An Adaptive On-line Framework For Novelty Detection in Data Streams

Jose M. Alves

Department of Electrical and Computer Engineering

Western University

London, Ontario, Canada, N6A 5B9

jalves7@uwo.ca

ABSTRACT

Machine learning algorithms applied to data streams should be able to learn from the off-line data, and from the data streams where data profiles are hardly ever constant. The task of learning new or unknown concepts from data is known as novelty detection, which is a challenging problem exceptionally when negatives examples are not present to be used during the machine learning training. This work proposes an on-line novelty detection framework that addresses the issue of identifying a pattern change in an on-line data streams. In this paper, only "normal" data is available for off-line training, and the framework uses supervised and unsupervised learning to training an autoencoder model that can adapt to changes in the data distribution. The results find the proposed machine learning framework can use off-line data to learning a data stream profile and it can adjust automatically to cope with changes and new concepts emerged on the on-line data stream.

Index Terms: CS [Artificial Intelligence]: Novelty Detection—Data Streams;

1 INTRODUCTION

Machine learning techniques have frequently been applied to real-world applications. Data streams are one of most popular and challenging context where these models have been implemented. Usually, data streams are generated from non-stationary distributions, which means data streams profiles are always changing along the time and new concepts can emerge from the data. This scenario can be more complicated, in cases where the data streams contain only examples of the "normal" data behavior, or negative instances and anomalies are few or missing in the dataset. In this cases, the task of identifying new concepts or changes in the data pattern is challenging to the traditional machine learning techniques.

Novelty detection is one of the approaches used in machine learning to classify new data that is different in some aspects from the data used during the training phase. In this technique, a machine learning model is constructed to recognize the "normal" data behavior in scenarios where the counter-examples are scarce, or it is not possible to training models with negative instances. Applications of novelty detection include data streams provided by critical systems where it is common the presence of sizable "normal" dataset while faults or negative examples are few observations due to the high measurement costs or low frequency.

The application of on-line novelty detection in data streams is a challenging problem considering it involves the challenges of learning just from "normal" instances, on-line machine learning and data streams that are characterized by the high-speed, non-stationary distribution, and in some cases unlimited size.

This work presents an autonomous on-line novelty detection framework designed to perform off-line and on-line training to identify a pattern change in a data streams and adapt the machine learning model to the new concept emerged from the data.

The paper is organized as follows: Section 2 introduces related work, Section 3 provides background knowledge on novelty detection, autoencoders neural networks and on-line training, Section 4 provides an overview of the experimental methodology, Section 6 covers experimental implementation details, and in Section 7 the experimental results are examined and discussed.

2 RELATED WORK

Several machine learning techniques have been applied to perform novelty detection. In the last years, the topic has been reviewed in different works [1–5]. In this section, the related works are presented according to the following five general categories proposed in [4]: (i) probabilistic, (ii) distance-based, (iii) reconstruction-based, (iv) domain-based, and (v) information-theoric techniques.

Probabilistic novelty detection: These methods are focused on the construction of an estimate of the density function from the "normal" class data. Parametric approaches where the data distribution is modeled using General Mixture Models (GMM) were proposed by [6–8]. Non-parametric techniques consider the architectures of a model is not inflexible, which means, the model can change to fit the complexity of the data. Examples of simple non-parametric models are

the application of histograms to check if a new data is an outlier or not. Erdogmus *et al.* [9] proposed a kernel density framework that apply Parzen windows [10] to calculate marginal distributions from the data. Subramaniam *et al.* [11] described an approach that uses kernel density estimator to calculate a similarity to multi-dimension data distributions in a sensor network context.

Distance-based novelty detection: This approach is one of the most used in novelty detection. Distance-based novelty detection involves the theories of nearest-neighbor and clustering. The original idea behind these methods is that "normal" data are always clustered, while new classes appear isolated from the nearest neighbors. Bay and Schwabacher [12] showed in their work a nearest neighbor-based algorithm that searches for the nearest neighbor for a data point and then defines a threshold based on the outliers found so far in the dataset. several others nearest neighbor-based approaches were introduced by Aniulli and Pizzuti [13] , Boriah *et al.* [14], Breunig *et al.* [15], Chandola *et al.* [16], and Zhang and Wang [17].

Clustering-based approach was used by Syed *et al.* [18] in a novelty-detection application to classify high-risk medical patients. A clustering technique was used in [19] for clustering frequent itemsets identified in a time window. Itemsets present in more than one window are considered normal. A framework to monitor the jet gas-turbine was proposed by Clifton *et al.* [20]. In their work, a standard deviation is applied to identify the distance between a test point and its closest cluster center. Clustering is a popular technique, and numerous publications have used this approach [21–24].

