Received November 21, 2018, accepted December 4, 2018, date of publication December 19, 2018,
date of current version January 7, 2019.

Digital Object Identifier 10.1109/ACCESS.2018.2886457

DeepAnT: A Deep Learning Approach for Unsupervised Anomaly Detection in Time Series

MOHSIN MUNIR[1,2], SHOAIB AHMED SIDDIQUI[1,2], ANDREAS DENGEL[1,2], AND SHERAZ AHMED[2]
[1]Fachbereich Informatik, Technische Universität Kaiserslautern, 67663 Kaiserslautern, Germany
[2]German Research Center for Artificial Intelligence (DFKI GmbH), 67663 Kaiserslautern, Germany

Corresponding author: Mohsin Munir (mohsin.munir@dfki.de)

This work was supported in part by the BMBF project DeFuseNN under Grant 01IW17002 and in part by the NVIDIA AI Lab (NVAIL) Program.

ABSTRACT Traditional distance and density-based anomaly detection techniques are unable to detect periodic and seasonality related point anomalies which occur commonly in streaming data, leaving a big gap in time series anomaly detection in the current era of the IoT. To address this problem, we present a novel deep learning-based anomaly detection approach (DeepAnT) for time series data, which is equally applicable to the non-streaming cases. DeepAnT is capable of detecting a wide range of anomalies, i.e., point anomalies, contextual anomalies, and discords in time series data. In contrast to the anomaly detection methods where anomalies are learned, DeepAnT uses unlabeled data to capture and learn the data distribution that is used to forecast the normal behavior of a time series. DeepAnT consists of two modules: time series predictor and anomaly detector. The *time series predictor* module uses deep convolutional neural network (CNN) to predict the next time stamp on the defined horizon. This module takes a window of time series (used as a context) and attempts to predict the next time stamp. The predicted value is then passed to the *anomaly detector* module, which is responsible for tagging the corresponding time stamp as normal or abnormal. DeepAnT can be trained even without removing the anomalies from the given data set. Generally, in deep learning-based approaches, a lot of data are required to train a model. Whereas in DeepAnT, a model can be trained on relatively small data set while achieving good generalization capabilities due to the effective parameter sharing of the CNN. As the anomaly detection in DeepAnT is unsupervised, it does not rely on anomaly labels at the time of model generation. Therefore, this approach can be directly applied to real-life scenarios where it is practically impossible to label a big stream of data coming from heterogeneous sensors comprising of both normal as well as anomalous points. We have performed a detailed evaluation of 15 algorithms on 10 anomaly detection benchmarks, which contain a total of 433 real and synthetic time series. Experiments show that DeepAnT outperforms the state-of-the-art anomaly detection methods in most of the cases, while performing on par with others.

INDEX TERMS Anomaly detection, artificial intelligence, convolutional neural network, deep neural networks, recurrent neural networks, time series analysis.

I. INTRODUCTION

Anomaly detection has been one of the core research areas for a long time due to its ubiquitous nature. In everyday life, we observe the abnormalities that are the focus of our attention. When something deviates largely from rest of the distribution, it is labeled as an anomaly or an outlier. In the context of this paper, anomalies and outliers are used interchangeably as stated in [1]. In computer science, anomaly detection refers to the techniques of finding specific data points, that do not conform to the normal distribution of the data set. The most relevant definition of an anomaly with respect to computer science is given by Grubbs [2]: "An outlying observation, or 'outlier', is one that appears to deviate markedly from other members of the sample in which it occurs". The term 'anomaly', is widely used and it refers to different problems in different domains. For example, an anomaly in network security system could be an activity related to a malicious software or a hacking attempt [3]. Whereas, in the manufacturing domain a faulty product is considered as an anomaly. It is very important to detect anomalies as early as possible to avoid big issues like financial system hack, total machine failure, or a cancerous tumor in human body.

Companies from different sectors including manufacturing, automotive, healthcare, lodging, traveling, fashion, food, and logistics are investing a lot of resources [4], [5] in collecting big data and exploring the hidden anomalous patterns in them to facilitate their customers. In most of the cases, the collected data are streaming time series data and due to their intrinsic characteristics of periodicity, trend, seasonality, and irregularity, it is a challenging problem to detect point anomalies precisely in them. Furthermore, in most of real life scenarios, it is practically impossible to label enormous amount of data, therefore, we are using an unsupervised method. Although many unsupervised methods are available, they don't handle the intrinsic characteristics of time series data. For example, traditional distance based anomaly detection techniques do not incorporate context of a time series, due to which they are unable to find point anomalies occurring in cycles. The proposed unsupervised approach incorporates context, seasonality, and trend into account for detecting anomalies. This approach can be adapted for different scenarios and use cases, and works on data from different domains.

This paper presents DeepAnT, a novel unsupervised deep learning based anomaly detection approach for streaming data. This approach doesn't rely on labeling of anomalies rather it leverages the original time series data even without removing anomalies (given that the number of anomalies in the data set is less than 5% [3]). DeepAnT employs CNN as its forecasting module. This module predicts the next time stamp of a given time series window. Subsequently, the forecasted value is passed to a detector module, which compares that value with the actual data point to detect anomalies in real-time. The approach is realistic and suitable even for domains where time series data are collected from heterogeneous sources and sensors. DeepAnT achieves good generalization capabilities in data scarce scenarios where less training data are available. Only a few number of training samples (depending on the data set, e.g. 568 data points from Yahoo data set and 140 data points from Ionosphere data set) are sufficient to build a prediction model due to its effective parameter sharing during feature extraction. DeepAnT when tested on publicly available anomaly detection benchmarks, outperformed the state-of-the-art anomaly detection methods in most of the cases. Instead of classifying whole time series as normal or abnormal (as done in [6]–[9]), DeepAnT's objective is to robustly detect point anomalies. In particular, following are the main contributions of this paper:

1) To the best of our knowledge, DeepAnT is the first deep learning based approach which is capable of detecting point anomalies, contextual anomalies, and discords in time series data in an unsupervised setting.
2) The proposed pipeline is flexible and can be easily adapted for different use cases and domains. It can be applied to uni-variant as well as multi-variant time series.
3) In contrast to the LSTM based approach, CNN based DeepAnT is not data hungry. It is equally applicable to big data as well as small data. We are only using 40% of a given time series to train a model.
4) We gathered different anomaly detection benchmarks at one place and provided extensive evaluation of 15 state-of-the-art methods in different settings on 10 data sets (covering both steaming and non-streaming cases) which contain 433 time series in total. DeepAnT has gained the state-of-the-art performance on most of the data sets.

The rest of the paper is organized as follows. Section II provides an overview of existing methods for anomaly detection. The state-of-the-art anomaly detection methods are mentioned and summarized in Section III, which are evaluated and compared with the proposed technique in Section V. Section IV provides details about the presented approach for anomaly detection in time series data. Section V provides a detailed evaluation of the DeepAnT along with a solid comparison with other state-of-the-art anomaly detection methods on different benchmarks. This section is further divided into sub-sections which elaborates on the details of the used data sets and the experimental settings of the state-of-the-art methods. Finally, Section VI concludes the paper and sketches direction for possible future work.

II. LITERATURE REVIEW OF ANOMALY DETECTION METHODS

Due to the large variety of scenarios and algorithms, anomaly detection problem is categorized in many ways. The most common categorization is based on the level of supervision required by the algorithm; supervised, semi-supervised, and unsupervised. Another categorization, followed by Aggarwal [10], is based on the underlying used methods. Examples of such methods for outlier detection are probabilistic models, statistical models, linear models, proximity based models, and outlier detection in high dimensions. In addition, anomaly detection methods also exist based on different machine learning and deep learning techniques. In this section, an overview of commonly used anomaly detection techniques is provided. First, we talk about anomaly detection techniques which are widely used for point anomalies. Then, an overview of anomaly detection techniques designed for time series data is given. In the end, anomaly detection techniques based on deep neural networks are discussed.

Statistical anomaly detection techniques are most commonly employed to detect anomalies. k-NN anomaly detection method is the simplest and most widely used unsupervised global anomaly detection method for point anomalies. This distance based algorithm calculates the anomaly score based on k-nearest-neighbors distance [11]. This technique is computationally expensive, highly dependent on the value of k, and may fail if normal data points do not have enough neighbors. Breunig *et al.* [12] presented the most widely used unsupervised method for local density-based anomaly detection known as Local Outlier Factor (LOF). In LOF, k-nearest-neighbors set is determined for each instance by computing the distances to all other instances. The basic assumption of this algorithm is that the neighbors of the data instances are distributed in a spherical manner. However, in some application scenarios, where normal data points are distributed in a linearly connected way, the spherical estimation of density becomes inappropriate [3]. Tang *et al.* [13] proposed an improved version of LOF known as Connectivity based Outlier Factor (COF), which improves the linear structure taken into account. A shortcoming of this algorithm is incorrect outlier score estimation in some cases when clusters with different densities are very close to each other. In such cases, instances at the border of the low-density clusters are local outliers with respect to the high density clusters [3]. This shortcoming is further resolved in Influenced Outlierness (INFLO) [14] algorithm.

Other than nearest neighbor based algorithms, clustering based algorithms are also used for unsupervised outlier detection. As name suggests, Cluster-Based Local Outlier Factor (CBLOF) [15] is a clustering based anomaly detection algorithm, in which data points are clustered using k-means (or any other) clustering algorithm. The anomaly score of an instance is the distance to the next large cluster. As this approach is based on clustering algorithm, the problem of choosing the right number of clusters arises, and reproduction of the same anomaly score also becomes impossible due to non-deterministic nature of clustering algorithms.

