SALAD: An Exploration of Split Active Learning based Unsupervised Network Data Stream Anomaly Detection using Autoencoders

Christopher Nixon, Mohamed Sedky, and Mohamed Hassan

Abstract—Machine learning based intrusion detection systems monitor network data streams for cyber attacks. Challenges in this space include detection of unknown attacks, adaptation to changes in the data stream such as changes in underlying behaviour, the human cost of labeling data to retrain the machine learning model and the processing and memory constraints of a real-time data stream. Failure to manage the aforementioned factors could result in missed attacks, degraded detection performance, unnecessary expense or delayed detection times. This research evaluated autoencoders, a type of feed-forward neural network, as online anomaly detectors for network data streams. The autoencoder method was combined with an active learning strategy to further reduce labeling cost and speed up training and adaptation times, resulting in a proposed Split Active Learning Anomaly Detector (SALAD) method. The proposed method was evaluated with the NSL-KDD, KDD Cup 1999, and UNSW-NB15 data sets, using the scikit-multiflow framework. Results demonstrated that a novel Adaptive Anomaly Threshold method, combined with a split active learning strategy offered superior anomaly detection performance with a labeling budget of just 20%, significantly reducing the required human expertise to annotate the network data. Processing times of the autoencoder anomaly detector method were demonstrated to be significantly lower than traditional online learning methods, allowing for greatly improved responsiveness to attacks occurring in real time. Future research areas are applying unsupervised threshold methods, multi-label classification, sample annotation, and hybrid intrusion detection.

Index Terms—Active Learning, Online Learning, Autoencoders, Anomaly Detection, Intrusion Detection System.

1 Introduction

Intrusion Detection Systems (IDS) monitor a computer network for cyber attacks. Traditional intrusion detection techniques rely on human subject matter experts to carefully produce signatures that can accurately detect a cyber attack at the network layer. For over a decade research has focused on improving IDS with machine learning (ML) methods in order to reduce the overall demand for human effort [1]. The majority of this research has centred around misuse detection whereby the ML based IDS is trained using a data set in which all cyber attacks are labeled, the drawback of this being that only the labeled attacks will be known to the model, missing unknown or new attacks, and that labeling of the initial data set is a time consuming and complex task prone to human error. An alternative to misuse detection is to use an anomaly detector whereby only the 'normal' network data is learned and any significant deviations treated as an anomaly meaning that new attacks will be detected, a challenge with this approach is the potential for false positives.

IDS capture network packet data directly from the network, requiring efficient real-time processing of each new packet as part of a continuous data stream. This network data stream is non-stationary and can change over time, a characteristic known as concept drift, which requires the ML model to adapt in order that detection performance is not degraded [2]. Adaptation requires detecting a change in the posterior probability of a class label, necessitating the ground truth to be known. Active learning (AL) is an attempt to lower the labeling cost, and speed up the adaption times, of change detection by employing uncertainty or random strategies according to a labeling budget [3]

An hypothesis that this research aims to test is that anomaly detectors monitoring non-stationary network data streams will experience increased false positives over time, which can be corrected by applying adaptation techniques to update the anomaly detector. This will be expanded by a further hypothesis that active learning strategies can provide good adaptation with minimal labeling cost, and reduced learning times, for anomaly detection.

Unsupervised learning allows for a model to be trained without all the class labels being known, typically achieved by learning a representation of the underlying data structure. Common unsupervised techniques, such as clustering, are impeded by high degrees of time complexity and memory usage [4]. Models based on neural networking are gaining increased attention in the IDS field and a type of feed-forward neural network, the autoencoder, is able to learn the representation of data without class labels by encoding a latent representation of the data, which can be utilised for anomaly detection by calculating the error of the decoded output from the original, and comparing to a predetermined anomaly threshold [5]. This research aims to test the hypothesis that autoencoders provide an effective online anomaly detector for network data streams when combined with active learning methods.

The remainder of this paper is organised as follows: Section 2, introduces related work; Section 3, describes the proposed Split Active Learning Anomaly Detector (SALAD)

method; Section 4, presents the evaluation results; Section 5, discusses how SALAD provides a low cost anomaly detector for network data streams; and Section 6, presents conclusions.

2 Related Work

2.1 Neural Networking Anomaly Detection

Intrusion detection systems can be either *anomaly* based or *misuse* based, where the former learns the normal behaviour and detects deviations, allowing for detection of previously unseen, unknown attacks, and the latter learns known attack signatures resulting in high levels of detection accuracy [6]. A challenge with network data streams is that they generate large volumes of data that become increasingly expensive for a human expert to analyse and correctly label. Anomaly detectors are beneficial because they only need to learn the representation of a single 'normal' class from which anomalies can be distinguished meaning that new, previously unseen, attacks can be detected without requiring new data labels and re-training of the model [6]. Unsupervised machine learning methods are well suited to the anomaly detection task as they can learn the representation of the underlying data to determine normal and anomaly classes [6], as well as learning useful features that better separate the classes. Buczak and Guven [1] have provided a comprehensive survey of IDS machine learning techniques, including anomaly detection, in most cases misuse and anomaly detection are combined into a *hybrid* system. This review briefly introduces recent studies within the unsupervised anomaly detection space, adopting neural networking methods familiar to the visual processing area, for comparison to the proposed approach.

Alrawashdeh and Purdy [7] evaluated Restricted Boltzmann Machines (RBM) arranged into a deep belief network combined with a logistic regression classifier trained using back propagation. Although the study claims to be 'anomaly' based the model is actually trained to identify known classes so would be more 'misuse' based in its approach. The accuracy of their model, with the 10% KDD Cup 1999 data set, is 97.91% [7]. The authors further build on there work by replacing the RBM activation function with a novel 'Adaptive Linear Function' (ALF) for intrusion detection with the aim of improving accuracy and convergence time [8]. Evaluated with KDD Cup 1999 and NSL-KDD data sets, the accuracy was 98.59% and 96.2% respectively [8].

Roshan et al. [9] proposed a novel intrusion detection approach using a Clustering Extreme Learning Machine (CLUS-ELM) method. This method allows for both unsupervised and supervised updates to the model, using a decision maker element to perform informed change detection based on the cluster output, in this design unsupervised refers to guessing the correct cluster for a given data sample as opposed to being told the label by a 'human expert'. The mean square error calculation used by the decision maker will still require the ground truth to be known. Results were evaluated using the NSL-KDD data set, with a detection rate for known attacks of 84% and 81% for unsupervised and supervised modes, 77% and 84% for unknown attacks, where the false positive rate was less than 3% [9]. The author remarks that the better unsupervised detection rates for known attacks compared to the supervised ones are unexpected and could be due to inaccuracies in the NSL-KDD data set [9].

Chen, Cao and Mai [10] proposed an offline anomaly detection method whereby Convolutional Neural Networks (CNN) are used to extract features which are then condensed into a spherical hyperplane by a deep Support Vector Data Description (deep-SVDD) technique. The method is trained on normal samples only so that such normal samples concentrate around the center of the sphere and attack samples concentrate on the outside as outliers allowing them to be detected as a one-class anomaly detector. Their method was evaluated with the KDD Cup 1999 data set, achieving an accuracy of 96% when all attack types are present.

Hassan et al. [11] proposed a combined CNN for feature reduction and Weight Dropped, Long Short Term Memory (WDLSTM) network for representation of dependencies among features, using the connection drop out regularisation method. The proposed supervised learning network was evaluated with the UNSW-NB15 data set, returning an F1-Score of 0.88 for abnormal samples and overall accuracy of 97.17% via offline holdout training.

The reviewed studies all demonstrate different network topologies for cyber intrusion detection, all of which have elements of supervised learning and traditional offline batch training. They do not address the problem of a truly unsupervised anomaly detector for online data streams as will be explored in this paper.

