# Intrusion Detection of Imbalanced Network Traffic Based on Machine Learning algorithms

D. Sai Sumanth[1], T. Sai Sushanth[2], P. Sai Vamshi[3], G. Sai Vardhan[4], S. Sai Varshith[5], S. Sai Varun[6], R A Manikandan[7]

[1,2,3,4,5,6]*School of Engineering, MallaReddy University*

[7]*Guide, Professor, Department of AIML, School of Engineering, MallaReddy University*

**Abstract: This project focuses on advanced machine learning for detecting intrusions in imbalanced network traffic. Despite challenges posed by data imbalance, the study employs cutting-edge algorithms and diverse preprocessing to glean insights from normal and malicious patterns. Findings highlight machine learning's potential in managing imbalanced network traffic, emphasizing tailored preprocessing and algorithm selection. Ultimately, the project advances intrusion detection by showcasing machine learning's role in enhancing security through swift threat identification and mitigation. In conclusion, this research underscores the pivotal role of machine learning in addressing imbalanced network scenarios, paving the way for a safer digital landscape.**

## I.INTRODUCTION

The rapid evolution and widespread application of 5G, IoT, Cloud Computing, and other technologies have led to increasingly complex and extensive network scales, as well as real-time traffic. Consequently, cyber-attacks have also grown in complexity and diversity, posing significant challenges to cybersecurity. As a second line of defense following the firewall, the Network Intrusion Detection System (NIDS) is crucial for accurately identifying malicious network attacks, providing real-time monitoring, implementing dynamic protective measures, and formulating effective strategies.

In 1980, James Anderson initially proposed the concept of intrusion detection, and subsequently, some scholars attempted to integrate machine learning methods into intrusion detection [1]. However, due to limitations in computer storage and processing power at that time, machine learning failed to gain attention. With the rapid advancement of computers and the rise of Artificial Intelligence (AI) and other technologies, numerous scholars have employed machine learning methods in network security and achieved noteworthy results.

In the actual cyberspace scenario, normal activities constitute the majority of traffic data, while malicious cyber-attacks represent only a small portion, resulting in a high imbalance between categories. This imbalance poses substantial pressure on intrusion detection. Cyber-attacks can easily conceal themselves within a large volume of normal traffic, making it challenging for machine learning algorithms to adequately learn the few instances of attacks, often leading to misclassification.

Since Lecun et al. introduced the theory of Deep Learning as a pivotal subset of machine learning, deep learning has exhibited exceptional performance in domains like Computer Vision (CV) and Natural Language Processing (NLP). Intrusion detection techniques based on deep learning have garnered significant attention in both academia and industry. Deep learning methodologies aim to extract potential features from high-dimensional data through model training, transforming network traffic anomaly detection problems into classification tasks. Training on a large volume of data samples enables adaptive learning of normal and abnormal behavior differences, substantially enhancing real-time intrusion processing performance.

Nonetheless, the issue of imbalanced classification persists in the context of multi-classification of network traffic. Faced with imbalanced network traffic data, we propose a novel Difficult Set Sampling Technique (DSSTE) algorithm to address the class imbalance problem in network traffic. This method effectively alleviates the imbalance issue by reducing majority samples and augmenting minority samples in the difficult set, enabling the classifier to better learn

the differences during training.

Our study utilizes classic machine learning and deep learning algorithms to validate on two benchmark datasets: the NSL-KDD and the current CSECIC-IDS2018. We conduct detailed analysis and data cleansing on these datasets. Our contributions include:

(1) Thorough analysis and data cleansing of benchmark datasets NSL-KDD and CSECIC-IDS2018.

(2) Introduction of the DSSTE algorithm, targeting class imbalance in intrusion detection by handling difficult samples through reduction of majority samples and augmentation of minority samples.

(3) Implementation and comparison of classification models such as Random Forest (RF), Support Vector Machine (SVM), XGBoost, Long Short-Term Memory (LSTM), AlexNet, and Mini-VGGNet. Our experiments comprise 30 methods aimed at tackling the challenge of imbalanced network traffic data in intrusion detection.

