Autoencoders: A Low Cost Anomaly Detection Method for Computer Network Data Streams

Christopher Nixon
Staffordshire University
Stoke-on-Trent, UK
nr106584@student.staffs.ac.uk

Mohamed Sedky
Staffordshire University
Stoke-on-Trent, UK
M.H.Sedky@staffs.ac.uk

Mohamed Hassan
Staffordshire University
Stoke-on-Trent, UK
Mohamed.Hassan@staffs.ac.uk

ABSTRACT

Computer networks are vulnerable to cyber attacks that can affect the confidentiality, integrity and availability of mission critical data. Intrusion detection methods can be employed to detect these attacks in real-time. Anomaly detection offers the advantage of detecting unknown attacks in a semi-supervised fashion. This paper aims to answer the question if autoencoders, a type of semi-supervised feed-forward neural network, can provide a low cost anomaly detector method for computer network data streams. Autoencoder methods were evaluated online with well known KDD Cup 1999 and UNSW-NB15 data sets, and it was demonstrated that running time and labeling cost is significantly reduced compared to traditional online classification techniques for similar detection performance. Further research would consider the trade-off between single vs stacked networks, multi-label classification, concept drift detection and active learning.

CCS CONCEPTS

• **Computer systems organization** → **Embedded systems**; *Redundancy*; Robotics; • **Networks** → Network reliability.

KEYWORDS

anomaly detection, intrusion detection, autoencoders, online learning

ACM Reference Format:
Christopher Nixon, Mohamed Sedky, and Mohamed Hassan. 2018. Autoencoders: A Low Cost Anomaly Detection Method for Computer Network Data Streams. In *Proceedings of 2020 4th International Conference on Cloud and Big Data Computing (ICCBDC 2020).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/1122445.1122456

1 INTRODUCTION

Computer networks are vulnerable to cyber attacks that threaten the confidentiality, integrity and availability of mission critical data. Networks are comprised of diverse typologies, protocols and devices that are open to a wide range of vulnerabilities across relevant use cases including, but not limited to: industrial control systems, public cloud and Internet of Things (IoT) networks.

ICCBDC 2020, June 03–05, 2018, Liverpool, UK

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/1122445.1122456

Intrusion detection systems (IDS) allow for both known (*misuse*) and unknown (*anomaly*) detection of cyber attacks [4]. IDS can use *machine learning* (ML) methods to automatically map the posterior probability of a normal or attack class y given an observation X: $p(y|X)$. A common challenge is that computer networks can be considered to be a continuous, infinite, data stream. Over time, the data stream will change due to changes in the underlying network or new attack types, affecting the posterior probability, this is known as *real concept drift* [17]. Other applications of ML rely on batch or *offline* training of the model, whereby the model is trained and validated on the whole data set, potentially more than once. This is not possible with data streams whereby each data observation should be processed only once in order to conserve memory. Additionally when processing a data stream in real-time, the running time of the model is important to provide timely detection. Processing of data streams is broadly known as *online learning* [6], and requires new methods to evaluate performance.

Anomaly detection offers advantages over misuse detection in both the ability to detect unknown attacks and to operate in a semi-supervised fashion, whereby only normal examples are required to train the model. *Autoencoders* are a type of feed-forward neural network that can be arranged to perform anomaly detection by learning a stochastic representation of the input 'normal' samples and further detecting abnormal samples by monitoring the *reconstruction error*: $x - \hat{x}$, compared to a predetermined anomaly threshold [2].

The aim of this research is to evaluate Autoencoders as anomaly detectors with well-known IDS data sets, comparing performance to well documented online classification methods, in order to answer the question: do autoencoders provide a low cost anomaly detection method for computer network data streams? Here low cost refers to the running time and labeling effort required to effectively detect attacks within a stream of network data. The primary contribution of this research is the demonstration of autoencoder methods as a low cost anomaly detector for data streams.

The remainder of this paper is organised as follows: section 2, introduces autoencoder anomaly detection core concepts; section 3, summarises related autoencoder IDS studies; section 4, describes the autoencoder methods that were evaluated; section 5, presents the evaluation results for KDD Cup '99 and UNSW-NB15 data sets; section 6, discusses how autoencoders provide a low cost anomaly detector for network data streams; and section 7, presents conclusions.

