Journal of Big Data

METHODOLOGY Open Access

Towards a deep learning-based outlier detection approach in the context of streaming data

Asmaa F. Hassan*, Sherif Barakat and Amira Rezk

*Correspondence:
asmaa.fawzy@std.mans.edu.eg

Department of Information Systems, Faculty of Computers and Information, Mansoura University, Mansoura, Egypt

Abstract

Uncommon observations that significantly vary from the norm are referred to as outliers. Outlier detection, which aims to detect unexpected behavior, is a critical topic that has attracted significant attention in a wide range of research areas and application domains, including video surveillance, network intrusion detection, disease outbreak detection, and others. Deep learning-based techniques for outlier detection have currently outperformed machine learning and shallow approaches on streaming data, which are big and complicated datasets. Despite the fact that deep learning has been successfully applied in a variety of application domains, developing an effective and appropriate model is a difficult task due to the dynamic nature and variations of real-world applications and data. Hence, this research proposes a novel deep learning model based on a deep neural network (DNN) to handle the outlier detection problem in the context of streaming data. The proposed DNN model is developed with multiple hidden layers to improve feature abstraction and capabilities. Extensive experiments performed on four real-world outlier benchmark datasets, available at the UCI repository, and comparisons to state-of-the-art approaches are used to evaluate the proposed model's performance. Experiment results demonstrate that it outperforms both machine learning algorithms and deep learning competitors, resulting in significant performance gains. Particularly, when compared to other algorithms, the evaluation results clearly demonstrated the efficacy of the proposed approach, with much higher accuracy, recall and f1-score rates of 99.63%, 99.014% and 99.437%, respectively.

Keywords: Outlier detection, Data streams, Deep learning, Anomaly detection, Deep neural networks

Introduction

The growth of connectivity and increased internet usage in recent years, especially over the past decade, have produced massive amounts of data. This type of big data is known as streaming data, which is a potentially infinite collection of data points arranged in ascending temporal order [1]. Thus, a data point in a data stream is not stationary; rather, it evolves over time. Hence, data streams are often processed using time window models because their size is potentially infinite and can't be totally stored in memory. Windows can be *data-driven*, i.e., every 100 instances, or *time-based*, every 20 s, for

example, where they estimate the change incrementally throughout a data window or over time. The most common timing window types include sliding window, landmark window, and damped window models [2]. In the *sliding window model,* every window has a fixed size *w* of the time *t*. Thus, each window includes just current data points while discarding older ones, and all these points within the active window have the same priority. In contrast, the *landmark window model* processes a data stream from a time-stamp called landmark to now, keeping one of its bounds fixed at a given time instant and allowing the other to follow the time evolution. A *damped window model*, on the other hand, assigns weights to the data in the stream, with each item assigned a weight based on its arrival time, and assigns larger weights to current data than to previous data, and this weight then declines exponentially over time in accordance with some ageing function. Streaming data is vulnerable to anomalous data points, which are called *outliers* as a result of the large amount of data involved. Hawkins [3] defined an outlier as "*an observation that deviates so significantly from other observations as to arouse suspicion that it was generated by a different mechanism"*. In addition, outliers are also known as anomalies, abnormalities, or deviants in the statistics and data mining literature [4]. Outlier data is commonly detected in order to provide early warning of danger or to uncover novel possibilities. *Outlier detection* is the process of finding data items that do not follow predicted or normal behavior. Detection of outliers over data streams is gaining popularity because it has a wide range of applications, including, but not limited to, fraud detection in the banking and finance sector, fault detection in the industry sector, and abnormal vitals and early disease detection in the healthcare and medical sector. On the other hand, outlier detection in data streams is a significant difficulty to overcome since the underlying distribution of this data may not be known in advance. Hence, this process over streaming data may be done in an offline or online mode. Outliers in historical data are detected offline using *batch processing,* which is the simultaneous processing of a huge volume of data. In online mode, instead, fresh data points come in while outliers are detected and this is called *real-time processing*. Real-time mode ensures that data is executed within a short time after it is received. In this study, the sliding window setting is adopted through the real-time streaming processing mode.

Deep neural networks (DNNs) have lately emerged as the go-to solution for tackling a wide range of problems in many domains, such as computer vision, natural language processing (NLP), and network security, etc. They dominate in these areas due to their deep design, which enables them to provide numerous representations for learning complicated features for effective prediction. In addition, the deep architecture of DNN addresses the constraints of classic ML techniques in terms of the manual feature engineering that is required, generalization to new changes in data, and scalability. Therefore, these characteristics of DNN have made it a suitable approach for outlier detection [5, 6]. A deep neural network is a supervised learning approach that trains the model over multiple layers. The term "deep" denotes the number of layers used to extract the data features. Figure 1 depicts the overall structure of a DNN, which comprises an input layer, a number of hidden layers, and an output layer. A layer is a group of neurons that are layered in various ways, such as convolutional and pooling, fully connected, and recurrent layers, where these layers may be used to construct a variety of popular neural networks, such as Convolutional neural

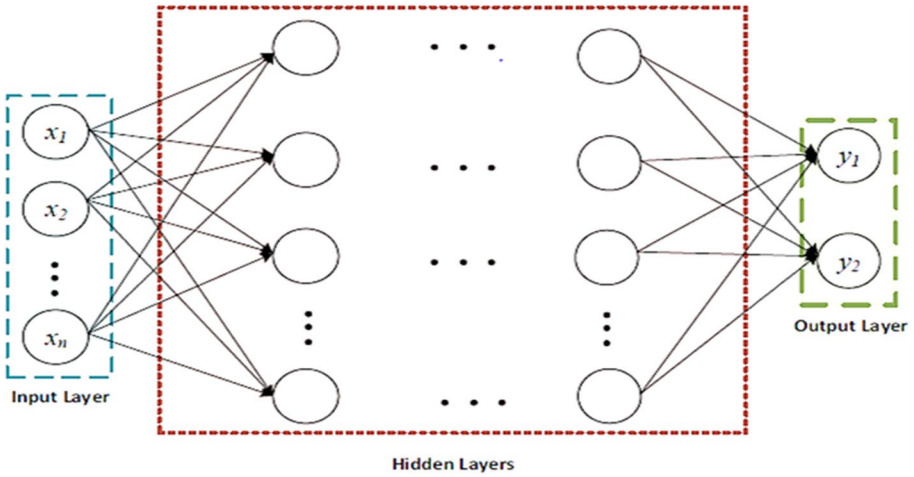

Fig. 1 The Deep Neural Network Architecture

networks (CNN), multilayer perceptron networks (MLP), and Recurrent Neural Networks (RNN). DNNs develop expressive representations by using complicated combinations of linear or nonlinear functions that may be represented by a computational graph. Such functions are known as "activation functions" because they determine the output of computational graph nodes, or neurons in neural networks, given specific inputs. Common activation functions include sigmoid, ReLU (Rectified Linear Unit), linear, and tanh.

