Online Anomaly Detection with Concept Drift Adaptation using Recurrent Neural Networks

Sakti Saurav*
IIIT Delhi
New Delhi, India
sakti14093@iiitd.ac.in

Pankaj Malhotra
TCS Research
New Delhi, India
malhotra.pankaj@tcs.com

Vishnu TV
TCS Research
New Delhi, India
vishnu.tv@tcs.com

Narendhar Gugulothu
TCS Research
New Delhi, India
narendhar.g@tcs.com

Lovekesh Vig
TCS Research
New Delhi, India
lovekesh.vig@tcs.com

Puneet Agarwal
TCS Research
New Delhi, India
puneet.a@tcs.com

Gautam Shroff
TCS Research
New Delhi, India
gautam.shroff@tcs.com

ABSTRACT

Anomaly detection in time series is an important task with several practical applications. The common approach of training one model in an offline manner using historical data is likely to fail under dynamically changing and non-stationary environments where the definition of normal behavior changes over time making the model irrelevant and ineffective. In this paper, we describe a temporal model based on Recurrent Neural Networks (RNNs) for time series anomaly detection to address challenges posed by sudden or regular changes in normal behavior. The model is trained incrementally as new data becomes available, and is capable of adapting to the changes in the data distribution. RNN is used to make multi-step predictions of the time series, and the prediction errors are used to update the RNN model as well as detect anomalies and change points. Large prediction error is used to indicate anomalous behavior or a change (drift) in normal behavior. Further, the prediction errors are also used to update the RNN model in such a way that short term anomalies or outliers do not lead to a drastic change in the model parameters whereas high prediction errors over a period of time lead to significant updates in the model parameters such that the model rapidly adapts to the new norm. We demonstrate the efficacy of the proposed approach on a diverse set of synthetic, publicly available and proprietary real-world datasets.

KEYWORDS

Online Anomaly Detection, Time Series, Recurrent Neural Networks, Incremental Learning, Concept Drift Adaptation, Change Point Detection

ACM Reference format:
Sakti Saurav, Pankaj Malhotra, Vishnu TV, Narendhar Gugulothu, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2018. Online Anomaly Detection with Concept Drift Adaptation using Recurrent Neural Networks. In *Proceedings of The ACM India Joint International Conference on Data Science & Management of Data, Goa, India, January 11–13, 2018 (CoDS-COMAD '18),* 10 pages.
https://doi.org/10.1145/3152494.3152501

1 INTRODUCTION

In the current Digital Era, streaming data is ubiquitous and growing at a rapid pace. There is renewed focus on Industrial Internet of Things [10], which has motivated all industries to not only install more sensors in the machines, but also to capture the sensor data at scale. One of the key anticipation from IoT technology is to automate the remote monitoring of industrial operations. For example, in the enterprise data centers time-taken to run a daily batch-job is recorded as a time series. Such time series are observed to have non-stationary behavior due to periodic (e.g. weekly, monthly, seasonal) changes in transaction volumes. It is desirable to detect the situations when the time-taken is abnormally high or low. Similar non-stationary patterns are observed in power-plants due to seasonal variations in power demand, different operating modes (full-load vs partial-load), varying quality of inputs (such as coal quality in thermal power plants), etc. Automatically detecting anomalies such as increased temperature or vibration becomes challenging under such dynamic conditions.

*Work done during an internship at TCS Research

CoDS-COMAD '18, January 11–13, 2018, Goa, India

ACM ISBN 978-1-4503-6341-9/18/01…$15.00
https://doi.org/10.1145/3152494.3152501

An important characteristic of above-mentioned real world time series is that the definition of normal behavior changes over time. In other words, the "concept" of normal changes over time. This is sometimes referred to as concept drift in literature (e.g. [32, 36]). The ability of models to adapt to the changing concepts is referred to as Concept Drift Adaptation [12]. As the distribution of data changes, the model built on old data becomes inconsistent with the new data since a model trained on one input distribution is not guaranteed to generalize to the changed new input distribution. For instance, the sensor data collected from machines is often non-stationary due to several reasons such as multiple operating conditions, varying control settings, seasonal variations, and gradual degradation. Also, there are manufacturing differences across different instances of the same machine model. Under these situations, relying on one model trained in an offline manner on historical sensor data may not be appropriate. Under such conditions, an online anomaly detection model for time series is desirable.

The definition of anomaly is not standard and varies depending upon the problem and domain. In general, it can be considered as behavior that is not normal. Traditional approaches for anomaly detection in time series (e.g. [17, 23, 27]) learn a model of normal behavior in an offline manner and use this model to detect anomalies in new unseen time series data. As the definition or the "concept" of normal behavior changes, so does the definition of anomalous. In such cases, the anomaly detection models need to adapt to the changing concept of normal behavior. *In this paper, we focus on the task of anomaly detection under concept drift such that the anomaly detection model is able to detect and accordingly adapt whenever there is a concept drift.*

Recently, Recurrent Neural Networks (RNNs), particularly those based on gated units such as Long Short Term Memory (LSTM) Units [19] and Gated Recurrent Units (GRUs) [9], have been shown to be very effective for sequential data applications yielding state-of-the-art results across domains such as speech [14], text [5, 9], video [34, 46] as well as data from other domains with real-valued time series such as sensor data from machines [15]. Various architectures based on RNNs have been proposed with applications to anomaly detection in general [24, 27, 43] and domain-specific applications such as fault detection [11], machine health monitoring [25, 37, 48], modeling customer behavior from transacations in e-commerce [4], and so on.

Most of the approaches for time series anomaly detection using RNNs (e.g. [11, 24, 27, 47]) rely on offline models learned using historical data assuming that the training and testing data come from the same distribution. However, in practice, most time series data is non-stationary in nature. In this paper, we extend one of the popular offline approaches for anomaly detection using RNNs in [27] and propose *Online RNN-AD* where we use a RNN trained incrementally as new data arrives for time series anomaly detection under concept drift. RNN is trained to predict values for multiple time steps using historical readings. The prediction error is used to detect anomalies and change points as well as appropriately adapt to changes. Through experiments on several simulated, publicly available and proprietary real-world time series datasets, we show that:

- A simple idea of *multi-step ahead prediction* and using the prediction error thus obtained to tune the model can be used to obtain a robust online anomaly and change detection model for time series.
- Online RNN-AD can automatically adapt to various types of changes (concept drifts) such as: i) sudden, ii) incremental, iii) gradual, and iv) continuous, in periodic as well as aperiodic time series.
- Along with multi-step prediction, *local normalization* over a window can be used to effectively deal with non-stationary time series.

