A Hybrid Online Offline System for Network Anomaly Detection

Murugaraj Odiathevar
Engineering and Computer Science
Victoria University of Wellington
New Zealand
murugaraj.odiathevar@ecs.vuw.ac.nz

Winston K.G. Seah
Engineering and Computer Science
Victoria University of Wellington
New Zealand
winston.seah@ecs.vuw.ac.nz

Marcus Frean
Engineering and Computer Science
Victoria University of Wellington
New Zealand
marcus.frean@ecs.vuw.ac.nz

***Abstract*—With the advancement in technology, normal network traffic is becoming more heterogeneous. In this scenario, the problem of detecting anomalies is intensified. In the literature, offline methods see more data and can be optimised to achieve lower false positive rates. However, they cannot readily adapt to changing network conditions or capture *concept-drift*. This necessitates an incremental online learning model. On the other hand, online training is easily affected by noise. In this paper, we propose a hybrid Online Offline system in which the Offline model retains general characteristics of network traffic while the Online model continuously learns. The Offline model acts as a bias for the Online model to select new data to learn from. The Online model retains its knowledge and adapts to the changing ground truth. They are put to work together to detect anomalies. We implement this idea with an Online Support Vector Machine (SVM) which retains its support vectors and shifts its decision boundary guided by an Offline Radius Nearest Neighbor (Rad-NN). The method is evaluated on the NSL-KDD 2009 dataset. This relatively simple model achieves over 95% accuracy on known anomalies and over 60% detection rate on most of the unknown anomalies.**

***Index Terms*—Anomaly detection, Concept-drift, Incremental learning, Support Vector Machine, Radius Nearest Neighbour**

I. Introduction

One of the main challenges in anomaly detection is the notion of *concept-drift*. Concept-drift refers to a change in the underlying distribution of the data. With the advancement in technology, new applications are being created almost daily around the world. Also with the recent advances in the Internet of Things (IoT), normal traffic profiles display large variations and heterogeneity. Many network anomaly detection techniques miss this aspect of the network and focus on algorithms. As such, there is a need for systems to be able to adapt to the changing ground truth. And our proposed method attempts to fill this gap.

Traditionally, machine learning models are trained from a dataset and then deployed in the real-world. Whenever the model's efficiency withers, it needs to be retrained with new data. In the supervised learning context, obtaining new data and labelling them manually is painful. Network traffic changes faster than one can label and retrain a model. Unsupervised learning method for anomaly detection mainly consist of outlier detection. Unfortunately, new normal traffic may also be outliers at a different point in time such as *Flash-Crowd* traffic. To combat the weaknesses of each model, hybrid models are used. There have been hybrids between supervised and unsupervised and between misuse-based and behaviour-based intrusion detection systems [1]. Combination between offline and online methods in the literature usually involve an offline step and then online learning. Offline learners are the traditional machine learning models. They can see more data and pick out patterns that are not easily recognisable by humans. Though they are optimised, they suffer from lengthy training times. Online learners are in the form of incremental learners where they learn in time windows as new data arrives. They are more suitable in tracking concept-drift. However, they are required to be quick and thus are difficult to optimise in real time. Furthermore, they suffer from noisy or irrelevant data points during training. Most of the online methods in the literature are generally unsupervised methods. Our approach is a form of semi-supervised learning where the Online model is trained using labelled data with a confidence score. Other semi-supervised learning methods include the use of data from only the normal class such as the one-class Support Vector Machine (SVM) [2] or the use of a few labelled data points [3]. This paper presents a hybrid Online Offline system to address the inadequacies of each individual model. The main challenge is enabling an Online and an Offline model to work together in a logical and effective manner to detect anomalies because these methods are classifier dependent. We address this issue with the use of confidence scores.

Another aspect of machine learning models is that be it supervised, unsupervised or hybrid, they have a bias-variance trade-off. Models with high bias are over simplified and may not have captured the regularities of the data while models with high variance are over-fit to training data and do not generalise well to unseen data. In the network anomaly context, both false positive and false negatives can be costly and this is one of the reasons why machine learning for network anomaly detection is not readily implemented in the real world [4].

This paper attempts to overcome the above mentioned limitations and fill in the gap of adapting to normal network traffic by a novel Hybrid Online Offline System. The proposed contributions of this paper include

(i) A method to enable an Offline and an Online model to work together

(ii) A technique to train an Online model effectively
(iii) An approach to address concept-drift
(iv) An approach to split and handle bias-variance trade-off separately

The Offline model we use is a Radius Nearest Neighbor (Rad-NN) which learns the general characteristics of the network data. It classifies data points with a confidence score. This acts as an inductive bias for the Online model to select data points to learn from. The Online model which is a SVM, retains its support vectors and learns incrementally in batches to address concept-drift should it occur. It helps to improve the detection rate of the system. With this hybrid system, we want both the stability for ground concepts and flexibility to deal with a changing environment [5] and to tackle both bias and variance separately. Furthermore, this approach is relatively easier to implement compared to many other methods in the literature. To the best of our knowledge, this is the first time such an approach is used for network anomaly detection. We test our model on the NSL-KDD 2009 dataset [6] and the initial results are promising.

The remainder of this paper is organised as follows: Section 2 surveys other related state of the art works; Section 3 introduces the framework and methodology of our proposed approach; Section 4 explains the experiment; Section 5 discusses the experimental results; Section 6 sees some future directions we can take this approach and we conclude the paper in Section 7.

II. Related Work

There are many papers on network anomaly detection in the literature and they can be categorized as either online, offline or a combination of them. A comprehensive review on the many techniques used can be found in [1]. used. The idea of combining an Online and an Offline model is utilised by Fischer *et al.* [5]. Here the authors implement the concept for image recognition. They use an offline Localised Generalised Matrix learning Vector Quantization (LGMLVQ) and online incremental classifiers such as incremental online LVQ and incremental SVM with a reliability measure for each classifier and choose the output of the more reliable classifier.