2.2 Autoencoder Anomaly Detection

An autoencoder is a type of feed-forward neural network that uses an encoding function to produce a latent code representation of the input data, and a decoding function to reconstruct the input from the code representation [12]. The mean square error between the reconstructed output and original input can be calculated using equation 1, where f is the encoding function and g is the decoding function [12], which can then be compared to an anomaly threshold to label a sample as either normal or anomalous.

$$\begin{aligned} \hat{X} &= g(f(X)) \\ \text{RE} &= \frac{1}{n}\sum_{j=1}^{n}(X_j - \hat{X}_j)^2 \end{aligned} \tag{1}$$

In our previous work [12], we reviewed autoencoder based anomaly intrusion detection methods, whereby single layer denoising models [13], Long Short Term Memory (LSTM), Recurrent Neural Network [14], [15], ensembled stacked autoencoders [16], [17], and sparsely connected networks [18], [15] were demonstrated across a range of IDS data sets. Vaiyapuri and Binbusayyis [19] evaluated a number of autoencoder network architectures for anomaly detection, finding the use of a *contractive penalty* to regulate the network provided the best performance when evaluated offline using the NSL-KDD and UNSW-NB15 data sets.

A number of methods were proposed in the literature to determine the anomaly threshold, an important parameter in deciding whether to label a sample as a positive

detection. The threshold can be set to the average RE value observed during training [19]. Naïve Anomaly Threshold (NAT) sets the threshold at the maximum observed RE during training [16]. Stochastic Anomaly Threshold (SAT) [13] sets the threshold based on the best observed accuracy when stepping through threshold values between the mean and 3 * standard deviation of the normal sample distribution. Nicolau and McDermott [13] proposed an anomaly threshold method using Kernel Density Estimation.

Aiming to find an optimal network configuration, we evaluated in [12], an undercomplete autoencoder, regulated with connection dropout, with a prequential online test using the KDD Cup 1999 and UNSW-NB15 data sets. Applying a single layer autoencoder with dropout probability of 0.1, using the Stochastic Anomaly Threshold method, provided an accuracy of 98% and F1-score of 0.812, using the KDD Cup 1999 data set, with a significantly improved running time compared to traditional Naïve Bayes (NB) and Hoeffding Adaptive Tree (HAT) online methods. Evaluation on the UNSW-NB15 data set using a 3-layer network and dropout probability of 0.2 returned an accuracy of 79.1% and F1-score of 0.703. The results showed that the SAT threshold performed better than the NAT, and that more complex data sets benefit from experimenting with the number of layers and regularisation of the network.

2.3 Concept Drift Detection with Active Learning

Non-stationary network data streams may experience real concept drift [2], whereby the posterior probability of classes will change over time due to changes in network behaviors, the cause of which could be either benign or adversarial in nature. The posterior probability is defined as $p(y|X)$ which represents the probability of class y given an observation X [2]. Autoencoders determine outliers using the RE-score, based on the hypothesis that adversarial behaviour deviates from the learned 'normal' representation resulting in scores above the anomaly threshold. Real concept drift presents a challenge that the aforementioned hypothesis will weaken overtime, with changing benign data also scoring above threshold, raising the false positive rate. Increasing the anomaly threshold does not present an optimal solution as although the false positive rate may lower, the false negative rate could increase and so is not recommended. The hypothesis of this research was that a change in underlying 'benign' network behaviour will result in a raised false positive rate and that learning the representation of the new behaviour will remedy this effect. Note that the change in benign activity could be from an unplanned change such as a network fault, in which case the usefulness of the anomaly detector is extended to a fault detector, however for the purposes of this research this will not be considered further.

Change detection is a set of methods that proactively monitor the data stream for concept drift [2]. Traditional methods such as adaptive windowing and statistical process control (SPC) [2], rely on fully supervised labels and are therefore not well suited to applications where data labeling is expensive, such as network data streams. Moreover unsupervised techniques that rely solely on monitoring a change compared to a reference distribution will not always detect real concept drift [20]. Sethi and Kantardzic [21] proposed a semi-supervised Margin Density Drift Detector (MD3) to reduce labeling costs through an *active learning* approach. First, using an unsupervised method, samples that fall below an uncertainty threshold are added to the margin. Density of the margin is compared to a training reference distribution to detect drift before confirming by testing accuracy with data labels, sensitivity can be adjusted through a varying factor of the reference distribution's standard deviation. A fading factor is utilised to give greater importance to more recent samples within a moving average of margin density [21]. MD3 can work with ensembles, calculating if a sample should be included within the margin by comparing the distance between the mean predicted class probabilities to the margin threshold (θ), given by equation 2. A possible benefit of this approach would be that the change in density of uncertain samples that are borderline outliers could indicate a concept drift that requires further analysis, prompting further action such as re-training. As the anomaly detector only requires labeled normal data to re-train, this would be a cheaper approach to other methods that require fully labeled data. A possible drawback is that the frequency of drifts could demand increased human expertise. Evaluation with the NSL-KDD data set reported an accuracy of 89.4 and 89.9 % using the SVM and random subspace ensemble methods, respectively where the first 15% of the data stream is used as a training set. The total labeling cost was 7.9%.

$$(p(\hat{y}_{c1}|X) - p(\hat{y}_{c2}|X)) \leq \theta \tag{2}$$

Shan et al. [22] also proposed an AL change detection strategy based on margin uncertainty, 'OALEnsemble', however in this approach the ensemble members are trained on different windows of the data set, with a stable classifier and a series of short window 'dynamic' classifiers that are continually replaced as new blocks of the data stream are processed, to balance the detection of both sudden and gradual concept drifts. Similar to [21], labeling is restricted to samples within the uncertainty margin, with the addition of a random labeling algorithm to randomly include samples outside of the margin where drift may also be occurring [22]. The stable classifier is incrementally trained with all new data, whilst dynamic classifiers are only trained on the most recent block and given a weight, providing importance to more recent data [22]. The incremental update of the stable classifier is restricted to models that feature *local replacement* such as very fast decision trees (VFDT) [2], and so would not be appropriate for autoencoder methods. The labeling rate is constrained by pro-actively adjusting the sensitivtiy threshold in order to manage the cost of the algorithm during periods of high uncertainty. Random sampling is desirable as it enables the classifier to be trained from the whole distribution, reducing bias [3]. The idea of gradually retraining the autoencoders with new 'normal' data in response to concept drift, whilst retaining the previous models for a period of time, moderating their importance with a weight scheme, could allow for the detection of both gradual and sudden changes in benign behaviour, however the problem of *global replacement* must be carefully considered as training on small data sets could degrade the autoencoders ability to represent normal data.

Dang [23] evaluated AL for IDS, using a novel strategy with the Naïve Bayes classifier, selecting instances with the greatest distance from the population distribution of probabilities under the hypothesis that a bigger change of $P(A|B)$ reflects a rare event that should be learned. The method was evaluated with the CICIDS 2012 data set, achieving an AUC-score of 90% compared to 85% with the uncertainty strategy with 10% of labeled data, and performance decreasing beyond this. The author argues that this indicates that good quality data is more important over larger volumes of data [23]. It may also be true that the method reduces class imbalance by proactively sampling examples with weaker performance that could reflect minority classes.

Zhang et al. [24] evaluted an Open-CNN method trained by AL labeling the 'unknown' detected attacks. Accuracy with the CTU data set was near equivalent to 100% label cost at just 1% of labeled attacks using an uncertainty strategy, demonstrating that only a low label cost is necessary to train the ML model.

Žliobaitė et al. [3] discussed three requirements for AL strategies: 1) balancing the labeling budget over time, 2) detect changes anywhere within the problem space and 3) preserve the distribution for unbiased change detection. A number of strategies were evaluated against these requirements, including *fixed uncertainty* as demonstrated by [21], and *uncertainty with randomisation*, whereby the sensitivity threshold is randomly selected from a standard distribution to occasionally include samples outside of the uncertainty margin. Fixed uncertainty is only able to satisfy requirement one, and randomised uncertainty satisfies requirement one and two, but neither can preserve the probability density of labeled data compared to the original distribution, which can bias the model [3]. A further *split* strategy is introduced which satisfies all three requirements by splitting the the data stream into two, using uncertainty and random strategy exclusively on either stream. Both streams are used for training, but only the randomised stream is used for change detection [3]. Shan et al. [22] presents a split strategy, although in this approach adaptation is *blind*, based on incrementally updating the ensemble members with both uncertainty and random labels, offering no proactive change detection, this could reduce overall adaptation speeds [2].