Reconstruction-based novelty detection: In this approach, a model is built to represent a "normal" training dataset, and when the test dataset is evaluated on this model, the difference between the test dataset and the output of the model is defined as the reconstruction error. The reconstruction error can be used to define a score to identify and classify the novel concepts. I novelty detection task, numerous approaches using neural network have been proposed [2]. Japkowictz *et al.* [25] present an approach where an autoencoder neural network is used to reproduce the input at the output layer. The autoencoder model is trained just with positive instances, and during the training, classification is possible because the negatives instances are expected to present a higher reconstruction error than the positive instances. Replicator Neural Networks (RNN) are applied in [26, 27]. Generalised radial basis (RBF) neural networks also have been applied for reconstruction-based novelty detection [28–30]. Another interesting neural network approach is presented by Marsland *et al.*. In his work, the novelty detection method uses a self-organizing network that Grows When Required (GWR). Hopfield networks were used in work proposed by Crook *et al.* [31].

Domain-based novelty detection: These methods try to create a boundary that defines a domain around "normal" data. Novel concepts are identified by their location on the boundary. SVMs are a common technique applied in domain-based novelty detection [32–34]. Another approach based in SVM known as support vector data description was introduced in [35]. Support vector regression approach is presented by ma and Perkins in [36]. A variation of SVM called Robust Support Vector Machine (RSVM) was used to solve an over-fit challenge in [37, 38].

Information-theoretic novelty detection: Information-theoretic measures such as entropy are calculated from the "normal" dataset, and these measures are used to classify new concepts. This classification is possible because a novel class should change the information content measures in a dataset. Typically, it is necessary a large number of novel data in the dataset for these methods detect a new concept. This is a disadvantage of this approach where the information-theoretic measure chosen affects deeply the overall performance of the method [4]. Previous works have applied information-theoretic novelty detection methods are described in [39–45]

3 Background

3.1 Novelty Detection in Data Streams

Novelty detection is one of the approaches used in machine learning to classify new data that is different in some aspects from the data used during the training phase. In this technique, a machine learning model is constructed to recognize the "normal" data behavior in scenarios where the counter-examples are scarce, or it is not possible to training models with negative instances. Applications of novelty detection include data streams provided by critical systems where it is common the presence of sizeable "normal" dataset while faults or negative examples are few observations due to the high measurement costs or low frequency. Numerous previous works have addressed novelty detection in off-line data [1, 2, 46], but, this assumption is not valid when we are dealing with data streams. A data stream is a type of data that arrive continuously. Data streams are generated from non-stationary distributions, and the consumer systems do not have control over the order and length of the data. Considering that in data streams scenarios the concepts are continually changing, the application of novelty detection has several challenges [5], which includes concept drift and outlier detection.

3.2 Autoencoders

Autoencoder is a neural network architecture that is trained to try to reproduce its input to its output. The network architecture consists of two parts: an encoder architecture and decoder architecture, where the number of neurons in the first layer of the decoder is equal to the number of neurons in the last layer of the encoder [47]. If an autoencoder is over-fitted, which means it can represent all inputs exactly, the model is not useful. Autoencoders are designed to be able to copy only inputs that can be used to represent the training data, and the model is forced to learn just the important aspects from its inputs. In this way, autoencoders learn the most important properties from the input data. The idea of use autoencoders

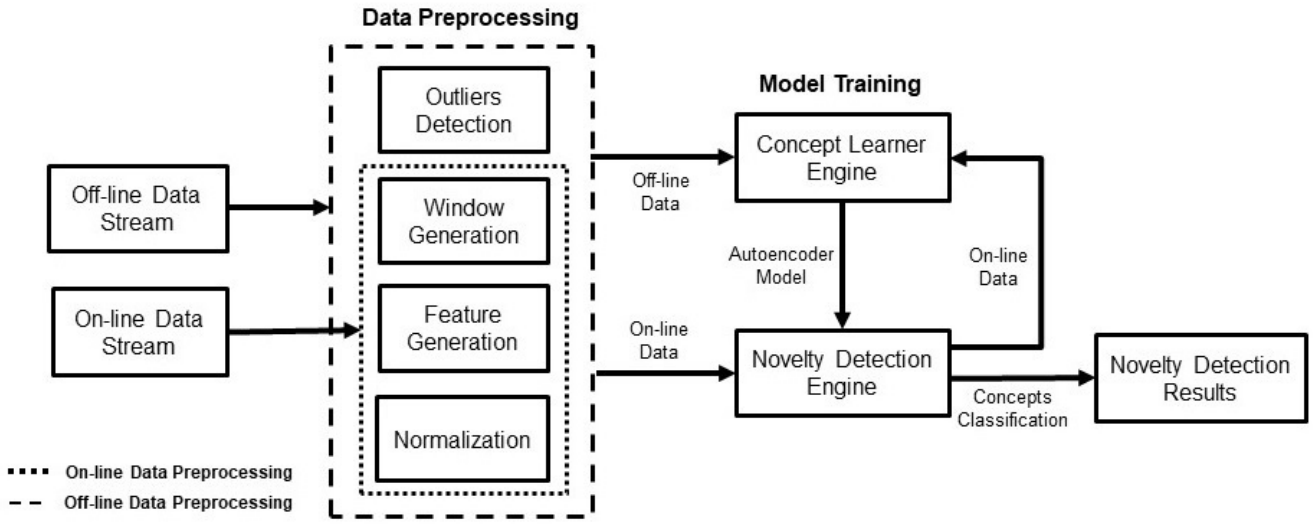

Figure 1: Adaptive on-line novelty detection framework.

