HEAL: Heterogeneous Ensemble and Active Learning Framework

Anubhav Chhabra[†,*], Tirumala Sree Akhil Nandyala[†], Paula Branco[†]
[†] School of Electrical Engineering and Computer Science, University of Ottawa

Abstract

Network Intrusion Detection is a well-known, relevant problem that has gained even more interest due to the exponential growth of technologies, systems and volume of data. The constantly evolving attacks require a continuous effort towards the development of novel and robust detection solutions. In this paper we propose a Heterogeneous Ensemble and Active Learning (HEAL) system, a novel tool that incorporates the implementation of a dynamic heterogeneous ensemble model with active learning capabilities. This ensures a solution that: i) adapts to changes in data through time, ii) remains robust providing good performance, iii) handles a continuous flow of data, and iv) requires less human intervention when compared against pure active learning solutions. HEAL system uses multiple individual base models to build a heterogeneous ensemble learner that adapts to the specific data characteristics. Then, active learning is applied to the ensemble so that it is retrained and re-evaluated with respect to time and new instances. Instances where the model has a low confidence are labeled by a domain expert. A new model is retrained with these instances and its performance is evaluated. The deployed model is replaced when the new model exhibits performance advantages. Finally, an experimental comparison of the performance at different stages is carried out in a case study using the well-known NSL-KDD data set. In this study we show the advantages of using HEAL system.

Keywords: network intrusion detection system, ensemble learning, active learning

1. Introduction

Novel and advanced ways to intrude networks and access business data or personal user's information are constantly being developed. The evolving nature of the attacks and increase in network traffic, requires the proposal of new techniques to effectively detect these intrusions. Many machine learning solutions have been proposed. Still, only a few exist that cope with the continuous flow of network traffic data. In this paper, we address important open issues of network intrusion detection, namely in terms of real-time systems capable of detecting time evolving threats while supported by a controlled and reduced intervention of domain experts. We propose an heterogeneous ensemble with an automatic mechanism for deciding when a model update is necessary. This solution works in real-time and makes use of an active learning procedure. Instances for which the heterogeneous ensemble has low confidence are manually classified by a domain expert being then re-used to train a new model. We developed a framework that implements our novel and automated HEAL solution mirroring a real-world system. Our main contributions are: i) build a dynamic heterogeneous ensemble model adapted to the predictive task; ii) incorporate an active learning procedure with the developed heterogeneous ensemble by automatically selecting low confidence samples and allowing the model to self update to cope with time evolving events when necessary and beneficial; and iii) develop and evaluate a framework that incorporates the developed solution and is able to deal with a continuous flow of data.

[*] achha096@uottawa.ca

2. Related Work

Several researchers have applied ensemble methods for tackling intrusion detection problems. In [1] an ensemble composed of 3 different algorithms is used while in [2] experiments are carried out with an ensemble of 6 classifiers. In [3] several ensemble classifiers for supervised anomaly-based network intrusion detection were also evaluated. In [4] various homogeneous and heterogeneous online ensemble models for network intrusion detection are compared, including one homogeneous ensemble and 3 heterogeneous ensemble.In [5] an online network intrusion detection system is implemented with an anomaly detection module containing a stacking ensemble which used an Auto-encoder, One-class SVM, and Random Forest as the base models.

Active learning is applied when only a small amount of labeled data is available, using a human annotator for providing labels for carefully selected unlabelled cases. Some research has been conducted in the intrusion detection using active learning methods. In [6] the TCM-KNN algorithm is presented and is integrated with uncertainty sampling for the active learning module.

3. The Heterogeneous Ensemble with Active Learning (HEAL) system

HEAL system contains two main modules that are implemented in a framework that enables real-time predictions. The first module builds an heterogeneous ensemble model for problem while the second module introduces active learning as a process to improve both the effectiveness and computational time of the models deployed.

3.1. Heterogeneous Ensemble Module

The heterogeneous ensemble model uses the features extracted from the network traffic packets. This model is trained on a set of instances and is deployed to give out predictions. The framework is responsible for updating this model with time as the data arrives.

Our first step consists of applying several pre-processing techniques to the initial data. This includes data cleaning, feature engineering and dealing with the class imbalance problem. We tackled the class imbalance problem through SMOTE (Synthetic Minority Oversampling Technique) [7] by generating new synthetic cases for the rare class by interpolating two existing cases of the minority class. This helps to balance the two existing classes which in turn helps the learner to also focus on the rare class.

To develop the heterogeneous ensemble we carry out a set of experiments with different machine learning models, compared their performance and implemented a soft voting scheme that uses the best models previously obtained. The heterogeneous ensemble model is build by selecting the two algorithms with better performance from a set of four standalone algorithms (Logistic Regression, KNN, Linear SVC, Decision Tree) and four ensemble algorithms (Random Forest, Bagging Classifier, Gradient Boosting, XGB Classifier).