An objective of this research was to satisfy all three AL requirements outlined by Žliobaitė et al. [3]. MD3 [21] will be biased towards uncertain samples and will miss change occurring outside of the margin which will affect overall detection performance. The work of Shan et al. [22] could be further improved by introducing pro-active change detection method to the randomly labeled data as suggested by Žliobaitė et al. [3] in order to increase adaptation time. In this research random, uncertainty, variable uncertainty, split and blind strategies are compared. The proposed hypothesis is that only the split strategy with informed change detection approach will be able to satisfy all three requirements and that the change detection approach will offer faster adaptation times to a blind approach. The informed approach can use a well known change detector such as Drift Detection Method (DDM) [25] to monitor the classification error of the anomaly detector.

3 Methods

The aim of this research was to explore that autoenoders can provide a low cost online anomaly detection solution when combined with AL methods. In our previous work [12] we evaluated dropout probability, NAT with decay and SAT anomaly thresholds, and single vs stacked network structure, to find optimal autoencoder parameters. Building on this work, in this paper, we further introduced a novel Adaptive Anomaly Threshold (AAT) method and also evaluated an AL based Active Stream Framework (ASF) [3] with which we compared blind, random, uncertainty, variable uncertainty and split AL strategies. The uncertainty strategy was adapted for use with autoencoders using a novel distance from RE method. All methods were evaluated using a prequential, interleaved test-then-train method [2], whereby the model is first tested on a previously unseen sample before training in a chunk wise fashion [12], after an initial period of pre-training. Results were compared against traditional Naïve Bayes (NB) and Hoeffding Adaptive Tree (HAT) online learning methods using the KDD Cup 1999[1] 10% [26] and UNSW-NB15[2] [27] data sets.

The Keras[3] neural networking [28], version 2.3.1, and Scikit-Multiflow[4] stream learning [29], 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, kappa and total running time. For prequential evaluation the scikit-multiflow default of updating evaluation metrics every 200 samples was used.

3.1 Adaptive Anomaly Threshold

From evaluating the make up of the data stream and performance achieved with both the NAT and SAT threshold methods [12] a proposed hypothesis was that chunks of the data stream that contained only normal samples benefit from a naïve approach whereby the maximum RE is used, therefore all samples will fall below this value, giving an accuracy of 100%. For anomaly samples the second hypothesis was that between the maximum value and the mean observed RE a threshold can be found that best splits normal and anomaly samples, similar to the stochastic approach. A third hypothesis was that the mean RE will change overtime due to concept drift, and so will become less sensitive to more recent samples when taken over a long stream.

To address the above three hypothesis an *'Adaptive Anomaly Threshold'* (AAT) method was proposed that combines the NAT, SAT and Fading Factor [30] methods. The proposed method is given in algorithm 1. Normal samples were used to update the fading average RE-score over the stream, using a fading factor α [30] in order to give more importance to more recent sample values, satisfying hypothesis 3 above. The maximum RE of normal samples over the data stream is also recorded and used to find the first value of the anomaly threshold ϕ. If the initial

1. http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
2. https://www.unsw.adfa.edu.au/unsw-canberra-cyber/cybersecurity/ADFA-NB15-data sets/
3. https://keras.io/
4. https://scikit-multiflow.github.io/

maximum value of ϕ achieves an accuracy of 1.0 or 100%, then this fulfilled the first hypothesis that all samples are normal and no further action was required. Otherwise hypothesis 2 is assumed and a stochastic approach was then used to step through potential threshold values until the highest accuracy is found.

Algorithm 1: Adaptive Anomaly Threshold

Input : autoencoder m, X, y, threshold ϕ, step size $v \leftarrow [> 0]$, fading factor α
Output: ϕ

```
/* Initialise fading sum, fading
   increment, and max RE variables   */
```
$S_0 \leftarrow 0; N_0 \leftarrow 0; RE_{\text{max}} \leftarrow 0;$
```
/* Find the fading mean RE of normal
   samples                           */
```
$X_{y \leftarrow 0} \subseteq X;$
$\text{RE}_i \leftarrow$ `predictRE(`m, $X_{y \leftarrow 0}$`);`
$S_i \leftarrow \text{RE}_i + \alpha * S_{i-1};$
$N_i \leftarrow 1 + \alpha * N_{i-1};$
$\text{RE}_{\mu\alpha} \leftarrow \frac{S_i}{N_i};$
```
/* Find the maximum normal sample RE of
   the data stream                   */
```
if $RE_i > RE_{MAX}$ **then**
  $\text{RE}_{\text{MAX}} \leftarrow \text{RE}_i;$
end
```
/* Set threshold to the maximum
   observed RE                       */
```
$\phi \leftarrow \text{RE}_{\text{MAX}};$
```
/* Calculate accuracy, only attempt to
   find a lower threshold if accuracy
   is not 100%                       */
```
$\hat{y} \leftarrow$ `predict(`m,ϕ,X`);`
$\text{acc}_w \leftarrow$ `calcAccuracy(`$\hat{y}$,y`);`
if $acc_w < 1.0$ **then**
  ```
  /* Step through to the fading mean
     of the stream RE to find
     threshold that yeilds the highest
     accuracy                        */
  ```
  $\phi_w \leftarrow \phi;$
  while $\phi > RE_{\mu\alpha}$ **do**
    $\phi \leftarrow \phi - v;$
    $\hat{y} \leftarrow$ `predict(`m,ϕ,X`);`
    $\text{acc} \leftarrow$ `calcAccuracy(`$\hat{y}$,y`);`
    if $acc > acc_w$ **then**
      $\phi_w \leftarrow \phi;$
      $\text{acc}_w \leftarrow \text{acc};$
    end
  end
  $\phi \leftarrow \phi_w;$
end

The proposed autoencoder anomaly detector is depicted in figure 1. The sample X is inputted to the autoencoder network from which a Reconstruction Error (RE) is produced based on the loss value between the approximate output and the original input. The RE is compared to an anomaly threshold value with samples scoring above threshold being labeled as an 'anomaly' and those below being 'normal' or benign. If a label Y is provided then the anomaly threshold is updated using a novel adaptive anomaly threshold method, which also maintains a memory of the population mean RE throughout the data stream by using a fading factor [2] memory mechanism to prioritise more recent samples for faster adaptation. The adaptive anomaly threshold is demonstrated to be superior to fixed and other threshold determination methods from the literature. Note that the use of labels to find the anomaly threshold results in a semi-supervised method.

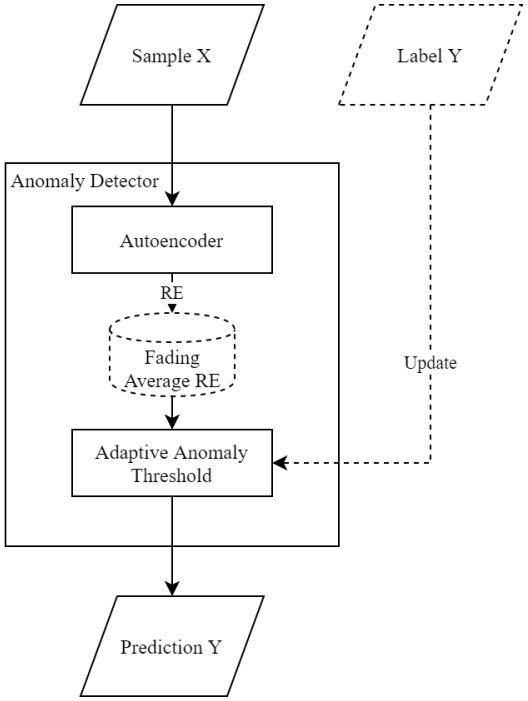


Fig. 1: Autoencoder Anomaly Detector

3.2 Active Stream Framework

The proposed autoencoder anomaly detector is a semi-supervised method requiring class labels to be known. Class annotation is also important to detect changes in the data stream that require learning to occur in order for the model to adapt. Given the infinite nature of a data stream, labeling all samples is infeasibly expensive, therefore AL methods were explored to minimise the labeling cost for both updating the model and threshold, whilst identifying and adapting to changes in the data stream.

Žliobaitė et al. [3], proposed an *active stream framework*, which combines change detection with a labeling strategy and a fixed budget B. Algorithm 2 gives the active stream framework evaluated in this research. The active learning *strategy* is an important part of the framework as it determines whether or not the current data sample X_i, y_i should be labeled. Blind, random, uncertainty, variable uncertainty and split strategies were evaluated in this research [3], [21], [22]. The framework maintains a running estimate of label usage $\hat{u}_i$ over a fading window, calculated by equation 3, where w is the size of the fading window and label_i is the labeling decision either 0 or 1 at time i. The spending

estimate $\hat{b}$ is then calculated from $\hat{u}_i$ over w, given in equation 4 [3]. During this evaluation, w was set to 1000.