to reconstruct positives input instances in a novelty detection application was introduced decades ago by Japkowicz *et al.* [25]

3.3 Numenta Anomaly Benchmark Dataset

The Numenta Anomaly Benchmark is a dataset proposed by Lavin *et al.* [48] with the goal to provide a standard benchmark dataset for the evaluation of anomaly detection algorithms in data streams. In the dataset, there are more than 50 labeled real-world and artificial time-series data files. In this dataset, the perfect anomaly detection algorithm should be able to detect all anomalies in a short time, no false alarms should be triggered, different data domains should be supported, and it should deal automatically with concept drift.

3.4 Important definitions

This section presents the definitions of important concepts that are often correlated with novelty detection.

Anomaly detection and outlier detection: In some scenarios, these terms are used with the same meaning as novelty detection, but, in fact, the three terms can be distinguished. Anomaly detection is a task used to identify an undesired pattern that can happen, and it should be investigated, which means, the unexpected behavior is of interest to the analysis process [49].

Outlier detection is the task to recognize data that it can be responsible for producing imprecise results affecting the systems and models negatively [50]. Outlier detection and anomaly detection are more similar concepts considering their objectives. However, novelty detection has a different goal that identifies a newly emerged concept that needs to be added to the regular pattern [3].

Concept Drift: Concept drift is an event that occurs when there is an important change in the data distribution. Gamma *et al.* [51] identified four types of concept drift: sudden, incremental, gradual, and recurring. In machine learning, concept drift is an essential problem, and there are several application tasks where this issue is relevant such as prediction, classification, detection, and clustering [52]. In this context, novelty detection is a machine learning application task that has to manage concept drift in order to identify and adapt to new concepts. Novelty detection algorithms should be able to cope with concept drift and update their decision model to detect the new classes and changes in the ”normal” behavior. Also, novelty detection applications should deal with outliers and noisy data [3].

4 Methodology

The architecture of the adaptive on-line novelty detection framework is showed in Fig. 1. The architecture is composed of three main components: data preprocessing, concept learner engine, and novelty-detection engine. The components on-line data stream, and off-line data represent the data inputs, while the component novelty detection results refer to the classification outputs of the framework. The three main components are explained below.

4.1 Data Preprocessing

The dataset used in this work was preprocessed with the goal to generate more features that could help to improve the overall efficiency of the novelty detection system. Also, an outlier detection process was conducted with the objective to remove noisy data that could affect the model negatively over the off-line training.

Outlier Detection: In this step, an interquartile test is executed to remove samples that could represent a noisy data. The outlier detection was performed grouping measures from the same hour and time, and the observation was removed if it

was above the upper bound or below the lower bound defined by the following formulas:

$$lowerBound = Q1 - IQR * 1.5 \quad (1)$$

$$upperBound = Q3 + IQR * 1.5 \quad (2)$$

Q1 is the 25th quantile, Q3 is the 75th quantile and IQR is the interquartile interval of the data.

Normalization: Usually, in datasets where the scales for different features are wildly different, normalization is used to facilitate the combination of features, and the converge of neural networks models. The features were normalized into the range [0,1] using Eq. (3) , where **X** is the dataset and N is the number of observations:

$$\tilde{x_n} = \frac{x_n - min(\mathbf{X})}{max(\mathbf{X}) - min(\mathbf{X})} \quad (3)$$

Window Generation: The samples were rearranged by using the time window concept. In this way, each input instance was represented by the sliding windows $w_{i,j}$ of $j = 1, ..., J$ observations. In this paper, $J = 21$ instances, and in each sliding window, three observations were skipped to decrease the similarity between sliding windows samples. [53].

Feature Generation: Temporal contextual features hour and minute of the day were added to the input instances because the data streams measures can have temporal seasonality. These features were transformed using a unit circle representation [53]. This data transformation is essential because date type features are inherently cyclical, and this encoding type preserves the distance between points. Moreover, it is necessary just two additional features per date type feature, while, 1-hot encoding requires 24 and 60 further elements to represent day and minute features.

Statistical features of each sliding window were also generated and used as features. The interquartile range (IQR) and mean were added to augment the features in the input instances. Furthermore, the difference between the means of the $w_{i,j}$ and $w_{(i-1),j}$ sliding windows, and the difference between the means of $w_{(i+1),j}$ and $w_{i,j}$ sliding windows were used to represent the moving average between consecutive sliding windows [54]. The final result of preprocessing process is described in Table 1.