Histogram-Based Outlier Score (HBOS) [16] is another statistical unsupervised anomaly detection algorithm. This algorithm is computationally far less expensive as compared to nearest neighbor and clustering based anomaly detection methods. HBOS works on arbitrary data by offering a standard fixed bin width histogram as well as dynamic bin width (fixed amount of items in each bin).

Semi-supervised and unsupervised variants of anomaly detection algorithms exist based on One-Class Support Vector Machine (OCSVM). Unsupervised variant of OCSVM was introduced by Amer *et al.* [17]. Based on the idea of [18], no prior training data are required for this technique. It attempts to learn a decision boundary that achieves the maximum separation between the points and the origin. This technique is also used for detecting anomalies in activities of daily life for example sleeping, sitting, and walking patterns [19]. Another time series anomaly detection technique based on OCSVM was proposed by Hu *et al.* [20]. In this technique, six meta-features on actual univariate or multivariant time series are defined first and then OCSVM is applied on meta-feature-based data space to find abnormal states. In general, OCSVM is sensitive to the outliers when there are no labels. It is also used as a novelty detection technique. Liu *et al.* [21] proposed an approach to detect outliers based on Support Vector Data Description (SVDD) [22].

Shyu *et al.* [23] proposed an approach for anomaly detection based on Principle Component Analysis (PCA), where predictive model is constructed from the major and minor principle components of the normal instances. Kwitt and Hofmann [24] proposed another variation of this technique, in which Minimum Covariance Determinant (MCD) is used for calculation of covariance and correlation matrix instead of standard estimators.

To incorporate time series characteristics, there exist different anomaly detection techniques which are designed to find anomalies specifically for streaming time series data. Netflix open-sourced it's anomaly detection function called Robust Anomaly Detection (RAD) in 2015 [25]. The function is based on Robust Principle Component Analysis (RPCA) to detect anomalies. To detect anomalous time series in multi-terabyte data set, a disk aware algorithm is proposed in [26]. Statistical autoregressive-moving-average (ARMA) model and its variations such as ARIMA and ARMAX are used widely for time series prediction and anomaly detection. Yu *et al.* [27] presented an anomaly detection technique for traffic control in wireless sensor networks, which is based on ARIMA model. They proposed that short step exponential weighted average method is the key to make better anomaly detection judgment in the network traffic. In the same domain, Yaacob *et al.* [28] proposed a technique for early warnings detection of Denial-of-Service (DoS) attacks. By comparing actual network traffic with the predicted patterns generated by ARIMA, anomalous behaviors are identified.

Nowadays, Artificial Neural Networks (ANN) have been successfully employed in a wide range of domains, such as hand writing recognition, speech recognition, document analysis, activity recognition, and many more; mainly for classification and prediction purposes. Different ANN architectures have been successfully leveraged for time series analysis. The anomaly detection technique proposed by Malhotra *et al.* [6] is based on stacked LSTMs. Their predictive model is trained on normal time stamps, which is further used to compute error vectors for given sequences. Based on the error threshold, a time series sequence is marked as normal or anomalous. Chauhan and Vig [8] used similar approach to detect anomalies in ECG data. They used RNN, augmented with LSTM, to detect 4 different types of anomalies. Another deep learning based anomaly detection technique was recently proposed by Kanarachos *et al.* [29], in which they combine wavelet and Hilbert transform with deep neural networks. They aim to detect anomalies in time series patterns.

Lipton *et al.* [7] used LSTM to classify a time series as normal or abnormal. They applied their technique on a clinical data set and demonstrated that LSTM trained on only raw time series with target replication outperforms MLP trained on hand engineered features. Zheng *et al.* [30] used CNN for multivariate time series classification. They proposed Multi-Channel Deep CNN (MC-DCNN) where each channel takes a single dimension of multivariate time series data as input and learns the features individually. This is followed by a layer of MLP to perform classification. Experimental results show that MC-DCNN outperforms competing baseline methods which are K-nearest neighbor (based on Euclidean Distance and Dynamic Time Wrapping). All of the aforementioned deep learning based time series anomaly detection techniques are used for classifying a sequence or a subsequence as normal or abnormal.

Autoencoder is a type of neural network which is trained to reproduce its input. Typically, autoencoders are used for dimensionality reduction which helps in classification and visualization tasks. Due to its efficient data encoding in an unsupervised manner, it is also gaining popularity for anomaly and novelty detection problems. Amarbayasgalan *et al.* [31] proposed a novelty detection technique based on deep autoencoders. Their approach gets compressed data and error threshold from deep autoencoders and apply density-based clustering on the compressed data to get novelty groups with low density. Schreyer *et al.* [32] also used deep autoencoders to detect anomalies in large-scale accounting data in the area of fraud detection.

III. THE STATE-OF-THE-ART METHODS USED FOR COMPARISON

This section summarizes the state-of-the-art methods used for comparison with the proposed approach. Twitter Inc. open-sourced it's anomaly detection[1] package in 2015, which is based on Seasonal Hybrid ESD (S-H-ESD) algorithm [33]. This technique is based on Generalized Extreme Studentized Deviate (ESD) test [34] to handle more than one outliers, and Seasonal and Trend Decomposition using Loess (STL) [35] to deal with the decomposition of time series data and seasonality trends. Twitter Anomaly Detection can detect both global and local anomalies. They have provided two anomaly detection functions for detecting anomalies in seasonal univariate time series:

(i) ***AnomalyDetectionTS*** function is used when input is a series of $< timestamp, value >$ pairs.
(ii) ***AnomalyDetectionVec*** function is used when input is a series of observations.

Another anomaly detection method, EGADS [36], which detects anomalies in large scale time series data was released by Yahoo Labs.[2] EGADS (Extensible Generic Anomaly Detection System) consists of two main components: Time series Modeling Module (TMM) and Anomaly Detection Module (ADM). For a given time series, TMM models the time series and produces an expected value at time stamp t. ADM compares the expected value with the actual value and computes number of errors E. Automatic threshold is determined on E and most probable anomalies are given as output. There are seven time series models which are supported by TMM and three anomaly detection models.

[1]Source code of Twitter Anomaly Detection: https://github.com/twitter/AnomalyDetection/releases

[2]EGADS Java Library: https://github.com/yahoo/egads

ContextOSE [37] is based on Contextual Anomaly Detection (CAD) method. As name indicates, CAD is based on the contextual/local information of time series instead of global information. This unsupervised approach takes a set of similar time series and a window size. First, a subset of time series is selected and then centroid of the selected time series is calculated. The centroid values are further used along with other time series features to predict the values of time series.

Numenta and NumentaTM [38], [39] are two variants of Numenta's anomaly detection method based on Hierarchical Temporal Memory (HTM). These techniques model the temporal sequences in a given data stream. At a given time t, HTM makes multiple predictions for next time-stamp. These predictions are further compared with actual value to determine if a value is normal or anomalous. For each time stamp, anomaly likelihood score is calculated which is thresholded to finally reach a conclusion regarding the presence or absence of anomaly.

Skyline [40] is a real-time anomaly detection method developed by Etsy, Inc. This method ensembles votes from different expert approaches. They make the use of different simple detectors which vote to calculate the final anomaly score.

Isolation Forest (iForest) [41] is a model based anomaly detection technique, which is built on the idea of random trees. Here, 'isolation' means separating an anomalous instance from the rest of the instances. iForest isolates instances by random partitioning of a tree followed by random selection of the features. This random partitioning produces shorter paths for anomalies. The path length from the root node to the terminating node is averaged over a forest of random trees.

Twitter anomaly detection method is specifically designed to detect seasonal anomalies in the context of social network data. This technique performs good when anomalies arise in periodic data which are not much different from the previous data. But, it struggles in finding anomalies when a time series trend is changing over time. Availability of different time series models makes EGADS a good candidate for a general purpose anomaly detection method. This method is capable of adapting itself to different use-cases and its parallel architecture enables the detection of anomalies in real-time anomaly. ContextOSE leverages the contextual information which is very important to detect time series anomalies. NumentaTM is capable of detecting spatial and temporal anomalies as it is based on an online sequence memory algorithm. The results provided in their study are based only on NAB score. This score is designed to evaluate the early detection of anomalies and cannot be directly used for point anomalies comparison.

IV. DeepAnT: THE PROPOSED APPROACH FOR ANOMALY DETECTION IN TIME SERIES

The proposed DeepAnT consists of two modules. The first module, **Time Series Predictor** predicts time stamps for a given horizon and the second module, **Anomaly Detector** is responsible for tagging the given time series data points as normal or abnormal. Deep learning has been employed in a wide range of applications primarily because of its capability to automatically discover complex features without having any domain knowledge. This automatic feature learning capability makes the neural network a good candidate for time series anomaly detection problem. Therefore, DeepAnT employs CNN and makes use of raw data. Also, it is robust to variations as compared to other neural networks and statistical models. It is shown in literature [42], [43] that LSTM performs well on temporal data due to its capability to extract long-term trends in the encountered time series. However, we have shown in this study that CNN can be a good alternate for uni-variate as well as multi-variate time series data due to its parameter efficiency. Generally, CNN and LSTM are used for time series classification problem in literature [7], [30], but we are using CNN (and LSTM for a comparison) for a time series regression problem.