A. Offline methods

Offline methods are the traditional methods which are trained and tested offline and then deployed in a system to perform anomaly detection. Ingre *et al.* [7] evaluate the performance of the NSL-KDD 2009 dataset using Artificial Neural Networks (ANN) with different optimisation algorithms. ANN have been proven to be able to estimate discriminative functions even with high dimensional data. Yin *et al.* [8] propose a deep learning approach using Recurrent Neural Networks (RNN). The authors experimented with different number of hidden nodes and varying learning rates. RNN is able to capture the temporal characteristics of recent data. Javaid *et al.* [9] propose Self-Taught Learning, a deep learning based approach to extract features of data. They use a Sparse Autoencoder(AE) to extract representations from an unlabelled training set. Thereafter, they train a soft-max regression on these new representations of the labelled training set. This is a form of unsupervised feature learning followed by supervised training. AEs are deep neural networks that are trained to regenerate the input as an output having learnt a latent representation in the hidden layer [10]. The representation in the hidden layer is taken as input to the anomaly detection system or a full AE trained on only normal data is also used. The rationale is that this AE will not be able to replicate an anomalous data point and give higher reconstruction error on anomalies. Aygun *et al.* [11] propose an Autoencoder and a Denoising Autoencoder in similar fashion but determine the threshold for the reconstruction error stochastically. For the above papers, after training, the weights of these models are fixed and as such they are unable to address concept-drift effectively. Sun *et al.* [12] uses double layer detection to firstly detect one category of network traffic using Gradient Boosting Decision Tree. Upon detection, of a specific class of anomalies, the authors use k-Nearest Neighbor (k-NN) to classify into subclasses. To detect other types of network traffic, the authors use a stacking ensemble of six classifiers.

Offline methods are able to achieve higher detection rates on known anomalies but are not efficient in detecting new anomalies or variants of existing anomalies. They cannot readily adapt to changing network conditions as they need to be completely retrained from scratch with new data.

B. Online methods

Online models are trained with new data as it comes. They are mostly continuous learners, unsupervised and learn from data grouped in time windows.

Bigdeli *et al.* [13] experiment with a two-layer cluster based approach. The authors represent each cluster by a Gaussian Mixture Model (GMM) and new instances are collected in batches and checked for similarity with existing clusters. If the new GMM is close to one of the existing cluster GMMs, they are merged to update the cluster. The strategy helps to stream out noise but merging only similar clusters could lead to loss of information. Self-Organising map (SOM) is a type of ANN used to perform dimensional reduction in an unsupervised manner. Huang *et al.* [14] propose an adaptive model to learn majority patterns under a dynamic changing environment. The authors use SOM to extract features from network traffic and a single-layer feed forward neural network for incremental learning. Ippoliti *et al.* [15] propose an adaptive growing hierarchical SOM which addresses concept-drift compared to the traditional SOM. The authors implement an enhanced threshold based training which accounts for patterns that differ only on one or two features whereas traditional SOM uses mean error with respect to all features. Data is normalized dynamically using observed value ranges instead of ranges in the training data and the error thresholds are changed over time. Alrawashdeh *et al.* [16] propose a fast activation function called Adaptive Linear Function (ALF) to allow the deep neural network to converge using fewer epochs for real-time applications. The methods above use unsupervised learning

which addresses the lack of labelled data for network traffic. They are better equipped to handle concept-drift but they are susceptible to noise during training. In this paper, we propose to train a model online with labels based on a confidence score instead of using completely unlabelled data.

C. Combination methods

There are few methods in the literature that combines offline and online methods. Existing methods consist of an offline step to select necessary features in a supervised or an unsupervised manner followed by online learning.

Su *et al.* [17], [18] improves k-NN based classifiers using clustering to identify DoS attacks. To improve the number of clusters, a recursive algorithm is proposed. It first computes the dispersion amongst each cluster. For the cluster with the highest dispersion, two points are computed based on distances between points as new centers. The cluster is split into two. This partition is valid if the average dispersion is reduced. The final resulting centroids are used instead of individual instances for k-NN classification. Alaei *et al.* [19] propose an anomaly detection method using an offline and an online step. The data is preprocessed offline to select features for each type of attack in the data set. A naive Bayesian module incrementally predicts the probability that the new point belongs one of the attack classes. The authors employ an active learning strategy where a random threshold is selected and the labels for instances near this threshold are inquired and the classifiers are updated where necessary. Else, the point is classified according to maximum certainty of the classifiers. This method does not account for shifting normal traffic. An incremental k-NN SVM method is explored by Xu *et al.* [20]. The data is split into batches and clustered using K-Means clustering. The cluster centers are classified using k-NN from a database of points. All points in the cluster are then labelled with the label of the cluster center. If the k neighbors do not have the same label, an SVM is trained on the neighbors before classification. The issue with k nearest neighbors as we will explain in the next section is that the neighbors may not necessarily be "near" enough.

III. Framework and Methodology

One of the methods to enable an Online and an Offline classifier to work together is using confidence scores. In our work, the Online classifier helps to classify if the Offline classifier's confidence does not meet a specified threshold. The Online model is required to have low training time. It also needs to be adaptive and continuously change as the ground truth changes. To improve its training, the Online model only accepts points that are classified with high confidence for training. Figure 1 depicts the system overview.

F denotes the data to be classified. The Offline model is trained offline with labelled data. The Online model is trained with recent data that has traversed through the network and their corresponding labels are as accurate as the confidence of the system. F is classified by the Offline model with confidence C^*. If C^* is lower than a specified threshold, $thres$, or the Offline model is unable to classify it ($C^* = 0$), the Online model steps in to classify F with confidence $C\hat{}$. The label, $L(C)$ then depends on the model with the higher confidence, $C = max(C^*, C\hat{})$. Thereafter, we can obtain the data classified by the system with high confidence, $C > u$ for an upper threshold u. The Online model will retrain itself by retaining some knowledge and using high confidence points when a retraining criteria is met. To implement our system, we chose Rad-NN as the Offline model and SVM as the Online model. We justify our selection as follows.