Table 1: Generated features

Feature	Description
Hour - $\{\sin\}$	Circle sine projection with $p_k = 24$
Hour - $\{\cos\}$	Circle cosine projection with $p_k = 24$
Minute - $\{\sin\}$	Circle sine projection with $p_k = 60$
Minute - $\{\cos\}$	Circle cosine projection with $p_k = 60$
$w_{i,j}, j = 1, ..., J$	J - size of sliding window
$\overline{w}_{i,j}$	Mean of CPU measures in each window i
$\overline{w}_{i,j} - \overline{w}_{(i-1),j}$	Difference between the means of i^{th} and $(i-1)^{st}$ sliding window
$\overline{w}_{(i+1),j} - \overline{w}_{i,j}$	Difference between the means of $(i+1)^{st}$ and $i-1^{th}$ sliding window
IQR	Interquartile range of CPU measures in each sliding window

4.2 Concept Learner Engine (CLE)

Concept learner engine is composed of an autoencoder neural network model that is used to reconstruct the off-line and on-line input instances. This component also is responsible for determining the threshold that is used to classify the instances in "normal" or " abnormal." This threshold is used by the novelty detection engine to identify when there is a change in the data pattern, and a new concept needs to be learned. This component is used initially for training the autoencoder model with off-line data, and secondly, it is used over the on-line operation when on-line data is used to training the model.

Off-line Phase: Over the initial learning phase, supervised training is performed, and historical data stream data is used to train the autoencoder model. In this training, only "normal" data is available and the autoencoder is trained to learn the simple objective function $f(\overrightarrow{x}) = \overrightarrow{x}$. After the training, the mean squared error (MSE) of the positives instances are calculated using Eq. (4).

$$MSE = \frac{1}{n}\sum_{i}^{k}(x_i - u_i)^2 \quad (4)$$

In the next step, a threshold is computed based on the sum of the mean and standard deviation of MSE of the positive instances. The threshold can classify unlabeled examples $\overrightarrow{x}$ over the on-line training because the MSE for positive instances should be low, while on-line inputs that never have been trained in the autoencoder should have a higher MSE. The mode how the threshold affects the data classification can be seen in Fig. (2), where the receiver characteristic curve (ROC) curve is plotted. The ROC curve plots the true positive rate (TPR) and the false positive rate (FPR) according to all possible thresholds. Higher and lower thresholds are not desired because they increase the number of false positive and false negative, respectively.

On-line Phase: In on-line phase, the concept learner engine is trained with on-line data when the novelty detection engine identifies a change in the data pattern and the model to be trained to recognize a new concept in the data stream. Over this step, on-line training is performed with on-line data sent by the novelty detection engine. After the on-line training, the threshold is calculated again based on the sum of the mean and standard deviation of the MSE of the data used in the on-line learning.

4.3 Novelty Detection Engine (NVE)

The novelty detection engine detects if there is a change in the data pattern and a new concept is emerging on the on-line dataset, which is also known as concept drift [51]. If a change in the data pattern is identified, the NVE triggers a request to start on-line training in the CLE component using the on-line data. The NVE use a fixed window to define if on-line training should be performed. Let a be some consecutive abnormal behavior identified on the on-line dataset, and p be a parameter chosen to represent the number of consecutive abnormal behavior that defines a change in the data pattern. Over the on-line operation, if a is equal to p, the novelty detection engine triggers on-line training to be executed by CLE. The number of input instances used over the on-line training is equal to the parameter p.

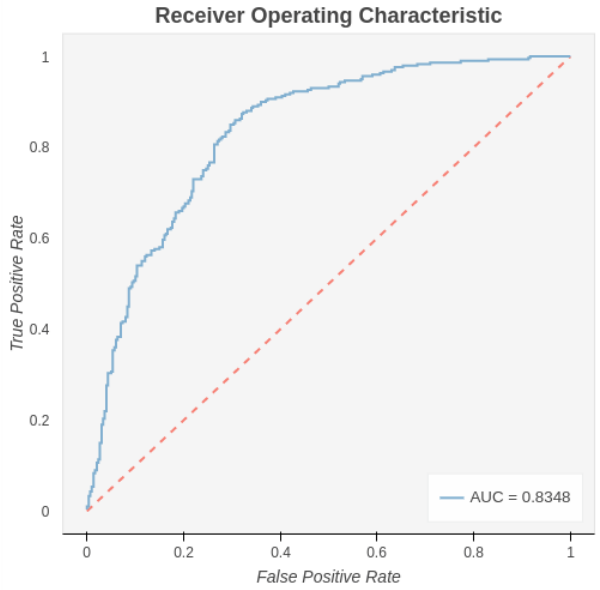

Figure 2: ROC Curve after of-line training

5 Experimental Results and Discussion

The proposed adaptive on-line novelty detection framework has been evaluated using an anomaly detection benchmark dataset called NAB. NAB dataset is composed of more than 50 different data files, but in this work, one specific file with measures from CPU utilization was used. The data used in the experiments contained 4010 samples. The measures were collected every five minutes from a real server hosted on Amazon AWS service.