The labeled samples are then used to train the model and perform change detection. If a warning signal is received then a new autoencoder (AE_L) is trained with the most recent examples, and when a change is signaled, the current model is replaced with AE_L, completing adaptation to the new concept. For this evaluation the Drift Detection Method (DDM) [25] change detector was used.

$$\hat{u}_i = \hat{u}_{i-1} * \frac{(w-1)}{w} + \text{label}_i \tag{3}$$

$$\hat{b} = \frac{\hat{u}_i}{w} \tag{4}$$

Algorithm 2: Active Stream Framework

Input : Autoencoder AE, Labeling budget B, budget window w, strategy(parameters), Change Detector D
Output: AE

```
/* Initialise active stream framework */
```
$\hat{b} \leftarrow 0; \hat{u}_0 \leftarrow 0;$
```
/* Check if current spending estimate is below budget and strategy decides to label */
```
if $\hat{b} < B$ *AND* `strategy(`*parameters*`) = 1` **then**
  Update label estimate $\hat{u}_i$ (equation 3) where $label_i = 1$;
  ```
  /* Incrementally fit the autoencoder and predict current label */
  ```
  AE $\leftarrow$ `partialFit(`AE, X_i, y_i`);`
  $\hat{y}_i \leftarrow$ `predict(`AE, X_i`);`
  ```
  /* Update the change detector statistics */
  ```
  `updateChangeDetector(`$D, y_i \neq \hat{y}_i$`);`
  if AE_L **then**
    ```
    /* If an alternative AE exists then update this with the labeled samples */
    ```
    AE$_L \leftarrow$`partialFit(`$AE_L X_i, y_i$`);`
    ```
    /* If change is detected then replace the current AE with the alternate */
    ```
    if `changeSignalled(`D`)` **then**
      Replace AE with AE$_L$;
    end
  else if `warningSignalled(`D`)` **then**
    ```
    /* If warning is signalled then create new alternate AE and train */
    ```
    Create new AE$_L$;
    AE$_L \leftarrow$`partialFit(`AE_L, X_i, y_i`);`
else
  Update label estimate $\hat{u}_i$ (equation 3) where $label_i = 0$;
end
Update spending estimate $\hat{b}$ (equation 4);

3.3 Active Learning Strategies

The following section outlines the active learning strategies evaluated in this research. Žliobaitė et al. [3] outlined three objectives of active learning strategies, which will need to be met by any proposed strategies:

1) balance the labeling budget B over infinite time;
2) detect changes anywhere in the instance space;
3) preserve the distribution of incoming data for detecting changes.

A *random* active learning strategy randomly selects a sample to label based on Bernoulli probability with a given budget B. The random strategy satisfies all three objectives of [3].

The *uncertainty strategy* labels a sample based on the level of uncertainty from the classifier compared to a threshold, and attempts to label the samples where there is the least confidence [3]. A common approach is to use the classifier's predicted probability for class c compared to the threshold θ: $P(y_c|X) \leq \theta$ [3], [21], [22].

Autoencoders do not provide a direct class probability, instead they provide a reconstruction error from which a normal or anomaly classification decision can be made. This research proposed a novel method whereby the RE squared difference from the anomaly threshold ϕ is used as a measure of uncertainty, equation 5, assuming the hypothesis that the lower the difference compared to the average of the population, then the greater the uncertainty for the sample. The difference is squared to make all values positive.

$$d_i = (\phi - RE_{Xi})^2 \tag{5}$$

In order to accommodate changes in the data stream and avoid a scenario where the strategy stops learning due to high variance, a fading factor α was used to produce a fading average of differences d_{avg}, calculated using equation 6. This allowed for the more recent samples to have a greater bearing on the strategy outcome.

$$\begin{aligned} S_i &= d_i + \alpha * S_{i-1} \\ N_i &= 1 + \alpha * N_{i-1} \\ d_{avg} &= \frac{S_i}{N_i} \end{aligned} \tag{6}$$

Using d_{avg} the fading standard deviation d_{std} of the stream was calculated using equation 7.

$$\begin{aligned} V_i &= (d_i - d_{avg})^2 + \alpha * V_{i-1} \\ d_{std} &= \sqrt{\frac{V_i}{N_i}} \end{aligned} \tag{7}$$

Finally, the strategy returned a labeling decision of 1 where $d_i < d_{avg} - d_{std}\theta$, equation 8, requiring a sample to be below the average by so many θ standard deviations, where θ was the confidence threshold. $\theta = 2$ should capture samples where the difference is the lowest 5% of all samples.

$$\text{labeling} = \begin{cases} 1, & d_i < d_{avg} - d_{std}\theta \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

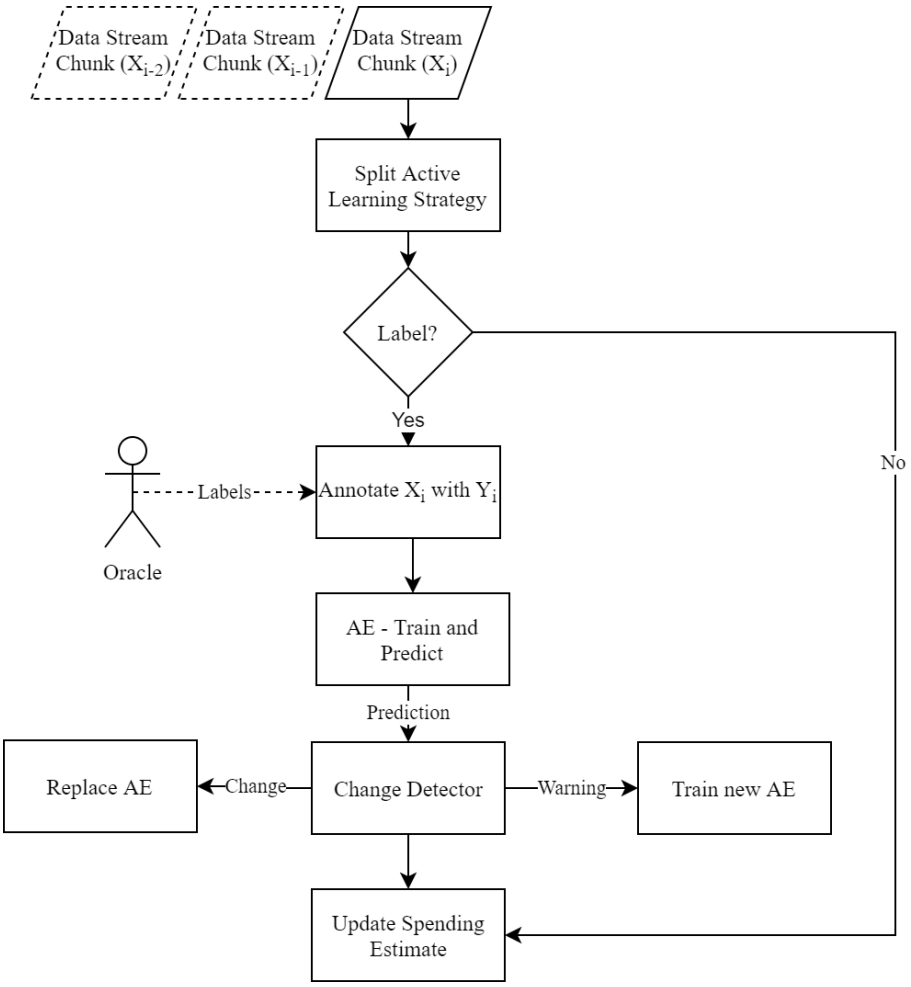


Fig. 2: Split Active Learning Anomaly Detector

The uncertainty strategy algorithm is given in 3, whereby the autoencoder AE model is used to predict the RE for sample X_i, and the fading average and standard deviation of the difference from the anomaly threshold ϕ over the stream used to provide a label output of 0 or 1 based on equation 8. On its own, an uncertainty strategy cannot satisfy all three active learning objectives as: the number of labeled samples will depend on the amount of uncertainty within the data stream and could vary above the intended budget, this is instead limited by line 2 of algorithm 2; only samples within the uncertainty margin are labeled, changes occurring outside of the margin will be missed; and change detection will be based on the distribution of uncertain samples that are trained on [3]. The strategy should reflect regions where real concept drift is occurring as higher uncertainty could reflect a change, resulting in faster adaptation times [21], [22].

