SN Computer Science (2021) 2:475
https://doi.org/10.1007/s42979-021-00866-4

ORIGINAL RESEARCH

Deep Quantile Regression for Unsupervised Anomaly Detection in Time-Series

Ahmad Idris Tambuwal[1] · **Daniel Neagu**[1]

Received: 16 January 2021 / Accepted: 8 September 2021 / Published online: 30 September 2021

Abstract

Time-series anomaly detection receives increasing research interest given the growing number of data-rich application domains. Recent additions to anomaly detection methods in research literature include deep neural networks (DNNs: e.g., RNN, CNN, and Autoencoder). The nature and performance of these algorithms in sequence analysis enable them to learn hierarchical discriminative features and time-series temporal nature. However, their performance is affected by usually assuming a Gaussian distribution on the prediction error, which is either ranked, or threshold to label data instances as anomalous or not. An exact parametric distribution is often not directly relevant in many applications though. This will potentially produce faulty decisions from false anomaly predictions due to high variations in data interpretation. The expectations are to produce outputs characterized by a level of confidence. Thus, implementations need the Prediction Interval (PI) that quantify the level of uncertainty associated with the DNN point forecasts, which helps in making better-informed decision and mitigates against false anomaly alerts. An effort has been made in reducing false anomaly alerts through the use of quantile regression for identification of anomalies, but it is limited to the use of quantile interval to identify uncertainties in the data. In this paper, an improve time-series anomaly detection method called deep quantile regression anomaly detection (DQR-AD) is proposed. The proposed method go further to used quantile interval (QI) as anomaly score and compare it with threshold to identify anomalous points in time-series data. The tests run of the proposed method on publicly available anomaly benchmark datasets demonstrate its effective performance over other methods that assumed Gaussian distribution on the prediction or reconstruction cost for detection of anomalies. This shows that our method is potentially less sensitive to data distribution than existing approaches.

Keywords Time-series · Anomaly detection · Prediction interval · Deep neural networks · Long short-term memory · Quantile regression

Introduction

Fast advances in Industry 4.0 technologies generate enormous amount of data from large number of sensors [1] and other devices within an increasing number of industrial applications. The generated data are time-ordered measurements processable as time-series data. Industries often collect and exploit such data for a number of critical applications including anomaly detection. Anomaly is defined as unexpected point in time-series (e.g., a sudden sensor drift), or an anomalous subsequence within the time-series (e.g., a continuous change in the sensor readings), or points that are anomalous based on defined context, or an anomalous time-series within the entire time-series database [2]. Anomaly detection methods are needed for early fault detection, with potential contributions to avoid total system failure. This includes providing an early evidence for detection of mechanical faults [3] and sensor faults [4] in automotive vehicles during usage.

Anomaly detection techniques are generally categorized based on their output [2] into labeling and scoring techniques. The labeling techniques, also called classification-based methods, directly assign class labels (e.g., normal or abnormal) to the test instances using a trained model [5–7]. For supervised-learning purposes, such techniques rely on

✉ Ahmad Idris Tambuwal
A.IdrisTambuwal@bradford.ac.uk

Daniel Neagu
D.Neagu@bradford.ac.uk

[1] Faculty of Engineering and Informatics, University of Bradford, Bradford, UK

pre-labeled data sets. However, labeling large volumes of data is often difficult, costly, and sometimes incomplete due to big data resource challenges. In contrast, the advantage brought by scoring techniques that assign anomaly scores based on a degree the data instance shows anomaly profile is similar to applications using unsupervised or semi-supervised learning. These techniques include statistical-based [8, 9] and distance-based methods [10–15]. However, due to the temporal nature of time-series, distance-based anomaly detection methods are difficult to be used due to their sensitivity to noise and distance calculation function. In addition, they are not suitable for cases without sufficient relevant data, which are relative to the task at hand.

As such, other prediction-based methods such as Autoregression [16], ARMA [17], ARIMA [18], Kalman Filters [19], and general regression [20, 21] were proposed. Neither of these methods use sequential models that exploit the temporal nature of time-series data which makes their predictions less accurate.

Recent advancement in deep learning methods applications to big data collections opens also opportunities to study their applicability to anomaly detection [22]. These methods used sequential models, and their performance in sequence analysis reported in multimedia applications [23–26] enable them to learn the hierarchical discriminative features and time-series temporal nature [27]. In addition, for each time-series point, an anomaly score is calculated which helps in handling the type of anomaly detection problem. Deep learning-based anomaly detection techniques using long short-term memory (LSTM) [28–30] and other forms of recurrent neural network (RNN) [31, 32], convolutional neural network (CNN) [33], and autoencoder [15, 34–39] demonstrate higher performance over the previously mentioned prediction-base anomaly detection techniques. Despite their performance and unsupervised learning approach, they compute the anomaly score by assuming a distribution usually of Gaussian type on the prediction error, which is either ranked, or threshold to label data instances as anomalous or not. However, an exact parametric distribution is often not directly relevant in some applications, and the assumption of any distribution will lead to false anomaly alerts due to high prediction variance. To solve this problem, Prediction Interval (PI) in regression tasks is considered [40]. PI quantifies the level of uncertainty associated with the point forecasts, thereby offering an interval of confidence for a prediction of lower and upper bounds. Such approach is most welcome in applications requesting better-informed decisions and mitigate against false anomaly alerts, particularly and consistent required in industrial applications.

Within the collection of approaches that used PI for anomaly detection, valuable solutions range from Bayesian approaches [41] to interpret dropout as performing variational inference [40, 42] at the cost of either higher computational demands or strong requirements. In contrast, a probabilistic forecasting model that returns a full conditional distribution was proposed in [43]. The probabilistic forecasts were achieved by assuming also a distribution usually Gaussian on the prediction error. However, as stated earlier, an exact parametric distribution is often not relevant in applications which will lead to false anomaly alerts due to high prediction variance.

To address the above challenges, quantile regression method is used in [44] which is limited to the use of quantile interval to identify only uncertainties in the data. In this paper, an improve unsupervised anomaly detection method called deep quantile regression anomaly detection (DQR-AD) is proposed that go further to used quantile interval as anomaly score and compare it with a threshold value to detect anomalies in time-series data. The proposed method consists of three processing modules: first, a segmentation module that segments the time-series into overlapping windows. Second, a prediction module that uses quantile regression to forecast quantile values. Finally, an anomaly detection module to compute an anomaly score for individual point using quantile interval (QI) as a range between upper and lower quantiles forecasted in the first module. The tests run of this method with publicly available anomaly benchmark datasets shows that it outperforms other baseline techniques that assume Gaussian distribution on prediction error to identify anomalies. In this paper:

1. An improved unsupervised deep learning-based anomaly detection method (DQR-AD) is proposed for detection and classification of anomalies in time-series data. Unlike previous approach [44] which used quantile interval to identify uncertainties in time-series data, DQR-AD uses quantile interval as anomaly score and compares it with threshold value to identify anomalies in the data. In this way, a more comprehensive picture of anomalous points can be achieved and classified from the normal points.
2. No assumption of any data distribution is required. Unlike other baseline deep learning-based anomaly detection techniques that assume Gaussian distribution on prediction error to compute anomaly score, DQR-AD uses QI to quantify the level of uncertainty to compute anomaly score which reduced the rate of false positives. Experimental results in "Experiments" confirm this on a case basis.
3. Unlike the work in [44] that evaluate the performance of their method using only a single dataset and without any comparison with other methods, this paper conducts an extensive evaluation and comparison with six baseline methods using two benchmarks of both real and synthetic datasets.. This demonstrates the ability of our proposed method to detect higher number of anomalies

with low false-positive rates and its applicability on multivariate time-series as confirmed in "Conclusion and Future Work".

The rest of the paper is organized as follows: "Literature Review" reviews related works. In "The Proposed DQR-AD Method", the proposed method and its algorithm depiction are detailed. "Experiments" describes the experiments testing the proposed process. Finally, "Conclusion and Future Work" contains conclusions and future studies generated from this work.

Literature Review

This section provides the review of related literature about time-series anomaly detection methods.

Many time-series anomaly detection techniques have been studies in the recent years [45]. These techniques include statistical-based [8, 9, 46] and distance-based methods [10–15]. Statistical-based methods use the distribution of the data which is not directly relevant in many applications. Distance-based methods compute anomaly scores by calculating the distance between sample points, thereby assuming normal points exist in the same area which is far apart from the abnormal data points. Moreover, due to the temporal nature of time-series, distance-based anomaly detection methods are difficult to be used due to their sensitivity to noise and distance calculation function. In addition, they are not suitable for cases without enough normal data, which are relative to the task at hand.