The dataset was split into three parts. 50% of the data was used for the off-line training, 15% was used for the test, and 35% was used to simulate on-line operation. Over the initial off-line learning, the outliers were removed, and feature generation and normalization were performed on the off-line data. In the test and on-line data, only feature generation and normalization were performed.

Table 2: Autoencoder model settings

Layer	Neurons	Activation Function
Encoder L1	26	Relu
Encoder L2	13	Relu
Encoder L3	6	Relu
Decoder L1	6	Relu
Decoder L2	13	Relu
Decoder L3	26	Sigmoid

In the off-line data (2010 samples), the dataset only contains samples from which is considered "normal" behavior. In the on-line operation, the data pattern change to a new pattern and this new data pattern remains in the next 800 samples. In the last part of the on-line dataset, the data changes the pattern again. In this way, the last 600 measures have a similar pattern to the off-line data. Overall, the dataset presents two changes in the data distribution, and this was important to validate if the novelty detection framework could adapt to these changes during the on-line training. The changes in the data distribution are large enough to be noted graphically in the Fig. (3).

5.1 Implementation Details

The implementation was developed in Python using Keras and Tensorflow. The autoencoder network was build using three layers for the encoder and three layers for the decoder, as detailed in Table 2.

A dropout rate of 0.75 and L1 regularization rate of 10e-5 was used on all dense layers. The neural network weights were initialized using Xavier initialization [55]. In off-line training, autoencoder model was trained for 20 epochs with a batch size of 20, and during on-line training, the number of epochs used was 20 with a batch size of 5.

In the novelty detection engine, the number of consecutive abnormal behaviors detected to trigger on-line training was 6. This parameter was chosen empirically through successive tests and evaluations. It was found the framework can work with higher and lower values than the value chosen, and trade-offs among quantity of on-line training, false alarms rate, time to detection and so on should be analyzed with more details in future works.

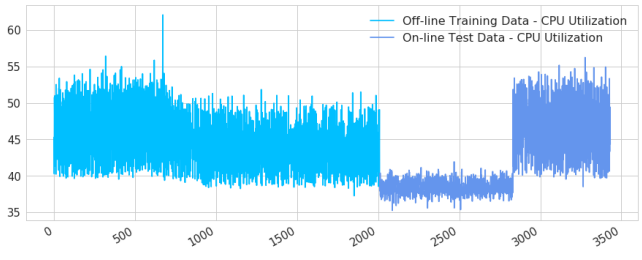

Figure 3: Off-line and On-line Data

5.2 Experiments

The experiments involved the training of model with off-line data and simulation of an on-line operation. Considering that the data distribution changed two times in the on-line dataset, three evaluations were performed to measure the framework results during its operation.

The test dataset used in the evaluations were composed of 600 samples. The data present in the test dataset represent the two types of data distribution present in the off-line and on-line dataset. In this way, the test dataset can be used to evaluate the framework in different moments, because it always has data that represent the normal behavior and the abnormal behavior. Before the evaluations, the samples in the test dataset were classified in normal and abnormal, and then, the evaluation was performed.

The first evaluation was performed after the off-line training. In the off-line dataset, the instances have the same data distribution, and there is no a change in the data pattern. When the on-line data start to arrive, the next 800 samples have a different data distribution if we compare with off-line data. In this way, it expected the autoencoder model changes along the time to adapt to this new pattern. Therefore, the second evaluation was performed when the sample 801st arrives, and the data distribution starts to change again, how it is possible to check in Fig. (4).

In the last part of the on-line dataset, from the 801st sample until the end of the data, there is a new change in the data pattern. A third evaluation is performed at the end of on-line training to check if the framework was able to cope with this last change.

In all evaluations, the objective was to examine the precision, recall, and accuracy of the framework. Recall, Eq. (5), also known as sensitivity, is a metric that represents the true positive rate of the model (TPR). Precision, Eq. (6), also called positive predictive values, represents the rate of relevant predicted instances. Accuracy represents the rate of correct predictions made by the model.

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

$$Precision = \frac{TP}{TP+FP} \tag{6}$$

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{7}$$

5.3 Results and Discussion

The performance of the framework was evaluated at three different moments, considering when the changes occurred in the data pattern and a novelty concept should be detected. The evaluation was performed using a test dataset that was not used on the off-line and on-line training.

From Table 3 it is possible to conclude, overall, the framework achieved an excellent performance. The evaluation 1, performed after the off-line training showed the framework presented a great accuracy rate.