## II. LITERATURE REVIEW

Intrusion detection involves recognizing and responding to malicious activities that threaten the security and functionality of computer networks. Network traffic data, comprising packets of information exchanged among various devices, serves as a vital source for Intrusion Detection Systems (IDS). However, an issue arises with class imbalance in network traffic data, where the volume of normal packets far surpasses that of malicious packets. This creates a hurdle for machine learning and deep learning techniques, which depend on balanced and representative data for effective model learning.

Several methodologies have emerged to tackle imbalanced network traffic data for intrusion detection. One common strategy involves employing sampling techniques that alter the original data distribution by reducing the majority class (normal packets), increasing the minority class (malicious packets), or both. For instance, a novel approach called the Difficult Set Sampling Technique (DSSTE) algorithm was proposed. This method divides the imbalanced training set into a 'difficult' set containing samples near the decision boundary and an 'easy' set consisting of samples distant from it. The DSSTE algorithm utilizes the Edited Nearest Neighbor (ENN) algorithm to identify the difficult set and employs the KMeans algorithm to compress majority samples in this set. Additionally, it manipulates continuous attributes of minority samples in the difficult set to generate new samples. By combining the easy set, compressed majority samples, minority samples, and their augmented versions, a new balanced training set is formed. This DSSTE algorithm demonstrated superior performance compared to other sampling methods on classic intrusion datasets, namely NSL-KDD and CSE-CIC-IDS2018, utilizing diverse classification models such as random forest, support vector machine, XGBoost, long and short-term memory, AlexNet, and Mini-VGGNet.

Another approach involves utilizing generative models that learn the inherent data distribution and produce synthetic samples resembling real data. Generative adversarial networks (GANs) are a prominent type of generative model comprising a generator and a discriminator competing against each other. By playing this adversarial game, GANs generate realistic samples augmenting the original data. For instance, an enhanced GAN called conditional tabular GAN (CTGAN) was proposed to generate synthetic samples for the minority class in imbalanced network traffic data. CTGAN, a variant capable of handling mixed attribute types, uses a conditional vector to control the generation process, thereby enhancing diversity and quality of the generated samples. Authors applied CTGAN to the CIDDS-001 dataset, a comprehensive intrusion dataset, and compared it with oversampling methods like SMOTE, ADASYN, and GAN. Results indicated that CTGAN improved the performance of various anomaly detection models such as isolation forest, one-class support vector machine, and autoencoder.

A third strategy involves employing attention mechanisms in deep learning models to focus on the most pertinent features or segments of data for the specific task. These mechanisms, widely used in natural language processing and computer vision, enhance data representation and interpretation. For instance, a proposed deep learning model for network intrusion detection (DLNID) combines attention mechanisms with bidirectional long short-term memory (Bi-LSTM) networks. The DLNID model extracts sequence features using a convolutional neural network (CNN), reallocates weights via the attention

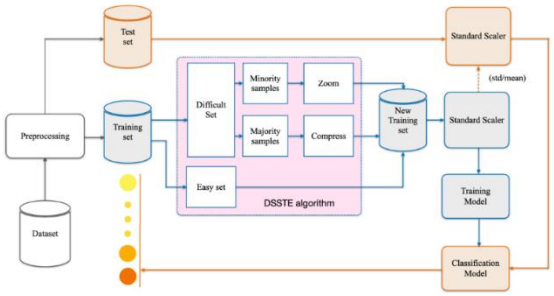
mechanism, and learns network sequence features using Bi-LSTM. This mechanism helps the model prioritize crucial features for intrusion detection, while Bi-LSTM captures temporal dependencies and context information. Authors evaluated the DLNID model on the NSL-KDD dataset, comparing it with other deep learning models like LSTM, CNN, and CNN-LSTM. Results indicated that the DLNID model achieved the highest accuracy and recall among the models evaluated.

## III. PROBLEM STATEMENT

Developing an Intrusion Detection System for Imbalanced Network Traffic through the Integration of Machine Learning and Deep Learning Techniques to Enhance Security and Mitigate Threats in Network
Methodology

## IV. METHODOLOGY

Workflow of Machine Learning ModelBuilding
There are some way involved in the methodology and every machine literacy model should follow this methodology.