As a result of above-mentioned problems, several prediction-based methods were proposed which includes classical regression models such as Autoregression [16], ARMA [17], ARIMA [18], Kalman Filters [19], and general regression [20, 21]. Neither of these methods use sequential models that exploit the temporal nature of time-series data which makes their predictions less accurate. Recent advancement in deep learning methods applications to big data collections open also opportunities to study their applicability to anomaly detection [22] where prediction-based anomaly detection methods uses deep learning algorithms for time-series anomaly detection. For example recently, recurrent neural networks (RNNs) particularly based on long short-term memory (LSTM) [47] or gated recurrent unit (GRU) [23] have been reported consistently for anomaly detection in published research.

Various RNN structures have been reported to be used for time-series anomaly detection [22]: Malhotra et al. [28] proposed a deep learning-based anomaly detection method (LSTM-AD) that used stacked LSTM as prediction model to learn time-series normal pattern and predict the future points. The model is evaluated using prediction error and likelihood of the point to be an anomaly is measured through an assumption of Gaussian distribution on the error. A similar approach was used by Chauhan and Vig [48] for detection of anomalies in ECG data. The method used RNN augmented with LSTM to detect 4 different types of anomalous behavior. In a similar context, Singh [49] explored the use of LSTM for anomaly detection in temporal data with a similar approach as in [28] by training the predictive model with normal time-series pattern by modeling the prediction error as Gaussian distribution to estimate the likelihood of an observed future value to be anomaly. In addition, the paper also investigates an alternative way of maintaining LSTM states and how that can affect its prediction and detection performance.

With a slightly different approach, Bontemps et al. [50] combined LSTM prediction model with circular array for detection of collective anomalies. In contrast to the previous point anomaly detection techniques that consider each time step separately, they trained the model to predict multiple time steps, thereby storing the prediction errors from those steps in a circular array with minimum attack time. This detection method faces the problem of identifying collective anomaly by defining a threshold using available labels in the validation data. Instead of using a circular array, Saurav et al. [31] developed an alternative procedure that make used of sliding window to handle time-series temporal and streaming nature. This technique used an incremental model that is trained whenever a new datum arrives. This enables the model to adapt changes (sudden, incremental, gradual, and continues concept drifts) in the time-series data. In addition, the authors used GRU units in the RNN architecture, that are simplified version of LSTM units. Although, the method used sliding window for multi-step ahead prediction and related prediction error for updating model parameters which enable online anomaly detection, the anomaly score of the window is computed as an average of the square of individual points estimated error. This will lead to false-positive result and increase in misidentification of anomalous points in real time where data instance arrives one after the other. To solve this problem, Shipmon et al. [51] proposed another method which combined LSTM-based prediction model to predict expected time-series value and Gaussian tail probability rule defined in [52] to estimate the likelihood of the predicted value to be an anomaly based on prediction history. In their paper, the distribution of an error is model as indirect probabilistic metric and was used to measure the likelihood that the current state is anomalous. Although, the distribution of prediction error is technically not Gaussian in some applications, they model it using normal distribution which may still result in false-positive result in a very noisy, unpredictable streams.

In an online fashion, hierarchical temporal memory (HTM) is used for anomaly detection in data streams [52].

This method models the temporal nature of the data stream at a given time and make predictions for the next time step. At each step, the actual instance is compared with the predicted instance to compute anomaly score which is thresholded to determine whether the point is anomalous or not. The likelihood of a point to be anomaly is determined by assumption of Gaussian distribution on prediction error. In another context, convolutional neural networks (CNN) are used as a prediction model as an unsupervised anomaly detection method (DeepAnT) for time-series data [33]. The DeepAnT method consists of two modules: (1) the prediction module for predicting the next time step using a sliding window; (2) the anomaly detection module responsible for data labeled as anomaly based on the prediction passed from the predictor module. This method also used the same procedure as the previous methods where anomaly score is computed by assumption of a normal distribution on prediction error.

However, the above-mentioned methods face two main challenges with real-life complex systems. First, not always the relevant variables are captured, and therefore, relevant existing or incomplete time-series could affect model performance. Second, an exact parametric distribution is often not directly relevant in some applications and assumption of any distribution will lead to false anomaly alerts due to high prediction variance.

To solve unpredictable nature of the time-series, an autoencoder is used for learning hidden representation and extraction of relevant features from the time-series data. autoencoder involves two parts, encoder and decoder. The encoder learns how to encode or compress the input data. The decoder reconstructs the encoded data back to input space. Efficient data encoding proposals that do not require additional supervised labeling become popular in current solutions for anomaly and novelty detection problems. Malhotra et al. [34] proposed an LSTM-based encoder–decoder for anomaly detection in multivariate time-series data. The encoder–decoder model learn the normal representation of the time-series by reconstructing it in an unsupervised manner. The authors assumed a model trained on time-series for normal behavior performs well for reconstruction normal time-series and poorly on abnormal time-series. This will lead to higher reconstruction error that can be used for computing an anomaly score to identify anomalies in the time-series data. Like the previous methods, a normal distribution is assumed on reconstruction error, which enhance the computation of anomaly score. In a similar context, deep autoencoders are proposed by Schreyer et al. [35] for fraud detection in large-scale accounting data. A different approach is proposed in [15] that combined autoencoder and clustering algorithm for novelty detection. This is achieved by computing an error threshold from the autoencoder model and pass it to the density-based clustering method. The compressed data from the autoencoder are clustered to identify novelty groups as low-density points. Similarly, Zong et al. [37] combine autoencoder with Gaussian mixture model (GMM) for unsupervised anomaly detection. The autoencoder is used for dimensionality reduction where the low-dimensional representation together with reconstruction error is pass to GMM for density estimation and identification of anomalies. Even though these methods demonstrate higher performance due to their unsupervised learning approach, they face similar challenge of assumption of a Gaussian distribution on the reconstruction error which is either ranked or threshold for anomaly detection. However, as mentioned earlier, an exact parametric distribution is often not directly relevant in some applications and assumption of any distribution will lead to false anomaly alerts due to high reconstruction variance.

In an attempt to mitigate the false-positive problem, Hundman et al. [53] introduced pruning procedure and learning from history of labeled anomalies in data streams. The pruning procedure, limited by maximum error/s, helps in ensuring the anomalous sequences which are not affected or depended by regular noise. Such procedures could improve precision of the detection method but are also affected by the use of maximum errors in their evaluation, thereby removing or reclassifying minimum error as nominal could lead to missed detections. To solve this problem, uncertainty is considered in the deep learning-based predictions for anomaly detection.

Various approaches have been developed to address uncertainty in deep neural networks for anomaly detection. They range from Bayesian approach [41] to interpreting dropout as performing variational inference [40, 42]. In these methods, prediction interval (PI) which quantifies the level of uncertainty associated with the point forecasts is considered. David et al. [54] proposed anomaly detection method that used auto regression model and its prediction interval to identify anomalies in environmental sensor streams. The authors calculate prediction interval that accounts for uncertainty in both input data and model parameters. They classified a data point as anomalous when it falls outside a given prediction interval. In contrast, Lingxue and Nikolay [40] argued that the measurement of prediction uncertainty does not depend on only input data and model parameters but rather combine with model misspecification and inherent noise. They achieved this by introducing an updated approach that used autoencoder for time-series predictions along with uncertainty estimates to forecast extreme events at Uber. In a similar context, Legrand et al. [42] investigated how inclusion of uncertainty in computation of anomaly score improves the performance of anomaly detection using autoencoder. They does that by developing a new anomaly score function that is weighted with uncertainty as opposed to the Bayesian score function introduce in [40]. This approach is computationally demanding or impose

