparable to semi-supervised models built using only the relevant sensors, suggesting that semi-supervised models are robust.
- C_2 *vs* C_1: We observe that S-ED performs better than ED suggesting that when numbers of sensors is large (e.g. 14 in GHL), an unsupervised approach for fault detection may not be able to capture the weak information from relevant sensors especially when the signature is present in only a few sensors (e.g. 1 in GHL).

6.2. SRS Evaluation

We consider the EDC classification models (Column C_5 in Table 3) for evaluating sensor relevance scoring approach. We evaluate our sensor relevance scoring algorithm using **Recall@K** metric: we choose the threshold over the estimates $\hat{y}$ for the 'Faulty' class for which F_1-score is maximum on the validation set. We consider the instances for which $\hat{c}$ is 'Faulty'. Then, for each 'Faulty' prediction, we find its neighbors (based on Euclidean distance as described in Section 5.2.1) from 'Normal' and 'Faulty' predictions in the train set. Then, we calculate the relevance scores for all the sensors for this prediction. The sensor relevance scoring for an instance is considered to be 'Correct' when the "actual" most relevant sensor is present in the set of "estimated" top-K relevant sensors (in descending order of sensor relevance score) as given by the SRS algorithm. Therefore, the metric Recall@K denotes the fraction of 'Faulty' test predictions for which the actual most relevant sensor is present in the set of estimated top-K relevant sensors.

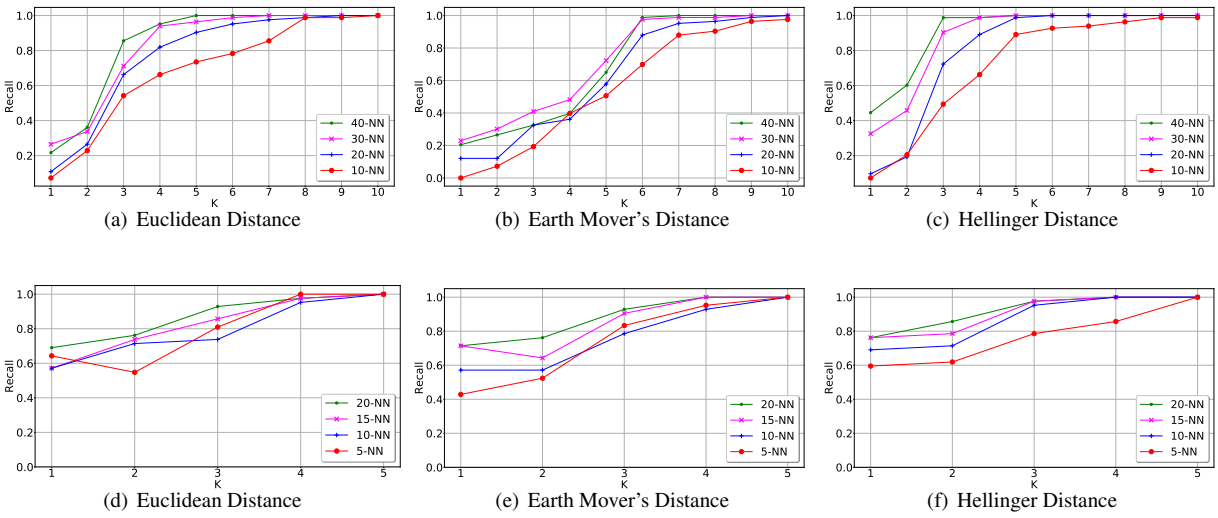
6.2.1. Observations

Figure 5 shows the performance of the SRS algorithm for various values of N (number of nearest neighbors) and the three histogram distance metrics considered. We observe that relevance scores based on Hellinger Distance yield the best performance in terms of Recall@K for both datasets. For GHL, we get a **Recall@3** of **0.99** when 40 nearest neighbors are considered, implying that while scoring the 14 sensors considered, the actual most relevant sensor is almost always present in the estimated top-3 sensors. Similarly, for the Pump dataset, we get a **Recall@3** of **0.98** when 20 nearest neighbors are considered, implying that out of the 5 sensors considered, the actual most relevant two sensors are almost always present in the top-3 estimated relevant sensors.