Data Acquisition
The dataset is collected from Kaggle website.
Data Pre-Processing
Upon extracting the dataset, it's common to encounter various issues like noisy data, duplicate entries, missing values, or outliers due to extraction or input errors. Therefore, our primary focus lies in preprocessing the data to ensure its quality and reliability. The key steps involved are:
1. Handling Duplicate Values: We meticulously remove duplicate entries within the dataset, retaining only one instance of each valid data point.
2. Addressing Outliers: Instances with missing values (NaN) or infinite values (Inf) are infrequent within the dataset. Consequently, we opt to eliminate these samples.
3. Feature Manipulation and Deletion: In the CSE-CIC-IDS2018 dataset, specific features like 'Timestamp,' 'Destination Address,' 'Source Address,' and 'Source Port' are removed. Additionally, for 'Init Bwd Win Byts' and 'Init Fwd Win Byts' features with a value of -1, we introduce two check dimensions. The presence of -1 is marked as 1, otherwise as 0. In NSL-KDD, we employ the OneHot encoder to transform data. For instance, protocol types like 'TCP,' 'UDP,' and 'ICMP' are converted into binary vectors (1, 0, 0), (0, 1, 0), and (0, 0, 1), respectively. This process significantly expands the initial 41 dimensions feature vector to 122 dimensions, considering the different categories within protocol type, flag function (11 categories), and service function (70 categories)
4. Standardizing Numerical Data: To remove variations between indicators and expedite model convergence, we standardize the data using the Z-score method. This transformation ensures that the mean of each feature becomes 0, and the standard deviation becomes 1, thereby transforming the data into a standard normal distribution. The formula for standardization involves subtracting the mean (u) of each feature from the corresponding element (x0i) and dividing it by the standard deviation (s) of that feature.
Standardization Formula:

$$u = \sum_{i=1}^{N} x_i$$

$$s = \sum_{i=1}^{N} (x_i - u)^2$$

$$x_i' = \frac{x_i - u}{s}$$

Experimental Parameters
The methodology utilizes both Sklearn, a machine learning framework, and Tensorflow, a deep learning framework, to conduct experiments on the Google Colaboratory platform. While machine learning algorithms rely on CPU computations, deep learning algorithms harness TPU acceleration. Table 4 delineates the specific parameters utilized in these experiments. To mitigate overfitting, the data undergoes standardization. Within the machine learning domain, the integrated learning model strategically incorporates shallow trees as a countermeasure against potential overfitting.
To leverage TPU acceleration, we opted for larger batch sizes and adjusted the number of epochs accordingly. To mitigate overfitting, our strategy

involved vigilant monitoring of accuracy and loss trends throughout the training phase, employing suitable learning rates, and incorporating Dropout within the neural network layers.

In our utilization of machine learning algorithms, we conducted experiments using Sklearn's Random Forest Classifier, svm. LinearSVC, and XGB Classifier. Table 5 details the specific parameters employed in these experiments.

Table 3 :Distribution of benchmark datasets.

| Dataset | Type | Total | Imbalance ratio | Training set | Testing set |
|---|---|---|---|---|---|
| NSL-KDD | Normal | 77054 | - | 67343 | 9711 |
| | DoS | 53385 | 1.44 | 45927 | 7458 |
| | R2L | 3882 | 19.85 | 995 | 2887 |
| | Probe | 14077 | 5.47 | 11656 | 2421 |
| | U2R | 119 | 647.51 | 52 | 67 |
| CSE-CIC-IDS2018 | Benign | 40000 | - | 32000 | 8000 |
| | Bot | 20000 | 2.00 | 16000 | 4000 |
| | DDOS attack-LOIC-UDP | 1730 | 23.12 | 1384 | 346 |
| | DDOS attack-HOIC | 20000 | 2.00 | 16000 | 4000 |
| | DDoS attacks-LOIC-HTTP | 20000 | 2.00 | 16000 | 4000 |
| | DoS attacks-GoldenEye | 20000 | 2.00 | 16000 | 4000 |
| | DoS attacks-Hulk | 20000 | 2.00 | 16000 | 4000 |
| | DoS attacks-Slowloris | 9908 | 4.04 | 7926 | 1982 |
| | SSH-Bruteforce | 20000 | 2.00 | 16000 | 4000 |
| | FTP-BruteForce | 46 | 869.57 | 37 | 9 |
| | Infilteration | 20000 | 2.00 | 16000 | 4000 |
| | Brute Force -Web | 550 | 72.73 | 440 | 110 |
| | Brute Force -XSS | 227 | 176.21 | 181 | 46 |
| | SQL Injection | 82 | 487.80 | 66 | 16 |