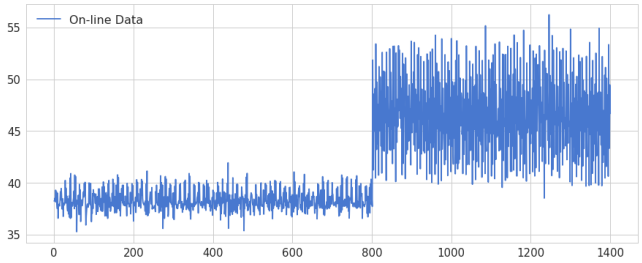

Figure 4: On-line Data

In the evaluation 2, the result reached was outstanding considering the framework learned how to classify the new data distribution during the on-line training. The accuracy rate achieved in the evaluation 2 was 0.94, which means the framework was able to detect a change in the data distribution and the on-line training was effective in adapting the model to the new concept. When the on-line operation started until the moment of evaluation 2, the framework trained the autoencoder model 9 times, Fig. (5). This result is interesting, and it shows the auto-encoder model was able to adapt to the change in the data pattern with a low number of retraining.

Notable is the drop in the accuracy and precision of the evaluation 3. The poor performance of the framework in this evaluation is most likely because it seems the autoencoder model did not "forget" the data pattern learned in the 800 first samples from the on-line data. The autoencoder model did not adapt so well to the pattern presented in the last 600 samples and because of that more instances are classified as false positive, which affects the precision and the overall accuracy of the model. From the 801st on-line sample until the end of the on-line data, the autoencoder was trained five times. The number of training is lower if we compare with the number of training performed to adapt to the old pattern and this point can be one of the reasons for the poor performance in the evaluation 3.

6 Conclusion and Future Work

In this research, an adaptive on-line novelty detection has been proposed. The framework uses off-line supervised training and on-line learning to create and update an autoencoder model that can learn a specific data pattern and change itself during on-line learning to adapt to a change in the data distribution. The results show that this framework can successfully to learn a data pattern and learn a new concept during on-line

Table 3: Evaluation Comparison

Evaluation	Recall	Precision	Accuracy
1	0.86	0.83	0.84
2	0.90	0.98	0.94
3	0.86	0.67	0.71

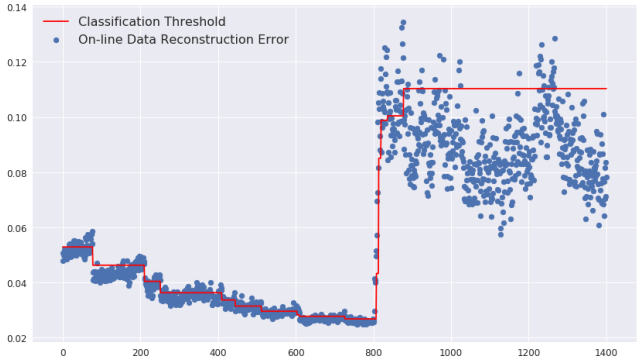

Figure 5: The changes in the red line indicates that a on-line training was performed and a new threshold was calculated. The blue points represent the reconstruction error of each input produced by the autoencoder model.

operation. Future works will explore improving the novelty detection engine, where an advanced heuristic to identify a change in the data pattern can be explored. Moreover, the framework can be compared with others on-line novelty detection techniques, and new datasets will be evaluated against the proposed framework.

REFERENCES

[1] Markos Markou and Sameer Singh. Novelty detection: a reviewpart 1: statistical approaches. *Signal processing*, 83(12):2481–2497, 2003.

[2] Markos Markou and Sameer Singh. Novelty detection: a reviewpart 2:: neural network based approaches. *Signal processing*, 83(12):2499–2521, 2003.

[3] Elaine R Faria, Isabel JCR Gonçalves, André CPLF de Carvalho, and João Gama. Novelty detection in data streams. *Artificial Intelligence Review*, 45(2):235–269, 2016.

[4] Marco AF Pimentel, David A Clifton, Lei Clifton, and Lionel Tarassenko. A review of novelty detection. *Signal Processing*, 99:215–249, 2014.

[5] Joao Gama. *Knowledge discovery from data streams*. CRC Press, 2010.

[6] Jarmo Ilonen, Pekka Paalanen, J-K Kamarainen, and H Kalviainen. Gaussian mixture pdf in one-class classification: computing and utilizing confidence values. In *Pattern Recognition, 2006. ICPR 2006. 18th International Conference on*, volume 2, pages 577–580. IEEE, 2006.

[7] Xiuyao Song, Mingxi Wu, Christopher Jermaine, and Sanjay Ranka. Conditional anomaly detection. *IEEE Transactions on Knowledge and Data Engineering*, 19(5):631–645, 2007.