2 AUTOENCODER ANOMALY DETECTION

An autoencoder is a type of unsupervised feed-forward neural network that aims to produce a stochastic representation of its

input data. Typically the network layout is symmetric consisting of an *encoding* function: $h = f(x)$ that encodes the input x into its code representation h, and a *decoding* function: $\hat{x} = g(h)$ that decodes h into the approximate reconstructed output $\hat{x}$ [7]. Figure 1 shows both an *undercomplete* network (a), whereby the hidden nodes are fewer than the visible ones to provide a compressed version of the input, and a *stacked* network (b), whereby there are multiple hidden layers, the *bottleneck layer* representing the encoded form of the input, in order to provide a better approximation. The network can be trained using *back-propagation* optimisation methods such as Stochastic Gradient Descent (SGD), or adaptive methods such as Adagrad.

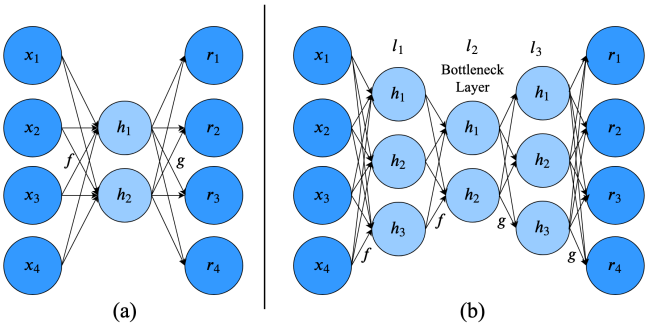


Figure 1: (a) Undercomplete Autoencoder. (b) Stacked Autoencoder

.

The network can be useful for both feature reduction and anomaly detection. Anomaly detection involves calculating the reconstruction error (RE) for a given observation x and comparing to a pre-determined anomaly threshold to determine if the observation belongs to the 'normal' or 'anomaly' class. Typically RE can be calculated as the Mean Squared Error (MSE) of the output nodes from the original input, given in equation 1.

$$\mathrm{RE} = \frac{1}{n}\sum_{j=1}^{n}(x_j - \hat{x}_j)^2 \tag{1}$$

An anomaly detection function a, (equation 2), compares the RE to the anomaly threshold ϕ and a sensitivity parameter β to determine whether to signal an anomaly [11]. In this paper sensitivity β was set to 1.18, in order to reduce false positives.

$$a(\mathrm{RE}) = \begin{cases} 1, & \text{if RE} \geq \phi\beta \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

3 RELATED WORK

Aygun and Yavuz [2] utilised a single layer Denoising Autoencoder (DAE) whereby the input data is partially corrupted to achieve a better stochastic representation. The anomaly threshold was determined using a novel Stochastic Anomaly Threshold (SAT) method, described in section 4.2.2 of this paper. The model was evaluated offline with the NSL-KDD data set, with an accuracy of 88.65% with 10% corruption of the input data.

Nicolau and McDermott [14] combined an autoencoder with the Kernel Density Estimation (KDE) method to provide an anomaly threshold based on the density model, comparing performance to an arbitrary threshold based on RE, with NSL-KDD multi-class ROC-AUC performance of 0.974, 0.891, 0.945 and 0.987 (DoS, R2L, USR and Probe) compared to 0.459 with the RE method.

Mirza and Cosan [12] built an autoencoder network with Long Short Term Memory (LSTM) layers, allowing for network packets represented as hexadecimal sequences to be evaluated. Testing on the ISCX IDS 2012 data set yielded an f1-score of 0.854.

Mirsky et al. [11] proposed Kitsune for online anomaly detection, an ensemble of autoencoders, bagging the feature space by clustering based on correlation distance. The network is stacked so that the output RE of each member is provided as an input to the *output* autoencoder to discriminate differences in the member RE values to detect an anomaly. Performance was evaluated using a novel data set, reporting ROC-AUC ranging from 0.58 to 0.99.

Chen et al. [5] developed RandNet, an ensemble of sparsely connected autoencoder networks in order to introduce variance between members. KDD Cup 1999 data set ROC-AUC performance was between 0.9 to 1.0.

Kieu et al. [8] expanded on the sparsity method of Chen et al. [5], incorporating sparsely connected Recurrent Neural Network (RNN) layers with each member to represent series learning. This was not evaluated using an IDS data set but was compared to RandNet across a number of alternate data sets, where greater performance is reported.

Li et al. [9] expanded on the Kitsune online ensemble method [11], using random forest for feature selection and GMM/K-Means to normalise the output RE which is calculated based on the Root Mean Squared Error (RMSE). The claim is that using random forest for feature selection is more efficient than the clustering approach used with Kitsune. The method was evaluated with the CSE-CIC-IDS 2018 data set, with ROC-AUC ranging from 0.538 to 1.0.