Table 4: Development environment.

| Model | NSL-KDD | | | | CSE-CIC-IDS2018 | | | |
|---|---|---|---|---|---|---|---|---|
| | Acc | Pre | Recall | F1 | Acc | Pre | Recall | F1 |
| RF | 0.7434 | **0.8137** | 0.7434 | 0.7015 | **0.9489** | **0.9481** | **0.9489** | **0.9481** |
| SVM | 0.7366 | 0.7384 | 0.7366 | 0.6966 | 0.9225 | 0.9261 | 0.9225 | 0.9126 |
| XGBoost | 0.7715 | 0.8107 | 0.7715 | 0.7365 | 0.9398 | 0.9449 | 0.9398 | 0.9340 |
| LSTM | **0.7824** | 0.7838 | 0.7823 | **0.7503** | 0.9375 | 0.9444 | 0.9370 | 0.9313 |
| AlexNet | 0.7618 | 0.8050 | 0.7611 | 0.7194 | 0.9376 | 0.9440 | 0.9369 | 0.9313 |
| miniVGGNet | 0.7605 | 0.8066 | 0.7594 | 0.7303 | 0.9388 | 0.9450 | 0.9384 | 0.9326 |
| RUS + RF | 0.7655 | 0.8220 | 0.7655 | 0.7304 | **0.9419** | **0.9454** | **0.9419** | **0.9428** |
| RUS + SVM | 0.7362 | 0.7510 | 0.7362 | 0.7058 | 0.8902 | 0.9087 | 0.8902 | 0.8926 |
| RUS + XGBoost | **0.7879** | 0.8177 | **0.7879** | 0.7468 | 0.9212 | 0.9362 | 0.9212 | 0.9234 |
| RUS + LSTM | 0.7705 | 0.7970 | 0.7704 | 0.7462 | 0.9150 | 0.9294 | 0.9139 | 0.9171 |
| RUS + AlexNet | 0.7834 | **0.8250** | 0.7814 | 0.7537 | 0.9294 | 0.9308 | 0.9268 | 0.9280 |
| RUS + miniVGGNet | 0.7827 | 0.8134 | 0.7826 | **0.7557** | 0.9337 | 0.9330 | 0.9329 | 0.9319 |
| ROS + RF | 0.7515 | 0.8125 | 0.7515 | 0.7066 | **0.9492** | **0.9484** | **0.9492** | **0.9483** |
| ROS + SVM | 0.7493 | 0.8005 | 0.7493 | 0.7300 | 0.9165 | 0.9311 | 0.9165 | 0.9073 |
| ROS + XGBoost | 0.7809 | 0.8196 | 0.7809 | 0.7532 | 0.9385 | 0.9448 | 0.9385 | 0.9320 |
| ROS + LSTM | **0.7872** | **0.8289** | **0.7866** | **0.7582** | 0.9348 | 0.9409 | 0.9346 | 0.9293 |
| ROS + AlexNet | 0.7850 | 0.8057 | 0.7849 | 0.7451 | 0.9299 | 0.9387 | 0.9295 | 0.9262 |
| ROS + miniVGGNet | 0.7626 | 0.7893 | 0.7626 | 0.7332 | 0.9362 | 0.9406 | 0.9359 | 0.9316 |
| SMOTE + RF | 0.7409 | 0.8070 | 0.7409 | 0.6977 | **0.9488** | **0.9481** | **0.9488** | **0.9480** |
| SMOTE + SVM | 0.7467 | 0.7987 | 0.7467 | 0.7275 | 0.9155 | 0.9302 | 0.9155 | 0.9062 |
| SMOTE + XGBoost | 0.7744 | 0.8142 | 0.7744 | 0.7421 | 0.9381 | 0.9449 | 0.9381 | 0.9318 |
| SMOTE + LSTM | 0.7509 | 0.7976 | 0.7508 | 0.7239 | 0.9345 | 0.9431 | 0.9344 | 0.9278 |
| SMOTE + AlexNet | **0.7875** | **0.8256** | **0.7851** | **0.7727** | 0.9324 | 0.9423 | 0.9308 | 0.9287 |
| SMOTE + miniVGGNet | 0.7651 | 0.7698 | 0.7646 | 0.7435 | 0.9366 | 0.9423 | 0.9366 | 0.9317 |
| DSSTE + RF | 0.8050 | **0.8468** | 0.8050 | 0.7863 | 0.9692 | 0.9739 | 0.9692 | 0.9698 |
| DSSTE + SVM | 0.7759 | 0.8076 | 0.7759 | 0.7658 | 0.9488 | 0.9497 | 0.9488 | 0.9463 |
| DSSTE + XGBoost | 0.8013 | 0.8349 | 0.8013 | 0.7761 | 0.9602 | 0.9641 | 0.9602 | 0.9611 |
| DSSTE + LSTM | 0.8178 | 0.8271 | 0.8177 | 0.8098 | 0.9638 | 0.9711 | 0.9636 | 0.9650 |
| DSSTE + AlexNet | **0.8284** | 0.8394 | **0.8278** | **0.8166** | 0.9653 | 0.9709 | 0.9625 | 0.9649 |
| DSSTE + miniVGGNet | 0.8127 | 0.8268 | 0.8132 | 0.8057 | **0.9699** | **0.9746** | **0.9697** | **0.9704** |