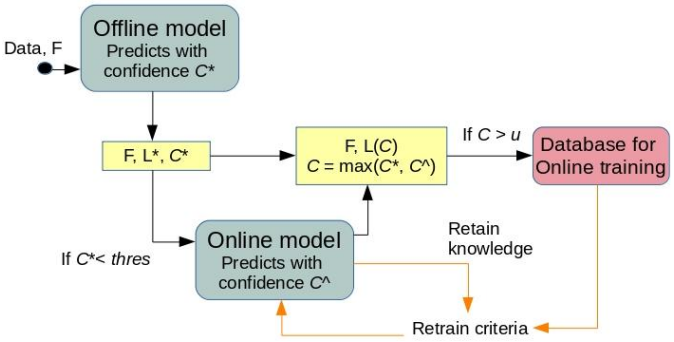


Fig. 1. System overview

Nearest Neighbor models are also known as lazy learners because they do not learn a discriminative function but "memorize" the dataset [21]. They simply maintain a knowledge base of points in hyperspace and when tasked with classifying a new point, F, the model calculates the distance between F and its neighbors. It then counts the votes or classes of the neighboring points and classifies F. This is an appropriate model to retain knowledge or useful points that encompass the general features of network traffic. The voting can be done based on equal weighting or distance weighting where closer points have a higher weight on their vote for F. Equal weighting might not be appropriate in the presence of class imbalance. The commonly used k-Nearest Neighbor (k-NN) model does this based on a user defined value k for the number of nearest neighbors to vote. An issue with k-NN is that sometimes a point might be quite far from all of the other points and yet it is able to classify the new point. Rad-NN counts the votes of neighboring points within a specified radius. In this context, if F is too far away, it will be an outlier and the model will be unable to classify it. The complexity for the search is of $O(d \times log_2(n))$, where n is the number of points and d is the number of features with an implementation of a *Ball tree* structure.

Upon classification, the confidence C^* of the Rad-NN's classification of point F is calculated using the following function.

$$C^*(F,k) = 1 - \frac{\sum_{i=1}^{k} D(F, NLN_i(F))}{\sum_{i=1}^{k} D(F, NUN_i(F))} \quad (1)$$

$D(a, b)$ represents the distance between points a and b. $NLN_i(F)$ represents the i^{th} nearest like neighbor of F while $NUN_i(F)$ represents its i^{th} nearest unlike neighbor [22]. The number of like points or unlike points are not constrained by the value of the radius. Since Rad-NN uses distance weighting for classifying, this measure will always be less than 1. This confidence scoring addresses two points. Firstly, the closer the target data point is to unlike points, it is highly likely to be near the decision boundary. As such, the denominator value will be low and the confidence score will be low. Secondly, if the target data point has few like neighbors close to it, then it is occurring in a sparse region. The numerator value will be high as it would have to take the distances of like points farther away and so the confidence score will be low. This scoring methodology is independent of the classification methodology.

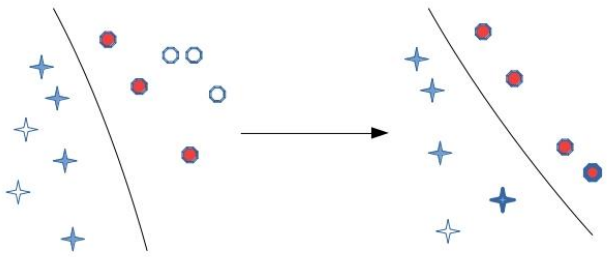
Nextly, for the Online model, we choose SVM. SVMs are simple and efficient machine learning models in high dimensional spaces. They use the kernal trick to map data into a higher dimensional subspace where a decision boundary, given by a decision function, is obtained [23]. SVMs can be trained with limited data. As normal data changes, SVM can incrementally learn with the new data and shift its decision boundary. We choose the Radial Basis Function (RBF) kernel because it does not assume any prior knowledge about the data, is invariant to translation and is able to fit every target value exactly [23]. Furthermore, the bias and variance can be adjusted using the C and γ parameters in the RBF kernel. C is the penalty parameter. A low value of C will give us a simpler decision function at the cost of training accuracy. γ is the inverse of the influence of each training point on the decision function and adjusts the spread of the decision region. In our case, the SVM can have high variance because it is trained on high confidence points. Training time is $O(max(n, d) \times min(n, d)^2)$ [24]. For a small number of samples, this does not pose an issue. Furthermore, once the SVM is trained, the support vectors are enough to describe the current decision boundary. These support vectors will be retained for the next retraining in the system. The SVM will be retrained in batches when the number of high confidence points in the database reaches a specified number. In this way, as depicted in figure 2, the decision boundary of the SVM will shift as new traffic arrives.

For the SVM, its confidence $C\hat{}$ can be computed using Platt scaling on the scores of the decision function. Platt scaling is an algorithm to give a probability estimate on the output of the decision function. This probability is calibrated using the data points used to obtain the decision function [25].

IV. Experiment and results

A. Dataset

The NSLKDD 2009 dataset is an improved version of the KDD Cup 99 dataset which was prepared using network traffic captured by 1998 DARPA IDS evaluation program [6], [26]. It contains four different types of attacks namely, DoS (Denial of Service), Probe, R2L (Remote to Local) and U2R (User to Root). The training dataset was collected for seven weeks and



The shaded items represent support vectors. Support vectors are retained on right diagram with the addition of 2 new points represented by thickness. The decision boundary shifts slightly and we obtain new support vectors.

Fig. 2. SVM decision boundary shift

the testing dataset was collected in the following 2 weeks. The testing data contains variations of the anomalies found in the training data to make the anomaly detection process realistic as most novel attacks are variants of known attacks. The KDD Cup 99 dataset contained too many redundant records biasing learning algorithms [27]. The NSLKDD was proposed to overcome the limitations of the KDD Cup 99. The dataset contains 41 features, of which 3 of them are nominal, 6 of them are binary and the remaining are numerical. Table I shows the count of each class in the dataset. In this paper, the labels of the attacks are replaced as anomaly.

