AE-NIDS : Automated Evolving SDN-based Network Intrusion Detection System

Suyeol Lee*
95leesu@kaist.ac.kr
KAIST

Seungwon Shin
claude@kaist.ac.kr
KAIST

Abstract

The network intrusion detection system(NIDS) with high accuracy and high recall has been considered as an unachievable goal owing to a trade-off between accuracy and recall. Some deep learning-based NIDS could have a high level of accuracy about their dataset, but these system's recall value about untrained real malicious packet data is very low. To address these limitations, we suggest a novel SDN-based NIDS, AE-NIDS, which enables high recall value keeping a high level of accuracy. AE-NIDS combines (semi) supervised and unsupervised learning, and trains deep neural networks during operation of the network with automatically collected malicious packet data. AE-NIDS could achieve following four unique features using supervised adversarial autoencoder(AAE) and balancing GAN; high recall about new data, efficiency, accelerated self-learning, scalability.

Challenges and Contributions

In the domain of NIDS, the classification performance of (semi) supervised[1][2] or unsupervised[3][2] deep learning-based NIDS outperforms traditional methods. However, here, we present four limitations of deep learning-based NIDS.

Trade off between high accuracy and high recall : Mirsky et al. suggested a novel network intrusion system with an ensemble of autoencoders. They could achieve a high level of accuracy with an adequate threshold, but the recall value is only around 30%[3]. To improve recall value, we introduce novel automated system; train supervised AAE, and detect malicious packet automatically during the operation of the network by calculating a distance from the cluster of the benign packet, and re-train model with BAGAN.

Lack of self-learning : While most of the existing works only consider a situation that pre-train the model before real operation of the network, Javaid suggested a concept of self-learning[2], but this model is too simple(only concept); No consideration of imbalanced data and catastrophic forgetting. To overcome these deficiencies and make accelerated self-learning, we introduce BAGAN to oversample imbalanced data and a continual learning algorithm.

Inefficient learning algorithm : Javaid et al. suggested a combination of feature learning and classification learning to improve detection ability[2], but this system separates neural network for feature and classification learning. To solve this limitation, we introduce supervised AAE, which can learn latent vector and classification simultaneously.

No consideration of distributed SDN environment : Software-Defined Networking (SDN) could have a distributed controller. However, there is no research to utilize the feature of a distributed SDN controller, although the application of federated learning can make a scalable system. Finally, we introduce federated learning for scalable learning.

System Design

AE-NIDS of each SDN controller has a feature extractor and trains three deep neural networks(supervised-AAE, Autoencoder, BAGAN) with pre-processed data from feature extractor. During all procedures, a federated learning algorithm updates the parameter of deep neural networks to utilize collected data from each distributed SDN controller. **1) pre-training procedure:** First, AE-NIDS trains the parameter of AAE through pre-processed data on each SDN controller. After training, AE-NIDS clusters and saves the latent vector of packet data. **2) Execution procedure:** During the real operation of the network, AE-NIDS classifies packet through AAE and distance from benign packet cluster, and stores feature of distance-based malicious packets automatically. **3) updating procedure:** If the number of the stored malicious packet is enough, autoencoder clusters feature data of the abnormal packet, which is gathered during execution procedure. Then, BAGAN generates new feature data, which can not be discriminated from original ones. And AAE utilizes augmented data to re-train the parameter with continual learning, and finally return to execution procedure.

Acknowledgments

This work was supported by IITP grants funded by the Korean government (MSIT) (No. 2015-0-00575, Global SDN/NFV Open-Source Software Core Module/Function Development)

References

[1] Bo Dong and Xue Wang. 2016. Comparison deep learning method to traditional methods using for network intrusion detection. In *ICCSN*. IEEE, 581–585.

[2] Ahmad Javaid, Quamar Niyaz, Weiqing Sun, and Mansoor Alam. 2016. A deep learning approach for network intrusion detection system. In *Proceedings of the 9th BIONETICS*. 21–26.