To generate the most suitable heterogeneous ensemble we employed soft voting' after selecting the 2 top performing algorithms. The soft voting technique incorporates each classifier uncertainty for determining the final prediction of the heterogeneous ensemble. Figure 1 displays the procedure used for building the heterogeneous ensemble model for this problem domain.

3.2. Active Learning-based Retraining Module

The second module of HEAL corresponds to the incorporation of active learning in the learning procedure while ensuring an intelligent model update. This module ensures a fast

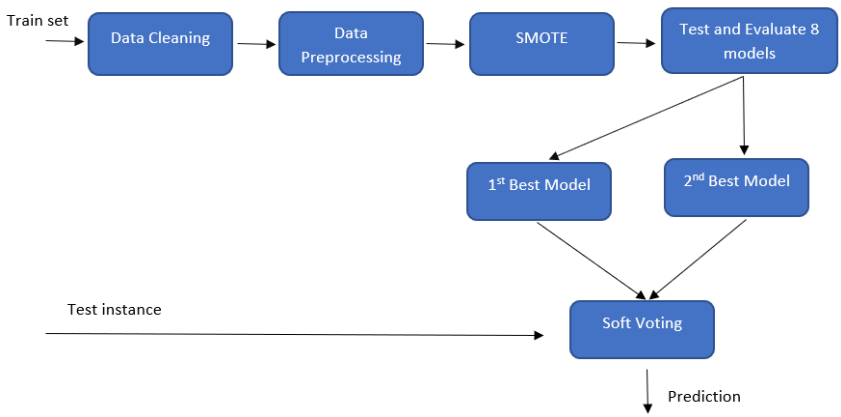

Figure 1. Heterogeneous Ensemble using soft voting technique

model training time as well as a fast prediction stage which is critical for our application problem.

Figure 2 displays the main functioning schema of this module. The confidence score of the predictions received from the previous heterogeneous module is calculated, and a number of instances with the lowest score (greater than a threshold) are selected for being manually labeled by the Oracle. The training set is updated with the newly labeled instances. This updated data set is used to retrain the model and the performance is evaluated and compared against the model currently being used. If the performance is improved, the new training set is used to update the model for the following iterations.

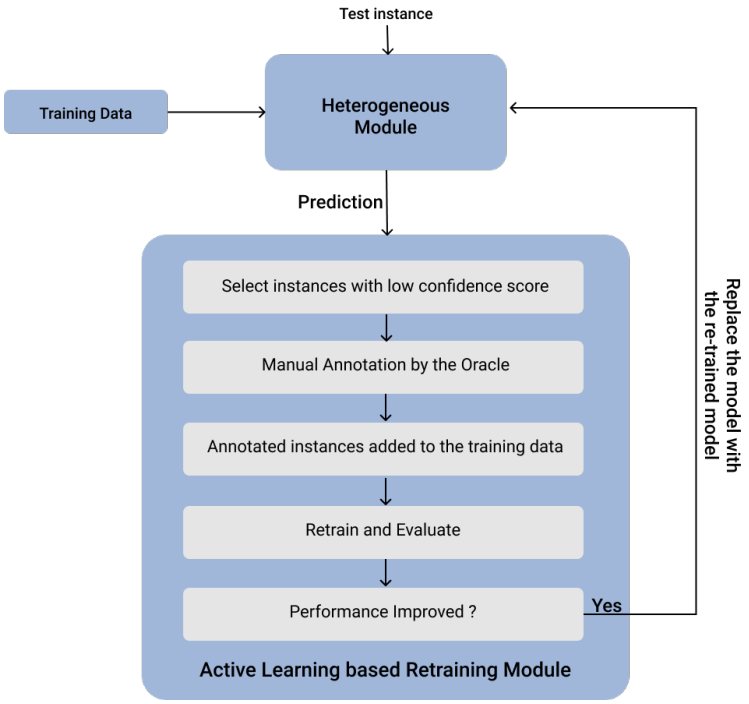

Figure 2. HEAL system modules and Active Learning-based Retraining module main steps.