Sample histograms for the top-scored and the bottom-scored sensors for the two datasets are shown in Figure 6, indicating a large difference in distribution of values taken by top-scored sensors for 'Normal' and 'Faulty' classes, while very similar distribution of values taken by bottom-scored sensors. This suggests that exploring the neighborhood of a 'Faulty' test instance through the final hidden layer outputs, can help understand the behavior of classifier. For a time series classified as faulty, sensor relevance score obtained based on the final hidden state of the RNN Classifier can be used to pin-point the sensors that are likely to have captured the information relevant for arriving at the decision. Such a system (along with histogram comparison visualization) can be used to aid the domain experts or remote monitoring engineers to easily check the relevant sensors to understand the fault and/or decide on the accuracy of the classification decision.

7. DISCUSSION

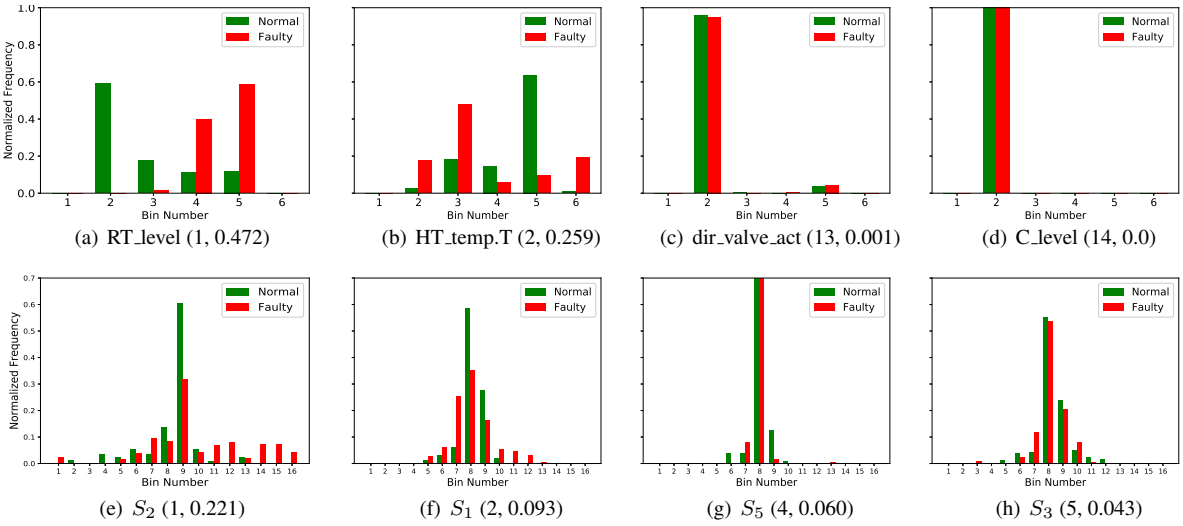
We highlight practical considerations for building fault detection systems and address challenges such as lack of labeled data and lack of interpretability of fault detection results. We propose a novel approach for obtaining relevance scores of input sensors to provide actionable insights into the results of a fault detection model based on RNN Classifier. Our approach for interpreting RNN predictions is generic and may be useful in other applications involving multivariate time series classification. Further, we observe that semi-supervised classification using pre-trained RNN autoencoders can provide signif-



(a) Euclidean Distance (b) Earth Mover's Distance (c) Hellinger Distance

(d) Euclidean Distance (e) Earth Mover's Distance (f) Hellinger Distance

Figure 5. Recall@K for GHL (a-c) and Pump (d-f) datasets. Total number of sensors (p): GHL=14, Pump=5. Number of relevant sensors: GHL=1, Pump=2.



(a) RT_level (1, 0.472) (b) HT_temp.T (2, 0.259) (c) dir_valve_act (13, 0.001) (d) C_level (14, 0.0)

(e) S_2 (1, 0.221) (f) S_1 (2, 0.093) (g) S_5 (4, 0.060) (h) S_3 (5, 0.043)

Figure 6. Sample histograms (GHL (a-d) and Pump (e-h)) comparing distribution of sensor values for most relevant and least relevant sensors for Normal and Faulty predictions. (x, y) in a sub-caption denotes relevance rank (x) and relevance score (y) for the sensor.

icant performance improvements in time series classification tasks, especially when data is sparsely labeled. The proposed approach to interpret RNN-classifier results is restricted to finding the most relevant sensors. In future, it would be interesting and useful to see how to highlight the relevant regions of a multivariate time series to further improve interpretability. Also, it may be interesting to see how a combination of classifier predictions and the insights from sensor relevance scoring can be used to improve the semi-supervised approach (Wei & Keogh, 2006) in an active learning setting. Further, an extension to the proposed approach to address the usually encountered non-stationarity, as addressed in Saurav et al. (2018), can be considered.