The rest of the paper is organized as follows: Section 2 provides a brief background to deep RNNs with gated recurrent units (GRUs). We provide details of Online RNN-AD in Section 3. Section 4 provides the experimental setup, evaluation and observations. Section 5 reviews the existing literature on online anomaly detection and Section 6 provides concluding remarks.

2 BACKGROUND: MULTILAYERED RECURRENT NEURAL NETWORKS

We use multilayered or deep RNNs to model the time series. Multilayered RNNs consist of multiple hidden layers of recurrent units stacked one on top of another. Deep RNNs have been shown to capture the temporal dependencies at different temporal resolutions [18, 27, 38] as well as dependencies across dimensions for multivariate time series data (e.g. dependencies across sensors over time). Multilayered RNNs can be used as generic feature extractors for time series data [26]. RNNs with gated units such as Long Short Term Memory Networks (LSTMs) have been shown to be capable of remembering relevant information from 1000s of time steps in the past by handling the vanishing gradient problem [19]. All these properties make RNNs an attractive choice for modeling time series data. In this paper, we consider Gated Recurrent Units (GRUs) [9] which are a simplified variant of LSTM units.

A GRU consists of two gates, namely *update gate* and *reset gate*, to control the flow of information by manipulating the *hidden state* of the unit. The reset gate is used to compute a proposed value for the hidden state at time t by using the hidden state at time $t-1$ and the hidden state of the units in the lower hidden layer at time t. The update gate decides as to what fractions of previous hidden state and proposed hidden state to use to obtain the updated hidden state at time t.

For a multilayered RNN with L hidden layers, the hidden state $\mathbf{z}_t^{(i)}$ at time t for i^{th} hidden layer is obtained from $\mathbf{z}_{t-1}^{(i)}$ and $\mathbf{z}_t^{(i-1)}$ as in Equation 1. The time series goes through the following transformations iteratively for $t = 1$ through T,

where T is length of the time series:

$$\begin{aligned} \textit{reset gate} &: \mathbf{r}_t^{(i)} = \sigma(\mathbf{W}_r^{(i)} \cdot [\mathbf{D}(\mathbf{z}_t^{(i-1)}), \mathbf{z}_{t-1}^{(i)}]) \\ \textit{update gate} &: \mathbf{u}_t^{(i)} = \sigma(\mathbf{W}_u^{(i)} \cdot [\mathbf{D}(\mathbf{z}_t^{(i-1)}), \mathbf{z}_{t-1}^{(i)}]) \\ \textit{proposed state} &: \tilde{\mathbf{z}}_t^{(i)} = \tanh(\mathbf{W}_p^{(i)} \cdot [\mathbf{D}(\mathbf{z}_t^{(i-1)}), \mathbf{r}_t \odot \mathbf{z}_{t-1}^{(i)}]) \\ \textit{hidden state} &: \mathbf{z}_t^{(i)} = (1 - \mathbf{u}_t^{(i)}) \odot \mathbf{z}_{t-1}^{(i)} + \mathbf{u}_t^{(i)} \odot \tilde{\mathbf{z}}_t^{(i)} \end{aligned} \tag{1}$$

where $\odot$ is Hadamard product, $[\mathbf{a}, \mathbf{b}]$ is concatenation of vectors $\mathbf{a}$ and $\mathbf{b}$, $\mathbf{D}(\cdot)$ is dropout operator that randomly sets the dimensions of its argument to zero with probability equal to dropout rate, $\mathbf{z}_t^0$ equals the value of the input time series at time t. $\mathbf{W}_r$, $\mathbf{W}_u$, and $\mathbf{W}_p$ are weight matrices of appropriate dimensions s.t. $\mathbf{r}_t^{(i)}, \mathbf{u}_t^{(i)}, \tilde{\mathbf{z}}_t^{(i)}$, and $\mathbf{z}_t^{(i)}$ are vectors in $\mathbb{R}^{c^{(i)}}$, where $c^{(i)}$ is the number of units in layer i. The sigmoid (σ) and $tanh$ activation functions are applied element-wise. The hidden state $\mathbf{z}_t^{(i)}$ is used to obtain the output via a linear or non-linear output layer. The parameters $\mathbf{W} = [\mathbf{W}_r, \mathbf{W}_u, \mathbf{W}_p]$ of the RNN consist of the weight matrices in Equations 1. Dropout is used for regularization [28, 33] and is applied only to the non-recurrent connections, ensuring information flow across time-steps.

3 APPROACH

We assume that a model that is able to predict the next few steps in a time series based on history is a suitable model for the time series. In other words, if a model is able to predict the values at next few steps in a time series well, it has effectively learned to capture the important temporal characteristics in the time series. In case of stationary time series, once trained, such a model is expected to predict the time series depicting normal behavior well but perform badly on the prediction task on time series corresponding to anomalous behavior. This idea has been effectively applied to several domains where a temporal model for normal behavior is learned using multilayered LSTM based RNNs (e.g. [6, 7, 27, 43]).

In the online setting, such a model of normal behavior needs to be updated in an incremental manner as new data arrives. We use the most recent prediction error(s) to serve two primary purposes: i) update the model parameters, ii) compute the anomaly score. We identify following key properties of a good online anomaly detection model based on prediction errors:

- If there is a drift in values being observed over a significant period of time, the model should quickly adapt to the new norm by updating the parameters.
- If there is a short subsequence of anomalous values, the model need not be updated. Rather the model updation step should be restrained.
- If there is a continuous drift in the normal behavior, the model should get updated continuously or model the continuous drift as part of the normal behavior.

We next describe our approach that caters to the aforementioned criteria. We show how multi-step predictions are obtained using RNNs and then used for anomaly score computation as well as incremental model updation. Overall steps of the algorithm are depicted in Figure 1.