In this paper, a novel deep learning-based model based on the deep neural network (DNN) is presented for detecting outliers in the context of streaming data. The proposed model uses the slide window approach in order to process streaming data to provide an online DNN-based framework for detecting outlier instances in real-time. In the proposed model, a DNN is developed with an input layer, three hidden layers, and an output layer with two neurons for recording an instance as an outlier or inlier. The ReLU function serves as an activation function for the three hidden layers, while the sigmoid function serves as an activation function for the output layer. The study's most notable contributions are as follows:

1. A novel online outlier detection approach based on the DNN technique is proposed in the context of streaming data.
2. The proposed model applies the sliding window time-based concept and real-time processing mode to process the flow of streaming data.
3. A review of the literature on outlier detection in data stream settings using both machine learning and deep learning approaches is conducted.
4. On four outlier detection benchmark datasets, an extensive experimental study of various machine-learning-based methods and state-of-the-art deep-learning-based outlier detection approaches was conducted.
5. The proposed model outperforms the state-of-the-art approaches, and it achieves the best tradeoff by achieving a high detection rate while having a low miss rate.

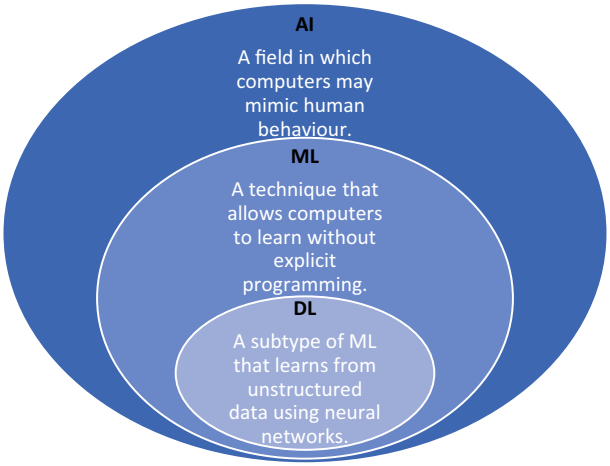

Fig. 2 Artificial Intelligence vs. Machine Learning vs. Deep Learning

The rest of this paper is structured as follows: "Related work" section provides an overview of the literature. "The proposed DNN model" section describes the proposed model adopted in this work. Experimental results and analysis Section includes a detailed explanation of the used datasets, experimental setup, and a discussion of the results. Finally, Conclusions and Future Work Section wraps up the proposed work by outlining future directions.

Related work

Outlier detection approaches in the setting of data streams have been divided into three categories: machine learning (ML), deep learning (DL), and hybrid techniques that combine ML and DL approaches [7]. As shown in Fig. 2, machine learning is a method that enables computers to learn without even being explicitly taught, in which the system may learn from the past or present and make a prediction or decision in the future [8]. Deep learning (DL), on the other hand, is a subtype of ML in the artificial intelligence (AI) field, where AI is the field that enables computers to act like humans and that develops neural networks to learn supervised, semi-supervised, or unsupervised from structured or unstructured data [9, 10]. In comparison to traditional ML approaches, recent advances in deep learning techniques have made it feasible to vastly increase outlier detection performance [11]. Deep learning for outlier detection entails using neural networks to develop feature representations or outlier scores for detecting outlier data [12]. It recently demonstrated exceptional abilities in learning expressive models of complex big data such as graph data, trajectories data, high-dimensional streaming data and temporal-spatial data.

Machine learning techniques are at the heart of a significant variety of existing outlier detection approaches. In the literature, several research methods on outlier detection for streaming data have been proposed, which were density-based [13–18], cluster-based [19–24], distance-based [25–28], classification-based [29–31], kernel-based [32–34] or ensemble-based techniques [35–40].

On the other hand, some recent deep learning-based outlier detection techniques have been presented. For instance, Chambers et al. [41] presented a unique

DNN-based technique called DeepStreamCE to identify concept evolution that used deep neural network activations in a streaming context. Another notable study is proposed in [42], where a framework for DNN-based anomaly detection is proposed, including explanations for any anomalies discovered. DeepAnT is a deep learning-based unsupervised anomaly detection approach proposed by the authors of [43]. It had two modules: the time-series predictor module, which was responsible for forecasting the next timestamp, and the anomaly detecting module, which was responsible for determining if a data point was normal or an outlier. DeepAnT trains CNN on raw data without eliminating abnormalities. They utilized two convolutional layers, which were followed by a max-pooling layer. A recent paper [44] proposes exploiting disturbances in the frequency domain's amplitude and phase spectrums for data augmentation in time series anomaly detection using a convolutional neural network (CNN).

In order to create effective models, researchers, in certain circumstances, turn to hybrid techniques that combine ML and DL methodologies. All of these techniques, in particular, inspect DL methods for complexity and feature reduction, followed by an ML predictor. Shone et al. [45], for example, used a hybrid technique of integrating autoencoder (AE) with the random forest technique by simply employing the encoder component of AE. Marir et al. created another hybrid technique by combining the deep belief network (DBN) with SVM using the ensemble approach and voting [46]. Furthermore, Yan et al. proposed another hybrid concept by combining sparse autoencoder and support vector machine (SVM), but this method had difficulty detecting minority outlier labels [47]. Munir et al. [48] proposed FuseAD, a hybrid unsupervised anomaly detection framework that combines statistical and deep-learning-based approaches. In particular, the Auto-regressive Moving Average (ARIMA) method and convolutional neural network (CNN) were used in the framework to address the anomaly detection problem over streaming sensors data. Federated Learning (FL) is an emerging branch of AI and machine learning in which a recent study [49] proposes bridging the gap between anomaly detection, federated learning, and data streams. None of the previous studies detected outliers using deep neural networks with high accuracy, precision, recall, and other metrics while minimizing false alarm and miss rates, as we did in this study. Although machine learning techniques have been at the heart of the outlier detection problem for many years, they have many limitations, such as the fact that they perform efficiently only on a predetermined set of features and become unreliable when new features are introduced into the system, whereas deep learning is efficient and works best in this situation. Deep learning-based outlier detection for streaming data is, however, still under-researched. Currently, Deep Neural Networks (DNN), Recurrent Neural Networks (RNN), Convolutional Neural Networks (CNN), Deep Belief Networks (DBN), Autoencoders (AE), Generative Adversarial Networks (GAN), and other DL approaches have been used by researchers to construct effective outlier detection models [50]. This is due to its superior learning capability from data, as well as its robustness and performance in solving difficult outlier detection problems for a wide range of real-world applications.