Algorithm 3: Uncertainty Strategy

Input : Confidence θ, Fading Factor α, X, autoencoder AE, Threshold ϕ
Output: label

1 $S_0 \leftarrow 0; N_0 \leftarrow 0; V_0 \leftarrow 0;$ label $\leftarrow 0;$
2 $\text{RE}_i \leftarrow$ `predictRE(`AE, X_i`)`;
3 Calculate difference d_i of RE_i from ϕ, using equation 5;
4 Calculate the fading average difference d_{avg}, using equation 6;
5 Calculate the fading standard deviation of differences d_{std} using equation 7;
6 **if** $d_i < d_{avg} - d_{std}\theta$ **then**
7 | label $\leftarrow 1$;
8 **end**

Variable uncertainty is based on the uncertainty strategy, but instead of using a fixed confidence θ, this is instead varied depending on the amount of labeling that is being requested from the strategy, so that more labels will increase the confidence and fewer will decrease to attenuate the labeling and better manage budget [3]. This approach also has the benefit that it is not limited to a fixed labeling ceiling and can better utilise higher budgets to accurately identify concept drift [22]. Similar to the uncertainty strategy this also does not satisfy all three requirements [3].

The *split strategy*, given in algorithm 4, combines the random and variable uncertainty strategies to benefit from their respective strengths of accessing the entire stream distribution for change detection, and adapting to potential

change in higher regions of uncertainty. Due to the incorporation of the random strategy, this also meets all three requirements of [3].

Algorithm 4: Split Strategy

Input : Label Budget B, Confidence θ, Fading Factor α, X, autoencoder AE, Threshold ϕ, Step s
Output: label

label $\leftarrow 0$;
if `randomStrategy(`B`)` = *True* **then**
  label $\leftarrow 1$;
else if `varUncertaintyStrategy(`θ,α,X_i, *AE*, ϕ,s`)` = *True* **then**
  label $\leftarrow 1$;

The proposed Split Active Learning Anomaly Detector (SALAD) method is depicted in figure 2. This method reduces the labeling cost of the data stream to a fixed budget by adopting an active learning strategy to determine which labels should be updated, satisfying the requirements of Žliobaitė et al. [3]. Labeled samples are used to train the anomaly detector and the predictions input to a change detector which monitors for real concept drift occurring in the data stream [2]. Where real concept drift occurs, the current anomaly detector is replaced with a new one that has been trained on samples since a warning signal was produced. The result of this method is faster training of the anomaly detector and the ability to quickly adapt to changes occurring in the data stream.

4 Results

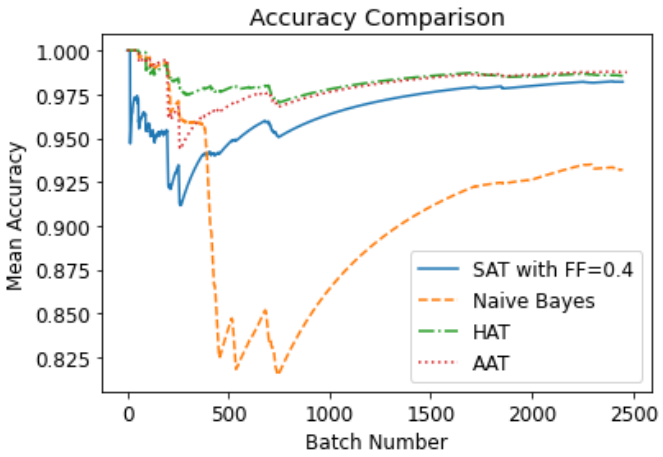
4.1 Adaptive Anomaly Threshold

The accuracy and F1-score of the Adaptive Anomaly Threshold method was compared to the Stochastic Anomaly Threshold with memory (SAT FF), HAT and NB algorithms. SAT FF is a novel modified version of the SAT algorithm to update the threshold based on a fading average [30] of previous thresholds to allow for memory when processing over a data stream. The parameter values for the autoencoder methods are given in table 1, where p represents the dropout probability; l is the number of hidden layers, h the ratio of hidden units to visible units; opt is the optimiser used to train the network with α learning rate; β is the threshold sensitivity; α is the fading factor; and v is the step size. NB and HAT algorithms used the scikit-multiflow default parameters [29].

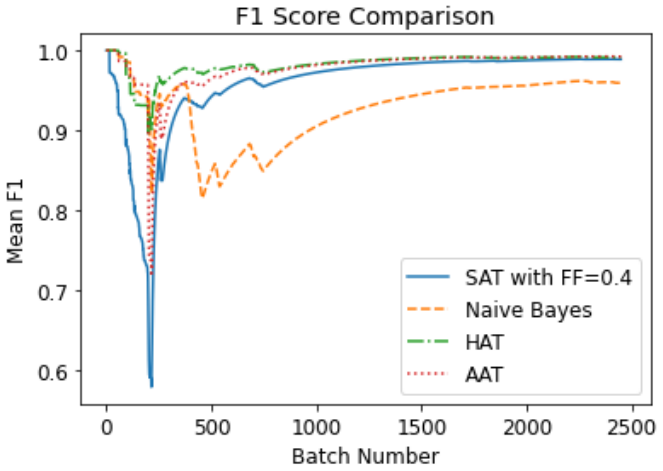
Method	Parameters
Prequential Evaluation	batch size = 100, pretrain size = 10000
Autoencoder	$l = 1, p = 0.1, h = 0.6, opt = adagrad$ ($\alpha = 0.01$)
SAT FF	$\beta = 1.1, v = 0.001, \alpha = 0.4$
Adaptive Anomaly Threshold	$\beta = 1.18, v = 0.001, \alpha = 0.4$

TABLE 1: Evaluation Parameters

The accuracy and F1 scores with the KDD Cup 1999 data set are plotted in figure 3. SAT FF and AAT are



(a) Accuracy



(b) F1-score

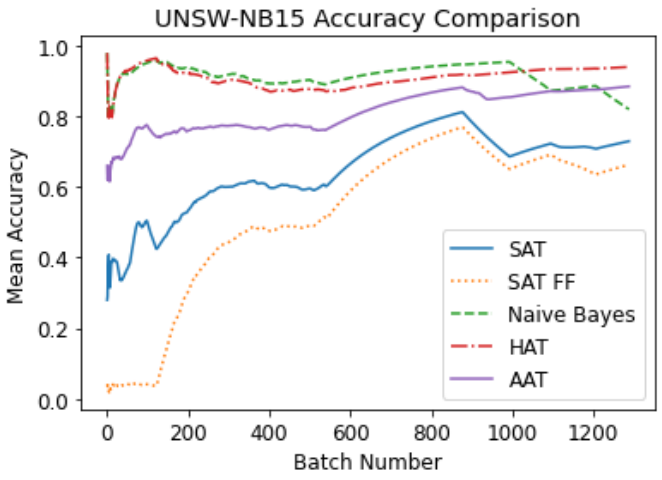
Fig. 3: KDD Cup 1999 AAT, SAT FF, NB and HAT accuracy and F1-score

close to HAT in terms of mean performance, with better kappa and F1 metrics when taken as an average across all batches, as shown in table 2. SAT FF and AAT were also significantly faster with a total running time (RT) of 14.04s and 19.18s, compared to 510.93s and 794.76s with NB and SAT, respectively. Note that running time will vary based on the underlying system performance and frameworks used, however the time of SAT FF is an order of magnitude better compared to both NB and HAT algorithms. Overall AAT returned the best mean accuracy and kappa results, an important metric for data stream learning.