A. TIME SERIES PREDICTOR

The predictor module of DeepAnT is based on CNN. CNN is a type of artificial neural network which has been widely used in different domains like computer vision and natural language processing in a range of different capacities due to its parameter efficiency. As the name indicates, this network employs a mathematical operation called *convolution*. Normally, CNN consists of sequence of layers which includes convolutional layers, pooling layers, and fully connected layers. Each convolutional layer typically has two stages. In the first stage, the layer performs the convolution operation which results in linear activations. In the next stage, a non-linear activation function is applied on each linear activation. In simplest form, convolution is a mathematical operation on two functions of real valued arguments to produce a third function. The convolution operation is normally denoted as asterisk:

$$s(t) = (x * w)(t) \tag{1}$$

This new function s can be described as a smoothed estimate or a weighted average of the function $x(\tau)$ at the time-stamp t, where weighting is given by $w(-\tau)$ shifted by amount t. In (1), function x is referred to as the input and function w is referred to as the kernel. The output is referred to as the feature map. One dimensional convolution is defined as:

$$s(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau) \tag{2}$$

In DeepAnt, similar to other well-known methods [44], [45], the output of a convolutional layer is further modified by a pooling function in a pooling layer. A pooling function statistically summarizes the output of the convolutional layer at a certain location based on its neighbors. Most commonly used *max-pooling* operation is used in DeepAnT which outputs the maximum activation in a defined neighborhood. Since there are more than one feature maps, individually the pooling function is applied on all of these feature maps.

After pair of convolutional and max-pooling layer, the final layer of connections in DeepAnT is a fully connected layer. In this layer, each neuron from a previous layer is connected to all output neurons. The activation for convolutional and fully connected layers is given in (4) and (6) respectively where k is defined as $\lfloor FilterSize/2 \rfloor$.

$$z_{ji}^{l} = \sum_{-k}^{k} W_{jk}^{l} a_{i-k}^{l-1} + b_{j}^{l} \tag{3}$$

$$a_{ji}^{l} = max\left(z_{ji}^{l}, 0\right) \tag{4}$$

$$z_{j}^{l} = \sum_{k=1}^{e} W_{jk}^{l} a_{k}^{l-1} + b_{j}^{l} \tag{5}$$

$$a_{j}^{l} = max\left(z_{j}^{l}, 0\right) \tag{6}$$

In (4), a_{ji}^{l} refers to the activation of the j^{th} neuron in the l^{th} layer at the i^{th} input location of a convolutional layer. Whereas, a_{j}^{l} refers to the activation of the j^{th} neuron in the l^{th} fully connected layer in (6).

Like other artificial neural networks, a CNN uses training data to adapt its parameters (weights and biases) to perform the desired task. In DeepAnT, parameters of the network are optimized using Stochastic Gradient Descent (SGD). The idea of training or learning of a neural network is to reduce a cost function C. In this predictor module, the cost function computes the difference between the network's predictions and the desired prediction. In the learning process, that difference is minimized by adapting the weights and biases of the network. The process of calculating the gradient, which is required to adjust the weights and biases, is called backpropagation. It is obtained by calculating the partial derivatives of the cost function with respect to any weight w or bias b as $\partial C/\partial w$ and $\partial C/\partial b$ respectively. Network weights are updated by SGD.

In order to leverage CNN for forecasting, time series data need to be changed in a compatible form for the system to operate on them. For each element x_t at time stamp t in a time series, next element x_{t+1} at time stamp $t+1$ is used as its label. Input data are transformed into several sequences of overlapping windows of size w. This window size defines the number of time stamps in history, which are taken into account (referred as a *history window*). It also serves as the context to x_t. The number of time stamps required to be predicted is referred to as *prediction window* (p_w). In some studies, prediction window is also called as (Forecasting) Horizon [46], [47].
Consider a time series:

$$\{x_0, x_1, \ldots, x_{t-1}, x_t, x_{t+1}, \ldots\}$$

For $w = 5$ and $p\_w = 1$, the sequence at index t will be as follow:

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

In a regression problem as ours, the left hand side is treated as input data and right hand side is treated as label. In this case,

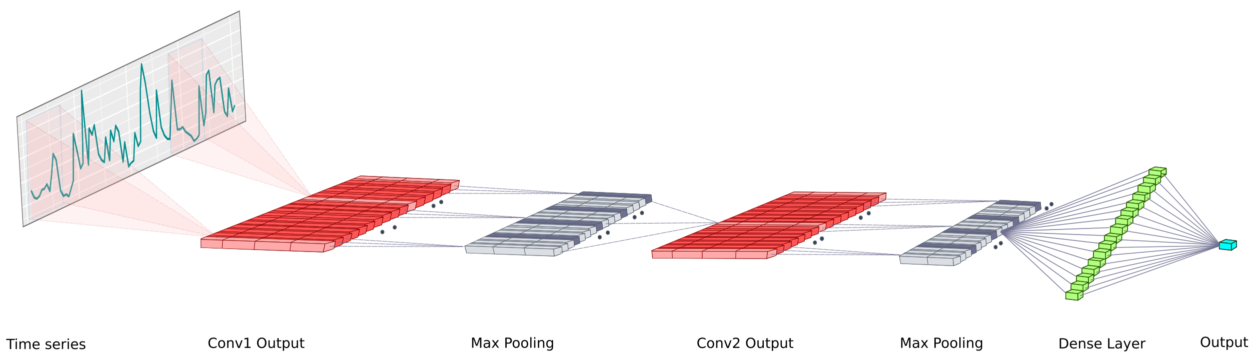

FIGURE 1. DeepAnT architecture for time series prediction: A convolutional neural network with two convolutional layers, two max pooling, and a fully connected layer.

it can be called as *many_to_one prediction*. When $p\_w > 1$, it can be called as *many_to_many prediction*.

1) ARCHITECTURE SUMMARY

We did extensive experiments to finalize the architecture and its hyperparameters. Two convolutional layers, each followed by a max-pooling layer, are used in this architecture as shown in Fig. 1. The input layer has w input nodes as we have converted the data into w vectors. Each convolution layer is composed of 32 filters (kernels) followed by an element-wise activation function, ReLU as given in (7). Last layer of the network is a fully connected (FC) layer in which each neuron is connected to all the neurons in the previous layer. This layer represents the network prediction for the next time stamp. The number of nodes used in the output layer are equal to p_w. In our case, we are predicting only the next time stamp, so the number of output node is 1. In later sections of this paper, when we are predicting a sequence instead of a single data point, the number of nodes in output layer is changed accordingly.

$$f(x) = max(0, x) \tag{7}$$

2) LOSS FUNCTION

Mean Absolute Error (MAE), given in (8) has been employed as an indicator of the discrepancy between the actual y_j and the predicted $\hat{y}_j$ output. By reducing the error between the actual and the predicted value, the network can learn to predict the normal behavior of the time series. We normalized each time series based on the training data.

$$MAE = \frac{1}{n}\sum_{j=1}^{n}\left|y_j - \hat{y}_j\right| \tag{8}$$

B. ANOMALY DETECTOR

Once the prediction of the next time stamp x_{t+1} is made by the *Time Series Predictor*, this module detects anomalies in a given time series. The value predicted by the predictor module is passed to this module and the difference between actual and predicted value is calculated. Euclidean distance given in (9) is used as a measure of the discrepancy.

$$(y_t, y'_t) = \sqrt{(y_t - y'_t)^2} \tag{9}$$

where y_t is actual value and y'_t is predicted value.

The Euclidean distance is used as anomaly score. A large anomaly score indicates a significant anomaly at the given time stamp. A threshold, based on the time series type needs to be defined for this module, which is required in most of the anomaly detection algorithms.

V. EXPERIMENTAL SETUPS

We have evaluated DeepAnT on 10 different data sets (433 time series) and provided a detailed comparison with 15 anomaly detection methods which include several state-of-the-art methods. Both synthetic and real time series data from different domains are used for experiments. We divide our experimental setup into several parts, because different anomaly detection methods in literature are evaluated on different benchmarks based on different metrics. The division of this section is based on Yahoo, NAB, classic anomaly detection benchmark, and NASA space shuttle valve data sets respectively. Detailed description of each benchmark and its evaluation setup are also provided in this section.

A. EXPERIMENTAL SETTING I: YAHOO DATA SET

1) DATA SET DESCRIPTION

Yahoo Webscope[3] data set is a publicly available data set released by Yahoo Labs. This data set consists of 367 real and synthetic time series with point anomaly labels. Each time series contains 1,420 - 1,680 instances. This anomaly detection benchmark is further divided into four sub-benchmarks namely *A*1 *Benchmark*, *A*2 *Benchmark*, *A*3 *Benchmark*, and *A*4 *Benchmark*.