The proposed DNN model

The proposed DNN-based model for detecting outliers over stream is a combination of three sequential phases, which are data preprocessing, DNN training, and detection phases. The workflow of the proposed DNN-based model is presented in Fig. 3. The approach applied to developing the DNN-based outlier detection model is thoroughly discussed in the next subsections.

Data preprocessing phase

Data preprocessing is a necessary stage that promotes the extraction of relevant insights from data by improving the quality of the data. Therefore, the data preprocessing phase is the first stage of the outlier detection model for processing and cleanup of the raw data to create and train the DNN-based model. This phase consists of five consecutive steps:

Importing the dataset

The dataset is imported that is collected from the stream systems.

Handling missing values of data

IT is critical to identify and handle missing values effectively, as missing data can significantly influence the conclusions that can be formed from the data. Hence, for variables with missing values, we will replace the missing values based on the data type of the variable: numerical or categorical. Specifically, if the missing data is numerical, the variable mean is used to fill in the blanks. If it is categorical, the variable mode is used instead. This neutrally removes the erroneous bias that might be produced by missing values.

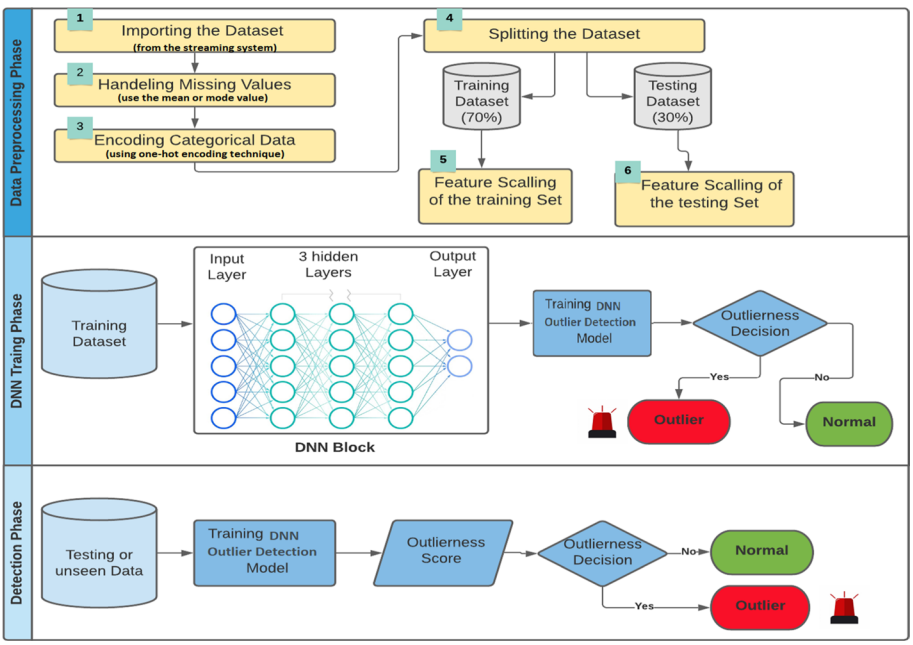

Fig. 3 The proposed DNN-based Outlier Detection Model

Encoding categorical data

DNNs basically work only with numerical data. Hence, these categorical variables are converted to numerical representations. The One-Hot Encoding technique is used for that conversion.

Splitting the dataset

The step of splitting the dataset, which divides the dataset into two subsets: training and testing sets, marks the completion of the data preparation phase. The training set is the dataset subset that is utilized to train the deep neural network. In contrast, the test set is the dataset subset that is used to test the model and evaluate it against an unknown test set. The data in this study is randomly divided so that the training set comprises 70% of the data and the testing set has 30% of the data.

Feature scaling for the training set

Feature scaling is a method for standardizing a dataset's independent variables within a certain range. In other words, feature scaling reduces the number of variables to a manageable number that can be compared on an equitable basis. Thus, each feature is standardized except for the target label in the training data based on Eq. (1).

$$x^{'} = \frac{x - mean(x)}{\sigma} \tag{1}$$

where $x^{'}$ is the data point's new value, x is the original data point value, and σ is the standard deviation.

Scaling of features for the test set

For the testing data, subtract the mean of the training data from the values of each feature and divide the resulting values by the training data's standard deviation.

DNN training phase

During the DNN training phase, a deep neural network comprising an input layer, three hidden layers, and an output layer is built to increase the abstraction features for more capability. The DNN used in this study is based on the feed-forward artificial neural network concept. The input layer nodes, in particular, accept the input training data and have the same number of neurons as the training data's number of features. The input layer *(I)* is linked to the hidden layer nodes through weights *(w)*, and the resulting link weights are generated using an activation function before being linked to the output layer nodes. As indicated in Fig. 4, the output layer nodes are two neurons for identifying instances as outliers or normal.

In this study, the activation function used for all three hidden layers is the *ReLU* function, which computes bias *(b)* from weighted inputs for the next layer of the network based on the result of each preceding layer, as given in Eq. (2).

$$Relu(x) = \max(0, x) \tag{2}$$

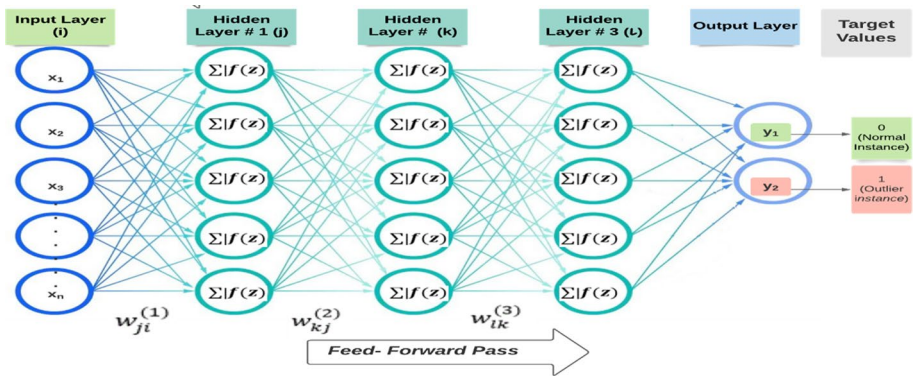

Fig. 4 DNN Block

More formally, Eq. (3) and Eq. (4) are used for determining the output of each hidden layer of the network:

$$x^l = f(z^l) \tag{3}$$

$$z^l = w^l x^{l-1} + b^l \tag{4}$$

where the output activation function $f(z^l)$ is calculated by the Relu function as in Eq. (2), l is the current layer, and w and b denote the hidden layer's weight and bias.