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

*The author will present the poster, student

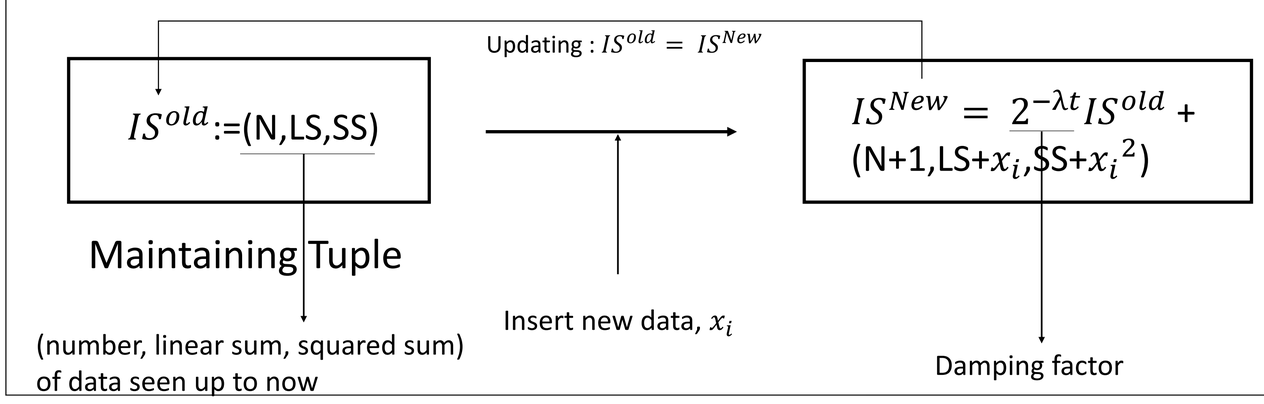
AE-NIDS : Automated Evolving SDN-based Network Intrusion Detection System

Suyeol Lee , Seungwon Shin, Korea Advanced Institute of Science and Technology, KAIST

1. Feature Extractor (From Kitsune, NDSS 2018)

Feature Extractor : extract the current behavior of a data stream

1. Damped Incremental Statistics : O(1) for updating



2. Extracted Features

- Bandwidth of the outbound traffic
- Bandwidth of the outbound and inbound traffic together
- Packet rate of the outbound traffic
- Inter-packet delays of the outbound traffic

2. Challenges and Contributions

Challenges

- Trade off between high accuracy and high recall
- Lack of self-learning
- Inefficient learning algorithm
- No consideration of distributed SDN environment