*A*1 *Benchmark* contains real Yahoo membership login data, which tracks the aggregate status of logins on Yahoo network [36], whereas, other three sub-benchmarks contain synthetic data. *A*2*Benchmark* and *A*3*Benchmark* contain only outliers, while *A*4*Benchmark* also contains

[3] ydata-labeled-time-series-anomalies-v1_0: https://webscope.sandbox.yahoo.com/catalog.php?datatype=s&did=70

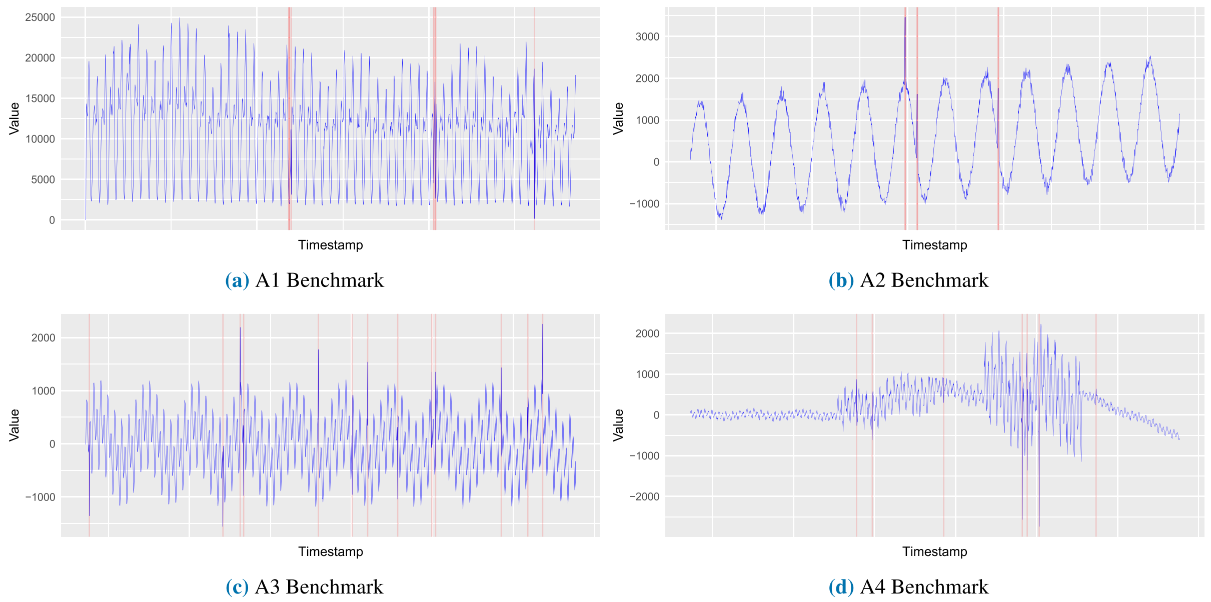

FIGURE 2. Sample time series from each Yahoo sub-benchmark are shown in this figure. Actual streaming data are shown in blue, whereas red vertical lines highlight anomalous data points based on the provided labels. Plots (a) and (b) have random seasonality, trend, and noise, whereas, plots (c) and (d) have trends with three pre-specified seasonalities.

change-point anomalies. In synthetic data, outliers are present on random positions. In each data file, there is a Boolean attribute – label – indicating if the value at a particular time stamp is considered as anomalous or normal. In addition to value and label, *A3Benchmark* and *A4Benchmark* contain additional fields such as change-point, trend, noise, and seasonality. However, we are discarding all the additional attributes and only using *value* attribute for all the experiments. Fig. 2 shows time series samples from the four sub-benchmarks. Actual data streams are shown in blue color and anomalous data points are highlighted by red vertical line. The main reason for selecting this data set for evaluation is the availability of the point anomalies labels, which are not commonly available in publicly available streaming data sets (available at [48]).

2) EVALUATION METRIC AND EXPERIMENTAL SETUP

F-score is most commonly used singleton metric which serves as an indicator of the model's performance. Therefore, we employed F-score (Eq. 10) as the evaluation metric for our models. All the anomaly detection methods in this experimental setting are applied on each time series separately. Average F-scores per sub-benchmark are reported for each method.

$$F\text{-}score = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{10}$$

We run all the algorithms on same machine having Intel Xeon(R) processor with 8 cores and NVIDIA GeForce GTX 1070. It took on average 0.076 seconds to get anomaly detection results on a given Yahoo Webscope time series which have 852 - 1008 instances in test data.

a: DeepAnT PARAMETERS

We are using only 40% of each time series as training set and rest of the 60% data as test set. We further split the training set and use 10% of it for validation. Since this is an unsupervised approach, we don't use any label information in training process. For each time series, only next time stamp is predicted and marked as either normal or anomalous data point. To compare the performance of CNN with LSTM in the context of anomaly detection, we have also used LSTM in DeepAnT's *Time Series Predictor* module. We used the same 40% training data scheme for training LSTM as we did for CNN. For LSTM based model, we used two LSTM layers (as done in [6]) of 32 memory cells each. For both techniques, we used same w for a sub-benchmark.

Finding the best threshold is very important for evaluation. Normally, each time series has its own characteristics, and finding a generic threshold which works for all of the time series is not a straightforward task. Since each Yahoo sub-benchmark shares common properties, therefore, we searched for the best threshold for each sub-benchmark based on the validation data. To get an automatic threshold on a single time series (a) parametric approach – $K\ \sigma$ *deviation* and (b) non-parametric approach – *density distribution* can be used as explained in [36]

History Window (w) is another hyperparameter that plays a vital role in improving the prediction model of DeepAnT. Again, there is no generic fixed window size which can be used for all of the time series. For the purpose of reproducibility, we list the combination of thresholds and window size yielding the best F-score in Table 1. We shortlisted the window sizes of 25, 35, and 45 after hyperparameter optimization. One can use grid search to find the best window size for new time series. The knowledge of period size in a

TABLE 1. This table shows the selected history window and thresholds which are used to evaluate DeepAnT on Yahoo data set.

Sub-benchmark	Threshold	History Window
A1	0.50	45
A2	0.75	45
A3	0.65	35
A4	0.55	35

particular time series can be a good starting point for the grid search to find the optimal window size. Fig. 3 shows the effect of w on average F-score in each sub-benchmark. These plots also show the importance of selecting the right number of w.

b: TWITTER ANOMALY DETECTION PARAMETERS

We used *AnomalyDetectionTS* function provided in Twitter anomaly detection for *A2Benchmark*, *A3Benchmark*, and *A4Benchmark*. For *A1Benchmark*, we used *AnomalyDetectionVec* function, because time stamps are replaced by integers with an increment of 1 in this data set by the publisher. We used all default parameters of this method except the following two:

(i) ***Alpha:*** This parameter defines the level of statistical significance with which to accept or reject anomalies. We used three values for this parameter i.e. 0.05, 0.1, and 0.2.

(ii) ***Direction:*** This parameter defines the directionality (positive or negative) of anomalies to be detected. We used 'both', as anomalies can be in any direction in this data set.

c: YAHOO EGADS PARAMETERS

We used *Olympic Model* in TMM and *EGADS ExtremeLowDensityModel Outlier* in ADM. Default values of all the other parameters are used. Both Twitter anomaly detection and EGADS calculate threshold themselves for each time series and give time stamps or indexes (if input data do not contain time stamp) of the anomalous data points as output. To evaluate these two methods, we used the same 60% test data of each time series which is used to evaluate DeepAnT.

3) RESULTS

DeepAnT anomaly detection results on a single time series are shown in Fig. 4. In this figure, the actual series is depicted in blue, the predictions on training data are depicted in yellow (not used in reported results) and the predictions on test data

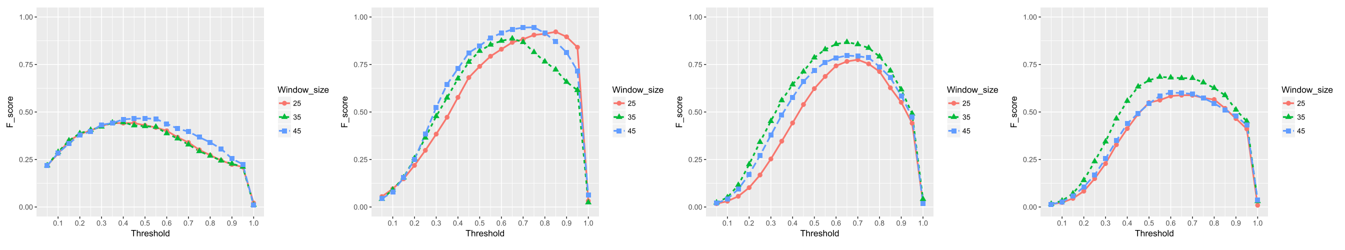

FIGURE 3. Average F-score of each Yahoo sub-benchmark is plotted per history window used in DeepAnT. Plots of three shortlisted windows per sub-benchmark are shown. For *A1Benchmark* and *A2Benchmark*, $w = 45$ provides better average F-score, but for *A3Benchmark* and *A4Benchmark*, $w = 35$ performs good. (1) *A1Benchmark*. (b) *A2Benchmark*. (c) *A3Benchmark*. (d) *A4Benchmark*.

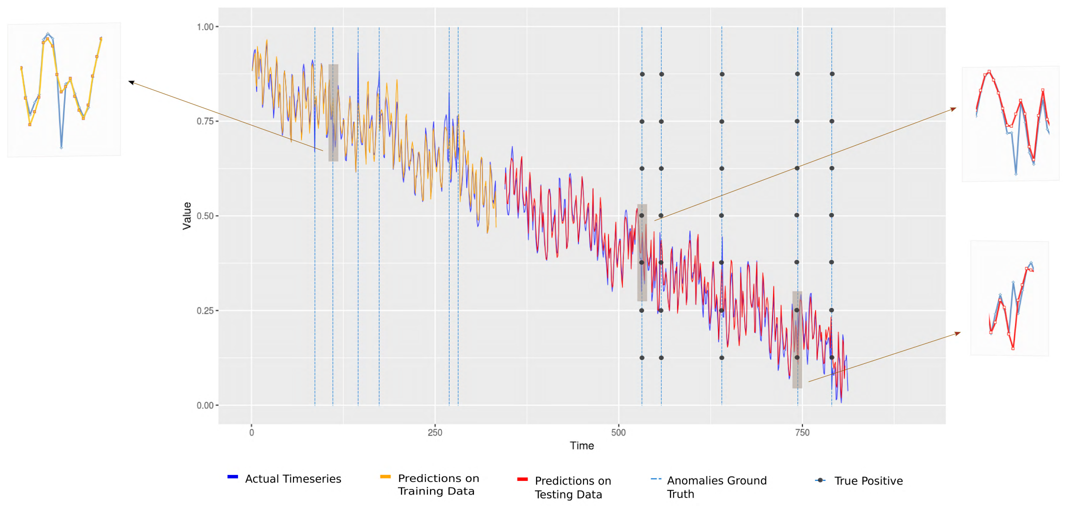

FIGURE 4. An example of time series prediction and anomaly detection using DeepAnT is shown in this figure. Actual time series is shown in blue. Predictions on training data are shown in yellow and predictions on test data are shown in red. Vertical blue lines are anomalies ground truth, and vertical blue lines with dotes on it show point anomalies detected by DeepAnT (true positive). DeepAnT F-score is 1 for this time series, whereas, EGADS and Twitter anomaly detection F-score is 0.