The final output decision layer is a fully-connected layer, which is fitted with a sigmoid activation function computed by Eq. (5). It is a sigmoid classification layer that considers two neurons to reflect the normal and outlier classification.

$$sigmoid(z) = 1/1 + e^{-z} \tag{5}$$

This sigmoid activation function is used for mapping the data instances $x = (x_1, x_2, \ldots, x_N)$ to an outlier score which represents the probability of being an outlier.

From the statistical perspective, normal instances have higher probabilities as they are frequent measurements, but outlier instances are prone to have lower probabilities as they are rare events. Therefore, this fact enables us to distinguish between outlierness and normality where the values of probability are in the range of [0, 1] to classify data instances to outliers or normal. Specifically, the higher probability denotes a greater degree of being an outlier.

The outlierness score is calculated by $\varepsilon(x|\theta)$ function in Eq. (6)

$$\varepsilon(x|\theta) = \sum_{i,j=1}^{k} w_{ij} z_i z_j \tag{6}$$

where z_i is a low-dimensional embedding of x's i^{th} feature value in the representation space Z, w_{ij} is a trainable parameter that represents the interaction's weight. The DNN training phase outcomes in a trained DNN model that can distinguish between outlier and normal training data instances.

Detection phase

The detection phase is the final stage of the proposed framework. The trained deep neural network model is used to categorize the unseen data as outliers or normal instances. Subsequently, the test data is tested on the trained DNN model, and then it assigns an outlierness score to a given unseen instance. The cumulative probability of the sigmoid response vectors is used to get the final outlierness score.

Experimental results and analysis

This section describes both the experimental setting and the results of experiments. The dataset that was used in the study and its description are presented first. Then, the evaluation metrics and experimental setup are elaborated. Finally, the results are presented and discussed.

Dataset description

Four benchmark datasets are utilized to assess the performance of the proposed framework: Breast-Cancer, Thyroid Disease, Musk, and Cardiotocography. All of these datasets are accessible in comma-separated values (CSV) format from the UCI machine learning repository [51]. Table 1 summarizes the number of instances of outliers and inliers in the benchmark datasets.

Breast Cancer

The Breast-Cancer dataset contains 9 attributes and a total of 683 instances, with 239 outlier instances, which are the cancer cases; these cases represent about 35% of the total dataset. All attributes are numerical data, so there is no need for the encoding process. Because of its low dimensionality, there is no need to apply feature selection or feature reduction. To handle missing values in this dataset, they are replaced with their mean values.

Annthyroid

This dataset is for thyroid disease cases, and it has 6 numerical attributes and 15 categorical attributes. There are three classes: hyperfunction, subnormal functioning, and not hypothyroid. The hyperfunction and subnormal functioning classes are considered outlier classes, but the not hypothyroid class is treated as an inlier class. Because this dataset has missing values, numerical attributes are replaced with the attribute mean while categorical attributes with the attribute mode. The technique of one-hot encoding is used to encode categorical data and convert it to numerical data.

Table 1 Description of datasets

Dataset	# of attributes	# of instances	# of outliers	Percentage of outlier (%)	# of inliers
Breast-cancer	30	683	239	34.99	444
Annthyroid	21	7200	534	7.42	6666
Musk	168	6598	211	3.2	6387
Cardiotocography	23	2126	204	9.6	1922

Musk

The musk dataset included information about whether a molecule was musk or non-musk. Its attributes are all numerical values, and there are approximately 211 outlier instances. There are several classes, but we combined non-musk classes (j146, j147, and 252) to form inliers, and musk classes (213 and 211) were added as outliers. Other courses were dropped.

Cardiotocography

The cardiotocography dataset has measurements of fetal heart rate and uterine contraction features. It contains 23 numerical attributes and 2126 instances. There are three classes in this dataset; the normal class (inlier), the pathologic class (outlier), and the suspect class that was discarded. Like the breast-cancer dataset, there is no need for the encoding process.

Evaluation metrics

The proposed deep learning-based model's performance and effectiveness are evaluated using accuracy, specificity or true negative rate, false alarm rate, false negative rate or miss rate, precision, recall or sensitivity, and F1-Score metrics, which are considered the standard metrics for evaluating any outlier detection method. These evaluation metrics are based on the four parts of the confusion matrix shown in Table 2, which are true positive (TP), false negative (FN), false positive (FP), and true negative (TN). In more detail, the true positive (TP) means that the model predicted the data point as an outlier and it was actually outlier data, the false negative (FN) means that the model predicted the data point as inlier data but it was actually an outlier instance, the false positive (FP) means that the model predicted the data point as an outlier data but it was actually an inlier instance, and finally, the true negative (TN) means that the model predicted the data point as an inlier data and it was actually inlier or normal. Therefore, the accurate detections of outlier and inlier occurrences are represented by TP and TN, respectively, which are shaded green in Table 2. Whereas, FP and FN are erroneous detections of the model, so they are represented in red shading. Table 3 defines in detail the standard metrics used to evaluate the proposed model.