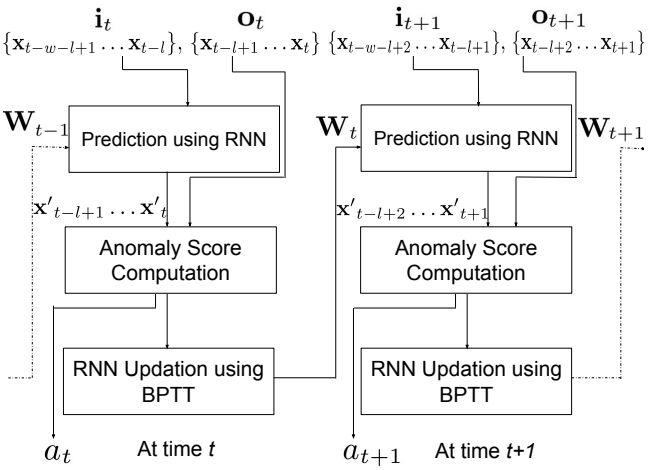


Figure 1: Steps in Online RNN-AD approach

3.1 Online RNN-AD

Consider a multivariate time series $\mathbf{x} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_t\}$, where t is the length of the time series, each point $\mathbf{x}_i \in \mathbf{R}^m$ ($i = 1 \dots t$) in the time series is an m-dimensional vector (e.g. each of the m dimensions corresponds to a different sensor). At time t, we use a past window of length w as raw input time series $\mathbf{i}_t$:

$$\mathbf{i}_t = \mathbf{x}_{t-w-(l-1)} \dots \mathbf{x}_{t-l} \tag{2}$$

The RNN model obtained at time $t-1$ with parameters $\mathbf{W}_{t-1}$ is used to obtain the l-steps ahead estimates $\mathbf{x'}_{t-l+1} \dots \mathbf{x'}_t$ corresponding to last l raw observed values $\mathbf{o}_t$:

$$\mathbf{o}_t = \mathbf{x}_{t-l+1} \dots \mathbf{x}_t \tag{3}$$

The raw input $\mathbf{i}_t$ and raw target output $\mathbf{o}_t$ are appropriately normalized before feeding it to the latest RNN network represented by weight matrix $\mathbf{W}_{t-1}$. The error in predicting $\mathbf{o}_t$ by processing $\mathbf{i}_t$ via the RNN $\mathbf{W}_{t-1}$ is used to obtain the anomaly score a_t and update the RNN to $\mathbf{W}_t$. We next describe these steps in detail:

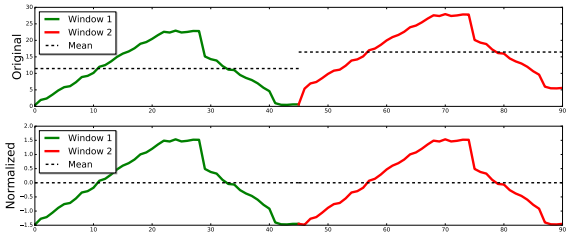


Figure 2: Effect of local normalization in case of sudden mean shift

Local normalization: The local normalization step is important for numerical stability of training the RNN else the time series can take unbounded values. Further, local normalization leads to more emphasis on the temporal patterns in the time series rather than the exact values and has been shown to useful for sequence mining applications [22]. Each point in the raw input $\mathbf{i}_t$ as well as the raw output $\mathbf{o}_t$ is normalized using the mean $\boldsymbol{\mu}_t$ and standard deviation $\boldsymbol{\sigma}_t$ computed over the raw input $\mathbf{i}_t$ (using idea similar to piecewise normalization [13]). Effectively, $x^d_{t'} \rightarrow (x^d_{t'} - \mu^d_t)/\sigma^d_t$, where $x^d_{t'}$ denotes the d-th dimension of $\mathbf{x}_{t'}$ and $t' \in \{t-w-(l-1), \ldots, t\}$. It is to be noted that $\boldsymbol{\mu}_t$ and $\boldsymbol{\sigma}_t$ are computed over values from time $t-w-(l-1)\ldots t-l$. A sample effect of local normalization to handle sudden mean shift is shown in Figure 2.

Multi-step prediction: There is one linear unit for each of the m dimensions in the input layer and $m \times l$ linear units in the output layer such that there is one unit for each of the l future predictions for each of the m dimension. The recurrent units in a hidden layer are fully connected through recurrent connections. We stack recurrent layers such that each unit in a lower hidden layer is fully connected to each unit in the hidden layer above it through feedforward connections. A sample RNN architecture is shown in Figure 3.

The RNN can be considered to be a function with parameters $\mathbf{W}_{t-1}$ that takes normalized input $\mathbf{i}_t$ and provides estimates for normalized output $\mathbf{o}_t$ after going through the set of computations in Equation 1 with a linear layer at the top, so as to provide estimates $\mathbf{o'}_t$ corresponding to observed values $\mathbf{o}_t$:

$$\mathbf{o'}_t = f_{RNN}(\mathbf{i}_t; \mathbf{W}_{t-1}) \tag{4}$$

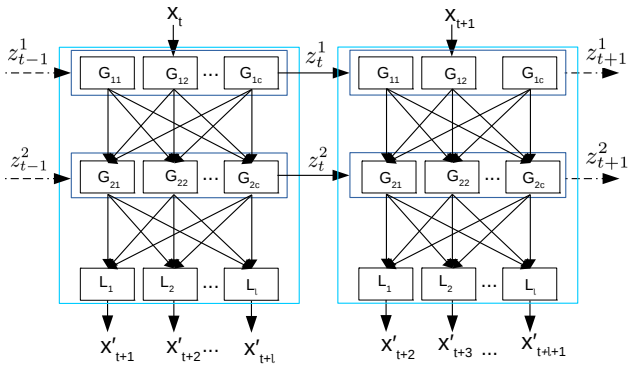


Figure 3: A sample RNN depicting l-step ahead prediction with two hidden layers with c GRU units each and a linear output layer

Anomaly Score Computation:

Given the estimated value $\mathbf{x'}_{t'}$ for $\mathbf{x}_{t'}$ (obtained using Equation 4), the error in estimate at time t' is given by:

$$e_{t'} = ||\mathbf{x}_{t'} - \mathbf{x'}_{t'}||_2, \ for\ t' = t-l+1 \ldots t \tag{5}$$

where $||.||_2$ denotes L_2-norm.