Alam et al. [1] combined online autoencoder with memristors to provide a power efficient anomaly detector at the cost of a reduction in accuracy, evaluated with the NSL-KDD dataset, achieving an accuracy of 92.91%.

4 EVALUATION METHODS

The aim of this paper was to evaluate the autoencoder method as a low cost anomaly detector for online computer network data streams. An objective was to achieve a simplistic, low cost, method that can easily fit into further online frameworks as required. Therefore the following aspects were evaluated: hidden node size and dropout probability, anomaly detection threshold determination methods, and single vs stacked networks. For comparison the results were further compared to well documented online learning classification methods: Naïve Bayes and Hoeffding Adaptive Tree, using the KDD Cup 1999 10% and UNSW-NB15 IDS data sets [15].

4.1 Dropout Probability

Dropout can be used within the network to reduce over fitting and allow a better stochastic representation to be achieved. During training the input of a hidden unit h for layer $l+1$ is included based on a Bernoulli distribution, as shown in equation 3, where parameter

p determines the probability of the output y of the current layer being thinned [16].

$$\begin{aligned} r_j^{(l)} &\sim \text{Bernoulli}(p) \\ \tilde{y}^{(l)} &= r^{(l)} * y^l \\ h_i^{(l+1)} &= w_i^{(l+1)} \tilde{y}^l + b_i^{(l+1)} \\ y_i^{(l+1)} &= f(h_i^{(l+1)}) \end{aligned} \tag{3}$$

The optimal dropout probability p and hidden units factor h were found using an exhaustive search, observing ROC-AUC on the NSL-KDD data set. A value of $p = 0.1, h = 0.6$ were selected, based on a ROC-AUC of 0.956

4.2 Anomaly Thresholds

4.2.1 Naïve Threshold Method with Decay. Naïve Anomaly Threshold (NAT) method [11] selects the maximum observed RE during training as the anomaly threshold. When evaluated with data streams, the initial RE is high and reduces overtime with training, this can incorrectly fix the threshold as high compared to the remaining distribution of RE samples, affecting performance. To correct this a decay factor α is used to gradually reduce the threshold overtime, given in algorithm 1. During evaluation α was set to 0.9.

Algorithm 1: Naïve Anomaly Threshold with Decay

Input : model m, X, y, threshold ϕ, decay $\alpha \leftarrow [0, 1]$
Output: ϕ

$X_{y \leftarrow 0} \subseteq X$;
RE $\leftarrow$ `predictRE(`$m, X_{y \leftarrow 0}$`)`;
if `MAX(`RE`)` $> \phi$ **then**
  $\phi \leftarrow$ `MAX(`RE`)`;
else
  $\phi \leftarrow \phi\alpha$
end

4.2.2 Stochastic Anomaly Threshold. Stochastic Anomaly Threshold (SAT) [2] finds the threshold between the mean and 3 * standard deviation of the normal sample distribution based on the observed accuracy. The method has been improved slightly from [2] to avoid storing all candidate thresholds by comparing against the previous highest accuracy acc_w during each iteration, given in algorithm 2. Note this method requires full labels. The step size v was set to 0.001 for this evaluation.

4.3 Single vs Stacked Network

A single layer network consists of one hidden layer, providing a shallow representation of the input data. This should mean fewer parameter updates and faster running times. A stacked network of 3 and 5 hidden layers was also compared, this should be able to better approximate the input data [7] to improve detection performance at the cost of memory and running time. For the single layer network the non-linear Sigmoid activation was used. For stacked networks, the linear ReLU function was used for the inner-most layers and Sigmoid for the outer layers in order to balance their individual large and small gradient update weaknesses [5].

All networks were trained using the Adagrad optimiser.