Experimental setup

The proposed model is developed by the open source Python programming language using the Keras API, which is an open source library for neural networks that operates on top of the TensorFlow framework and supports various DL techniques such as DNN,

Table 2 The confusion matrix

		Predicted Instances	
		Outlier (1)	Normal (0)
Actual Instances	Outlier (1)	TP	FN
	Normal (0)	FP	TN

Table 3 The evaluation metrics

Evaluation metric	Definition	Equation
Accuracy	The percentage of accurately predicted data to the total amount of data	$\frac{TP+TN}{TP+FP+FN+TN}$
True Negative Rate *(Specificity)*	The proportion of correctly predicted inlier data to total normal data	$\frac{TN}{FP+TN}$
False alarm rate	The proportion of outliers that were incorrectly predicted in comparison to all inlier instances	$\frac{FP}{FP+TN}$
False negative rate *(Miss rate)*	The proportion of incorrectly predicted inlier data to total outliers It indicates the probability that the model will miss the outlier cases	$\frac{FN}{TP+FN}$
Precision	The proportion of correctly predicted outliers to all predicted outlier instances	$\frac{TP}{TP+FP}$
Recall *(Sensitivity or hit rate)*	The proportion of correctly predicted outliers to all actual outliers	$\frac{TP}{TP+FN}$
F1-score	The harmonic mean of precision and recall values	$F = 2\left(\frac{Precision*Recall}{recision+Recall}\right)$

CNN, and RNN [5]. In addition, the implementation and performance evaluation of the proposed model are done in a Google Colab environment [52].

For the hyper-parameter settings, the optimizer was set to Adam with a learning rate of 0.001, the loss function was binary cross-entropy, and the batch size was selected to be 28. The activation functions utilized in this work are ReLU for the hidden layers and sigmoid for the output layer, as mentioned above. On the other hand, all datasets are divided into two groups: 70% training data and 30% test data. We specifically trained and tested the proposed model independently using fivefold cross-validation to reduce the proposed model's overfitting problem and increase its robustness at the same time, where a dataset is randomly partitioned into training and test sets. Overfitting can lead to an increase in the number of false positive detections of outliers, lowering the overall accuracy of the model; thus, we are attempting to avoid this issue. Furthermore, regularization is used to address this issue by introducing a penalty term into the loss function, i.e., a binary cross-entropy function is used as a loss function on the training set to decrease the complexity of the learned model.

Results and discussion

Experiments were conducted to evaluate the proposed DNN-based model for detecting outliers in data streams. More specifically, we assessed the classification accuracy and the aforementioned metrics to evaluate the model's ability to make the right decision. The proposed DNN-based model is compared with two different DL algorithms, which are DeepAnt [43] and RobustTAD [44]. Table 4 presents the results in terms of percentages for the performance evaluation measures evaluated in this study. It can be seen that when the ratio of outliers increases, the detection performance of DeepAnt and RobustTAD suffers significantly. However, the proposed model stays robust, such as in the case of Cardiotocography and Annthyroid datasets. Figures 5 and 6 also show the results of the performance measures. When compared to the other two DL methods, the proposed model had the highest accuracy scores of 98.354%, 98.644%, 99.241%, and 99.631%, while DeepAnt had the lowest accuracy scores of 91.037%, 93.618%, 90.392%,

Table 4 Performance evaluation of different DL methods (%)

Dataset	Algorithm	Accuracy	TNR	FAR	FNR	Precision	Recall	F1-score
Breast-Cancer	DeepAnT [43]	93.618	94.361	5.639	6.918	94.067	93.082	93.541
	RobustTAD [44]	95.864	96.938	3.062	4.789	96.243	95.211	95.642
	Proposed model	**98.354**	**96.544**	**3.456**	**2.215**	**98.579**	**97.785**	**98.108**
Annthyroid	DeepAnT [43]	91.037	92.304	7.696	9.164	91.210	90.836	91.120
	RobustTAD [44]	95.646	96.408	3.592	4.543	95.914	95.457	95.603
	Proposed model	**98.644**	**97.034**	**2.966**	**1.706**	**98.853**	**98.294**	**98.543**
Musk	DeepAnT [43]	90.392	92.143	7.857	10.195	90.491	89.805	90.098
	RobustTAD [44]	94.451	93.469	6.531	5.979	94.670	94.021	94.392
	Proposed model	**99.241**	**97.493**	**2.507**	**1.268**	**98.903**	**98.732**	**98.985**
Cardiotocography	DeepAnT [43]	90.816	91.362	8.638	10.368	90.984	89.632	90.620
	RobustTAD [44]	93.828	92.713	7.287	6.025	95.017	93.975	94.749
	Proposed model	**99.631**	**97.682**	**2.318**	**0.986**	**99.702**	**99.014**	**99.437**

The highest scores are highlighted in bold

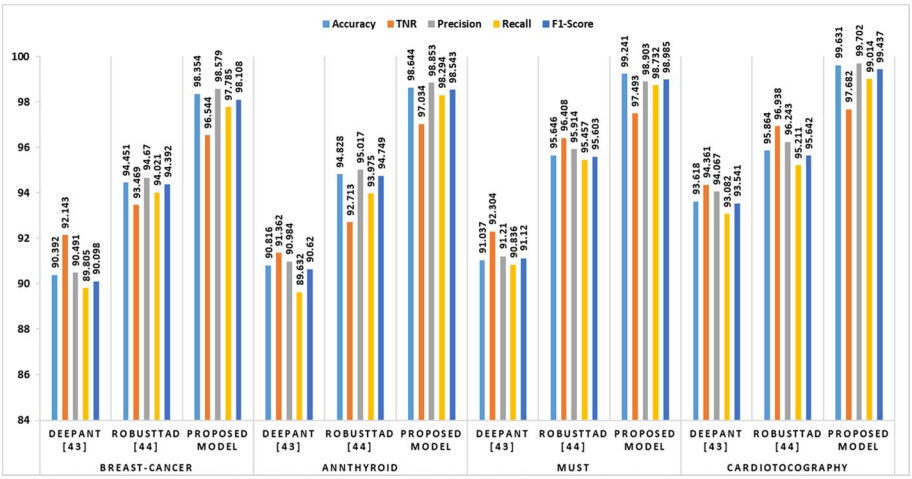

Fig. 5 Performance Metrics for different DL Methods on different Benchmark Datasets

and 90.816% for Breast-Cancer, Annthyroid, Musk, and Cardiotocography datasets, respectively. On the considered datasets, the proposed model correctly predicted outlier data with precision scores of 98.579%, 98.853%, 98.903%, and 99.702%. Furthermore, in terms of specificity, recall, and F1-score measures, the proposed model outperformed the DeepAnT and RobustTAD models.