TABLE I
NSLKDD 2009 DATASET

	Total	**Normal**	**DoS**	**Probe**	**R2L**	**U2R**
Training set	125973	67343	45927	11656	995	52
Testing set	22544	9711	7458	2421	2754	200

B. Evaluation measures

The results are evaluated based on false positive rate (FPR), true positive rate (TPR) also known as detection rate and accuracy (Acc). We will also use these measures on a validation set in the section IV-D to find optimal values for the system parameters.

$$\text{TPR} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2}$$

$$\text{FPR} = \frac{\text{FP}}{\text{FP} + \text{TN}} \tag{3}$$

$$\text{Acc} = \frac{\text{TP} + \text{TN}}{\text{Total number of data points}} \tag{4}$$

C. Data Preprocessing

We shall begin with one hot encoding of the nominal features to make them into a total of 90 binary features. Next, we decompose all binary features into 8 components using Principal Component Analysis (PCA). PCA transforms the feature set to a set of uncorrelated linear variables using orthogonal transformation. These 8 components explain at least 82% of the variability of these binary features. Taking

additional components accounts for marginal gains at implications of the curse of dimensionality.

We drop *'num_outbound_cmds'* feature as it only takes the value 0 in the entire dataset. Now we normalize the values to the range [0,1] so that they can be compared. For the 8 PCA components, we use Min-Max scaling. Min-Max scaling takes each value and subtracts the minimum value and divides over the feature range. However, we cannot use this method for the numerical features due to the effect of outliers which can highly distort the data. For the remaining 31 numerical features, we use the method used in [18].

$$\text{Normalization of } f_i = \frac{1 - e^{-lf_i}}{1 + e^{-lf_i}} \tag{5}$$

In this equation f_i is the observed value of the feature i. e represents the Euler number ($e \approx 2.7182818$) and l is a constant. Each numerical feature has its own l value. Since we want to adapt to normal traffic, the constant l is determined such that the average of feature i of the normal traffic instances is mapped to 0.5. If it is not possible, l is such that the average of feature i of all traffic instances in the training set is mapped to 0.5.

D. Model Training

To calculate the radius for Rad-NN and the distance between each point, we use the weighted euclidean distance.

$$D(\mathbf{a}, \mathbf{b}) = \sqrt{w_1(a_1 - b_1)^2 + \cdots + w_n(a_n - b_n)^2} \tag{6}$$

We use this metric because some of the features have more predictive power and our Rad-NN does not learn that some features are more discriminative than others. The feature weights are computed based on Chi Squared statistic [28] and rescaled to the range [0, 1] using Min-Max scaling.

Next, we take 20% of the training set to determine the value for the radius. For each point in this set, we compute the average of its distances from all other points. We take the 50th percentile of these distances as our radius because the distribution is skewed. Figure 3 shows the distribution. The smaller the radius, there is more responsibility on the online model for classification. The larger the radius, the system will obtain votes from points that are further away which is not appropriate as mentioned in the previous section.

The system contains a total of 6 parameters to specify. They are k from equation 1, the threshold value $thres$ for the acceptance of the classification of Rad-NN, u for selecting the point for online training, the number of points for online training and the C and γ parameters of the RBF kernal. We split the training set into 80%-20% proportion. The Rad-NN is trained on the 80% of the training set and the remaining 20% is used as the validation dataset to find optimal values for these parameters. The value for k in equation 1 and $thres$ are dependent. A larger k value gives a more robust confidence score for C^*. Calculating the confidence scores of each label classified by the Rad-NN, figure 4 shows that the minimum value tends towards 0 as k increases. Taking a high value of k however, is computationally costly as it takes longer to

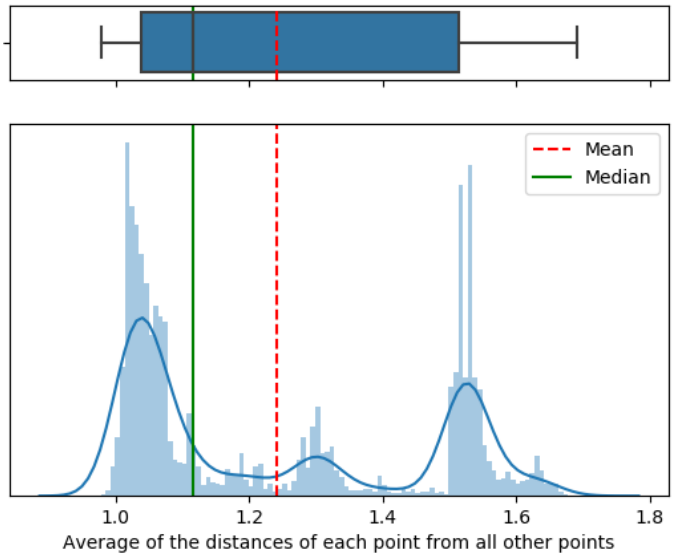


Fig. 3. Distribution of the average of the distances

calculate the confidence of the classified label and might not be feasible in real time. For each k, an optimal value for $thres$ is taken to be the maximum of the confidence scores of all the misclassified points in the validation set. In our experiment, we take $k = 10$ and the corresponding value for $thres$ is 92.2% as shown by the red point on figure 5. Subsequently, u must be greater than $thres$ as it is the second layer threshold to obtain high confidence points for online training. Using grid-search method on the system, we obtain the following parameter values. u =96% and 1300 for the number of points for online training gives the highest detection rate on the validation set as seen in figure 6. Similarly, using grid-search, $\gamma = 0.1$ gave the highest detection rate while $C = 100$ gave the lowest false positive rate. If a regularised SVM with a balance between bias and variance was used, the number of support vectors to retain for subsequent training will be large. As such, more high confident points need to be used at the expense of longer waiting time or the SVM be retrained regularly to ensure concept-drift is captured. Figures 7 and 8 shows the results of the validation process. For a low values of C or for $\gamma > 1$, the number of SVMs trained is very high if we fix retraining at 1300 points.