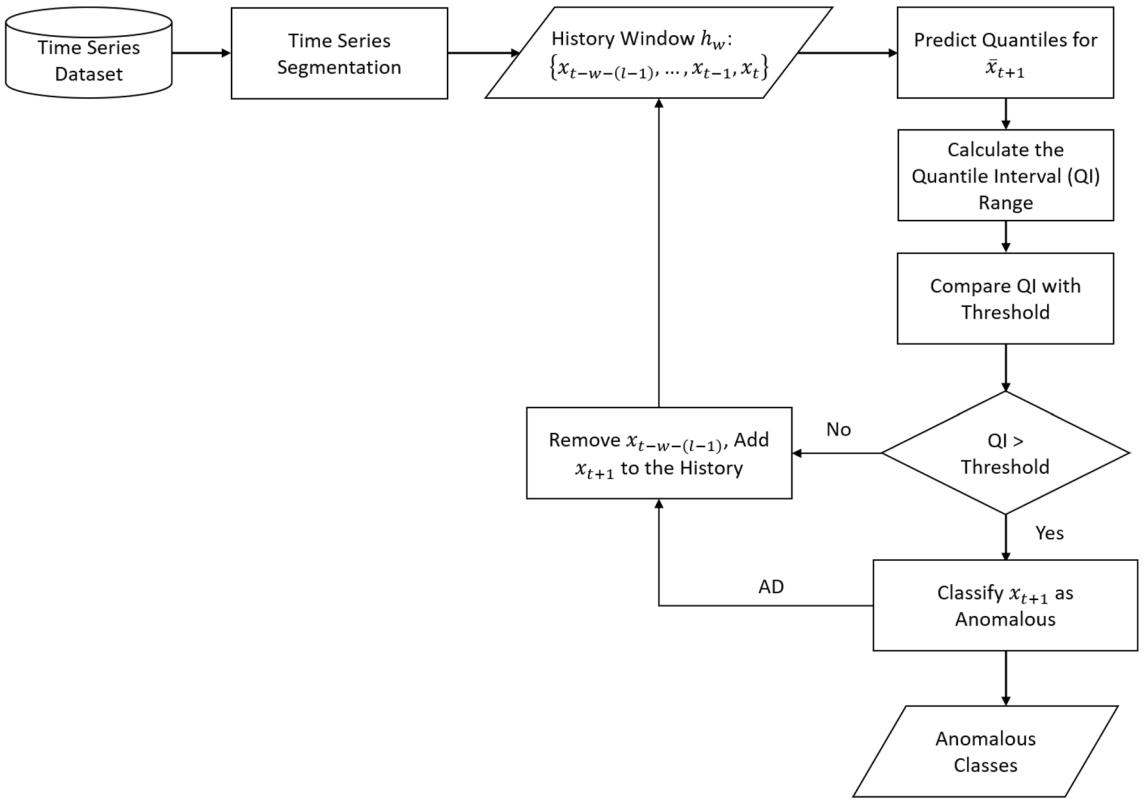

Fig. 1 Flowchart diagram of DQR-AD method that involves time-series segmentation, prediction, and anomaly detection

strong requirements. Reunanen et al. [38] proposed another unsupervised anomaly detection method that combine autoencoder and logistic regression for outlier detection and prediction in sensor data streams. The autoencoder reconstructs the input data and produces hidden representation of the input that can be used to create the required labels for logistic regression to classify anomalous points. The outlier is identified as extreme values that exceed a limit of three standard deviation defined using Chebyshev's inequality or any deviation in the correlation of data features. Although no assumption of the distribution of the data is required, but the method assumes the descriptive statistics (δ and μ) of the unknown normal values to be initially defined. It is also assumed the feature value of the initial data point to have non-zero variance often which the scaling limits for outlier detection cannot be define.

In contrast, a probabilistic forecasting model that returns a full conditional distribution was proposed in [43]. The authors introduced probabilistic forecasting using autoencoder model that directly output parameters of negative binomial and Gaussian likelihoods for real-valued time-series. This shows that the probabilistic forecasts were also achieved by assumption of a Gaussian distribution on the prediction error.

However, an exact parametric distribution is often not relevant in applications which will lead to false anomaly alerts due to high prediction variance. Also, none of the above-mentioned methods used quantile regression which predicts conditional quantiles and can produce probabilistic forecast without making any distributional assumption. An attempt was made to use quantile regression for anomaly detection [44], but it is limited to the use of quantile interval to identify only uncertainties in the data. This limitation leads to the development of an improve quantile regression anomaly detection method (DQR-AD) in this paper. The proposed method go further and used the identified uncertainties to set up a threshold which is compared with quantile interval to classified anomalies.

The Proposed DQR-AD Method

This section presents a detailed description of the proposed (DQR-AD) method. The proposed deep quantile regression anomaly detection (DQR-AD) process consists of three modules, which include time-series segmentation, time-series prediction, and anomaly detection. A detailed concept of these modules is depicted in Fig. 1 and described in the following subsections.

Time-Series Segmentation

This section presents time-series segmentation module which is used to segment time-series into overlapping windows which works as follows: consider a multivariate time-series $x = \{x_1, x_2, \ldots, x_t\}$, where t represent the time-series length and each point $x_i \in R^m$ (for $i = 1 \ldots t$) in the time-series have m-dimensions which correspond to the m features or sensor channels read at time t. A sliding window method is used to segment the time-series into two sequence of overlapping windows each with size l. First, is history window (h_w) $\{x_{t-l}, \ldots, x_{t-1}, x_t\}$ that defines the number of previous time steps which will be used as input to the model. Second, is the l-steps predicted window (p_w), that represent the number of time steps to be predicted where the number of dimensions d being predicted is $1 < d < m$. For example, consider a history window of five previous time steps $h_w = 5$ and one step ahead prediction window $p_w = 1$ as shown in (1):

$$x_{t-4}, x_{t-3}, x_{t-2}, x_{t-1}, x_t \rightarrow x_{t+1}. \tag{1}$$

The left-hand side of Eq. (1) serve as input data to the regression model and the right-hand side as the output which is treated as a label to the input data. This paper aims at predicting the next time step value with a single feature (i.e., $d = 1$ and $p_w = 1$) which resulted in a single scaler prediction $\hat{y}_t$ to be generated for the actual value at each step t. Detailed steps involved in time-series segmentation are given in Algorithm 1. The Algorithm receives as input a sequence of time-series, number of time steps which represent the size of history window, and number of features. It then loops though the sequence segmenting it into subsequence of history and prediction window which are stored in two different lists. The Algorithm returns these two lists as output.

To reduce the dynamic range of signal and enhance the performance of the regression model, the min–max normalization method is applied on the output from Algorithm 1 before passing to the next module. Normalization is important for numerical stability of training the deep neural networks, particularly because time-series prone to anomalies can have unbounded records. It also provides more emphasis on the temporal pattern of time-series and has been shown to be useful for sequence mining [55]. The sequence of both history and prediction windows are scaled between 0 and 1 ($x_{ij} \in [0,1]$) where $j = 1 \ldots m$, as shown in (2):

$$x_i = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}, \tag{2}$$

where $x_{\max}$ and $x_{\min}$ represent vectors of minimum and maximum values of the features. Each point in time will be scaled per each feature.

Time-Series Prediction

This section describes a time-series prediction module which involves deep quantile regression (DQR) that uses the history window (h_w) to forecast the quantiles of the next time step (p_w). The section starts with the description of DQR process which works as follows: the DQR model is built using multilayered LSTM-based RNNs. The choice of LSTM is motivated by its performance on forecasting time-series points and its ability to hold long-term temporal relationship within the time-series [56]. However, this paper focuses on developing a prediction model, which can forecast time-series points considering uncertainty. This is achieved via quantile regression that generate quantiles as oppose to point estimate in ordinary LSTM model [56]. Quantile regression is used, because it predicts conditional

Algorithm 1: Time Series Segmentation

1: Input: Sequence S, $\{\boldsymbol{S}_{\boldsymbol{t=0}}^{\boldsymbol{T}}\}$, n_steps, n_features
2: Output: x: list history window ($\boldsymbol{h_w}$) and y: list of prediction window ($\boldsymbol{p_w}$)
3: **x** ← **list**, **y** ← **list**
4: for i = **0 to len**(**S**) do
5: **end_ix** ← **i** + **n_steps**
6: if **end_ix** > (**len**(**S**)) − **1** break
7: else do
8: **x** ← **S**[**i**: **end_ix**, :]
9: **y** ← **S**[**end_ix**, **0**]
10: end if
11: end for
12: return x , y

quantiles of the target distribution which produces accurate probabilistic forecast without making any distributional assumptions [57]. To perform quantile regression on LSTM, the custom quantile loss (QL) function in Eq. (3) penalizes errors based on the quantile values generated and error's sign:

$$\mathcal{L}(\xi_i|\alpha) = \begin{cases} \alpha\xi_i & \text{if } \xi_i \geq 0 \\ (\alpha-1)\xi_i & \text{if } \xi_i < 0 \end{cases}, \tag{3}$$

where α is the required quantile with values between 0 and 1 and $\xi_i = y_i - f(x_i)$ with $f(x_i)$ as the predicted quantile and y_i as the observed value for the corresponding input x. To create a quantile loss for the entire dataset, the average of the quantile loss for an individual point is taken, as shown in (4):

$$\mathcal{L}(y,f|\alpha) = \frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(y_i - f(x_i)|\alpha). \tag{4}$$

In quantile regression, models are trained to minimize the QL, and by minimizing this loss, the model will have the ability to predict the normal behavior of the time-series. To estimate uncertainty, the DQR model focuses on predicting quantile values [lower (10th), upper (90th), and classical (50th) quantiles]. However, one of the complexities of estimating uncertainties in LSTM using QR is uncertainty due to internal weights initialization, which results in quantiles overlapping. To handle this problem, bootstrapping is employed. This allows the regression model to be iterated n times, thereby storing the predicted values in an array which is finally used to compute the desired quantiles. By predicting upper and lower quantiles, the model is considered to have covered the range of possible values where the difference or interval between these values will be small when the model is sure about the future and big otherwise. This behavior is used to enable model to detect abnormal values from test set as described in the next section. Detailed