Figure 6 compares the proposed model's performance to that of the other two DL-based techniques, DeepAnT and RobustTAD, in terms of FNR and FAR. When compared to other approaches, the proposed model demonstrated an extremely low FNR and FAR. It attained a false alarm rate of 3.456%, 2.966%, 2.50%, and 2.318%, respectively, while maintaining a very high detection rate. It can be observed that the competitor models exhibit greater FNR and FAR rates, demonstrating their inability to accurately detect outlier data instances.

Unlike the other two approaches, which tend to perform better on simple and low-dimensionality datasets such as the Breast-Cancer dataset, the benefit of the proposed

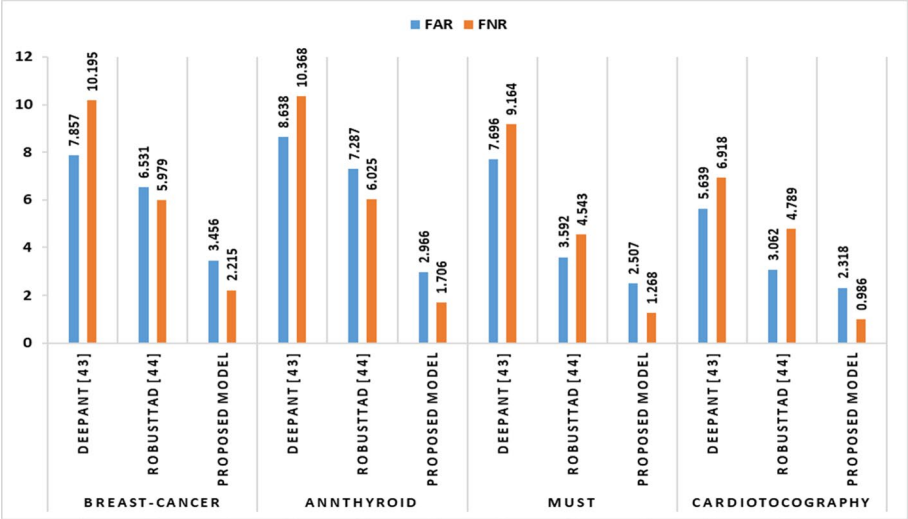

Fig. 6 Performance Metrics (FAR & FNR) of different DL Algorithms

Table 5 Performance evaluation of different ML methods (%)

Dataset	Algorithm	Accuracy	TNR	FAR	FNR	Precision	Recall	F1-Score
Musk	KNN	81.580	88.390	11.61	8.347	83.752	91.653	91.285
	SVM	84.451	90.813	9.187	5.208	84.610	94.792	92.204
	Random forest	89.413	93.684	6.316	4.989	90.688	95.011	93.942
	Proposed model	**99.241**	**97.493**	**2.507**	**1.268**	**98.903**	**98.732**	**98.985**
Cardiotocography	KNN	80.614	87.567	12.438	11.372	82.326	88.628	90.567
	SVM	83.828	89.651	10.349	9.963	85.711	90.037	90.930
	Random forest	85.653	92.874	7.126	7.606	90.014	92.394	91.507
	Proposed model	**99.631**	**97.682**	**2.318**	**0.986**	**99.702**	**99.014**	**99.437**

The highest scores are highlighted in bold

model becomes increasingly apparent on datasets of greater complexity and dimensionality. In particular, on the Annthyroid and Musk datasets, where Annthyroid has the most instances and Musk has the most features, the proposed model beats both alternatives. The proposed model, on the other hand, achieves the highest rates on both the Cardiotocography dataset, which has the highest outlier ratio, and the Musk dataset, which has the lowest outlier ratio.

In the next set of experiments, the proposed model is compared to the traditional machine learning algorithms such as KNN, SVM, and Random Forest. Table 5 summarizes the overall findings from the experiment, which was limited to two datasets, i.e., Annthyroid and Musk. These datasets were chosen because they have the highest dimensionality. Notably, as shown in Fig. 7, the proposed DNN-based model outperforms standard ML techniques across all metrics; this is due to the superior performance of DL techniques over ML techniques..

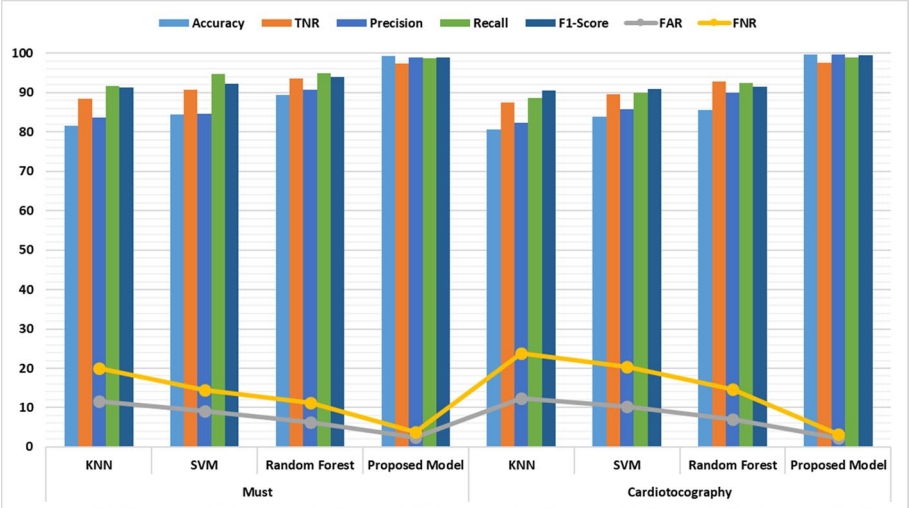

Fig. 7 Performance Comparison between ML methods and the proposed Model

Conclusions and future work

This research provides a novel outlier detection framework for streaming data settings based on deep neural networks (DNN) that can efficiently classify data flows as normal or outlier instances. The proposed model is evaluated on many real-world datasets and the experiment results demonstrated that it outperformed two state-of-the-art DL approaches in terms of detection accuracy, with a least false alarm rate that ranged from 2.3 to 3.4%. It was also revealed that the framework's detection rate, or recall, for detecting outliers varied from 97 to 99%. Furthermore, the proposed model definitely outperformed the classic ML techniques. However, because of the time required to train the model and its focus solely on detecting global outliers, the current version of the proposed model is not completely inadequate.

For future work, we plan to expand on this work by applying our approach to multiclass classification settings and utilizing additional DL techniques to more effectively solve the outlier detection problem in the context of data streams. Additionally, we intend to touch on the contextual outlier problem.