REFERENCES

Baehrens, D., Schroeter, T., Harmeling, S., Kawanabe, M., et al. (2010). How to explain individual classification decisions. *Journal of Machine Learning Research*, *11*(Jun), 1803–1831.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Dai, A. M., & Le, Q. V. (2015). Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems* (pp. 3079–3087).

de Bruin, T., Verbert, K., & Babuška, R. (2017). Railway track circuit fault diagnosis using recurrent neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(3), 523–533.

Ergen, T., Mirza, A. H., & Kozat, S. S. (2017). Unsupervised and semi-supervised anomaly detection with lstm neural networks. *arXiv preprint arXiv:1710.09207*.

Filonov, P., Lavrentyev, A., & Vorontsov, A. (2016). Multivariate Industrial Time Series with Cyber-Attack Simulation: Fault Detection Using an LSTM-based Predictive Data Model. *NIPS Time Series Workshop 2016, arXiv preprint arXiv:1612.06676*.

Gugulothu, N., TV, V., Malhotra, P., Vig, L., Agarwal, P., & Shroff, G. (2017). Predicting remaining useful life using time series embeddings based on recurrent neural networks. *International Journal on Prognostics and Health Management, IJPHM. arXiv preprint arXiv:1709.01073*.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, *9*(8), 1735–1780.

Isermann, R. (2005). Model-based fault-detection and diagnosis–status and applications. *Annual Reviews in control*, *29*(1), 71–85.

Jiang, L., Xuan, J., & Shi, T. (2013). Feature extraction based on semi-supervised kernel marginal fisher analysis and its application in bearing fault diagnosis. *Mechanical Systems and Signal Processing*, *41*(1-2), 113–126.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lipton, Z. C. (2016). The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*.

Malhotra, P., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). Lstm-based encoder-decoder for multi-sensor anomaly detection. *arXiv preprint arXiv:1607.00148*.

Malhotra, P., TV, V., Ramakrishnan, A., Anand, G., Vig, L., Agarwal, P., & Shroff, G. (2016). Multi-Sensor Prognostics using an Unsupervised Health Index based on LSTM Encoder-Decoder. *1st ACM SIGKDD Workshop on ML for PHM. arXiv preprint arXiv:1608.06154*.

Malhotra, P., TV, V., Vig, L., Agarwal, P., & Shroff, G. (2017). TimeNet: Pre-trained deep recurrent neural network for time series classification. In *25th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning.*

Malhotra, P., Vig, L., Shroff, G., & Agarwal, P. (2015). Long Short Term Memory Networks for Anomaly Detection in Time Series. In *ESANN, 23rd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning* (pp. 89–94).

Monroy, I., Benitez, R., Escudero, G., & Graells, M. (2010). A semi-supervised approach to fault diagnosis for chemical processes. *Computers & Chemical Engineering*, *34*(5), 631–642.

Ping Zhao, S., & Khorasani, K. (2007). A recurrent neural network based fault diagnosis scheme for a satellite. In *Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE* (pp. 2660–2665).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2016). Why should i trust you?: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 1135–1144).

Ribeiro, M. T., Singh, S., & Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI Conference on Artificial Intelligence (AAAI).*

Saurav, S., Malhotra, P., TV, V., Gugulothu, N., Vig, L., Agarwal, P., & Shroff, G. (2018). Online anomaly detection with concept drift adaptation using recurrent neural networks. In *Proceedings of the ACM India Joint International Conference on Data Science and Management of Data* (pp. 78–87).

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems* (pp. 3104–3112).

Vishnu, T., Gugulothu, N., Malhotra, P., Vig, L., Agarwal, P., & Shroff, G. (2017). Bayesian networks for interpretable health monitoring of complex systems. In *Workshop on AI for Internet of Things at IJCAI.*

Wang, J., Wu, G., Wan, L., Sun, Y., & Jiang, D. (2009). Recurrent neural network applied to fault diagnosis of underwater robots. In *Intelligent Computing and Intelligent Systems, 2009. ICIS 2009. IEEE International Conference on* (Vol. 1, pp. 593–598).