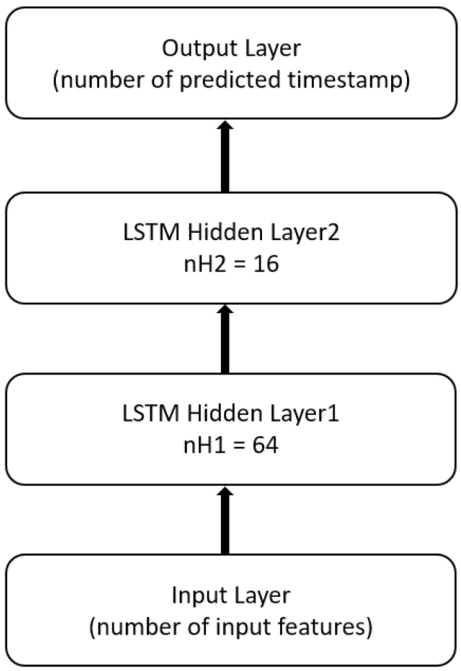

Fig. 2 DQR model architecture for time-series prediction

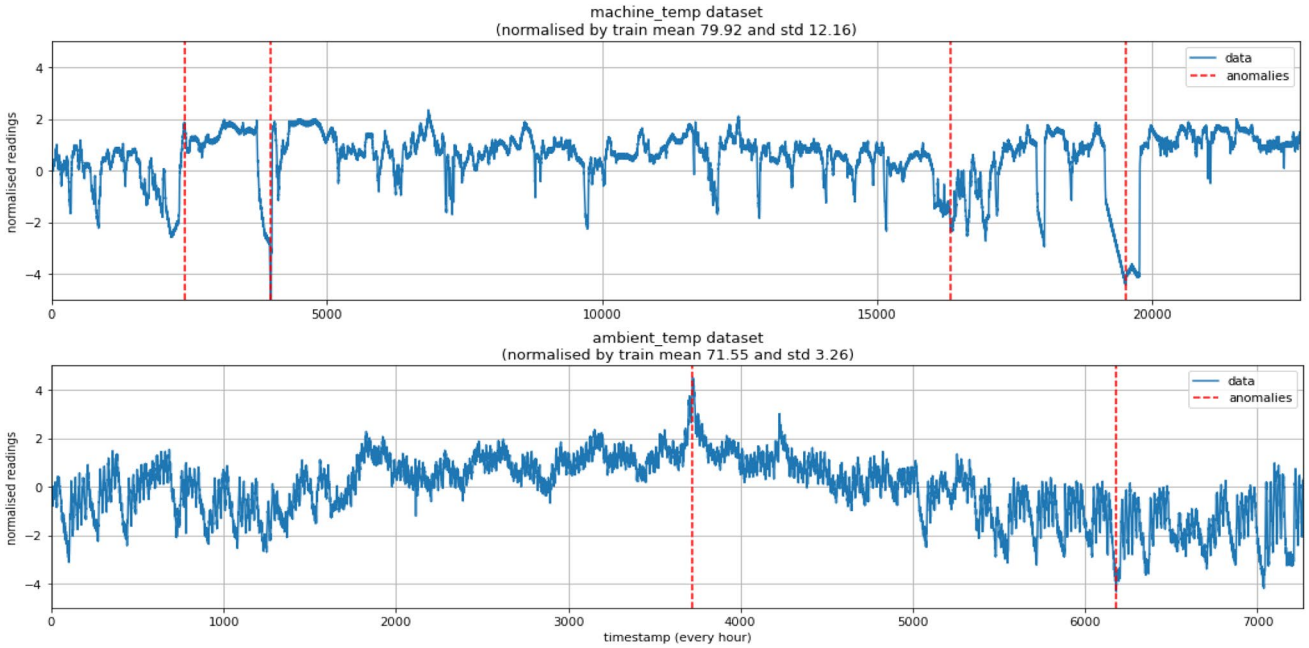

Fig. 3 Samples time-series with anomalies

steps of time-series prediction based on developed model are given in Algorithm 2. The Algorithm receives as input the list of history and prediction windows which is used to train and fit the model for quantile regression. The model predicts the quantiles values for each next time step based on the history. To achieve this, a quantile loss function is created which is integrated with LSTM model. To have an optimal result, bootstrapping is employed where the model is iterated 100 times predicting and storing the quantile values in an array. Thus, the output of this algorithm is an array of predicted quantiles.

Model Architecture Summary

An extensive experiment is carried out to finalize on the model architecture and its parameters. The model consists of four hidden layers that include input, output, and two hidden layers for prediction of quantile values, as shown in Fig. 2. The input layer has l input nodes corresponding to segmented time-series into l window vectors which

Algorithm 2: Time Series Prediction using DQR Model

```
Input: list x and list y
Output: array of lowerQuantile, array of classQuantile, array of upperQuantile
function q_loss(q, y, f)
    e ← (y − f)
 loss ← mean(maximum(q ∗ e, (q − 1) ∗ e), axi = −1)
return loss
end function
losses ← [lambda y, f: q_loss(0.1,y,f), lambda y, f : q_loss(0.5,y,f), lambda
y, f: quantile_loss(0.9,y,f)]
inputLayer ← (Z.shape[1], Z.shape[2])
lstmLayer1 ← LSTM(64, return_sequence=True, dropout=0.3)
(inputLayer, training=True)
lstmLayer2 ← LSTM(16, return_sequence=True, dropout=0.3)
(lstmLayer1, training=True)
lstmLayer3 ← Dense(50)(lstmLayer2)
output_10 ← Dense(1)(lstmLayer3)
output_50 ← Dense(1)(lstmLayer3)
output_90 ← Dense(1)(lstmLayer3)
lstmModel ← Model(inputLayer, [output_10, output_50, output_90])
lstmModel.compile(loss=losses, optimizer=adam, loss_weights=[0.3, 0.3, 0.3])
lstmModel.fit(x, [y, y, y])
forecastModel ← K.function([lstmModel.layers[0].input, K.learning_phase()],
[lstmModel.layers[-3].output, lstmModel.layers[-2].output, lstmModel.layers
[-1].output])
lowerQuantile ← [ ]
classQuantile ← [ ]
upperQuantile ← [ ]
for i = 0 to 100 do
    y` ← forecastModel([x, 0.5])
    lowerQuantile ← y`[0]
    classQuantile ← y`[1]
    upperQuantile ← y`[2]
end for
return lowerQuantile, classQuantile, upperQuamtile
```

represents the h_w. The first and second hidden layers are composed of 64 and 16 number of nodes, respectively, which is followed by an element-wise activation function ReLU. The output layer is a fully connected dense layer with its nodes connected to all nodes in the previous layers. This last layer calculates the predicted quantiles of the next time stamp with a number of nodes equal to the size of p_w. For the model predicting only next time stamp, the number of output nodes is 1.

Anomaly Detection and Classification

This section presents anomaly detection and classification module which classify anomalous points in time-series data. The anomaly detection process is described as follows: given the model predicted quantiles from previous section, a time-series point can then be identified as anomalous using the QI. The QI is computed as difference between upper and lower quantiles (90–10 quantiles range). The interval will be small when the model knows about the future and, on the contrary, to have uncertainties when there is a bigger interval. This is because the model is not trained to handle this type of scenarios and is unable to detect such changes in the data that can result in anomalies. The QI is used as anomaly score: large values flag up similarly relevant anomalies for the given timestamp. However, this module is expected to clarify the threshold based on the time-series type: a limitation and constraint for most anomaly detection techniques. When the QI related to a point in time is greater than the threshold, that point is classified and flagged up as anomaly. According to the literature [41, 54], two strategies are used for flagging abnormal data which include: anomaly detection (AD) and anomaly detection and mitigation (ADAM). To allow continues detection of anomaly, AD is chosen in this method as against ADAM, which faces the problem of false alarms for dynamic data [41]. Detailed steps of anomaly detection process are given in Algorithm 3. The Algorithm receives as input arrays of lower and upper quantile from the previous module, and a fixed threshold value. Quantile interval is then computed as a difference between upper and lower intervals. The QI is compared with the threshold to classify anomalies which is given as output. The output is given as an array of anomalous points.