Abbreviations

DNN	Deep neural network
NLP	Natural language processing
CNN	Convolutional neural network
MLP	Multilayer perceptron
RNN	Recurrent neural network
ReLU	Rectified linear unit
ML	Machine learning
DL	Deep learning
AI	Artificial intelligence
AE	Autoencoders
DBN	Deep belief networks
FL	Federated learning
ARIMA	Auto-regressive moving average
GAN	Generative adversarial networks

Acknowledgements

Not applicable.

Author contributions
Conceptualization, methodology, software, statistical analysis, data analysis, literature review, discussion, writing—original draft preparation: AFH.; data curation: AFH. and AR; writing—review and editing: all authors; visualization: AFH and AR; supervision: SB. and AR. All authors read and approved the final manuscript.

Funding
Open access funding provided by The Science, Technology & Innovation Funding Authority (STDF) in cooperation with The Egyptian Knowledge Bank (EKB).

Availability of data and materials
The code and software developed in this study are available on reasonable request from the corresponding author.

Declarations

Ethics approval and consent to participate
Not applicable.

Consent for publication
Not applicable.

Competing interests
The authors declare that they have no competing interests.

Received: 10 March 2022 Accepted: 29 November 2022
Published online: 16 December 2022

References

1. Kim T, Park CH. Anomaly pattern detection for streaming data. Expert Syst Appl. 2020;149:113252. https://doi.org/10.1016/j.eswa.2020.113252.
2. Mansalis S, Ntoutsi E, Pelekis N, Theodoridis Y. An evaluation of data stream clustering algorithms. Stat Anal Data Min. 2018;11(4):167–87. https://doi.org/10.1002/sam.11380.
3. Hawkins DM. Identification of outliers, vol. 11. Dordrecht: Springer; 1980.
4. Aggarwal CC. An Introduction to Outlier Analysis. In: Aggarwal CC, editor. Outlier Analysis. Cham: Springer International Publishing; 2017. p. 1–34. https://doi.org/10.1007/978-3-319-47578-3_1.
5. Nguyen G, et al. Machine learning and deep learning frameworks and libraries for large-scale data mining: a survey. Artif Intell Rev. 2019;52(1):77–124. https://doi.org/10.1007/s10462-018-09679-z.
6. Czum JM. Dive into deep learning. J Am Coll Radiol. 2020;17(5):637–8. https://doi.org/10.1016/j.jacr.2020.02.005.
7. Al-amri R, Murugesan RK, Man M, Abdulateef AF, Al-Sharafi MA, Alkahtani AA. A review of machine learning and deep learning techniques for anomaly detection in iot data. Appl Sci. 2021;11(12):5320. https://doi.org/10.3390/app11125320.
8. Gomes HM, Read J, Bifet A, Barddal JP, Gama J. Machine learning for streaming data: state of the art, challenges, and opportunities. SIGKDD Explor Newsl. 2019;21(2):6–22. https://doi.org/10.1145/3373464.3373470.
9. Zhang A, Lipton ZC, Li M, Smola AJ, Dive into deep learning, arXiv Prepr. arXiv2106.11342, 2021.
10. Vargas R, Mosavi A, Ruiz R, Deep Learning: A Review, Adv Intell Syst Comput, no. October, https://doi.org/10.20944/preprints201810.0218.v1. 2018.
11. Pang G, Shen C, Cao L, Van Den Hengel A. Deep learning for anomaly detection. ACM Comput Surv. 2021;54(2):1–38. https://doi.org/10.1145/3439950.
12. Xue F, Yan W, Wang T, Huang H, Feng B. Deep anomaly detection for industrial systems: a case study. Annu Conf PHM Soc. 2020;12(1):8. https://doi.org/10.36001/phmconf.2020.v12i1.1186.
13. Cao F, Estert M, Qian W, Zhou A, Density-based clustering over an evolving data stream with noise, in Proceedings of the 2006 SIAM International Conference on Data Mining, Apr. 2006;2006:328–339. https://doi.org/10.1137/1.9781611972764.29.
14. Constantinou V. PyNomaly: anomaly detection using local outlier probabilities (LoOP). J Open Source Softw. 2018;3(30):845. https://doi.org/10.21105/joss.00845.
15. Yang X, Zhou W, Shu N, Zhang H, A Fast and Efficient Local Outlier Detection in Data Streams, in Proceedings of the 2019 International Conference on Image, Video and Signal Processing, 2019;111–116. doi: https://doi.org/10.1145/3317640.3317653.
16. Huang JW, Zhong MX, Jaysawal BP. Tadilof: time aware density-based incremental local outlier detection in data streams. Sensors. 2020;20(20):1–25. https://doi.org/10.3390/s20205829.
17. Singh M, Pamula R. ADINOF: adaptive density summarizing incremental natural outlier detection in data stream. Neural Comput Appl. 2021;33(15):9607–23. https://doi.org/10.1007/s00521-021-05725-0.
18. Abid A, El Khediri S, Kachouri A. Improved approaches for density-based outlier detection in wireless sensor networks. Computing. 2021;103(10):2275–92. https://doi.org/10.1007/s00607-021-00939-5.
19. Hassan A, Mokhtar H, Hegazy O. A heuristic approach for sensor network outlier detection. Int J Res Rev Wirel Sens Netw. 2011;1(4):66–72.
20. Fawzy A, Mokhtar HMO, Hegazy O. Outliers detection and classification in wireless sensor networks. Egypt Informatics J. 2013;14(2):157–64. https://doi.org/10.1016/j.eij.2013.06.001.
21. Amini A, Saboohi H, Herawan T, Wah TY. MuDi-Stream: a multi density clustering algorithm for evolving data stream. J Netw Comput Appl. 2016;59:370–85. https://doi.org/10.1016/j.jnca.2014.11.007.