Before deployment in real systems, simulations would be done online prior to start up. To replicate this in our experiments, the online SVM is initialised using the first 1300 points in the testing set. The first 1300 points consist of 574 normal traffic, 453 DoS, 130 R2L, 130 Probe and 13 U2R points. The attacks are labelled as anomalies. The system will be evaluated on the remaining data points in the testing set.

E. Results

Rad-NN is trained on the entire training set and the following parameters which were determined in section IV-D are used in the system. SVM parameters are $C = 100$, $\gamma = 0.1$. Confidence threshold, $thres$ for Rad-NN is 92.2%.

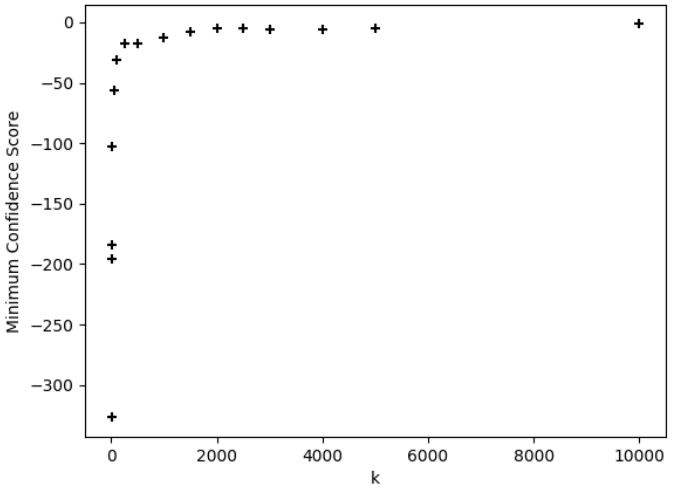


Fig. 4. Minimum C^* value against k

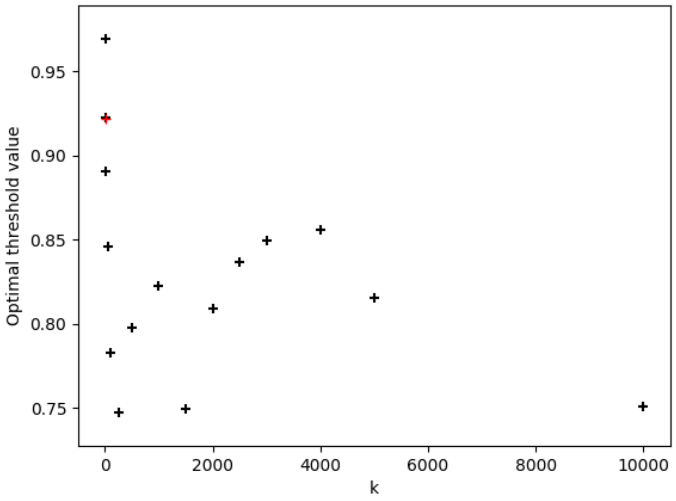


Fig. 5. Optimal *thres* value against k

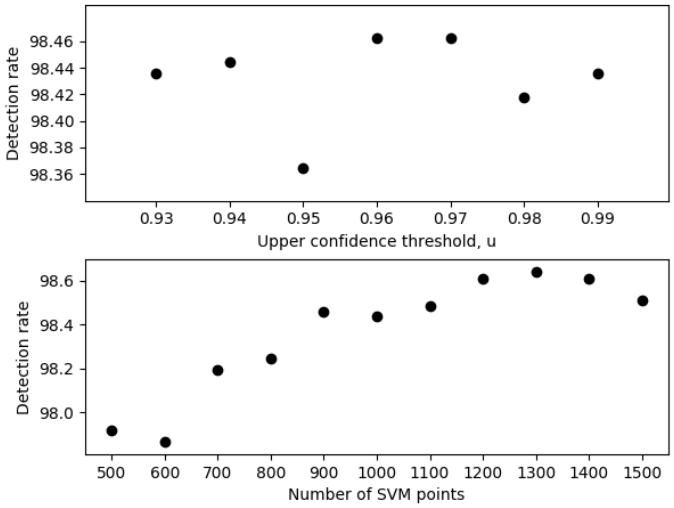


Fig. 6. Detection rate of validation set against u and number of points used in online training

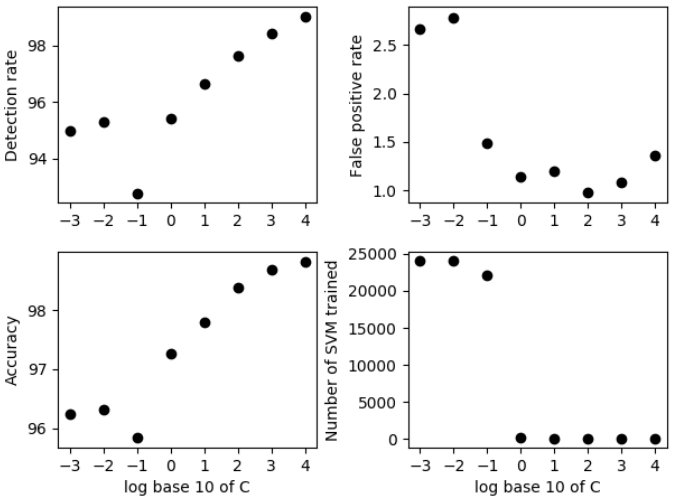


Fig. 7. Validation results for C in RBF kernel

k, to calculate its confidence is 10. Upper threshold, u for online training selection is 96%. The SVM is retrained when the number of previous support vectors and high confidence points reaches 1300. The results are as follows. The system achieves 95.55% accuracy, 94.42% detection rate and 2.96% false positive rate. The SVM was trained 11 times whilst predicting 21,244 data points in the testing set. Next we compare our system with other methods.

Table II compares our system with other offline + online algorithms implemented in a similar fashion. The Gaussian Naive Bayes (GNB) is used as the other online algorithm. In this experiment, all online models are retrained completely with no points being retained. There is no clear way to retain points for the GNB. We take the classification probabilities of GNB as its confidence measure. In table III, we compare our system with other offline methods. In table IV, we compare our system with other online methods.