Algorithm 3: Anomaly Classification

```
Input: threshold, lowerQuantile, upperQuamtile
Output: array of anomalous class
anomaly ← [ ]
QI ← (upperQuantile – lowerQuantile)
 if QI > threshold do
     anomaly ← 1
 else do
     anomaly ← 0
end if
return anomaly
```

Experiments

This section covers the extensive experiments conducted to test and compare the performance of DQR-AD with six other state-of-the-art anomaly detection methods that model prediction error using Gaussian distribution to identify anomalies. These methods include DeepAnT [33], NumentaTM [52], ContextOSE [58], EXPoSE [59], AE [38], and VAE-LSTM [39]. The experiment is conducted using real and synthetic datasets from different application domain. We divided the experiment into two parts, because AE is evaluated using different benchmark dataset based on a different metric. The division is based on NAB and sMAP benchmark datasets. Each experiment describes the dataset and experimental setups, and ends with results and discussion. All experiments are carried out on the same computer with Intel Pentium core i7 processor, Windows OS, and deep learning libraries on python anaconda 3.7.

Part I Experiment: NAB Dataset

Dataset Description

NAB (Numenta Anomaly Benchmark) [60] is a publicly available streaming benchmark dataset released by Numenta and available in their repository.[1] The benchmark dataset has been widely used in the literature for evaluating anomaly detection techniques [31, 52]. This dataset contains 58 data streams, each with 1000–22,000 records. The dataset contains real data from different application domains: road traffic, network utilization, online advertisement, sensors on industrial machines, and social media, and some artificially generated data files with artificial anomalies. Each dataset file records have time stamps and data values with anomaly labels stored in a separate set of files. An additional column is created in each dataset that hold anomalous label of each data point. The labels are created either based on the known root cause or as a result of labeling procedures defined in [52]. Although NAB contains labeled streaming anomaly detection datasets, it is faced with some challenges, which may affect point anomaly detection [61]. First, each label of an unusual point depends on the defined anomalous window. Which means when one data point within the window is an anomaly, all other data points in that window are also abnormal. Second, most of the data files have different data distribution where the distribution of some few timestamps is quite different from the distribution of the remaining time-series. The second challenge affects the choice of training and test sets which may come from different distribution and will have a negative impact on training and evaluation of time-series models. However, this characteristic becomes the main reason why we selected this dataset to evaluate the performance of our model in handling uncertainties due to model misspecification, as discussed in "The Proposed DQR-AD Method". Figure 3 shows time-series samples with anomalous points from the dataset.

Experimental Setup

In this subsection, we provide detail description of how the experiment is conducted. Two levels of the same experiment are conducted in this section. On the first level, DQR-AD is evaluated and compared with five time-series anomaly detection methods based on precision and recall using 20 NAB time-series from different domains. In this experimental work, the same time-series reported in [33] are used. On the second level of the experiment, detailed analysis of all the seven state-of-the-art methods and compare them with DQR-AD on the whole NAB benchmark datasets. For anomaly detection part, AE and VAE-LSTM are evaluated on the same settings and parameters with DQR-AD; while for the remaining methods, we used the same settings and parameters as stated in [33]. For this detailed evaluation, we used F-score (5) to report an overall performance of the anomaly detection methods. F-score is the most commonly used metric to evaluate the performance of anomaly detection methods [33]. Since NAB records have multiple time-series related to different application domains, average F_{score} is reported for each method in relation to each domain:

$$F_{\text{score}} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}. \tag{5}$$

To train all the models for time-series detection, records of each time-series are organized in 80% training set and 20% test set. From the training set, 20% of it will be used for validation. Both the training, test, and validation sets are segmented into history (h_{w}) and prediction window (p_{w}). The size of history window is set up by looking at the data to notice the presence of either daily, hourly, or weekly pattern in all the time-series. As such, a daily history window of size 24 timestamps (one observation every hour) and a prediction window of size 1 were chosen for predicting only one timestamp. No label information is required in this unsupervised training process. Instead, for each window of previous timestamps, only next timestamp is predicted and serve as the target. We also used the same 20% test split and the same window size for all the anomaly detection methods we used in our comparisons. DQR-AD, LSTM-AD, and VAE-LSTM models are trained using mini-batch gradient descent where a batch size of 128 is used and 50 number of epochs. For DQR-AD, we make use of bootstrapping by reactivating the dropout with value 0.3 and iterate for 100 times, thereby

[1] https://github.com/numenta/NAB.

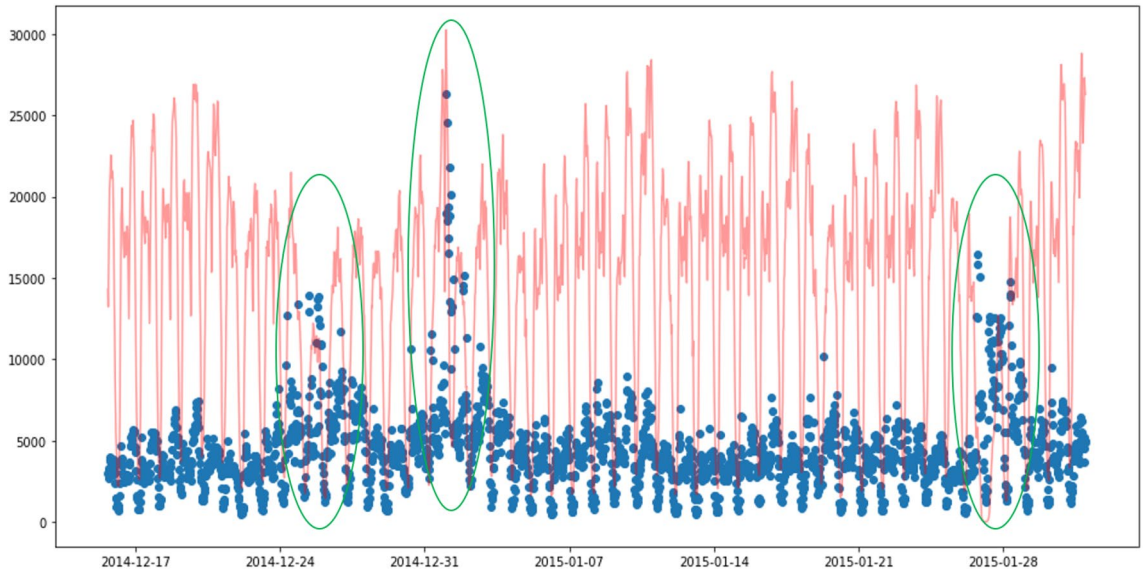

Fig. 4 For nyc_taxi dataset, actual time-series values (red) plotted against the QI (blue) to show periods of uncertainties (circled in green) in the test set

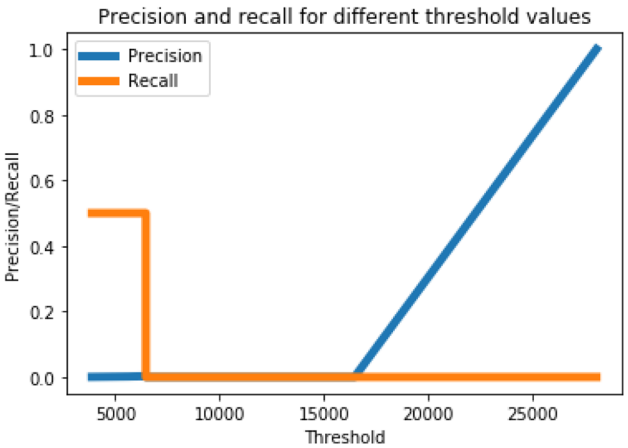

Fig. 5 A trade-off between precision and recall used for setting up a threshold value

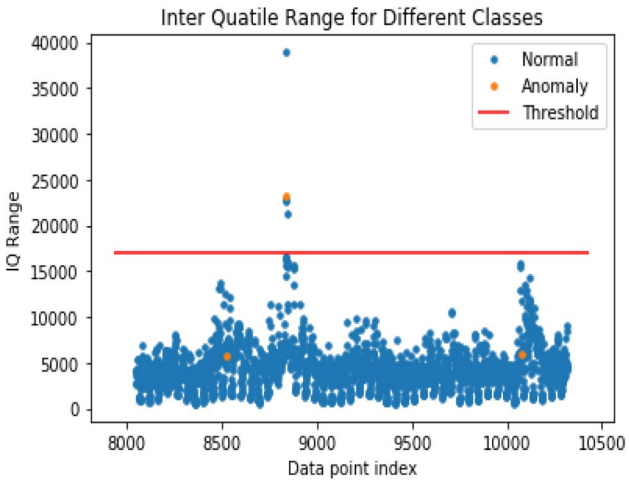

Fig. 6 Using a threshold value of 17,000 for classification

storing the predicted values in an array which is finally used to compute the desired quantiles.

As highlighted earlier, the limitation of finding best threshold is relevant in evaluating anomaly detection methods. Since each time-series in the NAB dataset is different, a generic threshold for all benchmark collection time-series is not easy to be found or defined. As such, we used the validation data to set a threshold for each time-series. This is achieved in many cases with a trade-off approach between precision and recall of the predicted results on validation data. The threshold value is then used to classify anomalous points in the test data.