[8] F Zorriassatine, A Al-Habaibeh, RM Parkin, MR Jackson, and J Coy. Novelty detection for practical pattern recognition in condition monitoring of multivariate processes: a case study. *The International Journal of Advanced Manufacturing Technology*, 25(9-10):954–963, 2005.

[9] D Erdogmus, R Jenssen, YN Rao, and JC Principe. Multivariate density estimation with optimal marginal parzen density estimation and gaussianization. In *Machine Learning for Signal Processing, 2004. Proceedings of the 2004 14th IEEE Signal Processing Society Workshop*, pages 73–82. IEEE, 2004.

[10] Emanuel Parzen. On estimation of a probability density function and mode. *The annals of mathematical statistics*, 33(3):1065–1076, 1962.

[11] Sharmila Subramaniam, Themis Palpanas, Dimitris Papadopoulos, Vana Kalogeraki, and Dimitrios Gunopulos. Online outlier detection in sensor data using non-parametric models. In *Proceedings of the 32nd international conference on Very large data bases*, pages 187–198. VLDB Endowment, 2006.

[12] Stephen D Bay and Mark Schwabacher. Mining distance-based outliers in near linear time with randomization and a simple pruning rule. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 29–38. ACM, 2003.

[13] Fabrizio Angiulli and Clara Pizzuti. Fast outlier detection in high dimensional spaces. In *European Conference on Principles of Data Mining and Knowledge Discovery*, pages 15–27. Springer, 2002.

[14] Shyam Boriah, Varun Chandola, and Vipin Kumar. Similarity measures for categorical data: A comparative evaluation. In *Proceedings of the 2008 SIAM International Conference on Data Mining*, pages 243–254. SIAM, 2008.

[15] Markus M Breunig, Hans-Peter Kriegel, Raymond T Ng, and Jörg Sander. Lof: identifying density-based local outliers. In *ACM sigmod record*, volume 29, pages 93–104. ACM, 2000.

[16] Varun Chandola, Shyam Boriah, and Vipin Kumar. Understanding categorical similarity measures for outlier detection. *Technical report 08–008, University of Minnesota*, 2008.

[17] Ji Zhang and Hai Wang. Detecting outlying subspaces for high-dimensional data: the new task, algorithms, and performance. *Knowledge and information systems*, 10(3):333–355, 2006.

[18] Zeeshan Syed, Mohammed Saeed, and Ilan Rubinfeld. Identifying high-risk patients without labeled training data: anomaly detection methodologies to predict adverse outcomes. In *AMIA Annual Symposium Proceedings*, volume 2010, page 772. American Medical Informatics Association, 2010.

[19] Daniel Barbará, Yi Li, Julia Couto, Jia-Ling Lin, and Sushil Jajodia. Bootstrapping a data mining intrusion detection system. In *Proceedings of the 2003 ACM symposium on Applied computing*, pages 421–425. ACM, 2003.

[20] David A Clifton, Peter R Bannister, and Lionel Tarassenko. Learning shape for jet engine novelty detection. In *International Symposium on Neural Networks*, pages 828–835. Springer, 2006.

[21] Zengyou He, Xiaofei Xu, and Shengchun Deng. Discovering cluster-based local outliers. *Pattern Recognition Letters*, 24(9-10):1641–1650, 2003.

[22] Suet-Peng Yong, Jeremiah D Deng, and Martin K Purvis. Novelty detection in wildlife scenes through semantic context modelling. *Pattern Recognition*, 45(9):3439–3450, 2012.

[23] Dantong Yu, Gholamhosein Sheikholeslami, and Aidong Zhang. Findout: finding outliers in very large datasets. *Knowledge and Information Systems*, 4(4):387–412, 2002.

[24] Kejia Zhang, Shengfei Shi, Hong Gao, and Jianzhong Li. Unsupervised outlier detection in sensor networks using aggregation tree. In *International Conference on Advanced Data Mining and Applications*, pages 158–169. Springer, 2007.

[25] Nathalie Japkowicz, Catherine Myers, Mark Gluck, et al. A novelty detection approach to classification. In *IJCAI*, volume 1, pages 518–523, 1995.

[26] Simon Hawkins, Hongxing He, Graham Williams, and Rohan Baxter. Outlier detection using replicator neural networks. In *International Conference on Data Warehousing and Knowledge Discovery*, pages 170–180. Springer, 2002.

[27] Graham Williams, Rohan Baxter, Hongxing He, Simon Hawkins, and Lifang Gu. A comparative study of rnn for outlier detection in data mining. In *Data Mining, 2002. ICDM 2003. Proceedings. 2002 IEEE International Conference on*, pages 709–712. IEEE, 2002.

[28] Christopher M Bishop. Novelty detection and neural network validation. *IEE Proceedings-Vision, Image and Signal processing*, 141(4):217–222, 1994.

[29] Alexandre Nairac, Timothy A Corbett-Clark, Ruth Ripley, Neil W Townsend, and Lionel Tarassenko. Choosing an appropriate model for novelty detection. 1997.