The anomaly score at time t is then given by:

$$a_t = \frac{1}{l} \sum_{t'=t-l+1}^{t} e^2_{t'} \tag{6}$$

RNN Updation: The anomaly score a_t is then used to update the network $\mathbf{W}_{t-1}$ to obtain $\mathbf{W}_t$ using backpropagation through time (BPTT) [41]:

$$\mathbf{W}_t = \mathbf{W}_{t-1} - \eta \nabla a_t(\mathbf{W}_{t-1}) \tag{7}$$

where η is the learning rate.

It is to be noted that the anomaly score controls the rate of update of network parameters. High anomaly score implies significant change in the parameters $\mathbf{W}$ of the RNN and vice-versa. Since the anomaly score is computed as an average over the last l steps, a point outlier or a very short term anomaly will lead to high anomaly score only for a very short period of time. This ensures no significant updates to the network parameters. On the other hand, a high anomaly score over a period of time due to concept drift leads to high anomaly score for a significantly large period of time leading to continuous changes till the network parameters are adapted to predict the new normal behavior well and the anomaly score drops. We discuss this in detail with the help of observations from empirical evaluation in the next section.

4 EXPERIMENTS

We consider several variants of non-stationary real-valued time series where mean or the frequency components of the time series values over a window changes with time. The change may happen suddenly, gradually, incrementally, or continuously. We first describe the datasets considered for experimental evaluation, then describe RNN model selection process followed by results and important observations.

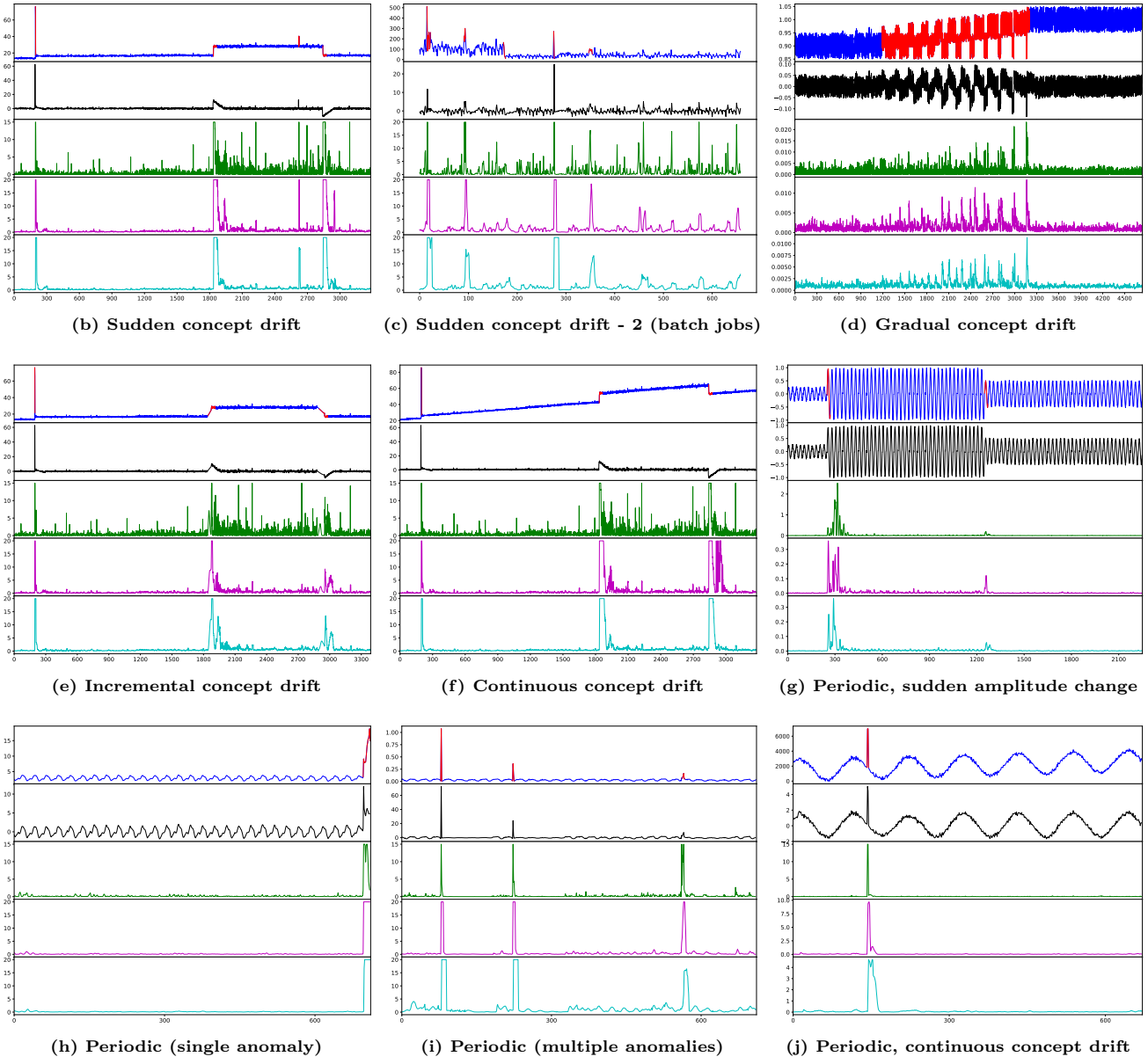
4.1 Datasets Description

We consider several synthetic and real-world time series - many of these time series are taken or derived from Numenta Anomaly Benchmark (NAB) [21] and Yahoo Labs Benchmark Dataset (Yahoo) [20]. The source for the cases considered from NAB and Yahoo datasets are provided in Table 1. We also consider a proprietary real-world time series of daily batch jobs time-taken on a server. The anomalies correspond to jobs that take more time than expected.

We consider anomalies and change points in amplitude domain (i.e., change in the range of values taken) as well as in frequency domains (i.e. change in periodicity for periodic time series). More specifically, the datasets are considered to cover at least one of the following scenarios (as identified in [12]):

- point anomalies or outliers (Figures 4b, 4e, 4f, 4h, 4i and 4j): we consider point anomalies in several aperiodic and periodic non-stationary time series.
- sudden concept drift due to mean change (Figures 4b, 4c, 4f).