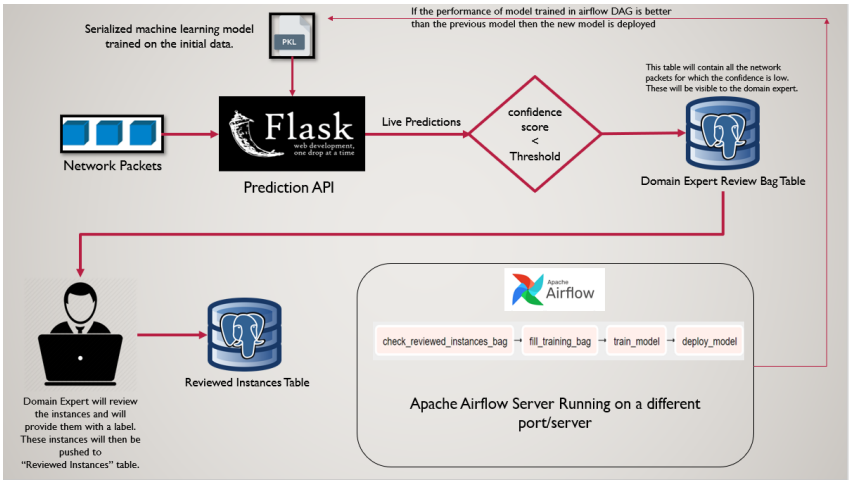

Figure 3. HEAL Architecture.

3.3. **Framework Development**

Our last step concerns the implementation of the proposed HEAL solution in a real-world scenario that considers a continuous flow of data.The following main technologies were used in our implementation: Flask, Apache Airflow and PostgreSQL.

Figure 3 shows the key implementation components of the HEAL framework. A continuous flow of network packets is made incident to the Flask servers prediction API. A Python simulation script makes a POST request to the API with the network packet data at random intervals of time. Flask internally uses the initial deployed machine learning model to obtain a prediction. After this step, the confidence score of the instance that is predicted is checked. If the confidence score obtained is less than a defined decision threshold, the instance is passed to the Oracle for reviewing and labeling. The newly labeled instances can then be used for retraining purposes. All the instances with low confidence score are appended to the Domain Expert Review Bag table. This table is accessible to the Oracle and contains the instances that need to be labelled. When labels are added to instances, they are removed from this table and are appended to the Reviewed Instances table being then used to retrain the model.

The model retraining and deployment is automated through Airflow DAG.At regular time intervals the DAG checks whether the number of instances in the Reviewed Instances table reaches a pre-defined threshold. When this happens we proceed with the retraining of the model. We evaluate the new model and compare its performance with the performance of the model already deployed. If the performance of the new model is better than that of the deployed model, the deployed model is replaced with the new one.

4. **A case study with the NSL-KDD data set**

4.1. **Data Set and System Configuration**

We selected the well known NSL-KDD[1] data set [8]. The motivation for selecting this data set is related with both the network intrusion detection goal of HEAL framework and the number of problems that this version has solved in comparison to the previous one (KDD'99). We used the version of the NSL-KDD data set built with 20% of the cases. It contains around 25000 instances and a total of 41 features. This is a binary

[1]Available at http://www.unb.ca/research/iscx/dataset/iscx-NSL-KDD dataset.html

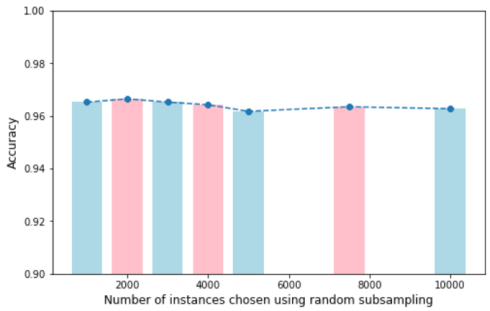

Figure 4. Performance when trained with increasing instances using random sub-sampling.

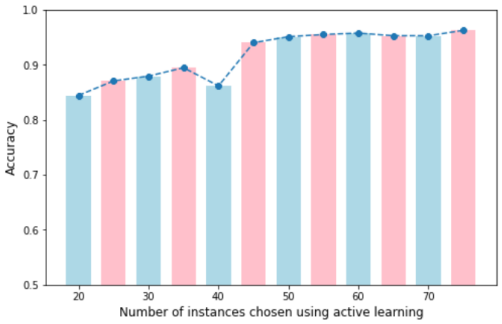

Figure 5. Performance when trained along with active learning.

classification problem with 13449 normal cases and 11743 anomaly cases. We carried out our experiments on 64-bit Windows system with a RAM of 8GB and an Intel Core i5 10th generation processor (1.60 GHz).

4.2. **Experimental Evaluation**

In these experiments, we used the default values of the learning algorithms for the first module which builds an heterogeneous ensemble through a soft voting procedure. A separate untouched test set was kept aside from the beginning to provide a more detailed analysis of the models' performance. Table 1 shows the intermediary results obtained for the 8 classifiers tested in the heterogeneous ensemble module. The two models that are selected are the Random Forest and the Gradient Boosting because they exhibited the best performance according to a user-defined metric.