22. Hyde R, Angelov P, MacKenzie AR. Fully online clustering of evolving data streams into arbitrarily shaped clusters. Inf Sci. 2017;382–383:96–114. https://doi.org/10.1016/j.ins.2016.12.004.
23. Bezerra CG, Costa BSJ, Guedes LA, Angelov PP. An evolving approach to data streams clustering based on typicality and eccentricity data analytics. Inf Sci. 2020;518:13–28.
24. Maia J, et al. Evolving clustering algorithm based on mixture of typicalities for stream data mining. Futur Gener Comput Syst. 2020;106:672–84.
25. Kontaki M, Gounaris A, Papadopoulos AN, Tsichlas K, Manolopoulos Y, Continuous monitoring of distance-based outliers over data streams, in Proceedings - International Conference on Data Engineering, 2011;135–146. https://doi.org/10.1109/ICDE.2011.5767923.
26. Tran L, Fan L, Shahabi C. Distance-based outlier detection in data streams. Proc ofthe VLDB Endow. 2016;9(12):1089–100.
27. Tran L, Fan L, Shahabi C, Fast distance-based outlier detection in data streams based on micro-clusters, ACM Int. Conf. Proceeding Ser, 2019; 162–169, https://doi.org/10.1145/3368926.3369667.
28. Tran L, Mun MY, Shahabi C. Real-time distance-based outlier detection in data streams. Proc VLDB Endow. 2020;14(2):141–53. https://doi.org/10.14778/3425879.3425885.
29. Bose B, Dutta J, Ghosh S, Pramanick P, Roy S, "Detection of Driving Patterns and Road Anomalies," in 2018 3rd International Conference On Internet of Things: Smart Innovation and Usages (IoT-SIU), 2018;1–7. https://doi.org/10.1109/IoT-SIU.2018.8519861.
30. Wu M, Song Z, Moon YB. Detecting cyber-physical attacks in cybermanufacturing systems with machine learning methods. J Intell Manuf. 2019;30(3):1111–23. https://doi.org/10.1007/s10845-017-1315-5.
31. Hasan M, Islam MM, Zarif MII, Hashem MMA. Attack and anomaly detection in IoT sensors in IoT sites using machine learning approaches. Internet Things. 2019;7:100059. https://doi.org/10.1016/j.iot.2019.100059.
32. Haque MA, Mineno H, Proposal of Online Outlier Detection in Sensor Data Using Kernel Density Estimation, Proc.—2017 6th IIAI Int Congr Adv Appl Informatics, IIAI-AAI 2017, 2017; July 2017: 1051–1052. https://doi.org/10.1109/IIAI-AAI.2017.41.
33. Daneshgadeh Çakmakçı S, Kemmerich T, Ahmed T, Baykal N. Online DDoS attack detection using mahalanobis distance and Kernel-based learning algorithm. J Netw Comput Appl. 2020;168:102756. https://doi.org/10.1016/j.jnca.2020.102756.
34. Bhattacharjee P, Garg A, Mitra P. KAGO: an approximate adaptive grid-based outlier detection approach using kernel density estimate. Pattern Anal Appl. 2021;24:1–22.
35. Iftikhar N, Baattrup-Andersen T, Nordbjerg FE, Jeppesen K. Outlier detection in sensor data using ensemble learning. Procedia Comput Sci. 2020;176:1160–9. https://doi.org/10.1016/j.procs.2020.09.112.
36. Kashef RF, Ensemble-based anomaly detection using cooperative learning, Proc Mach Learn. Res, 2017;71: 43–55, http://proceedings.mlr.press/v71/kashef18a/kashef18a.pdf
37. Ghomeshi H, Gaber MM, Kovalchuk Y, Ensemble Dynamics in Non-stationary Data Stream Classification, 2019;123–153https://doi.org/10.1007/978-3-319-89803-2_6.
38. Biswas P, Samanta T. Anomaly detection using ensemble random forest in wireless sensor network. Int J Inf Technol. 2021. https://doi.org/10.1007/s41870-021-00717-8.
39. Jayanthi N, Vijaya Babu B, Rao NS. An ensemble framework based outlier detection system in high dimensional data. Mater Today Proc. 2021;7(4):1162–75. https://doi.org/10.1016/j.matpr.2020.11.491.
40. Bii JK, Rimiru R, Mwangi RW. Adaptive boosting in ensembles for outlier detection: base learner selection and fusion via local domain competence. ETRI J. 2020;42(6):886–98. https://doi.org/10.4218/etrij.2019-0205.
41. Chambers L, Gaber MM, Abdallah ZS. DeepStreamCE: a streaming approach to concept evolution detection in deep neural networks, 2020;http://arxiv.org/abs/2004.04116
42. Amarasinghe K, Kenney K, Manic M, Toward explainable deep neural network based anomaly detection, Proc—2018 11th Int Conf Hum. Syst Interact HSI 2018, 2018;2:311–317. https://doi.org/10.1109/HSI.2018.8430788.
43. Munir M, Siddiqui SA, Dengel A, Ahmed S. DeepAnT: a deep learning approach for unsupervised anomaly detection in time series. IEEE Access. 2019;7(2019 January):1991–2005. https://doi.org/10.1109/ACCESS.2018.2886457.
44. Gao J, Song X, Wen Q, Wang P, Sun L, Xu H, "RobustTAD: Robust time series anomaly detection via decomposition and convolutional neural networks, Feb. 2020, http://arxiv.org/abs/2002.09545.
45. Shone N, Ngoc TN, Phai VD, Shi Q. A deep learning approach to network intrusion detection. IEEE Trans Emerg Top Comput Intell. 2018;2(1):41–50. https://doi.org/10.1109/TETCI.2017.2772792.
46. Marir N, Wang H, Feng G, Li B, Jia M. Distributed abnormal behavior detection approach based on deep belief network and ensemble SVM using spark. IEEE Access. 2018;6:59657–71. https://doi.org/10.1109/ACCESS.2018.2875045.
47. Khan N, Abdullah J, Khan AS. A dynamic method of detecting malicious scripts using classifiers. Adv Sci Lett. 2017;23(6):5352.
48. Munir M, Siddiqui SA, Chattha MA, Dengel A, Ahmed S. FuseAD : unsupervised anomaly detection in deep learning models. Sensors. 2019;19:1–15. https://doi.org/10.3390/s19112451.
49. Silva PR, Vinagre J, Gama J, Federated anomaly detection over distributed data streams, 2022, http://arxiv.org/abs/2205.07829
50. Mathew A, Amudha P, Sivakumari S. Deep learning techniques: an overview. In: Hassanien AE, Bhatnagar R, Darwish A, editors. Advanced machine learning technologies and applications. AMLTA 2020. Advances in intelligent systems and computing. Singapore: Springer Singapore; 2021. p. 599–608. https://doi.org/10.1007/978-981-15-3383-9_54.
51. Dua D, Gra C, UCI machine learning repository, 2017;http://archive.ics.uci.edu/ml.
52. Google Research Colaboratory, 2021; https://colab.research.google.com.

Publisher's Note