(a) Legend for subfigures (b)-(j)

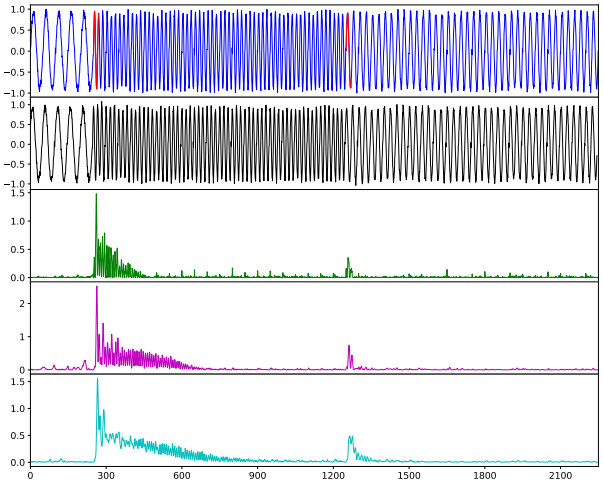


(b) Sudden concept drift (c) Sudden concept drift - 2 (batch jobs) (d) Gradual concept drift

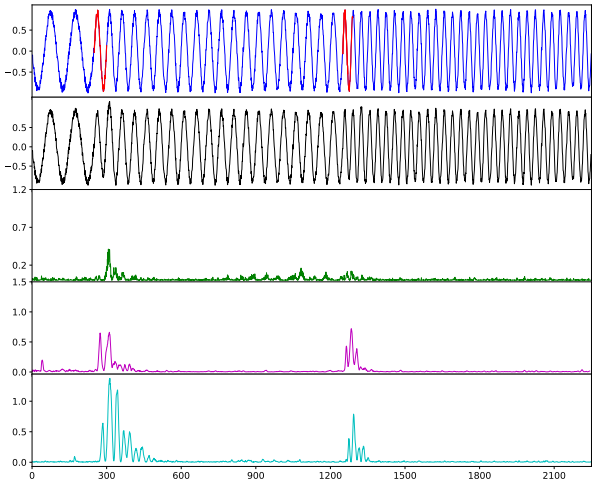
(e) Incremental concept drift (f) Continuous concept drift (g) Periodic, sudden amplitude change

(h) Periodic (single anomaly) (i) Periodic (multiple anomalies) (j) Periodic, continuous concept drift

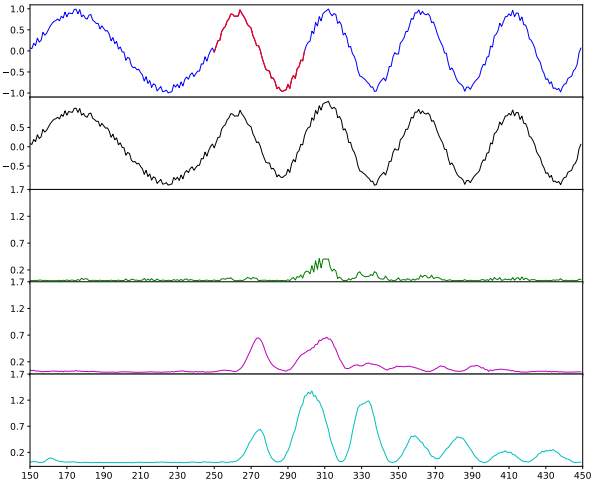
Figure 4: Results for Anomaly Detection under Concept Drift using Online RNN-AD. All anomaly scores are clipped at 20. (Best seen in color and on zooming.)



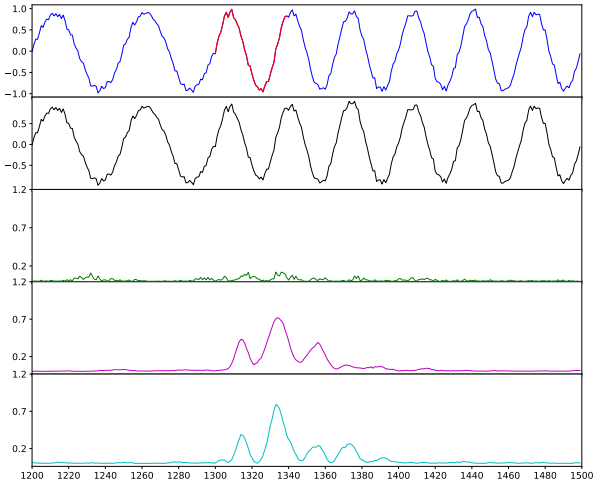
(a) Periodic, sudden frequency change - 1 (frequency increases, then frequency decreases)



(b) Periodic, sudden frequency change - 2 (frequency increases twice)



(c) Periodic, sudden frequency change - 2 (zoomed-in view for first frequency change in Figure 5b)



(d) Periodic, sudden frequency change - 2 (zoomed-in view for second frequency change in Figure 5b)

Figure 5: Frequency Domain Concept Drift Adaptation for Periodic Time Series. Refer Figure 4a for legends.

- gradual concept drift in range of values taken (Figure 4d): the fraction of points from one concept (one range of values) decreases over time while the fraction of points from second concept (another range of values) increases over time.
- incremental change in amplitude domain (Figure 4e): consisting of several intermediate ranges of values while changing from one range of values to another over a significant period of time. It is shown in concept drifts around time steps 1800 and 3000 in Figure 4e.
- sudden concept drift due to amplitude change in periodic series(Figure 4g).
- continuous concept drift (Figures 4f, 4j): continuous change in the normal range of values.
- sudden concept drift due to change in frequency for periodic time series (Figures 5a, 5b).