TABLE 2. Average F-Score of Twitter Anomaly Detection, Yahoo EGADS, DeepAnT, and LSTM (DeepAnT using LSTM as time series predictor) on Yahoo data set is given in this table. Bold F-scores are the best scores for corresponding Yahoo sub-benchmark.

Sub-benchmark	Yahoo EGADS	Twitter Anomaly Detection Alpha = 0.05	Twitter Anomaly Detection Alpha = 0.1	Twitter Anomaly Detection Alpha = 0.2	DeepAnT	LSTM
A1	0.47	**0.48**	**0.48**	0.47	0.46	0.44
A2	0.58	0	0	0	0.94	**0.97**
A3	0.48	0.26	0.27	0.30	**0.87**	0.72
A4	0.29	0.31	0.33	0.34	**0.68**	0.59

are depicted in red. Vertical blue lines are anomalies ground truth in training and testing data. Whereas, blue vertical lines with dotes show point anomalies detected by DeepAnT (true positive). It can be seen in this example that there are anomalies in training data, but the network correctly captured those data generating distribution disregarding the anomalies. Predicted data points (red) are super imposed on the actual time series in order to highlight the generalization capabilities of our model. The observed time series is a combination of periodicity and a trend. In such cases, anomaly is not just a spike which is clearly distinguishable, but it can be a data point which is locally deviated from the actual cycle. These local deviations are hard to detect robustly. A couple of such anomalies are magnified in Fig. 4. It can also be seen in this figure that w data points are missing from the beginning of training and testing data sets. In both of the cases, this is the starting sequence (history window) after which the predictions are made.

On a detailed level, Table 2 shows a comparison of DeepAnT with EGADS, Twitter, and LSTM (DeepAnT using LSTM as a predictor) on the whole data set. DeepAnT outperforms other methods in two sub-benchmarks and for the rest, it is runner up. $A1Benchmark$ consists of anomalies where there is no trend and seasonality effect. Mostly, the anomalies are just the spikes in $A1Benchmark$. Since we are computing F-score based on the sub-benchmark level threshold, therefore, DeepAnT is not on top. Whereas, other methods are computing threshold separately for each time series. For $A3Benchmark$ and $A4Benchmark$, F-scores for DeepAnT are significantly better than other methods. Twitter anomaly detection didn't work at all on $A2Benchmark$. This table also shows that the parameter 'Alpha' does not have a significant impact in this case. It is also important to note in this table that CNN based DeepAnT performs better than LSTM on three sub-benchmarks and performs slightly poor in one sub-benchmark. It shows that CNN could be used in the cases when only limited amount of training data is available.

As DeepAnT's *Anomaly Detector* module is dependent on the *Time Series Predictor* module, so good forecasting performance results in better anomalous points detection. Fig. 5 shows a plot of actual values (ground truth) vs. predicted values of a time series. Ideally, it should be a smooth diagonal line because actual and predicted values should be same or close to each other. But, in practice, it is an uneven diagonal line due to minor errors in the prediction model. When there is a difference between the actual and the predicted value,

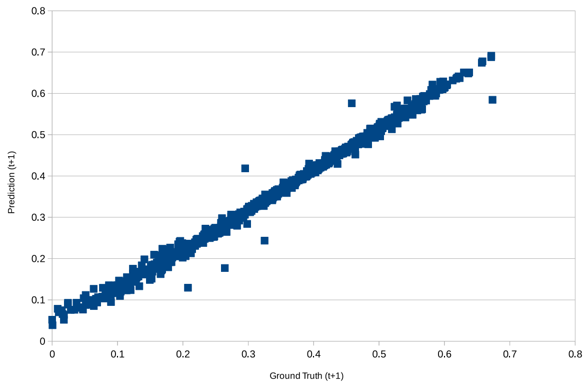

FIGURE 5. For a single time series, actual time series values are plotted against the time series predictions to show the accuracy of prediction model. When actual and predicted values are same (or close to each other), they form a diagonal line. Whereas, when actual and predicted values are not same (or not close to each other), they end up away from the diagonal line – which represent anomalies in the observed time series.

then the data point lies away from the diagonal line – showing anomaly at a particular time stamp.

B. EXPERIMENTAL SETTING II: NAB DATA SET

1) DATA SET DESCRIPTION

NAB (Numenta Anomaly Benchmark) [38] is a publicly available streaming anomaly detection benchmark, released by Numenta.[4] This data set consists of 58 data streams, each with $1,000$ - $22,000$ instances. This data set contains streaming data from different domains including road traffic, network utilization, on-line advertisement, and internet traffic. The data set is labeled either based on the known root cause of an anomaly or as a result of following the defined labeling procedure (described in [38]). Each data file consists of time stamps and actual data values. Anomaly labels of each data file are given in a separate set of files.

Although NAB provides a diverse labeled streaming anomaly detection data set, there are a few challenges [49] which make it hard to be used as a practical anomaly detection benchmark. Each data point with ground truth anomaly label is centered by a defined anomaly window (10% the length of a data file), which makes the ground truth label of normal data points also as anomalous. For example, for an anomaly window of size 350, all of the 350 data points in a data stream are labeled anomalous, whereas there are just 2-3 actual anomalies in the middle of this anomaly window. This kind

[4]https://numenta.org/

TABLE 3. Comparative evaluation of different state-of-the-art algorithms and the proposed algorithm on 20 NAB time series from different domains. Precision and recall are reported in this table.

Time series		ContextOSE		Numenta		NumentaTM		Skyline		ADVec		DeepAnT	
		Pre.	Rec.	Pre.	Rec.	Pre.	Rec.	Pre.	Rec.	Pre.	Rec.	Pre.	Rec.
Real Ad Exchange	exchange-2-cpc-results	0.5	0.006	0	0	0	0	0	0	0	0	0.03	0.33
	exchange-3-cpc-results	0.75	0.02	1	0.013	1	0.007	0	0	1	0.02	0.71	0.03
Real AWS Cloud Watch	ec2-cpu-utilization-5f5533	1	0.005	1	0.007	1	0.01	1	0.002	1	0.002	1	0.01
	rds-cpu-utilization-cc0c53	1	0.005	1	0.002	1	0.002	1	0.1	0.62	0.012	1	0.03
Real Known Cause	ambient-temperature-system-failure	0.33	0.001	0.42	0.004	0.5	0.006	0	0	0	0	0.26	0.06
	cpu-utilization-asg-misconfiguratio	0.12	0.001	0.64	0.018	0.52	0.01	0	0	0.74	0.01	0.63	0.36
	ec2-request-latency-system-failure	1	0.009	1	0.009	1	0.009	1	0.014	1	0.02	1	0.04
	machine-temperature-system-failure	1	0.001	0.58	0.004	0.27	0.004	0.97	0.01	1	0.02	0.8	0.001
	nyc-taxi	1	0.002	0.77	0.007	0.85	0.006	0	0	0	0	1	0.002
	rogue-agent-key-hold	0.33	0.005	1	0.005	0.5	0.005	0	0	0	0	0.34	0.05
	rogue-agent-key-updown	0	0	0.14	0.002	0	0	0	0	0.11	0.002	0.11	0.001
Real Traffic	occupancy-6005	0.5	0.004	0.33	0.004	0.2	0.004	0.5	0.004	0.5	0.004	0.5	0.004
	occupancy-t4013	1	0.008	0.4	0.008	0.66	0.008	1	0.04	1	0.02	1	0.036
	speed-6005	0.5	0.004	0.66	0.008	0.25	0.008	1	0.01	1	0.01	1	0.008
	speed-7578	0.57	0.03	0.66	0.03	0.6	0.02	0.86	0.16	1	0.01	1	0.07
	speed-t4013	1	0.008	1	0.01	0.8	0.01	1	0.06	1	0.01	1	0.08
	TravelTime-387	0.6	0.01	0.25	0.004	0.33	0.004	0.62	0.07	0.2	0.004	1	0.004
	TravelTime-451	1	0.005	1	0.005	0	0	0	0	0	0	1	0.009
Real Tweets	Twitter-volume-GOOG	0.75	0.002	0.36	0.003	0.38	0.005	0.59	0.02	0.81	0.01	0.75	0.01
	Twitter-volume-IBM	0.37	0.002	0.15	0.002	0.22	0.005	0.22	0.01	0.5	0.009	0.5	0.005

of labeling helps in calculating good NAB score and leaves the recall very low. NAB score is introduced in [38] as an anomaly detection score which is designed to reward early anomaly detections and penalize later detections based on the true and false detections within an anomaly window.