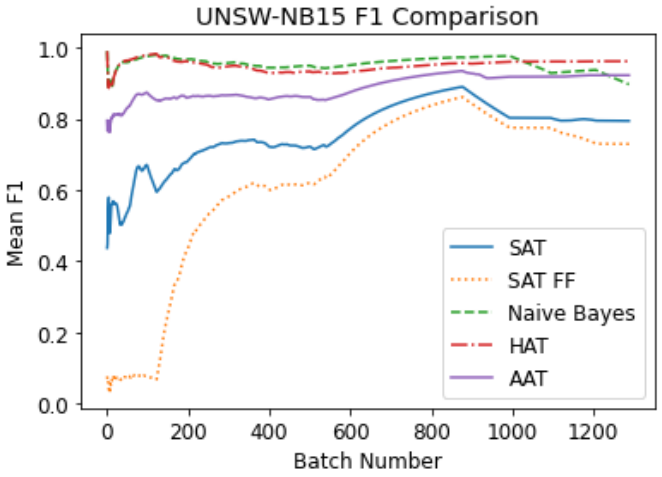
Algorithm	Accuracy % $\mu\pm$SD	Kappa $\mu\pm$SD	F1-score $\mu\pm$SD	RT (s)
AE AAT	**98.78±7.88**	**0.954±0.202**	0.802±0.395	19.18
AE SAT FF	98.16±8.65	**0.854±0.360**	0.812±0.387	**14.04**
NB	93.34±20.22	0.721±0.445	0.810±0.380	510.93
HAT	98.57±0.60	0.820±0.379	0.811±0.383	794.76

TABLE 2: KDD Cup 1999 AAT, SAT FF, NB and HAT Results

As demonstrated in our previous work [31], the UNSW-NB15 data set proved to be more challenging for on-



(a) Accuracy



(b) F1-score

Fig. 4: UNSW-NB15 AAT, SAT, SAT FF, NB and HAT accuracy and F1-score

line learning, requiring the number of network layers and dropout probability to be adjusted to better provide separation between normal and anomaly class distributions, with $l = 3$ and $p = 0.2$ being selected. The accuracy and F1-score results of the AAT method compared to SAT, SAT FF, NB and HAT are plotted in figure 4. Table 3 gives average accuracy of the SAT and SAT FF algorithms as 70.39% and 62.96%, respectively, which is considerably lower than that of NB and HAT. AAT returned the highest overall accuracy of the anomaly threshold methods, at 86.31% with 3 layers and dropout probability of 0.2, although kappa was lower, demonstrating reduced confidence in the anomaly decision for all methods. The results show that AAT is able to provide near equivalent performance to NB and HAT methods with a significantly lower running time.

4.2 Active Stream Framework

4.2.1 Labeling Budget

The effects of the labeling budget was evaluated with the random strategy as this is the only strategy to maintain the sample distribution of the stream so as to not add any bias to the results. Budget B was evaluated at values of 0.2 (20%), 0.5 (50%) and 1.0 (100%). The results are given in table 5 and mean accuracy plotted against the blind adaption AAT approach for comparison in figure 5. The greater the labeling budget, typically the higher the accuracy, kappa and F1 scores, the exception being UNSW-NB15 where $B = 0.5$ has a slightly higher accuracy and kappa. The difference in accuracy between 20% and 100% labels is 0.76% (KDD'99) and 2.69% (UNSW-NB15), demonstrating a small loss in performance for an 80% saving in labeling cost and approximate running time reduction of 54-62%; this reflects the results of Žliobaitė et al. [3], where a small loss of accuracy was observed between a B of 100% and 10% when tested with a number of non-cyber data sets.

Comparing to the blind adaptation of previous experiments, whereby no active learning is used, a labeling budget of 0.5 achieved a higher accuracy and F1 for half the labeling cost on both data sets. ASF RAND 1.0 is equivalent to the blind approach with full labels, but with the addition of change detection, with average accuracy and F1 improved across both data sets, although lower towards the end of the UNSW-NB15 stream as shown in figure 5b. Note the lower running time of the blind approach due to use of a chunk size of 100 vs 10 which influences the number of gradient updates and hence training time of the network.

Algorithm	Accuracy % $\mu\pm$SD	Kappa $\mu\pm$SD	F1-score $\mu\pm$SD	RT (s)
AE AAT	**86.31±16.32**	0.298±0.411	**0.767±0.335**	18.55
AE SAT	70.39±32.71	0.364±0.443	0.613±0.390	12.14
AE SAT FF	62.96±38.95	**0.420±0.458**	0.528±0.418	**11.01**
NB	83.69±28.99	0.399±0.480	0.832±0.343	350.39
HAT	92.85±11.19	0.436±0.479	0.813±0.340	610.94

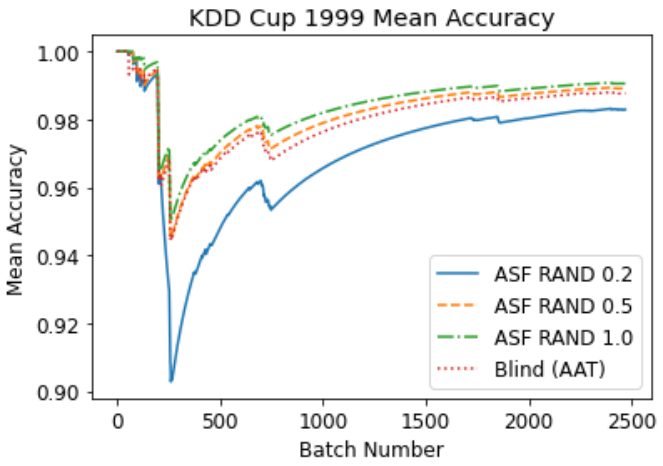
TABLE 3: UNSW-NB15 AAT, SAT, SAT FF, NB and HAT Results

Strategy	B	Accuracy % $\mu\pm$SD	Kappa $\mu\pm$SD	F1-score $\mu\pm$SD	RT (s)
KDD Cup 1999					
Random	0.2	98.32±8.50	0.932±0.217	0.811±0.381	55.9
Random	0.5	98.94±7.27	0.956±0.182	0.821±0.376	85.8
Random	1.0	99.08±7.02	0.962±0.176	0.825±0.374	145.4
Blind	1.0	98.78±7.88	0.954±0.202	0.802±0.395	19.18
UNSW-NB15					
Random	0.2	87.07±19.48	0.598±0.376	0.752±0.350	55.5
Random	0.5	90.85±12.16	0.619±0.265	0.791±0.338	84.0
Random	1.0	89.76±12.74	0.549±0.431	0.793±0.334	121.2
Blind	1.0	86.31±16.32	0.298±0.411	0.767±0.335	18.55

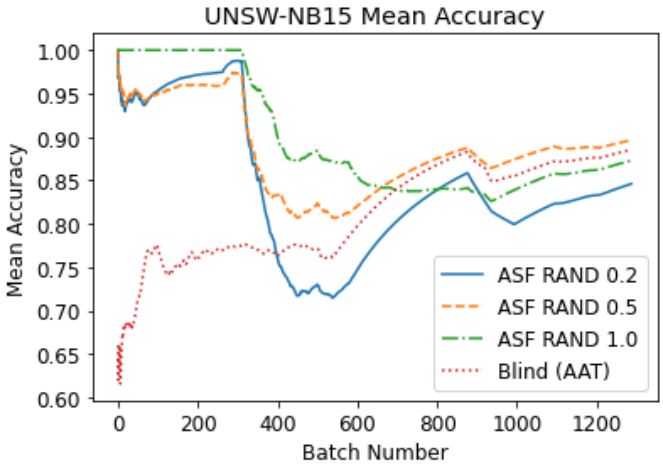
TABLE 4: Random Strategy Budget Size: KDD'99 and UNSW-NB15 Comparison

4.2.2 Active Learning Strategies

The results of each active learning strategy with a budget of 0.2 (20%) are given in Table 5, with accuracy and F1-score for both data sets plotted in figure 6. Each strategy was executed 5 times with the average and standard deviation presented. The worst performing strategy was the fixed uncertainty strategy, reflecting the results of Žliobaitė et al. [3], which was expected as the algorithm is biased only towards uncertain samples and cannot vary the amount of samples labeled, meaning that change occurring outside of



(a) KDD'99 Accuracy



(b) UNSW-NB15 Accuracy

Fig. 5: Labeling Budget Accuracy Comparison for Random Strategy

the fixed margin will be missed. It is also possible that the RE=value of normal samples outside of the margin may increase as the AE is trained more on uncertain samples, leading to higher false positives and lower F1-score.