4.2 Experimental Setup

For each dataset considered, we train the initial RNN model in an incremental fashion (continuously updating the network parameters at each time step) over first 750 time steps. The starting 500 points are considered as training set, and next 250 points are considered as validation set (except for batch jobs dataset where we only use first 65 points for training and next 10 points for validation, since the total time series length is only 735). The network with minimum average prediction error (average of the anomaly score, refer Equation 6) over the validation set is chosen as best model using grid search

Data Type	$l = 1$			$l = 5$			$l = 10$		
	L	c	AUC	L	c	AUC	L	c	AUC
Sudden concept drift [1]	2	40	0.939	3	20	**0.977**	1	40	0.967
Sudden concept drift (batch jobs)	3	40	0.747	2	40	0.852	1	40	**0.864**
Gradual concept drift	3	40	0.536	2	20	0.651	2	20	**0.690**
Incremental concept drift [1]	1	40	0.738	1	40	0.893	1	40	**0.944**
Continuous concept drift [1]	1	20	0.555	3	20	**0.708**	1	20	0.612
Periodic, sudden amplitude change	1	40	0.872	3	40	0.908	1	20	**0.931**
Periodic, sudden frequency change - 1	1	40	**0.871**	1	40	0.830	1	20	0.801
Periodic, sudden frequency change - 2	1	40	0.696	2	20	0.839	1	40	**0.899**
Periodic (single anomaly) [2]	2	40	**0.985**	3	20	0.959	1	40	0.967
Periodic (multiple anomalies) [3]	1	40	**0.879**	1	20	0.779	3	40	0.792
Periodic, continuous concept drift [4]	1	20	0.630	2	20	0.716	1	40	**0.869**

Table 1: RNN architecture details for various cases shown in subplots (b)-(g). Here L = number of hidden layers, c = number of GRU units per layer. Data taken/derived from *1) NAB realAWSCloudwatch/rds_cpu_utilization_e47b3b.csv, 2) Yahoo A1Benchmark/real_3.csv, 3) Yahoo A1Benchmark/real_9.csv, 4) Yahoo A2Benchmark/synthetic_13.csv*

over number of hidden layers $L = \{1, 2, 3\}$ and number of GRUs in each hidden layer $c = \{20, 40\}$. We used a dropout factor of 0.2 for regularization and learning rate $\eta = 0.0005$ for all cases. Further, we experimented with prediction length $l = 1, 5, 10$ such that there are six different models trained for each value of l. We choose fixed window length $w = 100$ for all experiments (except for batch jobs dataset where $w = 25$). The best architectures obtained for each dataset and the performance in terms of area under the receiver operating characteristic curve (AUC) are reported in Table 1 (assuming anomalous points and change points belong to the positive class). The best model chosen is used for testing on the remaining time series in an online fashion while being incrementally updated as described in Section 3. The results for each dataset considered are presented in Figures 4 and 5.

4.3 Observations

Following key observations can be made from the experiments:

(1) We observe that *local normalization* effectively converts a non-stationary time series to a stationary time series as can be seen in Figures 4f and 4j. This implies that the proposed approach is able to deal with continuously increasing (or decreasing) cases such as in Figure 4f and 4j. For cases with only mean shifts, it leads to similar pattern or range of values before and after the change point (e.g. Figure 4b, 4c).

(2) The network is effectively able to utilize continuous large anomaly score to *rapidly adapt to concept drifts without any external intervention* (e.g. Figure 4e between points 1800-2100).

(3) ***Pros and cons of multi-step prediction***:

(a) The *multi-step prediction* plays an important role in detecting anomalies and adapting to concept drifts. We observe that models for $l > 1$ perform better compared to models with $l = 1$ for most data types in terms of AUC as shown in Table 1. We observe that networks trained to predict for $l > 1$ are able to detect change points much earlier as compared to networks trained to predict only the next point with $l = 1$. This is particularly *useful for detecting changes that are of temporal nature* (e.g. frequency change) and cannot be detected by point-wise models: the network trained to predict only one point, i.e. with $l = 1$ is not able to detect frequency changes when they happen. It starts adapting much later when significant changes in input distribution are visible, as shown in third subplot in Figure 5c. The change in frequency starts around 250 in Figure 5c but is detected only after 290 when the first cycle of the changed frequency is almost completely observed. On the other hand, models with $l = 5$ and $l = 10$ are able to detect changes in frequencies much earlier as shown in fourth and fifth subplots in Figures 5c where the anomaly score goes high around 270. The models trained with $l = 1$ rely on the changes in input distribution to produce high anomaly scores whereas models trained on $l > 1$ ($l = 5$ and $l = 10$ in this case) detect the temporal changes in the target output distribution and are therefore able to detect concept drifts early.

(b) We also observe that *networks trained for $l = 1$ are very sensitive to noise and small changes in inputs*. For cases in Figure 4b, 4e and 4f, the anomaly score for models with $l = 1$ goes high several times ($a_t \geq 5$) due to noice-induced small peaks/dips in the input. Such an effect is absent in models for $l = 5$ and $l = 10$. This can be explained by an implicit smoothening/averaging effect of using prediction errors over last l steps (refer Equation 5), i.e. $e_{t-l+1} \dots e_t$ to compute the anomaly score a_t (refer Equation 6).

(c) *The model with $l = 1$ is not able to detect small changes in frequency*, e.g. change in cycle length from

50 to 33 between 1300-1350 in Figure 5d. However, the models for $l = 5$ and $l = 10$ are able to capture this small change in frequency as indicated by high anomaly score.

(d) Although $l > 1$ ensures early detection of concept drifts, it also leads to extended periods where the effect of a point anomaly is reflected in the anomaly score. This is clearly visible in Figures 4i and 4j, where point anomalies (sharp spikes) lead to continuous high anomaly scores at future time steps after the point anomaly is over for $l = 5$ and $l = 10$ cases. This suggests that *there is a trade-off between how early the change detection happens and how long do the false positives in case of point anomalies last depending on prediction length l.*

5 RELATED WORK

Anomaly/outlier detection in temporal data has been well studied in literature [17]. Our work falls under the category of "evolving prediction models" for online anomaly detection. The key idea of such approaches is that the parameters of the model are updated in an incremental manner as new data arrives in order to capture the changing normal trends in the data. Examples of models that fall under this category are Hierarchical Temporal Memory (HTM) based models [1], self-learning online dynamic clustering technique [40], and Kernel Density Estimates [2, 3], piecewise linear models of time series [45]. A probabilistic auto-regressive (AR) model to determine both outliers and change points in non-stationary contexts has been proposed in [44]. The model is learned incrementally so that it forgets the effect of past data gradually and learns the new normal.

Statistical techniques based on multinomial goodness-of-fit test and Tukey method have been proposed in [39]. A modified k-means clustering and scoring method to detect anomalies in an online manner for periodic time series has been described in [30]. The problem of rapid anomaly detection in computer network traffic was addressed in [35]. The method is based on the multi-cyclic (repeated) Shiryaev-Roberts [29] detection procedure. Relative entropy and Pearson correlation have also been used to dynamically detect anomalies in [31].