[30] Yuhua Li, Michael J Pont, and N Barrie Jones. Improving the performance of radial basis function classifiers in condition monitoring and fault diagnosis applications whereunknown'faults may occur. *Pattern Recognition Letters*, 23(5):569–577, 2002.

[31] Paul A Crook, Stephen Marsland, Gillian Hayes, and Ulrich Nehmzow. A tale of two filters-on-line novelty detection. In *Robotics and Automation, 2002. Proceedings. ICRA'02. IEEE International Conference on*, volume 4, pages 3894–3899. IEEE, 2002.

[32] Larry M Manevitz and Malik Yousef. One-class svms for document classification. *Journal of machine Learning research*, 2(Dec):139–154, 2001.

[33] Christopher P Diehl and John B Hampshire. Real-time object classification and novelty detection for collaborative video surveillance. In *Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on*, volume 3, pages 2620–2625. IEEE, 2002.

[34] Bernhard Schölkopf, Robert C Williamson, Alex J Smola, John Shawe-Taylor, and John C Platt. Support vector method for novelty detection. In *Advances in neural information processing systems*, pages 582–588, 2000.

[35] David MJ Tax and Robert PW Duin. Support vector domain description. *Pattern recognition letters*, 20(11-13):1191–1199, 1999.

[36] Junshui Ma and Simon Perkins. Online novelty detection on temporal sequences. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 613–618. ACM, 2003.

[37] Qing Song, Wenjie Hu, and Wenfang Xie. Robust support vector machine with bullet hole image classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 32(4):440–448, 2002.

[38] Wenjie Hu, Yihua Liao, and V Rao Vemuri. Robust anomaly detection using support vector machines. In *Proceedings of the international conference on machine learning*, pages 282–289, 2003.

[39] Zengyou He, Shengchun Deng, and Xiaofei Xu. An optimization model for outlier detection in categorical data. In *International Conference on Intelligent Computing*, pages 400–409. Springer, 2005.

[40] Zengyou He, Shengchun Deng, Xiaofei Xu, and Joshua Zhexue Huang. A fast greedy algorithm for outlier mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 567–576. Springer, 2006.

[41] Shin Ando. Clustering needles in a haystack: An information theoretic analysis of minority and outlier detection. In *Data Mining, 2007. ICDM 2007. Seventh IEEE International Conference on*, pages 13–22. IEEE, 2007.

[42] Michael Gamon. Graph-based text representation for novelty detection. In *Proceedings of the First Workshop on Graph Based Methods for Natural Language Processing*, pages 17–24. Association for Computational Linguistics, 2006.

[43] Maurizio Filippone and Guido Sanguinetti. Information theoretic novelty detection. *Pattern Recognition*, 43(3):805–814, 2010.

[44] Maurizio Filippone and Guido Sanguinetti. A perturbative approach to novelty detection in autoregressive models. *IEEE Transactions on Signal Processing*, 59(3):1027–1036, 2011.

[45] Laurent Itti and Pierre F Baldi. Bayesian surprise attracts human attention. In *Advances in neural information processing systems*, pages 547–554, 2006.

[46] Stephen Marsland, Jonathan Shapiro, and Ulrich Nehmzow. A self-organising network that grows when required. *Neural networks*, 15(8-9):1041–1058, 2002.

[47] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[48] Alexander Lavin and Subutai Ahmad. Evaluating real-time anomaly detection algorithms–the numenta anomaly benchmark. In *Machine Learning and Applications (ICMLA), 2015 IEEE 14th International Conference on*, pages 38–44. IEEE, 2015.

[49] Varun Chandola, Arindam Banerjee, and Vipin Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[50] Prasanta Gogoi, DK Bhattacharyya, Bhogeswar Borah, and Jugal K Kalita. A survey of outlier detection methods in network anomaly identification. *The Computer Journal*, 54(4):570–588, 2011.

[51] João Gama, Raquel Sebastião, and Pedro Pereira Rodrigues. On evaluating stream learning algorithms. *Machine learning*, 90(3):317–346, 2013.

[52] Indrė Žliobaitė, Mykola Pechenizkiy, and Joao Gama. An overview of concept drift applications. In *Big Data Analysis: New Algorithms for a New Society*, pages 91–114. Springer, 2016.

[53] Norman L Tasfi, Wilson A Higashino, Katarina Grolinger, and Miriam AM Capretz. Deep neural networks with confidence sampling for electrical anomaly detection. In *Proc. of the 3rd International Conference on Smart Data (SmartData-2017)*, 2017.

[54] Daniel B Araya, Katarina Grolinger, Hany F ElYamany, Miriam AM Capretz, and G Bitsuamlak. Collective contextual anomaly detection framework for smart buildings. In *Neural Networks (IJCNN), 2016 International Joint Conference on*, pages 511–518. IEEE, 2016.

[55] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010.