Experimental Result and Discussion

Figure 4 shows DQR-AD anomaly detection results on a single time-series: the red lines represent actual time-series and QI in blue dots. It can be seen in this example; the QI goes high in the period of uncertainties (circled in green). We used this behavior to set up a threshold value of 17,000 which is maximum trade-off value between precision and recall of the validation set, as shown in Fig. 5. The threshold value is then used for classification in the test data. The classification results on the test set are shown in Fig. 6, where the actual normal points are depicted in blue and abnormal points in orange with a red line indicating the threshold used

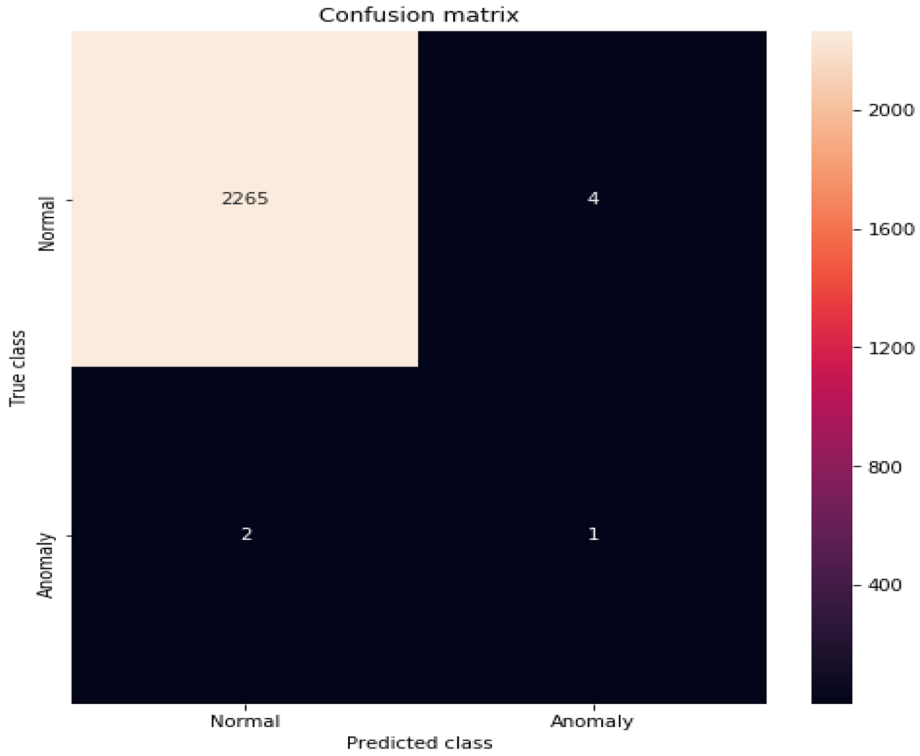

Fig. 7 Confusion matrix for DQR-AD

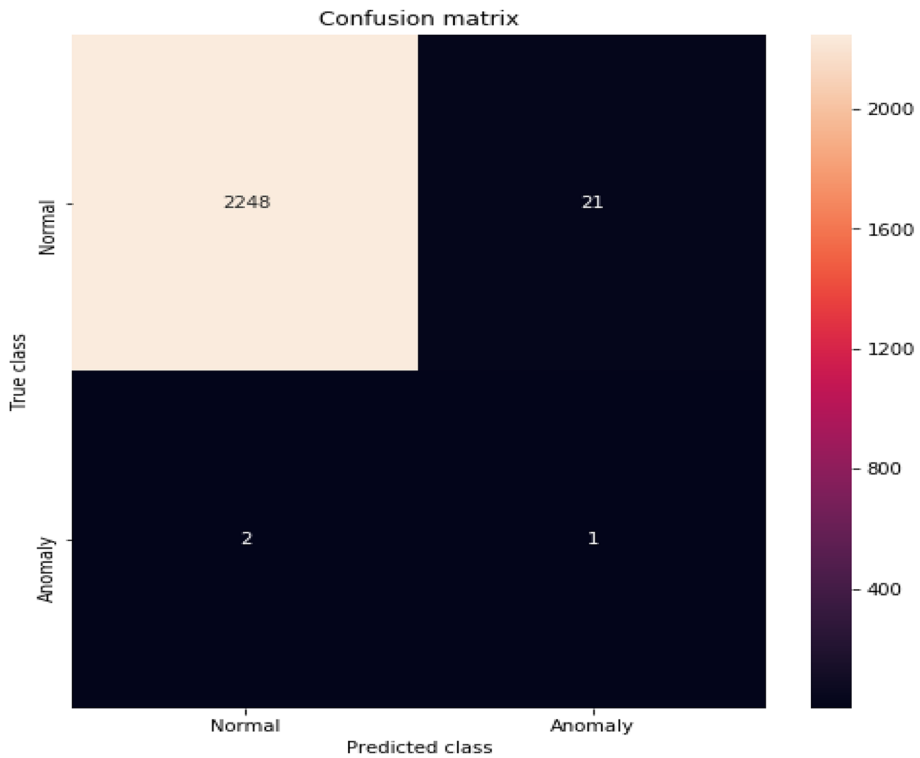

Fig. 8 Confusion matrix for DeepAnT

by DQR-AD for classification of anomalies. True-positive and false-positive values are depicted with points above the threshold line in orange and blue, respectively. To have a better look of the classification result, a confusion matrix is shown in Fig. 7. The confusion matrix shows when compared with DeepAnT method on the same time-series and under the same settings, it shows the same performance in terms of number of anomalies identified (true positive) but with higher rate of false-positive value equal to 41 as shown in the confusion matrix in Fig. 8.

Table 1 Comparative evaluation of DQR-AD with four anomaly detection methods (NumentaTM, ContextOSE, DeepAnT, and VAE-LSTM) on 20 NAB time-series from different domains

NAB dataset	NumentaTM		ContextOSE		DeepAnT		VAE-LSTM		DQR-AD	
	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall	Precision	Recall
Real known cause										
nyc_taxi	0.85	0.006	1	0.002	1	0.002	0.961	1	1	0.33
ambient_temperature	0.05	0.006	0.33	0.001	0.26	0.06	0.806	1	1	0.06
cpu_utilization	0.52	0.01	0.12	0.001	0.63	0.36	0.694	1	1	0.5
ec2_request_late	1	0.009	1	0.009	1	0.04	0.993	1	1	0.7
machine_temperature	0.27	0.004	1	0.001	0.8	0.001	0.559	1	0.5	0.5
rogue_agent_key_hold	0.5	0.005	0.33	0.005	0.34	0.05	0.02	0.1	1	0.05
rogue_agent_key_updown	0	0	0	0	0.11	0.001	0.11	0	0.5	0.1
Real Ad exchange										
exchange-2_cpc_results	0	0	0.5	0.006	0.03	0.33	0	0	0.7	0.11
exchange-3_cpc_results	1	0.007	0.75	0.02	0.71	0.03	0	0	1	0.33
Real tweets										
Twitter_volume_GOOG	0.38	0.005	0.75	0.002	0.75	0.01	0.03	1	0.83	1
Twitter_volume_IBM	0.22	0.005	0.37	0.002	0.5	0.005	0	0	1	0.1
Real traffic										
occupancy_6005	0.2	0.004	0.5	0.004	0.5	0.004	0.01	1	1	0.1
occupancy_t4013	0.66	0.008	1	0.008	1	0.036	0.11	0.5	1	0.48
speed_6005	0.25	0.008	0.5	0.004	1	0.008	0	0	1	0.5
speed_7578	0.6	0.02	0.57	0.03	1	0.07	0.01	0.33	1	0.25
speed_t4013	0.8	0.01	1	0.008	1	0.08	0	0	1	0.20
TravelTime_387	0.33	0.004	0.6	0.01	1	0.004	0	0	0	0
TravelTime_451	0	0	1	0.005	1	0.009	0.10	1	1	0.25
Real AWS cloud watch										
ec2_cpu_utilization_5f5533	1	0.01	1	0.005	1	0.01	0.06	1	1	0.23
rds_cpu_utilization_cc0c53	1	0.002	1	0.005	1	0.03	0	0	1	0.33

Precision and recall are reported in this table

Table 2 Comparative evaluation of DQR-AD with five other anomaly detection methods (ContextOSE, EXPoSE, NumentaTM, DeepAnT, and VAE-LSTM) applied to entire NAB dataset using mean F_{score} for each domain

Datasets	ContextOSE	EXPoSE	NumentaTM	VAE-LSTM	DeepAnT	DQR-AD
Real known cause	0.005	0.005	0.012	**0.629**	0.200	0.408
Real Ad exchange	0.022	0.005	0.035	0.006	0.132	**0.301**
Real tweets	0.003	0.003	0.010	0.011	0.075	**0.280**
Real traffic	0.02	0.011	0.036	0.060	0.223	**0.376**
Real AWS cloud watch	0.007	0.015	0.018	0.002	**0.146**	0.031
Artificial with anomaly	0.022	0.004	0.017	0.012	0.156	**0.276**