Many of these techniques do not take into account the temporal aspect of time series and rather use statistical properties over a time window to detect anomalies. Our proposed approach leverages properties of deep RNNs – an explicitly temporal model – which is capable of dealing with temporal dependencies as well as multi-dimensional dependencies.

Recently, a large amount of research has gone into using RNNs for anomaly detection in time series. Stacked LSTM networks for anomaly detection in time series have been recently proposed in [27]. RNNs with LSTM units are used to model the normal time series behavior in an offline manner on historical normal data by training for multi-step ahead prediction. The prediction errors are then modeled as a multivariate Gaussian distribution which is then used to estimate the likelihood of anomalous behaviour at a time instant. [7] proposes similar approach for anomaly detection in ECG signals. Similarly, LSTM-based Encoder-Decoder scheme for anomaly detection in multi-variate time-series has been proposed in [24]. The encoder learns a feature vector of the input time-series and the decoder uses this feature vector to reconstruct the time-series. The reconstruction error at any future time instance is used to compute the likelihood of anomaly at that point. A multi scale LSTM model for detecting anomalous internet traffic procedure was presented in [8]. Here the traffic is considered as a multi-dimensional time series and traffic patterns are learned from historical features using sliding time windows. While these approaches use RNNs for anomaly detection, they rely on a model trained in an offline manner and do not address the challenges of concept drift. They assume fixed normal behavior(s) and thus may not be suitable for non-stationary time series. Hence these methods would tend to detect a drift in normal behavior as anomalous and fail to adapt as they are not learning in an online fashion.

While applying a pre-trained RNN in an online manner is fairly straight-forward, training RNNs in an incremental manner for streaming data is a lesser explored area. A recent work in [16] explores incremental training of RNNs by using an adaptive gradient learning. The focus of this work is robust time series forecasting in the presence of noise including anomalies and change points. It explores local statistical features of time series to automatically weight the gradients of the loss of newly available observations with distributional properties of the data in real time. It uses adaptive Real Time Recurrent Learning (RTRL) [42] for online learning in RNN. Unlike our approach, the focus of this approach is primarily forecasting of time series and not anomaly detection.

Hierarchical Temporal Memory (HTM) for anomaly detection proposes a machine learning algorithm derived from neuroscience that models spatial and temporal patterns in streaming data [1]. HTMs aim to capture the structural and algorithmic properties of the neocortex and are continuously learning systems that automatically adapt to changing statistics. HTMs, however, do not use multi-step predictions for computing the anomaly score which may be important for early detection and hence early adaptation to certain types of changes (e.g. in frequency domain) for prediction based models like HTMs (as empirically observed in Section 4.3). To the best of our knowledge, our work is the first to consider deep RNNs trained in an incremental manner for multi-step prediction with application to online anomaly detection and concept drift adaptation.

6 DISCUSSION

We have proposed an approach using deep recurrent neural networks for anomaly detection in time series under non-stationary scenarios. The model is continuously updated as new data arrives. The proposed approach uses local normalization to adapt to concept drifts in the amplitude domain and multi-step prediction to adapt to concept drifts in both

amplitude and frequency domains. We show that the approach is able to adapt to different types of concept drifts including sudden, gradual, and incremental changes in mean value as well as adapt to changes in frequency for periodic time series. Through empirical evidence, we show that models (such as those based on RNNs) that use prediction errors of multiple recently observed values are able to deal with noise, and detect certain type of changes early compared to models that only use single-step predictions.

REFERENCES

[1] Subutai Ahmad and Scott Purdy. 2016. Real-Time Anomaly Detection for Streaming Analytics. *arXiv preprint arXiv:1607.02480* (2016).

[2] Tarem Ahmed. 2009. Online anomaly detection using KDE. In *Global Telecommunications Conference, 2009. GLOBECOM 2009. IEEE*. IEEE, 1–8.

[3] Tarem Ahmed, Mark Coates, and Anukool Lakhina. 2007. Multivariate online anomaly detection using kernel recursive least squares. In *INFOCOM 2007. 26th IEEE International Conference on Computer Communications. IEEE*. IEEE, 625–633.

[4] Gaurangi Anand, Auon H. Kazmi, Pankaj Malhotra, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2015. Deep Temporal Features to Predict Repeat Buyers. In *NIPS 2015 Workshop: Machine Learning for eCommerce*.

[5] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[6] Loïc Bontemps, James McDermott, Nhien-An Le-Khac, et al. 2016. Collective Anomaly Detection Based on Long Short-Term Memory Recurrent Neural Networks. In *International Conference on Future Data and Security Engineering*. Springer, 141–152.

[7] Sucheta Chauhan and Lovekesh Vig. 2015. Anomaly detection in ECG time signals via deep long short-term memory networks. In *Data Science and Advanced Analytics (DSAA), 2015. 36678 2015. IEEE International Conference on*. IEEE, 1–7.

[8] Min Cheng, Qian Xu, Jianming Lv, Wenyin Liu, Qing Li, and Jianping Wang. 2016. MS-LSTM: A multi-scale LSTM model for BGP anomaly detection. In *Network Protocols (ICNP), 2016 IEEE 24th International Conference on*. IEEE, 1–6.

[9] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[10] Li Da Xu, Wu He, and Shancang Li. 2014. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics* 10, 4 (2014), 2233–2243.

[11] Pavel Filonov, Andrey Lavrentyev, and Artem Vorontsov. 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* (2016).

[12] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. 2014. A survey on concept drift adaptation. *ACM Computing Surveys (CSUR)* 46, 4 (2014), 44.

[13] Martin Gavrilov, Dragomir Anguelov, Piotr Indyk, and Rajeev Motwani. 2000. Mining the stock market (extended abstract): which measure is best?. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 487–496.

[14] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*. IEEE, 6645–6649.

[15] Narendhar Gugulothu, Vishnu TV, Pankaj Malhotra, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2017. Predicting Remaining Useful Life using Time Series Embeddings based on Recurrent Neural Networks. *2nd ACM SIGKDD Workshop on ML for PHM. arXiv preprint arXiv:1709.01073* (2017).

