Literature Review: An Unsupervised Network Anomaly

Detection Model and Implementation

Research focus: network anomaly detection using deep-learning

Problems solved: lack of large amount of labeled data in model training

Difficult to detect unknown networks attacks or variant attacks

This paper collects and organizes a large amount of data and designs four types of network attack data, including new attack means, worms, system vulnerabilities and botnet

So they don’t just use traditional method like unsupervised k-means classification but added deep learning k-means classification.

The problem with supervised learning is that it requires lot of labelled data-so, they solved that by using unsupervised learning which is more suitable for unknown network environments.

Unsupervised learning does not rely on labels, saves the cost of adding labels, and is more suitable for unknown network environments

**📌 Workflow of the Paper (Unsupervised Anomaly Detection with Autoencoder + Clustering)**

1. **Problem Identification**
   * Recognized that supervised models require lots of labeled data (hard to obtain).
   * Decided to design an **unsupervised anomaly detection approach** that doesn’t need labels.
2. **Data Preprocessing**
   * Extracted **basic network traffic features**.
   * Applied **time correlation analysis** to capture temporal dependencies in traffic.
   * Used **hierarchical clustering** to reduce data size and dimensionality, making training/detection faster and less memory-intensive.
3. **Feature Learning**
   * Implemented an **Autoencoder** to learn representations of **normal traffic**.
   * Autoencoder trained to reconstruct normal patterns → deviations in reconstruction indicate anomalies.
   * Leveraged concepts from:
     + Sparse Autoencoders (Ng)
     + Denoising Autoencoders (Vincent)
     + Variational Autoencoders (Kingma)
     + Contractive Autoencoders (Rifai)
4. **Anomaly Detection**
   * Once the Autoencoder learns **normal traffic patterns**, new input traffic is compared.
   * If reconstruction error is **high**, it’s flagged as anomalous.
5. **Efficiency Optimization**
   * Combined **hierarchical clustering + time correlation features** to reduce computational cost (time + space complexity).
   * This makes the method more efficient and scalable.
6. **Validation / Testing**
   * Tested the model on benchmark datasets (e.g., **KDD Cup 1999** dataset in prior related work).
   * Demonstrated that the unsupervised model can detect abnormal traffic effectively without requiring labeled data.

**Why they used Autoencoder for anomaly detection?**

**Limitations of Common Unsupervised Algorithms**

* **Hierarchical Clustering:**
  + Very high computational complexity.
  + Tends to produce chain-like clusters, reducing accuracy.
* **K-Means:**
  + Sensitive to the choice of initial center points.
  + Often falls into local optima.
  + Can only effectively detect globular clusters, missing irregular patterns.
  + Not suitable for detecting unknown attacks.
* **DBSCAN (Density-Based Clustering):**
  + Works well for density-based patterns, but as data grows, computation and time complexity increase sharply.
  + Not practical for large-scale network traffic**.**

**✅ Why Autoencoder is Better for Anomaly Detection**

* Learns compressed representations of normal traffic automatically (no manual feature engineering needed).
* Can reconstruct input data; anomalies are detected when reconstruction error is high.
* Scales better to large datasets compared to clustering methods.
* More effective for capturing non-linear and complex traffic patterns.
* Suitable for detecting unknown or novel attacks, since it models normal behavior rather than predefined clusters.

Note: The main thing is to make the model computationally less expensive and make detect wide range of anomalies.

Main thing about this paper is that it only helps detects four types of anomalies: new attack means, worms, system vulnerabilities and botnets

So one thing we can do is make a dataset with more anomalies and detect it.

Question to ask: Are there better model than autoencoder for anomaly detection?

**Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection**

There are many network intrusion detection systems (NIDSs) but is the main problem in that?

 The study investigates the **impact of oversampling techniques** on training sample size and identifies the **minimal suitable training sample size**.

 It compares **two feature selection techniques**:

* Information Gain
* Correlation-based selection

 It examines their effect on **detection performance** and **time complexity**.

 The work also explores **hyperparameter optimization techniques** to improve NIDS (Network Intrusion Detection System) performance.

 Evaluation is done on **two recent intrusion detection datasets**:

* CICIDS 2017
* UNSW-NB 2015

 **Key results**:

* Training sample size reduced by **up to 74%**.
* Feature set size reduced by **up to 50%**.
* Hyperparameter optimization further improves performance.
* Achieves **over 99% detection accuracy** on both datasets.
* Outperforms recent literature with **1–2% higher accuracy** and **1–2% lower false alarm rate**.