To test our system further, we attempt to test our system to detect a new type of attack. We train the Offline model without each one of the attack types from the dataset. The respective attack is also removed from the first 1300 points used to initialise the Online model. The results are found in table V. We perform a similar experiment in table VI but this time initialise the Online model as before.

TABLE II
COMPARISON WITH OTHER OFFLINE AND ONLINE METHODS

Model Offline + Online	**FPR (%)**	**TPR (%)**	**Acc (%)**
Our System	**2.96**	**94.42**	**95.55**
Rad-NN + SVM[a]	3.80	88.82	92.00
Rad-NN + GNB	3.95	81.01	87.47
SVM + GNB	8.15	70.68	79.78

[a]SVM is purely incremental. Support vectors are not retained.

TABLE III
COMPARISON WITH OFFLINE METHODS

Model	FPR (%)	TPR (%)	Acc (%)
Our System	2.96	94.42	**95.55**
SVM (C=100, γ=0.1)	8.04	71.01	80.02
Rad-NN	**2.13**	59.33	75.90
Decision Tree	3.37	70.12	81.51
Gaussian Naive Bayes	3.26	62.04	76.95
k-NN (k=5,equal-weighted votes)	2.88	61.33	76.71
Artificial Neural Networks [7]	3.23	81.20	81.16
Deep Recurrent Neural Networks [8]	3.07	72.95	83.28
Self taught learning [9]	21.60[a]	**95.95**	88.39
Stochastically improved Denoising Autoencoder [11]	4.01[a]	83.08	88.65

[a]Derived from the confusion matrix of the published results

TABLE IV
COMPARISON TO OTHER ONLINE METHODS

Model	FPR (%)	TPR (%)	Acc (%)
Our System	**2.96**	**94.42**	**95.55**
SOM & Neural Network [14]	14.06	94.40	90.00
2 layer GMM clustering [13]	7.00	85.00	88.44[a]
Deep Belief Network with Adaptive Linear Function [16]	4.47[a]	95.2	95.34[a]

[a]Derived from the confusion matrix of the published results

TABLE V
DETECTION AFTER REMOVING EACH TYPE OF ATTACK DURING OFFLINE TRAINING AND ONLINE INITIALISATION

Training Data	% R2L detected	% DoS detected	% Probe detected	% U2R detected
Full System	78.39	99.60	97.68	85.03
Data without R2L	14.67	99.31	97.64	81.28
Data without DoS	73.89	79.69	98.52	83.42
Data without Probe	77.55	99.57	61.50	86.63
Data without U2R	77.93	99.60	97.95	80.21

TABLE VI
DETECTION AFTER REMOVING EACH TYPE OF ATTACK DURING OFFLINE TRAINING ONLY

Training Data	R2L % detected	DoS % detected	Probe % detected	U2R % detected
Full System	78.39	99.60	97.68	85.03
Data without R2L	73.66	99.60	97.42	81.81
Data without DoS	73.36	99.21	98.47	85.56
Data without Probe	77.25	99.44	93.28	86.10
Data without U2R	78.01	99.60	97.82	84.49

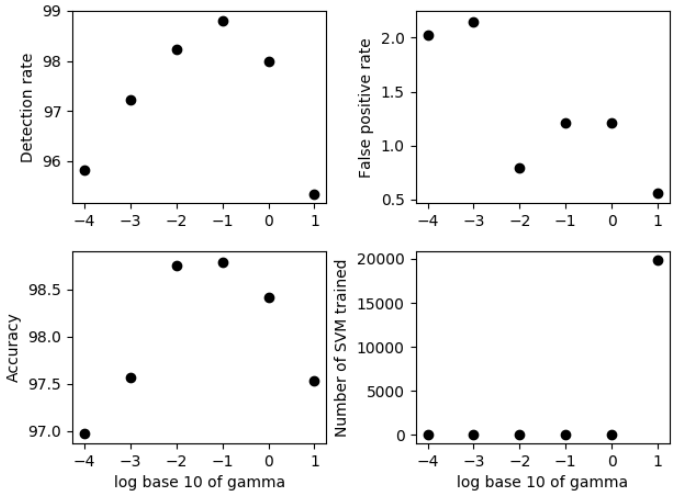


Fig. 8. Validation results for γ in RBF kernel

V. DISCUSSION

Firstly, retaining the support vectors as done in our system gives better overall results compared to Rad-NN + SVM in table II. Training without retention runs the risk of shifting the decision boundary too much, especially more since the data points are only as accurate as the confidence label.

Offline methods are not effective in capturing concept drift and higher detection rates comes at the expense of higher false positive rates as seen in table III. The Rad-NN by itself gives us the poorest detection rate. This is certainly expected as its role is to capture the general features of the network traffic. The SVM trained offline also does not perform well because it was trained with too many irrelevant points, i.e. noise. Though they do not perform as well individually but together, they are able to overcome their inadequacies. The performance of Rad-NN is improved when complemented with an online model in table II. However, the SVM trained purely offline shows better performance than SVM + GNB. This is mainly because the offline SVM misclassifies with higher confidence than the Rad-NN and more incorrectly labelled points are used in the training of the online model. Also, since the confidence scoring of Rad-NN is independent of its classification methodology, the Rad-NN does not face this issue. This is a caveat of our system and the results suggest that a strong offline classifier with a robust confidence scoring methodology is necessary.

Comparing table IV with table III, online methods have higher overall detection rates but they also have higher false positive rates. This is because it is difficult to optimise in real time. Online models do a better job at capturing concept drift and so they have higher detection rates.