As indicated in the above figures, the classification result of DQR-AD is significantly better than that of DeepAnT in terms of false-positive values which is the main problem that this paper addresses. On more detailed level, Table 1 shows results of the first level of the experiment. It can be observed from this table that DQR-AD obtained relatively better precision and recall than NumentaTM, ContextOSE, and DeepAnT in almost all the time-series except in Real-KnownCause time-series where VAE-LSTM have better performance only in recall. This demonstrates the ability of DQR-AD in detecting higher number of anomalies with low false-positive rates. On the other hand, Table 2 shows the result of the second-level experiment where mean F_{score} for the four algorithms is reported on the whole NAB dataset. As indicated in the table, DQR-AD outperforms other methods in all domains except in two domains (i.e., real known cause and real AWS cloud watch) where VAE-LSTM and DeepAnT, respectively, have better performance with relatively small margin. It is therefore clear that DQR-AD outperforms both other methods on 4 out of 6 domains (as

Table 3 Comparative evaluation of DQR-AD and AE ($h=2$, $h=10$) on five sMAP datasets (SMAP1 to SMAP5)

Datasets	AE ($h=2$)	AE ($h=10$)	DQR-AD
SMAP1	0.88	0.89	**0.98**
SMAP2	**0.89**	**0.89**	0.86
SMAP3	0.88	0.88	**0.95**
SMAP4	**0.88**	**0.88**	0.87
SMAP5	0.88	0.88	**0.95**

An average AUROC is reported on each dataset

indicated in bold) from the table. DQR-AD is approximately two-to-three times better than the DeepAnT which perform better than all the remaining four methods for different domains in the NAB dataset.

Part II Experiment: sMAP Dataset

Dataset Description

In this part of the experiment, we tested and evaluated our method using sMAP dataset which used in [38]. The dataset is retrieved using a python public front end.[2] sMAP is a real sensor data that consist reading of electrical sensors from an elevator in Cory Hall at University of California, Berkeley. The dataset contains five unlabeled time-series (SMAP1 to SMAP5) each one having 20,000 data points and four features. To label the time-series and ease the computation of evaluation metric for our algorithm, we used the same procedure as in [38] where the valued of the random features of a data point are swapped. This approach preserves the order of the data points and helps in providing an access to the anomalies during evaluation.

Experimental Setup

We used the same setting in part 1 experiment where our proposed model is train using 80–20 splitting pattern. Similarly, an additional 20% from the training set is used for evaluation. Both the training, test, and validation sets are segmented into history (h_w) and prediction window (p_w). The size of history window is set up daily window of size 24 timestamps (one observation every hour). A prediction window of size 1 was chosen for predicting only one timestamp where for each window of previous timestamps, only next timestamp is predicted. We maintained the unsupervised learning in the training process of both algorithms by removing the class label and consider for each window of previous timestamps, only the next timestamp is predicted and serve as the target. For AE, we used 2 and 10 hidden neurons to provide an extensive evaluation of the outlier detection algorithm as indicated in [38]. While on the other hand, we used the same parameter settings in the previous part of the experiment for DQR-AD. In this part of the experiment, AUROC is used as metric to measure the performance of our model. Similar to [38], the performance evaluation is repeated ten times which enable evaluation of our learning algorithm for data stream. An average AUROC for the ten repetitions is reported for each time-series.

Experimental Results and Discussion

Table 3 reports the outlier detection performance of DQR-AD compared with AE on the sMAP dataset (SMAP1 to SMAP5). Each cell of the table contains an average AUROC value of ten repetitions. It can be observed from the table that both algorithms achieved AUROC value that exceed 0.8 which shows that they all succeeded in detecting anomalies in sensor streams. However, the result in Table 3 demonstrate good performance of DQR-AD where out of the five-time-series (SMAP1 to SMAP5), DQR-AD is 10% better in performance than AE on three datasets (SMAP1, SMAP3, and SMAP5) and relatively equal performance on the remaining two datasets which have higher level of noise. This demonstrates the ability of DQR-AD in consistently providing an accurate result on different datasets and experimental settings. The use of sMAP time-series with four-dimensional features demonstrates the applicability of our proposed approach in multivariate time-series data.

Conclusion and Future Work

This paper presents a deep learning-based anomaly detection method for detection and classification of anomalies in time-series data. Deep quantile regression anomaly detection (DQR-AD) is unsupervised and does not require any assumption of regular data distribution to identify anomalous data points. Instead, the method used quantile interval which quantifies the level of uncertainties associated with the LSTM point forecasts and helps mitigates against false anomaly alerts. The proposed method can detect sudden spikes in time-series: this particular challenge is generally missed in reports listed in the literature review section that propose other distance and density anomaly detection methods.

In the first part of the experiment, DQR-AD is evaluated on the NAB benchmark dataset containing 58 real and synthetic time-series and compared with 6 other prediction-based anomaly detection methods that assume normal distribution on prediction or reconstruction error for identification of anomalies. The experimental results as shown in Table 1 indicate that DQR-AD obtained relatively better precision

[2] https://pythonhosted.org/Smap/en/2.0/python_access.html.

than all other methods. This demonstrates that DQR-AD is capable of detecting higher number of anomalous points with low false-positive rates. Similarly, the results in Table 2 show DQR-AD to be approximately two-to-three times better than the DeepAnT which performs better than all the remaining methods on all domains in the NAB dataset. This demonstrates that our approach can be practically applied on the time-series with large amount of unlabeled data. In the second part of the experiment, DQR-AD have 10% better performance than AE on three datasets (SMAP1, SMAP3, and SMAP5) and equal performance on the remaining two datasets (SMAP2 and SMAP4) with relatively higher level of noise. The use of sMAP time-series with four-dimensional features demonstrate the applicability of DQR-AD on multivariate time-series data.

The future work will be focused on optimizing this method to incorporate more scenarios for comparison. Also, to improve our model, we will add feature extraction module through autoencoder that will extract time-series representation before predictions. We will also work toward extending the method to identify both concept drift and anomalies in real-time sensor streams.

Funding This study was funded by Petroleum Technology Development Fund (PTDF) PhD Scholarship, Nigeria (Award Number: PTDF/ED/PHD/IAT/884/16).

Declarations

Conflict of interest The authors declare that they have no conflict of interest.

References

1. Da Xu L, He W, Li S. Internet of things in industries: a survey. IEEE Trans Ind Inform. 2014;10(4):2233–43.
2. Kandhari R, Chandola V, Banerjee A, Kumar V, Kandhari R. Anomaly detection: a survey. ACM Comput Surv. 2009;41(3):1–6.
3. Andreas Theissler ID. An anomaly detection approach to detect unexpected faults in recordings from test drives. In: Proc. WASET Int. Conf. Veh. Electron. Saf. 2013, Stock., vol. 7, no. 7, pp. 195–198, 2013.
4. Sangha MS, Yu DL, Gomm JB. Sensor fault diagnosis for automotive engines with real data evaluation. Multicr Int J Eng Sci Technol. 2011;3(8):13–25.
5. Fujimaki R. Anomaly detection support vector machine and its application to fault diagnosis. In: 2008 Eighth IEEE International Conference on Data Mining, 2008, pp. 797–802.
6. Sotiris VA, Tse PW, Pecht MG. Anomaly detection through a Bayesian support vector machine. IEEE Trans Reliab. 2010;59(2):277–86.
7. Sheikhan M, Jadidi Z. Flow-based anomaly detection in high-speed links using modified GSA-optimized neural network. Neural Comput Appl. 2014;24(3–4):599–611.
8. Holst A, Bohlin M, Ekman J, Sellin O, Lindström B, Larsen S. Statistical anomaly detection for train fleets. AI Mag. 2012;34(1):33.
9. Hill DJ, Minsker BS, Amir E. Real-time Bayesian anomaly detection in streaming environmental data. Water Resour Res. 2009;45(4).
10. Angiulli F, Pizzuti C. Fast outlier detection in high dimensional spaces. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2002, vol. 2431 LNAI, pp. 15–27.
11. Zhang J, Wang H. Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. Knowl Inf Syst. 2006;10(3):333–55.
12. Breunig MM et al. LOF: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data—SIGMOD '00, 2000, vol. 29, no. 2, pp. 93–104.
13. Huang H, Qin H, Yoo S, Yu D (2014) Physics-based anomaly detection defined on manifold space. ACM Trans Knowl Discov. Data (TKDD). 2014;9(2).
14. A least-squares approach to anomaly detection in static and sequential data. Pattern Recognit Lett. 2014;40:36–40.
15. Amarbayasgalan T, Jargalsaikhan B, Ryu K. Unsupervised novelty detection using deep autoencoders with density based clustering. Appl Sci. 2018;8(9):1468.
16. Fujimaki R, Yairi T, Machida K. An anomaly detection method for spacecraft using relevance vector learning. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2005, vol. 3518 LNAI, pp. 785–790.
17. Pincombe B. Anomaly detection in time-series of graphs using ARMA processes. IEEE J Sel Top Signal Process. 2005;24(4):2.
18. Zare Moayedi H, Masnadi-Shirazi MA. ARIMA model for network traffic prediction and anomaly detection. In: 2008 International Symposium on Information Technology, 2008, pp. 1–6.
19. Knorn F, Leith DJ. Adaptive Kalman filtering for anomaly detection in software appliances. In: Proceedings—IEEE INFOCOM, 2008, pp. 1–6.
20. Foxt AJ. Outliers in time-series. J R Stat Soc Ser B. 1972;34(3):350–63.
21 Seheult AH, Green PJ, Rousseeuw PJ, Leroy AM. Robust regression and outlier detection. J R Stat Soc Ser A. 1989;152(1):133.
22. Chalapathy R, Chawla S. Deep learning for anomaly detection: A survey. 2019. arXiv:1901.03407 [Online].
23. Cho K et al. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In: EMNLP 2014–2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 2014, pp. 1724–1734.
24. Ullah A, Ahmad J, Muhammad K, Sajjad M, Baik SW. Action recognition in video sequences using deep bi-directional LSTM with CNN features. IEEE Access. 2017;6:1155–66.
25. Graves A, Mohamed A, Hinton G. Speech recognition with deep recurrent neural networks. In: 2013 IEEE international conference on acoustics, speech and signal processing, 2013, pp. 6645–6649.