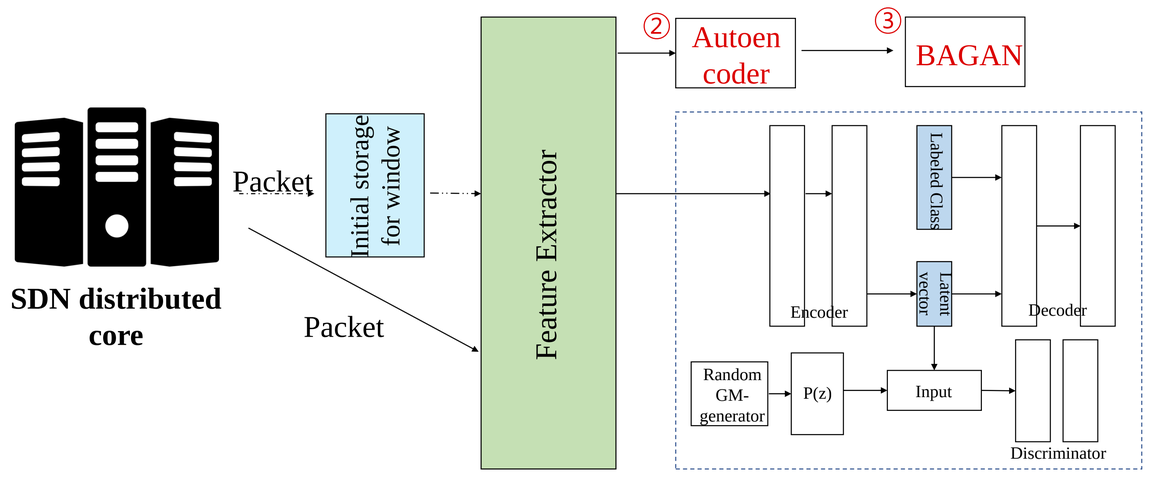
Contributions

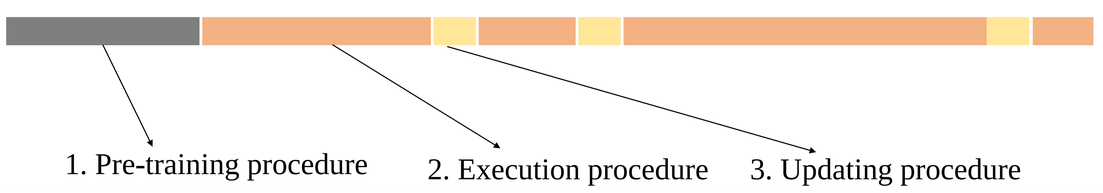
- New system with AAE and distance-based automated learning
- Introduce GAN for augmenting imbalanced data and continual learning(EWC)
- Introduce one tool for classification and latent vector learning; AAE
- Federated learning for scalabitiy

2. AE-NIDS design

AE-NIDS on each distribued SDN core

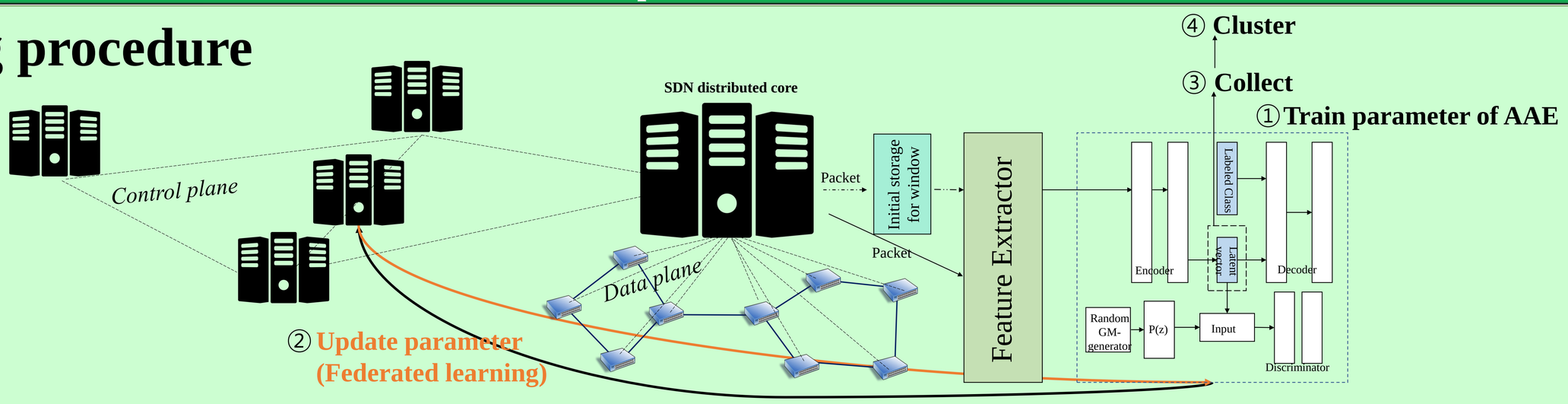
= Three deep neural network [1. Supervised – AAE 2. Autoencoder 3.BAGAN] + Feature Extractor





3. AE-NIDS procedures

1. Pre-training procedure



2. Execution procedure : During real operation of network

3. Updating procedure