The split strategy, returned the best results across both data sets, combining random and variable uncertainty strategies. Note that the total running time is between that of the random and variable uncertainty strategies, indicting time complexity savings where uncertain samples were first selected by the random strategy. The Kappa of the split strategy was observed as 0.717 (table 5) for the UNSW-NB15 data set, this is much higher than the performance of the blind AAT, NB, HAT and other AL strategies, indicating a higher level of confidence in the anomaly decisions.

5 Discussion

This research evaluated online anomaly detection in the form of a prequential evaluation method whereby the model is first tested on the next sample or chunk in the stream before training. The anomaly threshold is a key parameter for anomaly detection and finding an optimal threshold

Strategy	Accuracy % $\mu\pm$SD	Kappa $\mu\pm$SD	F1-score $\mu\pm$SD	RT (s)
KDD Cup 1999				
Random	98.32±8.50	0.932±0.217	0.811±0.381	**55.9**
Uncertainty	93.32±23.40	0.892±0.303	0.762±0.422	81.6
Var Uncert	98.61±8.65	**0.951±0.194**	0.817±0.379	74.1
Split	**98.85±7.55**	0.947±0.199	**0.819±0.378**	69.6
UNSW-NB15				
Random	87.07±19.48	0.598±0.376	0.752±0.350	55.5
Uncertainty	83.95±16.40	0.348±0.304	0.762±0.334	**53.8**
Var Uncert	87.51±16.26	0.452±0.368	0.768±0.339	64.6
Split	**90.88±14.96**	**0.717±0.363**	**0.791±0.343**	63.3

TABLE 5: Active Learning Strategy Comparison

for a data stream is non-trivial. A number of methods for finding the threshold were compared including fixed, naïve, stochastic and adaptive techniques. The adaptive anomaly threshold (AAT) was introduced as a novel hybrid of the naïve and stochastic methods in order to better adapt to chunks of normal or anomaly samples based on initial observed accuracy. Overall AAT outperformed other methods and is a recommended contribution of this research to be explored further.

The results observed with the KDD'99 data set and AAT threshold method provide strong evidence that the hypothesis of effective anomaly detection for network data streams can be supported by the autoencoder method with both strong detection and run time performance compared to traditional methods. UNSW-NB15 results could be strengthened by further design choices.

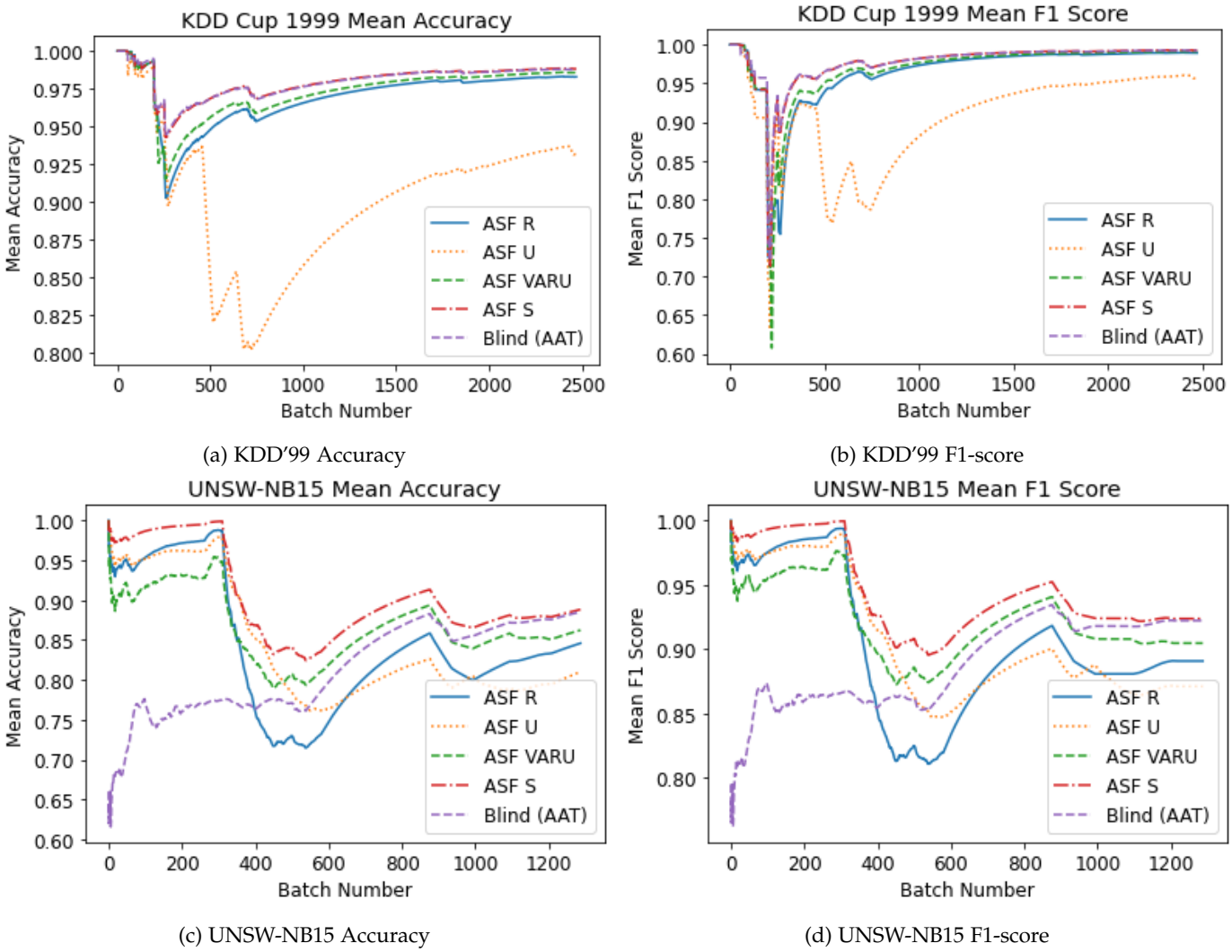
The AAT method makes use of *blind* adaptation, whereby the model is trained on all labeled samples. This has the drawback of high cost due to full labels and slow adaptation times to change occurring in the data stream. The research further explored change detection and active learning strategies, as outlined by Žliobaitė et al. [3], to further improve performance for a lower overall cost.

An ASF framework was implemented along with the random, uncertainty, variable uncertainty and split active learning strategies. With the uncertainty strategy, a new method for AE was proposed, whereby the average RE difference from the threshold is used as a baseline to detect samples with high uncertainty, defined as being in the proportion of the population with the smallest difference, tuned by a confidence parameter.

The use of ASF demonstrated that better accuracy, kappa and F1 scores can be achieved, compared to blind adaptation, with just 20% of the labeling cost, enabled by active learning of the most important samples to accelerate the learning process [3]. The results align to those presented by Žliobaitė et al. [3], with a split strategy being recommended as this fulfills all three active learning requirements to maintain a fixed budget, access to all samples within the stream and preserve the distribution of incoming data for detecting changes. Unlike Žliobaitė et al. [3], this research recommends inclusion of the uncertain samples with the change detection to improve per class performance.

6 Conclusion

The aim of this research was to explore semi-supervised online autoencoder methods for the task of anomaly in-



(a) KDD'99 Accuracy

(b) KDD'99 F1-score

(c) UNSW-NB15 Accuracy

(d) UNSW-NB15 F1-score

Fig. 6: ASF Strategy Comparison, $B = 20\%$

trusion detection on non-stationary network data streams, adapting to concept drift over time, with minimal labeling cost, by adopting an active learning change detection strategy. A unique contribution of this research was to compare a selection of anomaly threshold methods, proposing memory adaptations for data streams and a hybrid Adaptive Anomaly Threshold method which demonstrated superior performance. One of the more striking findings of the research is that the processing time of the autoencoder anomaly detector method is significantly lower when compared to traditional online learning techniques, making it well adjusted for high speed online network data streams, demonstrating an ability to detect an equivalent number of cyber attacks to traditional online learning methods, in a significantly reduced time frame. An area of future research would be to explore alternative threshold methods, such as clustering, which may allow for better identification of classes that overlap with normal samples and multi-label classification.