Wei, L., & Keogh, E. (2006). Semi-supervised time series classification. In *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 748–753).

Wulsin, D., Blanco, J., Mani, R., & Litt, B. (2010). Semi-supervised anomaly detection for eeg waveforms using deep belief nets. In *Machine learning and applications (icmla), 2010 ninth international conference on* (pp. 436–441).

Xu, L. D., He, W., & Li, S. (2014). Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, *10*(4), 2233–2243.

Yadav, M., Malhotra, P., Vig, L., Sriram, K., & Shroff, G. (2016). Ode-augmented training improves anomaly detection in sensor data from machines. *arXiv preprint arXiv:1605.01534*.

Zaremba, W., Sutskever, I., & Vinyals, O. (2014). Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*.

Zhao, Y., Ball, R., Mosesian, J., de Palma, J.-F., & Lehman, B. (2015). Graph-based semi-supervised learning for fault detection and classification in solar photovoltaic arrays. *IEEE Transactions on Power Electronics*, *30*(5), 2848–2858.

A. APPENDIX

A.1. Long Short Term Memory Unit

We use a variant of LSTMs (Hochreiter & Schmidhuber, 1997) as described in (Zaremba et al., 2014). Consider $T_{n_1,n_2} : R^{n_1} \rightarrow R^{n_2}$ is an affine transform of the form $\mathbf{z} \mapsto \mathbf{W}\mathbf{z} + \mathbf{b}$ for matrix $\mathbf{W}$ and vector $\mathbf{b}$ of appropriate dimensions. The values for input gate $\mathbf{i}$, forget gate $\mathbf{f}$, output gate $\mathbf{o}$, hidden state $\mathbf{z}$, and cell activation $\mathbf{c}$ at time t are computed using the current input $\mathbf{x}_t$, the previous hidden state $\mathbf{z}_{t-1}$, and memory cell value $\mathbf{c}_{t-1}$ as given by Eqs. 4-6.

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \mathbf{g}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ tanh \end{pmatrix} T_{m+n,4n} \begin{pmatrix} \mathbf{x}_t \\ \mathbf{z}_{t-1} \end{pmatrix} \tag{4}$$

$$\mathbf{c}_t = \mathbf{f}_t \mathbf{c}_{t-1} + \mathbf{i}_t \mathbf{g}_t \tag{5}$$

$$\mathbf{z}_t = \mathbf{o}_t tanh(\mathbf{c}_t) \tag{6}$$

A.2. Distance Metrics Considered

For two discrete probability distributions $P = (p_1, p_2, ..., p_K)$ and $Q = (q_1, q_2, ..., q_K)$, Hellinger distance is defined as:

$$H(P,Q) = \frac{1}{\sqrt{2}} \sum_{i=1}^{K} (\sqrt{p_i} - \sqrt{q_i})^2 \tag{7}$$

Earth Mover's Distance (EMD) is defined as:

$$\begin{aligned} \text{EMD}_0 &= 0 \\ \text{EMD}_i &= p_i + \text{EMD}_{i-1} - q_i \\ \text{EMD(P,Q)} &= \textstyle\sum_{i=1}^{K} |\text{EMD}_i| \end{aligned} \tag{8}$$

A.3. Datasets Details

GHL
GHL dataset (Filonov et al., 2016) contains data for faulty behavior (due to cyber-attacks) in a plant induced by changing the control logic of a gasoil plant heating loop. There are two types of faults in this dataset: i) unauthorized change of max RT level, (Fault IDs 1-24), ii) unauthorized change of max HT temperature (Fault IDs 25-48). If a sensor reading crosses a pre-defined threshold after an attack, then Danger sensor is set to 1. We use this sensor as ground truth for our experiments (1:Faulty, 0:Normal). We down-sample the original time-series by 4 using 4-point average, and then take a window of 100 points to generate time-series instances.

Pump
This is a proprietary real-world dataset with per-minute sensor readings over a period of two years for 28 pumps that have failed due to a particular fault. The data for most pumps is not available for entire life but only from mid-operational life. So that unsupervised models (ED) such as (Malhotra, Ramakrishnan, et al., 2016) are difficult to train as knowledge of normal operating region is not known. We downsample the time series data from the original one reading per minute to one reading per hour using one-hour average and then take a window of 48 hours to generate time-series instances, s.t. window-length $T = 48$. The dataset has five sensors while the signature for the fault is present in two of them.