In table V, after complete removal of each type of attack, the system still detected more than 50% of DoS, Probe and U2R data points as anomalies. However, it was only able to detect 14.67% of the R2L attacks. This may be because R2L attacks are similar to normal traffic and possibly closer to the boundary. Figure 9 shows a PCA analysis of the testing

set. In two dimensions, R2L and normal data points are more agglomerated compared to the other anomalies. In table VI, we gave the online model a head start with a few points of each attack trying to mimic the situation where the online model has already detected a new attack before. As such, we can see that there is a huge improvement in detecting each type of attack compared to table V. Hence, our system is not yet complete. We leave it to future work to be enable learning of new anomalies. Right now, the system can detect some new anomalies because the SVM boundary separates the normal and known anomalous points and tracks concept-drift of normal regions. However, to effectively detect them, it still needs to learn their profiles. Also, one weakness of our system is when new anomalies close to the SVM decision boundary such as R2L attack data points occur, the system does not perform well.

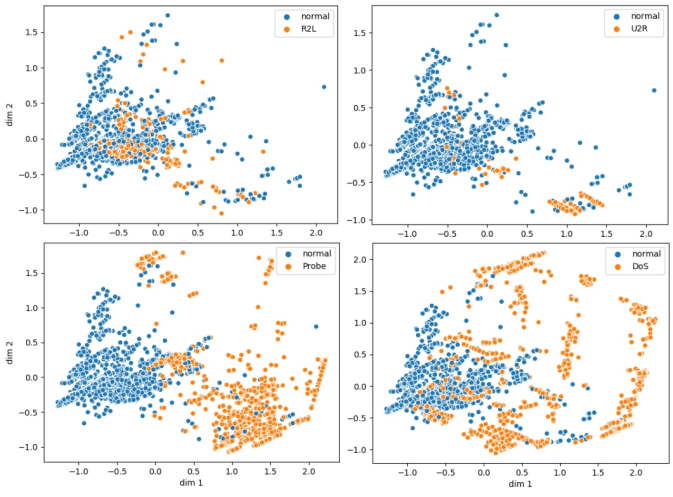


Fig. 9. PCA analysis of testing set

In tables V and VI, the system trained without certain attacks enables better detection of other anomalies while poorer detection for some others. For example in table V, without DoS attacks, Probe attacks have a higher detection rate while U2R attacks have a lower detection rate. There may be a few reasons for this phenomenon. Firstly, with fewer unrelated anomalies in the RadNN, there is less noise. On the contrary, the misclassification comes at a cost of training the SVM incorrectly. Furthermore, the SVM is overfit with a high penalty parameter C in the RBF kernel. The fluctuations are not consistent because part of the detection process depends on the shifting decision boundary of the online SVM and this depends on the recent points that have traversed through the system.

VI. Future Work

In this section, we shall discuss adaptation of the system and possible improvements as future work. Firstly, there is room for flexibility within the Rad-NN and SVM models. Another threshold can be placed to obtain low confidence data. Changes in confidence score distribution is able to show when we need to update the Offline model [29]. Another layer of machine learning model can be placed on the low confidence data points for this task. That being said, the Offline model can be updated at a later time with some of the points with low confidence after further processing. Some of the least used points in Rad-NN in confidence calculation or classification can also be removed to save memory and reduce order of complexity.

We used all features here for our system which introduces noise. If we are aware of the anomalies to detect, we can train our model on specific features. For example, to detect R2L attacks, we can use duration of connection, service requested and number of failed login attempts instead of all features [6]. For detecting DoS attacks, aggregating flows by source and destination instead of a 5 tuple and using number of sources and average packet sizes may be better suited [30]. That being said, this system can be deployed in parallel and be trained with specific features. Each instance can specialise to detect one type of anomaly as in [19] instead of trying to do everything together. This can also address the implications of the curse of dimensionality.

Another point to note is that we weighted the features based on the training data and we removed one feature because it did not have any predictive power. This may not be the best approach for detecting new anomalies because the preprocessing steps depend on the data we have. We can't detect new anomalies effectively considering features based on available data. Unsupervised feature learning methods such as in [9] might be appropriate. More research is required to find an appropriate method for feature weighting and selection that is independent of the training data set. In the same sense, the l value from equation 5 and the feature weights from equation 6 are computed based on data in our Offline model. With changing normal traffic, these would eventually also change. Hence, the Offline model also needs to update itself.

In this paper, we did not look at any other temporal characteristics of network traffic such as correlation between subsequent flows. We can improve on this model by incorporating time series analysis. This would enable us to strengthen the normal profile of network traffic. Moreover, we may find other optimal conditions to invoke the SVM retraining or enable hyperparameters such as C and γ to be functions of time or other network characteristics. In the real world, we cannot wait until a flow is completed to make a prediction. Thus, a model needs to classify using features of subflows where the data is collected in the first few seconds. Even though we experimented with full flows, we did not make any assumptions on the features for the model but taking feature values in the first few seconds may render certain features less effective. Hence, with appropriate time windows, normalizing and scaling of the features, this can be explored further. Furthermore, our model can be implemented at the application layer in a Software Defined Networking (SDN) framework as described in [31]. We leave it as future work to implement into a live system and account for other variabilities.

Instead of using Rad-NN or SVM, other machine learning

models can be explored as well and we leave this to future work. The main challenges are to implement robust confidence measures for each model and enabling them to work together in a seamless manner.

VII. Conclusion

In summary, this paper describes an Online Offline hybrid system. The Offline Rad-NN model stores knowledge in the sense of data points and its neighborhood. It classifies by obtaining votes from points within a radius. This paper also gives a method to calculate a confidence score for the Rad-NN model independent of the classification methodology. If the confidence is low, the online SVM aids with the classification. If the confidence is high, the online SVM selects this point for learning. The online SVM captures concept-drift by retraining itself in intervals. The model has relatively low order of complexity for real time detection compared to other systems in the literature. Our experimental evaluation shows that our system is equally effective as other models in the field. The system can be deployed in real time with avenues for further customisation. We also discussed the numerous directions in which we can improve and extend this model.

VIII. Acknowledgements

Murugaraj Odiathevar is supported by the Victoria Doctoral Scholarship.

References

[1] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo, "A comprehensive survey on machine learning for networking: evolution, applications and research opportunities," *Journal of Internet Services and Applications*, vol. 9, no. 1, p. 16, 2018.