Table 1. Base classifiers results for the NSL-KDD data on the heterogeneous ensemble module.

	Accuracy	Precision	Recall	F_1
KNN	0.91300	0.922131	0.91300	0.91206
Linear SVC	0.90125	0.912322	0.90125	0.899986
Decision Tree	0.91800	0.918873	0.91800	0.917809
Logistic Regression	0.93475	0.935401	0.93475	0.934629
Bagging	0.94200	0.942036	0.94200	0.941971
Random Forest	**0.94800**	0.948087	0.94800	0.947966
Gradient Boosting	**0.95450**	0.955301	0.95450	0.954412
XGBoost	0.93275	0.936549	0.93275	0.932379

In our next step we assessed the sensitivity of the performance of the obtained heterogeneous voting ensemble for different training sample sizes. We used random subsampling of our data set with increasing observations in the training set at each iteration. At each such iteration, the model is tested against an untouched test set. We started with 2,000 observations randomly sampled from the training set and increased this number up to 10,000 observations. Figure 4 shows the results of this experiment. We observe that the obtained heterogeneous ensemble not only provides a good performance but is also robust when using smaller training samples. Moreover, we also observe that our special-purpose heterogeneous ensemble exhibits an improved performance when compared against the base algorithms, showing the suitability and efficacy of our proposed heterogeneous ensemble model.

To check the impact of the integration of the active learning-based retraining module we split the initial 25000 cases into a initial train set of 20 observations, an evaluation set of

Table 2. Model Deployment History for 7 iterations of retraining.

id	Train_date	Accuracy	Is_currently_deployed
1	2021-12-07 4:17:57	0.91575	false
2	2021-12-07 4:33:49	0.937	false
3	2021-12-07 4:41:37	0.94425	false
4	2021-12-07 4:49:24	0.949	false
5	2021-12-07 4:58:18	0.9705	false
6	2021-12-07 5:01:36	0.971	false
7	2021-12-07 5:09:31	0.9725	false
8	2021-12-07 5:17:25	0.973	true

4000 observations and a test set with around 21000 observations. We set the confidence threshold to 0.7 and we selected accuracy as the evaluation metric. Figure 5 shows the performance obtained in the described experiment. We observe that the accuracy of our model increases rapidly from 0.8445 to 0.96275 after a few iterations. This shows that HEAL can take advantage of the embedded active learning procedure enabling: i) the use of less data for training; ii) keep the models' performance high; and iii) enabling an efficient and fast training and updating of the deployed model. Finally, we validated the complete framework for a continuous flow of data. Table 2 shows the obtained model deployment history. We observe that accuracy clearly increases after 7 iterations of retraining and re-deployment.

5. **Conclusion**

In this paper we propose HEAL, an innovative framework for network intrusion detection that embeds two complementary modules: a dynamic heterogeneous ensemble module and an active learning-based retraining module. These modules are incorporated in a framework that deals with a continuous flow of data. We demonstrate the robustness and efficiency of HEAL in a network intrusion case study using the NSL-KDD data set. Future research directions include the extension of the experiments to more data sets and the exploration of other methods for selecting the instances to be manually labeled.

References

[1] A. H. Mirza. "Computer network intrusion detection using various classifiers and ensemble learning". In: *2018 26th Signal Processing and Communications Applications Conference (SIU)*. IEEE. 2018, pp. 1–4.

[2] M. Zaman and C.-H. Lung. "Evaluation of machine learning techniques for network intrusion detection". In: *NOMS 2018-2018IEEE/IFIP Network Operations and Management Symposium*. IEEE. 2018, pp. 1–5.

[3] V. Timčenko and S. Gajin. "Ensemble classifiers for supervised anomaly based network intrusion detection". In: *2017 13th IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)*. IEEE. 2017, pp. 13–19.

[4] N. Martindale, M. Ismail, and D. A. Talbert. "Ensemble-Based Online Machine Learning Algorithms for Network Intrusion Detection Systems Using Streaming Data". In: *Information* 11.6 (2020), p. 315.

[5] Y.-F. Hsu, Z. He, Y. Tarutani, and M. Matsuoka. "Toward an online network intrusion detection system based on ensemble learning". In: *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*. IEEE. 2019, pp. 174–178.

[6] Y. Li and L. Guo. "An active learning based TCM-KNN algorithm for supervised network intrusion detection". In: *Computers & security* 26.7-8 (2007), pp. 459–467.

[7] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer. "SMOTE: synthetic minority over-sampling technique". In: *Journal of artificial intelligence research* 16 (2002), pp. 321–357.

[8] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani. "A detailed analysis of the KDD CUP 99 data set". In: *2009 IEEE symposium CISDA*. IEEE. 2009, pp. 1–6.