Algorithm 2: Stochastic Anomaly Threshold

Input : model m, X, y, threshold ϕ, step size $v \leftarrow (> 0)$
Output: ϕ

$X_{y \leftarrow 0} \subseteq X$;
RE $\leftarrow$ `predictRE(`$m, X_{y \leftarrow 0}$`)`;
$\phi \leftarrow \mu_{\text{RE}} + 3\sigma_{\text{RE}}$;
$\phi_w \leftarrow \phi$;
$\text{acc}_w \leftarrow -1$;
while $\phi > \mu_{RE}$ **do**
  $\hat{y} \leftarrow$ `predict(`m,ϕ,X`)`;
  acc $\leftarrow$ `calcAccuracy(`$\hat{y},y$`)`;
  if $acc > acc_w$ **then**
    $\phi_w \leftarrow \phi$;
    $\text{acc}_w \leftarrow$ acc;
  end
  $\phi \leftarrow \phi - v$;
end
$\phi \leftarrow \phi_w$;

4.4 Prequential Evaluation

To better assess the performance of online learning methods on data streams, a prequential evaluation can be used whereby the model is first tested on a new, previously unseen, data sample, or chunk of data samples, before then training on that sample [6]. The models were pre-trained on a number of samples prior to commencing prequential evaluation, whereby the data stream was then presented in 'chunks' of 100 samples at a time. Each chunk resulted in a network parameter update.

Both KDD Cup 1999 and UNSW-NB15 data sets were preprocessed using nominal to binary and min-max normalisation filtering. UNSW-NB15 training and test data sets were concatenated into one unshuffled data stream.

The Keras[1] neural networking, version 2.3.1, and Scikit-Multiflow[2] stream learning [13], version 0.4.1, frameworks for Python were used for this evaluation. Evaluations were ran on a Windows 10 64bit PC with Intel i7 1.8GHz processor and 8GB RAM.

Observed metrics during evaluation included: accuracy, f1-score and total running time. For prequential evaluation the scikit-multiflow default of updating evaluation metrics every 200 samples was used.

5 EVALUATION RESULTS

5.1 KDD Cup 1999

Autoencoder (AE) anomaly methods using both the Naïve Anomaly Threshold with Decay (NATD) and Stochastic Anomaly Threshold (SAT) were evaluated against the KDD Cup 1999 data set, along

[1] https://keras.io/
[2] https://scikit-multiflow.github.io/

with Naïve Bayes (NB) and Hoeffding Adaptive Tree (HAT) online classifiers. The results are provided in table 1, and the rolling average accuracy and F1 plotted for the data stream in figure 2.

The results of the AE SAT algorithm are comparable with HAT for both accuracy and F1 measures. The total running time for AE is significantly shorter than either NB and HAT, offering a much lower cost method for comparable detection performance, with a throughput of 19,368 samples per second. Note that the running time is highly dependent on the underlying system architecture and the algortithm implementation. For this evaluation all models were ran on the same architecture and NB and HAT algorithms using the scikit-multiflow default algorithms.

KDD Cup 1999 10% consists of 97,277 normal samples and 396,743 anomaly samples. AE only needs to train on the normal samples, representing 19.7% of the total data set, thus lowering the cost for equivalent performance.

Table 1: KDD Cup 1999 Results

Algorithm	Accuracy	F1 Score	Run Time (s)
AE SAT	0.980	**0.812**	25.3
AE NATD	0.954	0.776	**23.4**
NB	0.933	0.810	510.9
HAT	**0.986**	0.811	794.8

5.2 UNSW-NB15

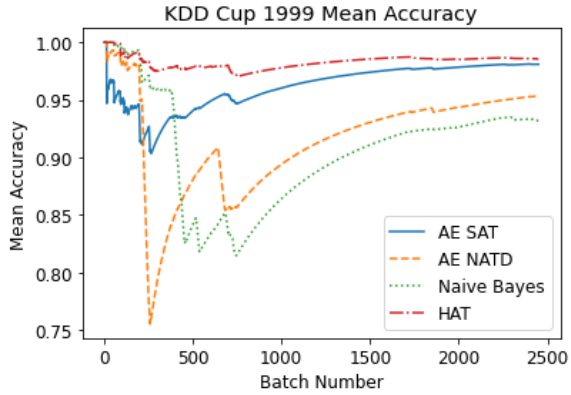
AE SAT was further evaluated with the UNSW-NB15 data set. This data set proved to be more challenging, with normal and anomaly samples bearing closer similarities, evidenced by the overlapping RE distributions shown in figure 4. To better separate normal and anomaly samples, stacked autoencoders were evaluated, (table 2), with a 3-layer network and dropout probability of 0.2 providing better results. Running time was increased by 70% compared to a singler layer network, although this is still much lower than NB and HAT methods. The sample rate of the single layer network was 21,289/s and 12,505/s for the 3-layer network. The rolling average accuracy and F1 scores are plotted in figure 3.