2) EXPERIMENTAL SETUP AND EVALUATION METRIC

A high NAB score shows that a particular algorithm has a higher tendency to detect early anomalies. However, it does not show how good that algorithm is in terms of true detections of anomalies and false alarms. In real life scenarios, in addition to early anomaly detection, it is equally important to detect correct number of anomalies. It is shown in [49] that for some cases NAB score is high, but the precision and recall is low, which means that the algorithm was not able to detect maximum number of anomalies. Two levels of same experiment are shown in this section. On the first level, we applied five time series anomaly detection algorithms, in addition to DeepAnT, on 20 NAB time series from different domains. We picked same time series as mentioned in [49]. The algorithms are evaluated on the basis of precision and recall. On the second level of this experiment, we have done the detailed analysis of 11 algorithms and compared them with DeepAnT on whole NAB benchmark. The evaluated algorithms include Twitter's Anomaly Detection (Twitter ADVec), context OSE, Skyline, Numenta, Multinomial Relative Entropy [50], Bayes changepoint detection [51], EXPoSE [52], and simple sliding threshold. All of these algorithms are used in same settings and with same parameters as mentioned in [39], as they have done extensive parameter tuning for each algorithm and used the optimal parameters. We have used F-score for this detailed evaluation so that an overall performance of an algorithm can be reported. We are not reporting NAB score here because we want to evaluate an algorithm on the basis of the number of detected and rejected anomalies and the other arguments made in [49]. Since NAB benchmark consists of multiple time series from different domains, therefore, we have reported the mean F-score per domain.

3) RESULTS

Table 3 shows results of the first level of our NAB experiment. In most of the cases, high precision is followed by low recall. The main reason of such low recall is the labeling mechanism used in the NAB data set. It can be observed in this table that each algorithm is capable of achieving the precision close to 1, but recall stays in between $0.001 - 0.36$. In such cases, algorithms detect $1-4$ anomalies out of $346-401$ anomalies. Whereas, DeepAnT gives relatively better recall with equivalent precision as other algorithms (e.g., ec2-request-latency-system-failure, speed-t4013). Table 4 shows mean F-scores of a wide range of algorithms on the whole NAB data set (results of second level). It can be noted here that DeepAnT outperforms other algorithms by significant margin. DeepAnT is $2 - 13$ times better than the best performing algorithm for different domains in the NAB data set.

TABLE 4. Comparative evaluation of the state-of-the-art anomaly detection methods on the NAB data set. Mean F-score is reported for each domain as each domain contains different number of time series. DeepAnT out-performs all the other methods on whole data set (best mean F-score in bold).

	Bayes Changepoint	Context OSE	EXPoSE	HTM JAVA	KNN CAD	Numenta	NumentaTM	Relative Entropy	Skyline	Twitter ADVec	WindowedGaussian	DeepAnT
Artificial no Anomaly	0	0	0	0	0	0	0	0	0	0	0	0
Artificial with Anomaly	0.009	0.004	0.004	0.017	0.003	0.012	0.017	0.021	0.043	0.017	0.013	**0.156**
Real Ad Exchange	0.018	0.022	0.005	0.034	0.024	0.040	0.035	0.024	0.005	0.018	0.026	**0.132**
Real AWS Cloud Watch	0.006	0.007	0.015	0.018	0.006	0.017	0.018	0.018	0.053	0.013	0.060	**0.146**
Real Known Cause	0.007	0.005	0.005	0.013	0.008	0.015	0.012	0.013	0.008	0.017	0.006	**0.200**
Real Traffic	0.012	0.02	0.011	0.032	0.013	0.033	0.036	0.033	0.091	0.020	0.045	**0.223**
Real Tweets	0.003	0.003	0.003	0.010	0.004	0.009	0.010	0.006	0.035	0.018	0.026	**0.075**

TABLE 5. Data properties of the anomaly detection benchmarks.

	No. of Instances	No. of Features	Anomaly Class	Anomaly Percentage
Shuttle	49097	9	classes $\neq$ 1 (class 4 removed)	7%
Pima	768	8	pos	34.9%
ForestCover	286048	10	class 4 (vs. class 2)	0.96%
Ionosphere	351	32	bad	36%
Http	567497	3	attack	0.39%
Smtp	95156	3	attack	0.03%
Mulcross	262144	4	2 clusters	10%
Mammography	11183	6	class = 1	2%

C. EXPERIMENTAL SETTING III: CLASSIC ANOMALY DETECTION BENCHMARK

1) DATA SET DESCRIPTION

In this section, we have used 7 natural and 1 synthetic data set which are most commonly used in classic anomaly detection setting. These multi-variant data sets are available at UCI Machine Learning Repository [53] and OpenML.[5] Known anomaly cases are marked as ground truth in these data sets. We have removed all non-continuous attributes as done in [41] and [54]. A brief description of each data set is given below:

1) ***Shuttle*** is NASA's shuttle data set which is already divided into train and test set by the publisher. As in [41], we have removed all the data instances which belong to class 4. Rest of the classes except class 1, are treated as anomalies.
2) ***Pima*** is a diabetes data set collected at the National Institute of Diabetes and Digestive and Kidney Diseases, USA. This diagnostic data shows if a patient has signs of diabetes or not. The target value 'pos' indicates that a patient is suffering from diabetes and the corresponding data point is treated as anomalous.
3) ***ForestCover*** data set (also known as Covertype in UCI repository) has target values in integers, which are different tree species. The data set is comprised of 54 features in total, where 44 features are categorical. Therefore, we only use 10 non-categorical features for training our model. Out of the 7 target classes, we use 2 classes as done in [41]. All the instances from class 4 are considered anomalous, while instances from class 2 are considered normal.
4) ***Ionosphere*** is a radar data set. The target attribute is ionosphere, which is considered as 'good' if radar shows evidence of some type of structure in the ionosphere, otherwise it is considered as 'bad'. 'Bad' ionospheres are considered as anomalous.
5) ***HTTP*** is a subset of KDD CUP '99 network intrusion data. A wide variety of anomalies (i.e. network attack) were hand-injected in the normal network data. This data set is used in a lot of studies. We have used this data set in the standard way described in [55].
6) ***SMTP*** is also a subset of KDD CUP '99 network intrusion data. This data set is also used as described in [55].
7) ***Mulcross*** data set is obtained from a synthetic data generator known as Mulcross [56]. It generates a multivariate normal distribution with a selectable number of anomaly clusters. We have used same settings (contamination ratio, distance factor, and anomaly clusters) for this data set as mentioned in [41].
8) ***Mammography*** data set is publicly available at *OpenML* and has 6 features. All the data instances with class 1 are considered anomalous.

Properties of these data sets are shown in Table 5. The number of features and anomaly percentage varies significantly between these data sets. The target class (anomaly) of each data set is also mentioned in this table.

2) EVALUATION METRIC AND EXPERIMENTAL SETUP

For the evaluation of different anomaly detection algorithms and DeepAnT, AUC measure has been utilized. AUC is used

[5]https://www.openml.org/u/3768/data

most commonly for reporting results of anomaly detection techniques for mentioned data sets. The evaluation is done in a semi-supervised fashion. In a semi-supervised setting (also known as novelty detection [54]), training data consist of only normal data. In this setting, all the anomalies from the training set are removed in a pre-processing step. We compare the results of three state-of-the-art anomaly detection methods with DeepAnT on the aforementioned data sets. For model based methods (iForest, OCSVM, and DeepAnT), 40% of the actual data are used for training and rest for testing. To train iForest model, we have used the default parameters i.e. $\psi = 128$ and $t = 100$, as suggested in [41]. For OCSVM, we have used RBF (Radial Basis Function) kernel. Commonly used setting of $k = 10$ is applied for LOF. For DeepAnT, history window of 2 is used, with the rest of the parameters being intact.

TABLE 6. Comparison of the state-of-the-art anomaly detection methods in semi-supervised (novelty detection) setting. DeepAnT performs best in most of the cases (best AUC in bold).

	iForest	OCSVM	LOF	DeepAnT
Shuttle	0.98	**0.99**	0.56	**0.99**
Pima	0.40	0.26	**0.48**	0.31
ForestCover	0.78	0.70	0.57	**0.85**
Ionosphere	0.82	0.84	0.83	**0.85**
Http	0.98	**0.99**	0.42	**0.99**
Smtp	**0.80**	0.67	0.41	0.75
Mulcross	**0.99**	**0.99**	0.59	**0.99**
Mammography	0.85	0.89	0.72	**0.99**

3) RESULTS

Evaluation results of semi-supervised or novelty detection setting are shown in Table 6. DeepAnT shows best AUCs for most of the used data sets. In novelty detection setting, OCSVM is considered the best method, however, DeepAnT outperforms it in most of the data sets. These results show that DeepAnT is capable of finding anomalies in multi-variant data set too.