[16] Tian Guo, Zhao Xu, Xin Yao, Haifeng Chen, Karl Aberer, and Koichi Funaya. 2016. Robust Online Time Series Prediction with Recurrent Neural Networks. In *Data Science and Advanced Analytics (DSAA), 2016 IEEE International Conference on*. Ieee, 816–825.

[17] Manish Gupta, Jing Gao, Charu C Aggarwal, and Jiawei Han. 2014. Outlier detection for temporal data: A survey. *IEEE Transactions on Knowledge and Data Engineering* 26, 9 (2014), 2250–2267.

[18] Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Advances in Neural Information Processing Systems*. 190–198.

[19] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[20] Nikolay Laptev, Saeed Amizadeh, and Y. Billawala. 2015. Yahoo Labs News: Announcing A Benchmark Dataset For Time Series Anomaly Detection [Online blog]. Available: http://labs.yahoo.com/news/announcing-a-benchmark-datasetfor-time-series-anomaly-detection.

[21] Alexander Lavin and Subutai Ahmad. 2015. Evaluating Real-Time Anomaly Detection Algorithms–The Numenta Anomaly Benchmark. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*. IEEE, 38–44.

[22] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. 2007. Experiencing SAX: a novel symbolic representation of time series. *Data Mining and knowledge discovery* 15, 2 (2007), 107–144.

[23] Junshui Ma and Simon Perkins. 2003. Time-series novelty detection using one-class support vector machines. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, Vol. 3. IEEE, 1741–1745.

[24] Pankaj Malhotra, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2016. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. *In Anomaly Detection Workshop at 33rd International Conference on Machine Learning. arXiv preprint arxiv:1607.00148* (2016).

[25] Pankaj Malhotra, Vishnu TV, Anusha Ramakrishnan, Gaurangi Anand, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 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* (2016).

[26] Pankaj Malhotra, Vishnu TV, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 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*. 607–612.

[27] Pankaj Malhotra, Lovekesh Vig, Gautam Shroff, and Puneet Agarwal. 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*. 89–94.

[28] Vu Pham, Théodore Bluche, Christopher Kermorvant, and Jérôme Louradour. 2014. Dropout improves recurrent neural networks for handwriting recognition. In *Frontiers in Handwriting Recognition (ICFHR), 2014 14th International Conference on*. IEEE, 285–290.

[29] Aleksey S Polunchenko and Alexander G Tartakovsky. 2012. State-of-the-art in sequential change-point detection. *Methodology and computing in applied probability* 14, 3 (2012), 649–684.

[30] Umaa Rebbapragada, Pavlos Protopapas, Carla E Brodley, and Charles Alcock. 2009. Finding anomalous periodic time series. *Machine learning* 74, 3 (2009), 281–313.

[31] Laura Rettig, Mourad Khayati, Philippe Cudré-Mauroux, and Michal Piórkowski. 2015. Online anomaly detection over big data streams. In *Big Data (Big Data), 2015 IEEE International Conference on*. IEEE, 1113–1122.

[32] Jeffrey C Schlimmer and Richard H Granger. 1986. Incremental learning from noisy data. *Machine learning* 1, 3 (1986), 317–354.

[33] Nitish Srivastava, Geoffrey E Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research* 15, 1 (2014), 1929–1958.

[34] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. 2015. Unsupervised learning of video representations using lstms. In *International Conference on Machine Learning*. 843–852.

[35] Alexander G Tartakovsky, Aleksey S Polunchenko, and Grigory Sokolov. 2013. Efficient computer network anomaly detection by changepoint detection methods. *IEEE Journal of Selected Topics in Signal Processing* 7, 1 (2013), 4–11.

[36] Alexey Tsymbal. 2004. The problem of concept drift: definitions and related work. *Computer Science Department, Trinity College Dublin* 106, 2 (2004).

[37] Vishnu TV, Narendhar Gugulothu, Pankaj Malhotra, Lovekesh Vig, Puneet Agarwal, and Gautam Shroff. 2017. Bayesian Networks for Interpretable Health Monitoring of Complex Systems. *Workshop on AI for Internet of Things at IJCAI* (2017).

[38] Jos van der Westhuizen and Joan Lasenby. 2017. Visualizing LSTM decisions. *arXiv preprint arXiv:1705.08153* (2017).

[39] Chengwei Wang, Krishnamurthy Viswanathan, Lakshminarayan Choudur, Vanish Talwar, Wade Satterfield, and Karsten Schwan. 2011. Statistical techniques for online anomaly detection in data centers. In *Integrated Network Management (IM), 2011 IFIP/IEEE International Symposium on*. IEEE, 385–392.

[40] Xing Wang, Jessica Lin, Nital Patel, and Martin Braun. 2016. A Self-Learning and Online Algorithm for Time Series Anomaly Detection, with Application in CPU Manufacturing. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 1823–1832.

[41] Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proc. IEEE* 78, 10 (1990), 1550–1560.

[42] Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation* 1, 2 (1989), 270–280.

[43] Mohit Yadav, Pankaj Malhotra, Lovekesh Vig, K Sriram, and Gautam Shroff. 2015. ODE-augmented Training Improves Anomaly Detection in Sensor Data from Machines. In *NIPS 2015 Time Series Workshop. arXiv:1605.01534*.

[44] Kenji Yamanishi and Jun-ichi Takeuchi. 2002. A unifying framework for detecting outliers and change points from non-stationary time series data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 676–681.

[45] Yuan Yao, Abhishek Sharma, Leana Golubchik, and Ramesh Govindan. 2010. Online anomaly detection for sensor systems: A simple and efficient approach. *Performance Evaluation* 67, 11 (2010), 1059–1075.

[46] Joe Yue-Hei Ng, Matthew Hausknecht, Sudheendra Vijayanarasimhan, Oriol Vinyals, Rajat Monga, and George Toderici. 2015. Beyond short snippets: Deep networks for video classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 4694–4702.

[47] Shuangfei Zhai, Yu Cheng, Weining Lu, and Zhongfei Zhang. 2016. Deep structured energy based models for anomaly detection. In *International Conference on Machine Learning*. 1100–1109.

[48] Rui Zhao, Ruqiang Yan, Jinjiang Wang, and Kezhi Mao. 2017. Learning to monitor machine health with convolutional bi-directional lstm networks. *Sensors* 17, 2 (2017), 273.