[2] D. Ippoliti, C. Jiang, Z. Ding, and X. Zhou, "Online adaptive anomaly detection for augmented network flows," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 11, no. 3, p. 17, 2016.

[3] R. A. R. Ashfaq, X.-Z. Wang, J. Z. Huang, H. Abbas, and Y.-L. He, "Fuzziness based semi-supervised learning approach for intrusion detection system," *Information Sciences*, vol. 378, pp. 484–497, 2017.

[4] R. Sommer and V. Paxson, "Outside the closed world: On using machine learning for network intrusion detection," in *Security and Privacy (SP), 2010 IEEE Symposium on*. IEEE, 2010, pp. 305–316.

[5] L. Fischer, B. Hammer, and H. Wersing, "Combining offline and online classifiers for life-long learning." in *IJCNN*, 2015, pp. 1–8.

[6] L. Dhanabal and S. Shantharajah, "A study on nsl-kdd dataset for intrusion detection system based on classification algorithms," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 4, no. 6, pp. 446–452, 2015.

[7] B. Ingre and A. Yadav, "Performance analysis of nsl-kdd dataset using ann," in *Signal Processing And Communication Engineering Systems (SPACES), 2015 International Conference on*. IEEE, 2015, pp. 92–96.

[8] C. Yin, Y. Zhu, J. Fei, and X. He, "A deep learning approach for intrusion detection using recurrent neural networks," *IEEE Access*, vol. 5, pp. 21 954–21 961, 2017.

[9] A. Javaid, Q. Niyaz, W. Sun, and M. Alam, "A deep learning approach for network intrusion detection system," in *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2016, pp. 21–26.

[10] S. Naseer, Y. Saleem, S. Khalid, M. K. Bashir, J. Han, M. M. Iqbal, and K. Han, "Enhanced network anomaly detection based on deep neural networks," *IEEE Access*, vol. 6, pp. 48 231–48 246, 2018.

[11] R. C. Aygun and A. G. Yavuz, "Network anomaly detection with stochastically improved autoencoder based models," in *2017 IEEE 4th International Conference on Cyber Security and Cloud Computing (CSCloud)*. IEEE, 2017, pp. 193–198.

[12] C. Sun, K. Lv, C. Hu, and H. Xie, "A double-layer detection and classification approach for network attacks," in *2018 27th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2018, pp. 1–8.

[13] E. Bigdeli, M. Mohammadi, B. Raahemi, and S. Matwin, "Incremental anomaly detection using two-layer cluster-based structure," *Information Sciences*, vol. 429, pp. 315–331, 2018.

[14] S.-Y. Huang, F. Yu, R.-H. Tsaih, and Y. Huang, "Network-traffic anomaly detection with incremental majority learning," in *Neural Networks (IJCNN), 2015 International Joint Conference on*. IEEE, 2015, pp. 1–8.

[15] D. Ippoliti and X. Zhou, "An adaptive growing hierarchical self organizing map for network intrusion detection," in *2010 Proceedings of 19th International Conference on Computer Communications and Networks*. IEEE, 2010, pp. 1–7.

[16] K. Alrawashdeh and C. Purdy, "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.

[17] M.-Y. Su, "Using clustering to improve the knn-based classifiers for online anomaly network traffic identification," *Journal of Network and Computer Applications*, vol. 34, no. 2, pp. 722–730, 2011.

[18] ——, "Real-time anomaly detection systems for denial-of-service attacks by weighted k-nearest-neighbor classifiers," *Expert Systems with Applications*, vol. 38, no. 4, pp. 3492–3498, 2011.

[19] P. Alaei and F. Noorbehbahani, "Incremental anomaly-based intrusion detection system using limited labeled data," in *Web Research (ICWR), 2017 3th International Conference on*. IEEE, 2017, pp. 178–184.

[20] B. Xu, S. Chen, H. Zhang, and T. Wu, "Incremental k-nn svm method in intrusion detection," in *Software Engineering and Service Science (ICSESS), 2017 8th IEEE International Conference on*. IEEE, 2017, pp. 712–717.

[21] S. Raschka and V. Mirjalili, *Python Machine Learning, 2nd Ed.* Birmingham, UK: Packt Publishing, 2017.

[22] R. Hu, S. Delany, and B. MacNamee, "Sampling with confidence: using k-nn confidence measures in active learning," in *Conference papers*, 2009, p. 50.

[23] C. M. Bishop, "Pattern recognition and machine learning (information science and statistics) springer-verlag new york," *Inc. Secaucus, NJ, USA*, 2006.

[24] O. Chapelle, "Training a support vector machine in the primal," *Neural computation*, vol. 19, no. 5, pp. 1155–1178, 2007.

[25] H.-T. Lin, C.-J. Lin, and R. C. Weng, "A note on platt's probabilistic outputs for support vector machines," *Machine learning*, vol. 68, no. 3, pp. 267–276, 2007.

[26] "Kdd cup 99."

[27] 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.

[28] D. P. Vivencio, E. R. Hruschka, M. do Carmo Nicoletti, E. B. dos Santos, and S. D. Galvao, "Feature-weighted k-nearest neighbor classifier," in *Foundations of Computational Intelligence, 2007. FOCI 2007. IEEE Symposium on*. IEEE, 2007, pp. 481–486.

[29] A. Haque, L. Khan, M. Baron, B. Thuraisingham, and C. Aggarwal, "Efficient handling of concept drift and concept evolution over stream data," in *2016 IEEE 32nd International Conference on Data Engineering (ICDE)*. IEEE, 2016, pp. 481–492.

[30] J. Dromard, G. Roudière, and P. Owezarski, "Online and scalable unsupervised network anomaly detection method," *IEEE Transactions on Network and Service Management*, vol. 14, no. 1, pp. 34–47, 2017.

[31] C. Song, Y. Park, K. Golani, Y. Kim, K. Bhatt, and K. Goswami, "Machine-learning based threat-aware system in software defined networks," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*. IEEE, 2017, pp. 1–9.