D. EXPERIMENTAL SETTING IV: DISCORD DETECTION

In the previous sub-sections, we have shown that DeepAnT has the capability of detecting point anomalies as well as contextual anomalies in streaming and non-streaming data. In this section, we show that DeepAnT is also applicable to time series discord detection. Time series discords are subsequences of a longer time series, which are different from rest of the subsequences [57]. Discords are considered as the anomalous sequences in a time series. For this experiment, we have picked NASA space shuttle valve data set [58]. The time series in this data set are current measurements on a *Marotta MPV-41* series valve. These valves are used to control flow of fuel on the space shuttle. In this data set, some subsequences are normal whereas a few are abnormal. Originally, each subsequence consists of 1, 000 data points at a rate of 1 ms per sample. But, we have down sampled this data set by 70% to show that time series discord can be detected on far less data using CNN. Normal subsequences

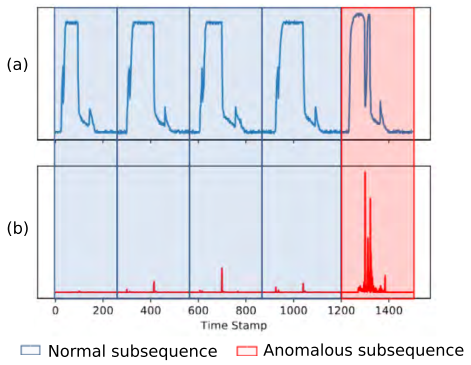

FIGURE 6. DeepAnT is also applicable to detect time series discords. Normal subsequences of a time series are highlighted in blue color in plot (a), whereas subsequence highlighted in red color is a discord. Lower plot (b) shows corresponding point wise anomaly score of a subsequence, which is accumulated (per subsequence) to detect a discord.

are shown in blue highlighted area of Fig. 6 (a), whereas the abnormal subsequence is shown in red highlighted area. Each subsequence is separated by a blue vertical line in this figure. Instead of extracting all the subsequences and converting them to some symbolic representation (as done in [57]), we simply train our predictor model on normal subsequences. Same DeepAnT architecture and parameters are used here except the history window and the horizon. On a given test data, the DeepAnT predictor tries to predict the whole subsequence. By aggregating the anomaly score calculated at each time stamp (shown in Fig. 6 (b)) of a subsequence, an anomaly score of whole subsequence is calculated. A discord is detected by applying a threshold on the calculated subsequence anomaly score. In addition to the discord detection, the behavior of subsequence which actually caused the discord can also be identified using DeepAnT. It can be observed in the red highlighted area of Fig. 6 (b), that the anomaly score of the corresponding abnormal behavior is much higher which actually caused the discord.

VI. CONCLUSION

We presented a deep learning based approach for the detection of anomalies in time series data. Since the approach is unsupervised, it requires no labels for anomalies. Instead, the method models the regular data distribution and marks data points which don't conform to this model as anomalous. The method is capable of handling minor data contamination (less than 5%). This technique is accurate even in the detection of small deviations/anomalies in time series cycles which are generally overlooked by other distance based and density based anomaly detection techniques.

We evaluated DeepAnT on 10 different data sets comprising of 433 time series in total and provided a detailed comparison with 15 state-of-the-art anomaly detection methods. To highlight the generic nature of the proposed technique, we tested it on real as well as synthetic datasets from different domains including road traffic [38], network utilization [38], on-line advertisement [38], internet traffic [55], space shuttle [58], and health [53]. In most of the cases, DeepAnT outperformed the state-of-the-art methods while remained on par with others. The proposed approach is capable of detecting point anomalies and contextual anomalies in time series data even with periodic and seasonality characteristics, and it is also applicable to time series discords detection. DeepAnT demonstrated generalization capabilities on small as well as big data scenarios.

This approach can be practically applied in situations where there is availability of a large amount of data without any possibility of labeling it. However, poor data quality can corrupt the data modelling phase. On the other hand, if the contamination level is too high (more than 5%), the system will try to model those instances, hence, considering them as normal at inference time. Another limitation is the selection of the network architecture and the corresponding hyperparameters. This can be circumvented by employing the recent architecture search techniques [59] trading human expertise with compute time. One of the most severe limitation is perhaps adversarial examples [60] limiting the usage of this method (and most of the prior data-driven methods) in security critical scenarios. Significant strides have been made in understanding and defending against these adversarial examples. However, no generic technique has yet been developed to circumvent this issue.

We are working on extending the model and using the concept of domain adaptation and transfer learning for anomaly detection in time series analysis. It will also be interesting to evaluate the impact of incorporating different pre-processing techniques on the final time series forecast.

ACKNOWLEDGMENT

The authors would like to thank the reviewers for their valuable feedback in improving the quality of this paper. They also appreciate the technical support provided by Mr. Muhammad Naseer Bajwa.

REFERENCES

[1] C. C. Aggarwal, "An introduction to outlier analysis," in *Outlier Analysis*. Cham, Switzerland: Springer, 2016, pp. 1–40.

[2] F. E. Grubbs, "Procedures for detecting outlying observations in samples," *Technometrics*, vol. 11, no. 1, pp. 1–21, 1969.

[3] M. Goldstein, "Anomaly detection in large datasets," Ph.D. dissertation, Dept. Comput. Sci., Univ. Kaiserslautern, München, Germany, Feb. 2014. [Online]. Available: http://www.goldiges.de/phd

[4] L. Columbus. (Dec. 2017). *53% of Companies Are Adopting Big Data Analytics*. [Online]. Available: https://goo.gl/tN5eNC

[5] J. Koetsier. (Dec. 2017). *IoT in the USA*. [Online]. Available: https://goo.gl/CPKYrc

[6] P. Malhotra, L. Vig, G. Shroff, and P. Agarwal, "Long short term memory networks for anomaly detection in time series," in *Proc. Eur. Symp. Artif. Neural Netw.*, vol. 23, 2015, p. 89.

[7] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell. (2015). "Learning to diagnose with LSTM recurrent neural networks." [Online]. Available: https://arxiv.org/abs/1511.03677

[8] S. Chauhan and L. Vig, "Anomaly detection in ECG time signals via deep long short-term memory networks," in *Proc. IEEE Int. Conf. Data Sci. Adv. Anal. (DSAA)*, Oct. 2015, pp. 1–7.

[9] V. Chandola, D. Cheboli, and V. Kumar, "Detecting anomalies in a time series database," Dept. Comput. Sci., Univ. Minnesota, Minneapolis, MN, USA, Tech. Rep. 09-004, 2009.

[10] C. C. Aggarwal, *Outlier Analysis*, 2nd ed. Cham, Switzerland: Springer, 2016.

[11] S. Ramaswamy, R. Rastogi, and K. Shim, "Efficient algorithms for mining outliers from large data sets," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 427–438, 2000.

[12] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, 2000.

[13] J. Tang, Z. Chen, A. W.-C. Fu, and D. W. Cheung, "Enhancing effectiveness of outlier detections for low density patterns," in *Advances in Knowledge Discovery and Data Mining*. Berlin, Germany: Springer, 2002, pp. 535–548.

[14] W. Jin, A. K. H. Tung, J. Han, and W. Wang, "Ranking outliers using symmetric neighborhood relationship," in *Advances in Knowledge Discovery and Data Mining*. Berlin, Germany: Springer, 2006, pp. 577–593.

[15] Z. He, X. Xu, and S. Deng, "Discovering cluster-based local outliers," *Pattern Recognit. Lett.*, vol. 24, nos. 9–10, pp. 1641–1650, 2003.

[16] M. Goldstein and A. Dengel, "Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm," in *Proc. KI-2012: Poster Demo Track*, 2012, pp. 59–63.

[17] M. Amer, M. Goldstein, and S. Abdennadher, "Enhancing one-class support vector machines for unsupervised anomaly detection," in *Proc. ACM SIGKDD Workshop Outlier Detection Description*, 2013, pp. 8–15.

[18] B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson, "Estimating the support of a high-dimensional distribution," *Neural Comput.*, vol. 13, no. 7, pp. 1443–1471, 2001.

[19] S. W. Yahaya, C. Langensiepen, and A. Lotfi, "Anomaly detection in activities of daily living using one-class support vector machine," in *Proc. U.K. Workshop Comput. Intell.* Cham, Switzerland: Springer, 2018, pp. 362–371.

[20] M. Hu *et al.*, "Detecting anomalies in time series data via a meta-feature based approach," *IEEE Access*, vol. 6, pp. 27760–27776, 2018.

[21] B. Liu, Y. Xiao, L. Cao, Z. Hao, and F. Deng, "SVDD-based outlier detection on uncertain data," *Knowl. Inf. Syst.*, vol. 34, no. 3, pp. 597–618, Mar. 2013, doi: 10.1007/s10115-012-0484-y.

[22] D. M. J. Tax and R. P. W. Duin, "Support vector data description," *Mach. Learn.*, vol. 54, no. 1, pp. 45–66, Jan. 2004.

[23] M.-L. Shyu, S.-C. Chen, K. Sarinnapakorn, and L. Chang, "A novel anomaly detection scheme based on principal component classifier," in *Proc. IEEE Found. New Directions Data Mining Workshop, Conjunction 3rd IEEE Int. Conf. Data Mining (ICDM)*, 2003, pp. 172–179.

[24] R. Kwitt and U. Hofmann, "Robust methods for unsupervised PCA-based anomaly detection," in *Proc. IEEE/IST WorNshop Monitor., Attacn Detection Mitigation*, Sep. 2006, pp. 1–3.

[25] J. Wong, C. Colburn, E. Meeks, and S. Vedaraman. (Feb. 2015). *Rad—Outlier Detection on Big Data*. [Online]. Available: http://techblog.netflix.com/2015/02/rad-outlier-detection-on-big-data.html

[26] D. Yankov, E. Keogh, and U. Rebbapragada, "Disk aware discord discovery: Finding unusual time series in terabyte sized datasets," in *Proc. 7th IEEE Int. Conf. Data Mining (ICDM)*, Oct. 2007, pp. 381–390.

[27] Q. Yu, L. Jibin, and L. Jiang, "An improved ARIMA-based traffic anomaly detection algorithm for wireless sensor networks," *Int. J. Distrib. Sensor Netw.*, vol. 12, no. 1, p. 9653230, 2016.

[28] A. H. Yaacob, I. K. Tan, S. F. Chien, and H. K. Tan, "ARIMA based network anomaly detection," in *Proc. 2nd Int. Conf. Commun. Softw. Netw. (ICCSN)*, Feb. 2010, pp. 205–209.

[29] S. Kanarachos, S.-R. G. Christopoulos, A. Chroneos, and M. E. Fitzpatrick, "Detecting anomalies in time series data via a deep learning algorithm combining wavelets, neural networks and Hilbert transform," *Expert Syst. Appl.*, vol. 85, pp. 292–304, Nov. 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0957417417302737

[30] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Proc. Int. Conf. Web-Age Inf. Manage.* Cham, Switzerland: Springer, 2014, pp. 298–310.