Table 2: UNSW-NB15 Results

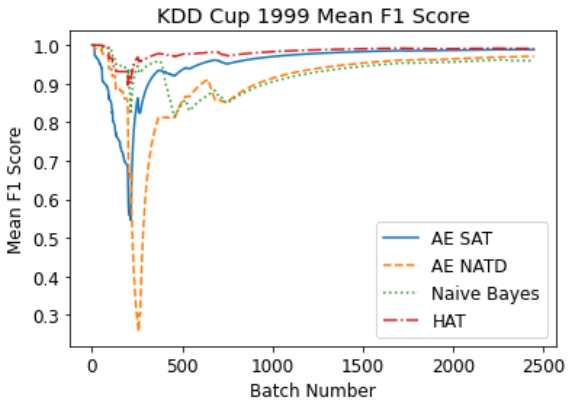
Algorithm	Accuracy	F1 Score	Run Time (s)
AE SAT L=1	0.739	0.613	**12.1**
AE SAT L=3	0.783	0.697	18.9
AE SAT L=5	0.736	0.644	20.1
AE SAT L=3, p=0.2	**0.791**	**0.703**	20.6
NB	0.837	0.832	350.4
HAT	0.929	0.813	610.9

6 DISCUSSION

Autoencoders can provide a low cost anomaly detection method for computer network data streams, and this has been demonstrated with the well known KDD Cup 1999 and UNSW-NB15 data sets.



(a) Mean Accuracy



(b) Mean F1 Score

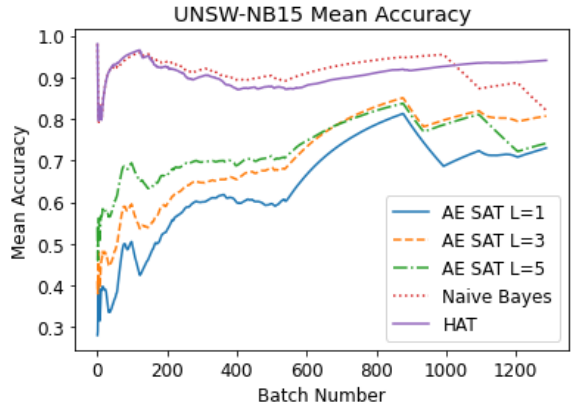
Figure 2: KDD Cup 1999 Evaluation

Running time is significantly improved compared to traditional online classification methods and labeling requirements lowered to that of the proportion of normal samples within the data stream. The memory size of the AE will also be constant as the shape of the weight matrices will be unchanged throughout the data stream, which is an advantage over a decision tree method such as HAT, that can grow with the stream [3].

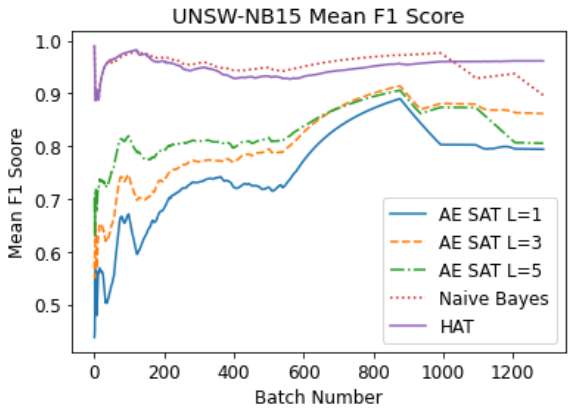
This paper has focused solely on the autoencoder, whereas other studies [9, 11, 14] have augmented the model with various pre and post processing stages, that could be unnecessary when considering the raw performance of the AE itself.

The following areas should be further considered when implementing the AE method:

- *Single vs Stacked AE*, represents a trade-off between detection performance and throughput speeds.
- *Multi-Label Classification*, combining the AE with a misuse classifier or unsupervised clustering in order to identify multiple classes.
- *Concept Drift and Active Learning*, labeling cost could be further reduced and performance improved by pro-actively monitoring the data stream for suspected concept drift [6] and enforcing a labeling budget to confirm drift [10].



(a) Mean Accuracy



(b) Mean F1 Score

Figure 3: UNSW-NB15 Evaluation

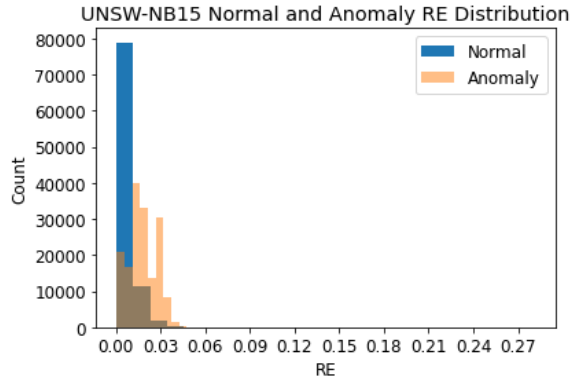


Figure 4: UNSW-NB15 RE Distribution

.