A further contribution of this research was to evaluate the autoencoder method with an Active Stream Framework, allowing the labeling cost of the data stream to be significantly reduced to a budget of 20%. A novel variable uncertainty strategy was proposed for autoencoders where the posterior probability is not available, instead tracking the distribution of sample RE distances from the anomaly threshold to determine uncertainty. An area of future research should be how to efficiently annotate samples, possibly by unsupervised clustering methods such as those demonstrated by [32].

Overall this research has demonstrated that the proposed Split Active Learning Anomaly Detector (SALAD) method can demonstrate high levels of performance with network data streams, which significantly reduced the labeling cost. The results are not perfect however, and it would be recommended to combine in a hybrid intrusion detection model whereby misuse detection is used before or after the anomaly detector to further identify classes, reduce false positives and better identify minority classes. Multi-label classification would be a further research area to expand on this work and provide additional context to detections.

References

[1] A. L. Buczak and E. Guven, "A survey of data mining and machine learning methods for cyber security intrusion detection," *IEEE*

Communications Surveys & Tutorials, vol. 18, no. 2, pp. 1153–1176, 2016.

[2] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, and A. Bouchachia, "A survey on concept drift adaptation," *ACM computing surveys (CSUR)*, vol. 46, no. 4, p. 44, 2014.

[3] I. Žliobaitė, A. Bifet, B. Pfahringer, and G. Holmes, "Active learning with drifting streaming data," *IEEE transactions on neural networks and learning systems*, vol. 25, no. 1, pp. 27–39, 2013.

[4] S. Mansalis, E. Ntoutsi, N. Pelekis, and Y. Theodoridis, "An evaluation of data stream clustering algorithms," *Statistical Analysis and Data Mining: The ASA Data Science Journal*, vol. 11, no. 4, pp. 167–187, 2018.

[5] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.

[6] S. Dua and X. Du, *Data mining and machine learning in cybersecurity*. Auerbach Publications, 2016.

[7] K. Alrawashdeh and C. Purdy, "Toward an online anomaly intrusion detection system based on deep learning," in *2016 15th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, 2016, pp. 195–200.

[8] ——, "Fast activation function approach for deep learning based online anomaly intrusion detection," in *2018 IEEE 4th International Conference on Big Data Security on Cloud (BigDataSecurity), IEEE International Conference on High Performance and Smart Computing,(HPSC) and IEEE International Conference on Intelligent Data and Security (IDS)*. IEEE, 2018, pp. 5–13.

[9] S. Roshan, Y. Miche, A. Akusok, and A. Lendasse, "Adaptive and online network intrusion detection system using clustering and extreme learning machines," *Journal of the Franklin Institute*, vol. 355, no. 4, pp. 1752–1779, 2018.

[10] X. Chen, C. Cao, and J. Mai, "Network anomaly detection based on deep support vector data description," in *2020 5th IEEE International Conference on Big Data Analytics (ICBDA)*. IEEE, 2020, pp. 251–255.

[11] M. M. Hassan, A. Gumaei, A. Alsanad, M. Alrubaian, and G. Fortino, "A hybrid deep learning model for efficient intrusion detection in big data environment," *Information Sciences*, vol. 513, pp. 386–396, 2020.

[12] C. Nixon, M. Sedky, and M. Hassan, "Autoencoders: A low cost anomaly detection method for computer network data streams," in *Proceedings of the 2020 4th International Conference on Cloud and Big Data Computing*, ser. ICCBDC '20. New York, NY, USA: Association for Computing Machinery, 2020, p. 58–62. [Online]. Available: https://doi.org/10.1145/3416921.3416937

[13] M. Nicolau and J. McDermott, "A hybrid autoencoder and density estimation model for anomaly detection," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 717–726.

[14] A. H. Mirza and S. Cosan, "Computer network intrusion detection using sequential lstm neural networks autoencoders," in *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE, 2018, pp. 1–4.

[15] T. Kieu, B. Yang, C. Guo, and C. S. Jensen, "Outlier detection for time series with recurrent autoencoder ensembles," in *28th international joint conference on artificial intelligence*, 2019.

[16] Y. Mirsky, T. Doitshman, Y. Elovici, and A. Shabtai, "Kitsune: an ensemble of autoencoders for online network intrusion detection," *arXiv preprint arXiv:1802.09089*, 2018.

[17] X. Li, W. Chen, Q. Zhang, and L. Wu, "Building auto-encoder intrusion detection system based on random forest feature selection," *Computers & Security*, p. 101851, 2020.

[18] J. Chen, S. Sathe, C. Aggarwal, and D. Turaga, "Outlier detection with autoencoder ensembles," in *Proceedings of the 2017 SIAM International Conference on Data Mining*. SIAM, 2017, pp. 90–98.

[19] T. Vaiyapuri and A. Binbusayyis, "Application of deep autoencoder as an one-class classifier for unsupervised network intrusion detection: a comparative evaluation," *PeerJ Computer Science*, vol. 6, pp. 1–26, 2020.

[20] B. Krawczyk, L. L. Minku, J. Gama, J. Stefanowski, and M. Woźniak, "Ensemble learning for data stream analysis: A survey," *Information Fusion*, vol. 37, pp. 132–156, 2017.

[21] T. S. Sethi and M. Kantardzic, "On the reliable detection of concept drift from streaming unlabeled data," *Expert Systems with Applications*, vol. 82, pp. 77–99, 2017.

[22] J. Shan, H. Zhang, W. Liu, and Q. Liu, "Online active learning ensemble framework for drifted data streams," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 2, pp. 486–498, 2018.

[23] Q.-V. Dang, "Active learning for intrusion detection systems," in *IEEE Research, Innovation and Vision for the Future*, 2020.

[24] Z. Zhang, Y. Zhang, J. Niu, and D. Guo, "Unknown network attack detection based on open-set recognition and active learning in drone network," *Transactions on Emerging Telecommunications Technologies*, vol. n/a, no. n/a, p. e4212. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4212

[25] J. Gama, P. Medas, G. Castillo, and P. Rodrigues, "Learning with drift detection," in *Brazilian symposium on artificial intelligence*. Springer, 2004, pp. 286–295.

[26] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*. IEEE, 2009, pp. 1–6.

[27] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 military communications and information systems conference (MilCIS)*. IEEE, 2015, pp. 1–6.

[28] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[29] J. Montiel, J. Read, A. Bifet, and T. Abdessalem, "Scikit-multiflow: a multi-output streaming framework," *The Journal of Machine Learning Research*, vol. 19, no. 1, pp. 2915–2914, 2018.

[30] J. Gama, R. Sebastião, and P. P. Rodrigues, "On evaluating stream learning algorithms," *Machine Learning*, vol. 90, no. 3, pp. 317–346, 2013.

[31] C. Nixon, M. Sedky, and M. Hassan, "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, 2019, pp. 1–5.

[32] Z. Cataltepe, U. Ekmekci, T. Cataltepe, and I. Kelebek, "Online feature selected semi-supervised decision trees for network intrusion detection," in *NOMS 2016-2016 IEEE/IFIP Network Operations and Management Symposium*. IEEE, 2016, pp. 1085–1088.

Christopher Nixon received his BEng(Hons) in Computer Science in 2008 and Masters by Research in Computing Science with distinction in 2020 from Staffordshire University. He has 11 years of experience in the financial services and telecoms industries, leading a team of cyber security experts in problem areas such as end user, cloud and network security. His research interests include machine learning and its application to cyber security.

Dr Mohamed Sedky is an Associate Professor in Artificial Intelligence and Machine Learning at Staffordshire University. After he received his BEng(Hons) in Electro-Physics and Communications in Alexandria University, Egypt, in 1996, Mohamed founded SKM communication systems. He finished his MSc degree in Communications and Electronics from the AAST in 2002. In 2009, he gained a PhD from Staffordshire University. He has led the Artificial Intelligence (AI) & Internet of Things (IoT) research group in Staffordshire University to conduct research in the development of artificial intelligent techniques and simulation tools for different Internet of things applications, robotic, and more recently applying his work in forensic science towards the detection and classification of microplastics.

Mohamed Hassan is leading the cyber security award at Staffordshire University. He holds a Master of Science degree in computer science with distinction and has extensive industry and academic experience in computing and cyber security. His teaching and research focus in the field of cyber security, machine/deep learning and digital forensics. He is also interested in other topics in cyber security, such as malware analysis and reverse engineering, security in embedded systems and IoT security.