[31] T. Amarbayasgalan, B. Jargalsaikhan, and K. H. Ryu, "Unsupervised novelty detection using deep autoencoders with density based clustering," *Appl. Sci.*, vol. 8, no. 9, p. 1468, 2018. [Online]. Available: http://www.mdpi.com/2076-3417/8/9/1468

[32] M. Schreyer, T. Sattarov, D. Borth, A. Dengel, and B. Reimer, "Detection of anomalies in large scale accounting data using deep autoencoder networks," *CoRR*, vol. abs/1709.05254, pp. 1–19, Sep. 2017. [Online]. Available: http://arxiv.org/abs/1709.05254

[33] A. Kejariwal. (Jan. 2015). *Introducing Practical and Robust Anomaly Detection in a Time Series*. [Online]. Available: https://blog.twitter.com/2015/introducing-practical-and-robust-anomaly-detection-in-a-time-series
[34] B. Rosner, "Percentage points for a generalized ESD many-outlier procedure," *Technometrics*, vol. 25, no. 2, pp. 165–172, May 1983.
[35] R. B. Cleveland, W. S. Cleveland, J. E. McRae, and I. Terpenning, "STL: A seasonal-trend decomposition procedure based on loess," *J. Off. Statist.*, vol. 6, no. 1, pp. 3–73, 1990.
[36] N. Laptev, S. Amizadeh, and I. Flint, "Generic and scalable framework for automated time-series anomaly detection," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2015, pp. 1939–1947.
[37] (2015). *Contextual Anomaly Detector*. [Online]. Available: https://github.com/smirmik/CAD
[38] A. Lavin and S. Ahmad, "Evaluating real-time anomaly detection algorithms—The Numenta anomaly benchmark," in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl. (ICMLA)*, Dec. 2015, pp. 38–44.
[39] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, "Unsupervised real-time anomaly detection for streaming data," *Neurocomputing*, vol. 262, pp. 134–147, Nov. 2017.
[40] (2013). *Skyline*. [Online]. Available: https://github.com/etsy/skyline
[41] F. T. Liu, K. M. Ting, and Z.-H. Zhou, "Isolation forest," in *Proc. 8th IEEE Int. Conf. Data Mining (ICDM)*, Dec. 2008, pp. 413–422.
[42] F. A. Gers, D. Eck, and J. Schmidhuber, "Applying LSTM to time series predictable through time-window approaches," in *Neural Nets WIRN Vietri-01*. London, U.K.: Springer, 2002, pp. 193–200.
[43] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[44] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet classification with deep convolutional neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2012, pp. 1097–1105.
[45] F. N. Iandola, S. Han, M. W. Moskewicz, K. Ashraf, W. J. Dally, and K. Keutzer. (2016). "SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and $<$ 0.5 MB model size." [Online]. Available: https://arxiv.org/abs/1602.07360
[46] T. T. Tchrakian, B. Basu, and M. O'Mahony, "Real-time traffic flow forecasting using spectral analysis," *IEEE Trans. Intell. Transp. Syst.*, vol. 13, no. 2, pp. 519–526, Jun. 2012.
[47] P. Du Jardin and E. Séverin, "Predicting corporate bankruptcy using a self-organizing map: An empirical study to improve the forecasting horizon of a financial failure model," *Decis. Support Syst.*, vol. 51, no. 3, pp. 701–711, 2011.
[48] Y. Chen *et al.* (Jul. 2015). *The UCR Time Series Classification Archive*. [Online]. Available: www.cs.ucr.edu/~eamonn/time_series_data/
[49] N. Singh and C. Olinsky, "Demystifying Numenta anomaly benchmark," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, May 2017, pp. 1570–1577.
[50] C. Wang, K. Viswanathan, L. Choudur, V. Talwar, W. Satterfield, and K. Schwan, "Statistical techniques for online anomaly detection in data centers," in *Proc. IFIP/IEEE Int. Symp. Integr. Netw. Manage. (IM)*, May 2011, pp. 385–392.
[51] R. P. Adams and D. J. C. MacKay. (2007). "Bayesian online changepoint detection." [Online]. Available: https://arxiv.org/abs/0710.3742
[52] M. Schneider, W. Ertel, and F. Ramos, "Expected similarity estimation for large-scale batch and streaming anomaly detection," *Mach. Learn.*, vol. 105, no. 3, pp. 305–333, 2016.
[53] D. Dheeru and E. K. Taniskidou. (2017). *UCI Machine Learning Repository*. [Online]. Available: http://archive.ics.uci.edu/ml
[54] N. Goix. (2016). "How to evaluate the quality of unsupervised anomaly detection algorithms?" [Online]. Available: https://arxiv.org/abs/1607.01152
[55] K. Yamanishi, J.-I. Takeuchi, G. Williams, and P. Milne, "On-line unsupervised outlier detection using finite mixtures with discounting learning algorithms," in *Proc. 6th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2000, pp. 320–324.
[56] D. M. Rocke and D. L. Woodruff, "Identification of outliers in multivariate data," *J. Amer. Stat. Assoc.*, vol. 91, no. 435, pp. 1047–1061, 1996.
[57] E. Keogh, J. Lin, and A. Fu, "Hot sax: Efficiently finding the most unusual time series subsequence," in *Proc. 5th IEEE Int. Conf. Data Mining*, 2005, pp. 226–233.
[58] B. Ferrell and S. Santuro. (2005). *NASA Shuttle Valve Data*. [Online]. Available: http://www.cs.fit.edu/~pkc/nasa/data/
[59] B. Zoph and Q. V. Le, "Neural architecture search with reinforcement learning," *CoRR*, vol. abs/1611.01578, pp. 1–16, Nov. 2016.
[60] A. Kurakin, I. J. Goodfellow, and S. Bengio, "Adversarial machine learning at scale," *CoRR*, vol. abs/1611.01236, pp. 1–17, Nov. 2016.

MOHSIN MUNIR received the master's degree in computer science from the University of Kaiserslautern, Germany. He is currently pursuing the Ph.D. degree in computer science with the German Research Center for Artificial Intelligence (DFKI GmbH) under the supervision of Prof. Dr. Prof. H. C. Andreas Dengel. The topic of his master's thesis was Connected Heating System's Fault Detection using Data Anomalies and Trends. He did internships at RICOH, Japan, and BOSCH, Germany, during his master's degree. His research topic is Time Series Forecasting and Anomaly Detection. His research interests are time series analysis, deep neural networks, forecasting, predictive analytics, and anomaly detection. During his Ph.D., he did a research internship at Kyushu University, Japan, under the supervision of Prof. Seiichi Uchida.

SHOAIB AHMED SIDDIQUI received the bachelor's degree in computer science from the National University of Sciences and Technology, Pakistan, in 2016. He is currently pursuing the M.S. degree leading to Ph.D. program with the German Research Center for Artificial Intelligence (DFKI GmbH) and Technische Universitaet Kaiserslautern, under the supervision of Prof. Dr. Prof. h.c. Andreas Dengel. His areas of interests include interpretability and robustness of deep learning models (including adversarial examples and defenses), document understanding, time-series analysis, and extreme classification. He is also a Reviewer for the *ICES Journal of Marine Science* and the IEEE ACCESS.

ANDREAS DENGEL received the Diploma degree in computer science from the University of Kaiserslautern and the Ph.D. degree from the University of Stuttgart. He is a Scientific Director with the German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern. In 1993, he became a Professor at the Computer Science Department, University of Kaiserslautern, where he holds the chair Knowledge-Based Systems. Since 2009, he has been a Professor (Kyakuin) with the Department of Computer Science and Information Systems, Osaka Prefecture University. He was also with IBM, Siemens, and Xerox Parc. Moreover, he has co-edited international computer science journals and has written or edited 12 books. He has authored more than 300 peer-reviewed scientific publications and has supervised more than 170 Ph.D. and master's theses. He is a member of several international advisory boards, has chaired major international conferences, and founded several successful start-up companies. He is an IAPR Fellow and has received prominent international awards. His main scientific emphasis is in the areas of pattern recognition, document understanding, information retrieval, multimedia mining, semantic technologies, and social media.

SHERAZ AHMED received the master's degree in computer science from the Technische Universitaet Kaiserslautern, Germany, and the Ph.D. degree from the German Research Center for Artificial Intelligence, Germany, under the supervision of Prof. Dr. Prof. H. C. Andreas Dengel and Prof. Dr. habil. Marcus Liwicki. His Ph.D. topic is Generic Methods for Information Segmentation in Document Images. He is a Senior Researcher with the German Research Center for Artificial Intelligence (DFKI GmbH), Kaiserslautern, where he is leading the area of time series and document analysis. Over the last few years, he has primarily worked for the development of various systems for information segmentation in document images. From 2012 to 2013, he visited Osaka Prefecture University, Osaka, Japan, as a Research Fellow, supported by the Japanese Society for the Promotion of Science, and from 2014 to 2014, he visited the University of Western Australia, Perth, Australia, as a Research Fellow, supported by the DAAD, Germany and Go8, Australia. His research interests include document understanding, generic segmentation framework for documents, gesture recognition, pattern recognition, data mining, anomaly detection, and natural language processing. He has more than 30 publications on the said and related topics, including three journal papers and two book chapters. He is a frequent Reviewer of various journals and conferences, including *Patter Recognition Letters*, *Neural Computing and Applications*, IJDAR, ICDAR, ICFHR, and DAS.