7 CONCLUSION

This paper has answered the question if autoencoders can provide a low cost anomaly detection method for computer data streams, and demonstrated that the total running time and labeling cost is much lower than traditional online learning classification approaches. Data streams with similarities between anomaly and normal samples present a challenge, which can be answered in part with a trade-off of AE network complexity, but further enhancements should be considered such as hybrid detection, ensemble methods, concept drift detection and active learning.

The following contributions have been made by this paper: the use of dropout with an AE network for computer network data streams; Naïve Anomaly Threshold with Decay approach to improve performance for data streams; improvement to the efficiency of the Stochastic Anomaly Threshold method introduced by Aygun and Yavuz [2]; and a demonstration of the raw performance of the AE for computer network data streams, allowing for further areas of enhancement to be better appreciated.

REFERENCES

[1] Md Shahanur Alam, B. Rasitha Fernando, Yassine Jaoudi, Chris Yakopcic, Raqibul Hasan, Tarek M. Taha, and Guru Subramanyam. 2019. Memristor Based Autoencoder for Unsupervised Real-Time Network Intrusion and Anomaly Detection. In *Proceedings of the International Conference on Neuromorphic Systems*. 1–8.

[2] R. Can Aygun and A. Gokhan Yavuz. 2017. Network anomaly detection with stochastically improved autoencoder based models. In *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, 193–198.

[3] Albert Bifet, Ricard Gavaldà, Geoff Holmes, and Bernhard Pfahringer. 2018. *Machine Learning for Data Streams with Practical Examples in MOA*. MIT Press. https://moa.cms.waikato.ac.nz/book/.

[4] Anna L. Buczak and Erhan Guven. 2016. A survey of data mining and machine learning methods for cyber security intrusion detection. *IEEE Communications Surveys & Tutorials* 18, 2 (2016), 1153–1176.

[5] Jinghui Chen, Saket Sathe, Charu Aggarwal, and Deepak Turaga. 2017. Outlier detection with autoencoder ensembles. In *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 90–98.

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

[7] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

[8] Tung Kieu, Bin Yang, Chenjuan Guo, and Christian S. Jensen. 2019. Outlier detection for time series with recurrent autoencoder ensembles. In *28th international joint conference on artificial intelligence*.

[9] XuKui Li, Wei Chen, Qianru Zhang, and Lifa Wu. 2020. Building Auto-Encoder Intrusion Detection System Based on Random Forest Feature Selection. *Computers & Security* (2020), 101851.

[10] Indrė Žliobaitė, Albert Bifet, Bernhard Pfahringer, and Geoffrey Holmes. 2013. Active learning with drifting streaming data. *IEEE transactions on neural networks and learning systems* 25, 1 (2013), 27–39.

[11] Yisroel Mirsky, Tomer Doitshman, Yuval Elovici, and Asaf Shabtai. 2018. Kitsune: an ensemble of autoencoders for online network intrusion detection. *arXiv preprint arXiv:1802.09089* (2018).

[12] Ali H. Mirza and Selin Cosan. 2018. Computer network intrusion detection using sequential LSTM Neural Networks autoencoders. In *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 1–4.

[13] Jacob Montiel, Jesse Read, Albert Bifet, and Talel Abdessalem. 2018. Scikit-multiflow: a multi-output streaming framework. *The Journal of Machine Learning Research* 19, 1 (2018), 2915–2914.

[14] Miguel Nicolau and James McDermott. 2016. A hybrid autoencoder and density estimation model for anomaly detection. In *International Conference on Parallel Problem Solving from Nature*. Springer, 717–726.

[15] Christopher Nixon, Mohamed Sedky, and Mohamed Hassan. 2019. Practical Application of Machine Learning based Online Intrusion Detection to Internet of Things Networks. In *2019 IEEE Global Conference on Internet of Things (GCIoT)*. IEEE, 1–5.

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research* 15, 1 (2014), 1929–1958.

[17] Xiaoming Yuan, Ran Wang, Yi Zhuang, Kun Zhu, and Jie Hao. 2018. A Concept Drift Based Ensemble Incremental Learning Approach for Intrusion Detection. In *2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData)*. IEEE, 350–357.