One of the reason we are doing the research:

Several protection mechanisms have been proposed such as firewalls, user authentication, and the deployment of antivirus and malware programs as a first line of defense . However, these mechanisms have not been able to completely protect the organizations’ networks, particularly with contemporary attacks

There are two types of intrusion detection system:

One is signature based and another is anomaly based:

Signature-based detection systems base their detection on the observation of pre-defined attack patterns. Thus, they have proven to be effective for attacks with wellknown signatures and patterns

While anomaly-based detection systems base their detection on the observation of any behavior or pattern that deviates from what is considered to be normal or signature based.

Therefore, these systems can detect unknown attacks or intrusions based on the built models that characterize normal behavior

This is particularly evident given the high volume of generated network traffic data, continuously evolving environments, vast amounts of features collected that form the training datasets (high dimensional datasets), and the need for real-time intrusion detection

having redundant or irrelevant features can have a negative impact on the detection capabilities of NIDSs as it slows down the model training process. Therefore, it is important to choose the most suitable subset of features and optimize the parameters of the machine learning (ML)-based detection models to enhance their performance

What they do in making the model is:

Determines the minimum suitable training size

Two different feature selection techniques, namely information gain and correlation-based feature selection

Does hyperparameter tuning

Test Dataset: CICIDS 2017 dataset [12] (which is the updated version of the ISCX 2012 dataset [13] used in our previous work [11]) and the UNSW-NB 2015 dataset [14]

Evaluation techniques: accuracy (acc), precision, recall, and false alarm rate (FAR)

What we can also do is: Optimize the already premade models of intrusion detection

The main contributions and differences between this work and our previous work in [11] can be summarized as follows:

• Propose a novel multi-stage optimized ML-based NIDS framework that reduces computational complexity and enhances detection accuracy.

• Study the impact of oversampling techniques and determine the minimum suitable training sample size for effective intrusion detection.

• Explore the impact of different feature selection techniques on the NIDS detection performance and time (training and testing) complexity.

• Propose and investigate different ML hyper-parameter optimization techniques and their corresponding enhancement of the NIDS detection performance.

• Evaluate the performance of the optimized ML-based NIDS framework using two recent state-of-the-art datasets, namely the CICIDS 2017 dataset [12] and the UNSW-NB 2015 dataset [14].

• Compare the performance of the proposed framework with recent works from the literature and illustrate the improvement of detection accuracy, reduction of FAR, and a reduction of both the training sample size and feature set size.

Different types of normalization, different types of feature selection techniques and different hyperparameter tuning values, and different filling missing values filling technique.

Doing different types of cross validation:

* [**K-Fold Cross-Validation:**](https://www.google.com/search?sca_esv=9477f2e8cc3fac1a&rlz=1C1GCEA_enUS1176US1176&sxsrf=AE3TifNAxFv2x1DEIIZOdMgtc0Wyt5APXA%3A1758495223887&q=K-Fold+Cross-Validation&sa=X&ved=2ahUKEwi_0KKZ-eqPAxUdl-4BHWDaDYIQxccNegUI1gEQAw&mstk=AUtExfBgxo-6zw3x407mY0MJHFLOUJTqQB_VFjmoPrco4R4KaOYK9MUkt_yxzJbiVSPcNatjFROHiKCRuMmnSzzrArNxB7QvqlkYt1dhAquhJJwis7Chi17U_3Lbrc8XluRRKXTl1PkNWS7bFDop7jL1ILN6Aq3sECMK34_-kWaZLRk7057ZQjk5I59tQOdeV71QoxN5D_ltiO_jwN16UHEivZofHFa_chnx0empBJF7HYP7yFfYbBxrxoq_M6O046fR6LJYtY3UuiZHpkPDd9menC7J&csui=3) The data is split into K folds; the model is trained K times, using each fold as a validation set once.
* [**Leave-One-Out Cross-Validation**](https://www.google.com/search?sca_esv=9477f2e8cc3fac1a&rlz=1C1GCEA_enUS1176US1176&sxsrf=AE3TifNAxFv2x1DEIIZOdMgtc0Wyt5APXA%3A1758495223887&q=Leave-One-Out+Cross-Validation&sa=X&ved=2ahUKEwi_0KKZ-eqPAxUdl-4BHWDaDYIQxccNegUIzAEQAQ&mstk=AUtExfBgxo-6zw3x407mY0MJHFLOUJTqQB_VFjmoPrco4R4KaOYK9MUkt_yxzJbiVSPcNatjFROHiKCRuMmnSzzrArNxB7QvqlkYt1dhAquhJJwis7Chi17U_3Lbrc8XluRRKXTl1PkNWS7bFDop7jL1ILN6Aq3sECMK34_-kWaZLRk7057ZQjk5I59tQOdeV71QoxN5D_ltiO_jwN16UHEivZofHFa_chnx0empBJF7HYP7yFfYbBxrxoq_M6O046fR6LJYtY3UuiZHpkPDd9menC7J&csui=3) **(LOOCV):** A special case of K-Fold where K equals the total number of data points. Each data point is used as a test set once.