26. Graves A, Schmidhuber J. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neural Netw. 2005;18(5–6):602–10.
27. Gugulothu N, Tv V, Malhotra P, Vig L, Agarwal P, Shroo G. Predicting remaining useful life using time-series embeddings based on recurrent neural networks. In: 2nd ML PHM Work. SIGKDD 2017, vol. 10, 2017.
28. Malhotra PAP, Vig L, Shroff G, Rinard M. Long short term memory networks for anomaly detection in time-series. In: Proceedings, European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning. Bruges (Belgium), 22–24 April 2015, 2015.
29. Aldosari MS. Unsupervised anomaly detection in sequences using long short term memory recurrent neural networks. PhD Diss. Georg. Mason Univ., p. 98, 2016.
30. Lipton ZC, Kale DC, Elkan C, Wetzel R. Learning to diagnose with LSTM recurrent neural networks. In: 4th International Conference on Learning Representations, ICLR 2016—Conference Track Proceedings, 2016.
31. Saurav S et al. 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—CoDS-COMAD '18, 2018, pp. 78–87.
32. Kanarachos S, Christopoulos S-RG, Chroneos A, Fitzpatrick ME. Detecting anomalies in time-series data via a deep learning algorithm combining wavelets, neural networks and Hilbert transform. Expert Syst Appl. 2017;85:292–304.
33. Munir M, Siddiqui SA, Dengel A, Ahmed S. DeepAnT: a deep learning approach for unsupervised anomaly detection in time-series. IEEE Access. 2019;7:1991–2005.
34. Malhotra P, Ramakrishnan A, Anand G, Vig L, Agarwal P, Shroff G. LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. In: Presented at ICML 2016 anomaly detection workshop. New York, NY. Available: arXiv:1607.00148. 2016 [Online].
35. Schreyer M, Sattarov T, Borth D, Dengel A, Reimer B. Detection of anomalies in large scale accounting data using deep autoencoder networks, CoRR, pp. 1–19. Available: arXiv:1709.05254. 2017 [Online].
36. Lu W, et al. Unsupervised sequential outlier detection with deep architectures. IEEE Trans Image Process. 2017;26(9):4321–30.
37. Zong B et al. Deep autoencoding Gaussian mixture model for unsupervised anomaly detection. In: 6th Int. Conf. Learn. Represent. ICLR 2018—Conf. Track Proc., pp. 1–19, 2018.
38. Reunanen N, Räty T, Jokinen JJ, Hoyt T, Culler D. Unsupervised online detection and prediction of outliers in streams of sensor data. Int J Data Sci Anal. 2020;9(3):285–314.
39. Lin S, Clark R, Birke R, Schonborn S, Trigoni N, Roberts S. Anomaly detection for time-series using VAE-LSTM hybrid model. In: ICASSP, IEEE International Conference on Acoustics, Speech and Signal Processing—Proceedings, 2020, vol. 2020–May, pp. 4322–4326.
40. Zhu L, Laptev N. Deep and confident prediction for time-series at uber. In: IEEE International Conference on Data Mining Workshops, ICDMW, 2017, vol. 2017–Novem, pp. 103–110.
41. Pang J, Liu D, Peng Y, Peng X. Anomaly detection based on uncertainty fusion for univariate monitoring series. Measurement. 2017;95:280–92.
42. Legrand A, Trannois H, Cournier A. Use of uncertainty with autoencoder neural networks for anomaly detection. Proc.—IEEE 2nd Int. Conf. Artif. Intell. Knowl. Eng. AIKE 2019, pp. 32–35, 2019.
43. Salinas D, Flunkert V, Gasthaus J, Januschowski T. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. Int J Forecast. 2019;36(3):1181–91.
44. Cerliani M. Anomaly detection with LSTM in Keras. towardsdatascience.com, 2019. [Online]. Available: https://towardsdatascience.com/anomaly-detection-with-lstm-in-keras-8d8d7e50ab1b. Accessed: 03 June 2021.
45. Gupta M, Gao J, Aggarwal CC, Han J. Outlier detection for temporal data: a survey. IEEE Trans Knowl Data Eng. 2014;26(9):2250–67.
46. Wang C, Viswanathan K, Choudur L, Talwar V, Satterfield W, Schwan K. Statistical techniques for online anomaly detection in data centers. In: Proceedings of the 12th IFIP/IEEE International Symposium on Integrated Network Management, IM 2011, 2011, pp. 385–392.
47. Hochreiter S, Schmidhuber JJ. Long short-term memory. Neural Comput. 1997;9(8):1–32.
48. Chauhan S, Vig L. Anomaly detection in ECG time signals via deep long short-term memory networks. In: Proceedings of the 2015 IEEE International Conference on Data Science and Advanced Analytics, DSAA 2015, 2015.
49. Singh A. Anomaly detection for temporal data using long short-term memory (LSTM). Stock. SWEDEN Inf. Commun. Technol., 2017.
50. Bontemps L, Cao VL, McDermott J, Le-Khac NA. Collective anomaly detection based on long short-term memory recurrent neural networks. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 2016, vol. 10018 LNCS, pp. 141–152.
51. Shipmon DT, Gurevitch JM, Piselli PM, Edwards ST. Time series anomaly detection; detection of anomalous drops with limited features and sparse examples in noisy highly periodic data. arXiv:1708.03665. 2017 [Online].
52. Ahmad S, Lavin A, Purdy S, Agha Z. Unsupervised real-time anomaly detection for streaming data. Neurocomputing. 2016;20:53.
53. Hundman K, Constantinou V, Laporte C, Colwell I, Soderstrom T. Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2018, pp. 387–395.
54. Hill DJ, Minsker BS. Anomaly detection in streaming environmental sensor data: a data-driven modeling approach. Environ Model Softw. 2010;25(9):1014–22.
55. Lin J, Keogh E, Wei L, Lonardi S. Experiencing SAX: a novel symbolic representation of time-series. Data Min Knowl Discov. 2007;15(2):107–44.
56. Gers FA, Eck D, Schmidhuber J. Applying LSTM to time-series predictable through time-window approaches. London: Springer; 2002. p. 193–200.
57. Wen R, Torkkola K, Narayanaswamy B, Madeka D. A multi-horizon quantile recurrent forecaster. 2017.
58. Hayes MA, Capretz MAM. Contextual anomaly detection in big sensor data. In: Proceedings - 2014 IEEE international congress on big data, BigData Congress; 2014. pp. 64–71.
59. Schneider M, Ertel W, Ramos F. Expected similarity estimation for large-scale batch and streaming anomaly detection. Mach Learn. 2016;105(3):305–33.
60. Lavin A, Ahmad S. Evaluating real-time anomaly detection algorithms—the Numenta anomaly benchmark. In: 14th Int. Conf. Mach. Learn. Appl. (IEEE ICMLA'15), vol. 28, no. 2, pp. 34–37, 2015.
61. Singh N, Olinsky C. Demystifying Numenta anomaly benchmark. In: Proceedings of the International Joint Conference on Neural Networks, 2017, vol. 2017–May, pp. 1570–1577.