Table 5:. Machine learning model related parameters.

| Project | Properties |
|---|---|
| OS | Ubuntu 18.04.3 LTS |
| CPU | Intel(R) Xeon(R) CPU @2.30GHz |
| TPU | - |
| Memory | 12.7 GiB |
| Disk | 64.0 GiB |
| Framework | Sklearn 0.22.2.post1 + TensorFlow 2.2.0 |

| Classfier | Parameters |
|---|---|
| RandomForestClassifier | n_estimators=200, criterion='gini', min_samples_plit=2, min_samples_leaf=1 |
| svm.LinearSVC | penalty='l2', loss='squared_hinge', dual=True, tol=0.0001, C=1.0, multi_class='ovr', fit_intercept=True, intercept_scaling=1, max_iter=1000 |
| XGBClassifier | objective='multi:softmax', booster='gbtree', verbosity=0, silent=0, learning_rate=0.1 |

## Evelution Metrics

We employ various metrics—Accuracy, Precision, Recall, and F1-Score—to assess the performance of the experimental model. These metrics provide insights into the accuracy of flow recognition in the intrusion detection system and its false alarm rates. The evaluation is based on categorizing the model's prediction results and the actual labels into four types: False Negative (FN) - a positive sample mistakenly categorized as negative; False Positive (FP) - negative samples wrongly identified as positive; True Negative (TN) - correctly identified negative samples; and True Positive (TP) - accurately identified positive samples. These metrics are computed using the following equations:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

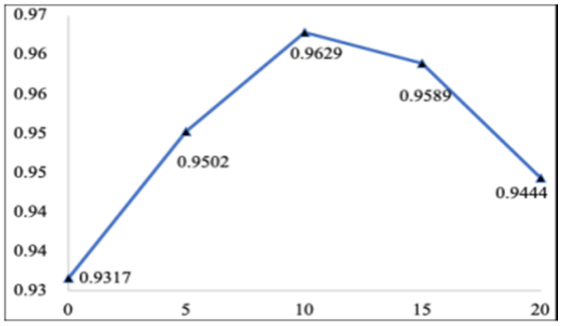
$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP + TN}{TP + TN + FP + FN}$$

$$F1\_Score = \frac{2 \times Precision \times Recall}{Precision + Recall}$$

These metrics collectively provide a comprehensive evaluation of the model's performance in identifying

positive and negative samples, offering a balanced assessment of its effectiveness in intrusion detection.

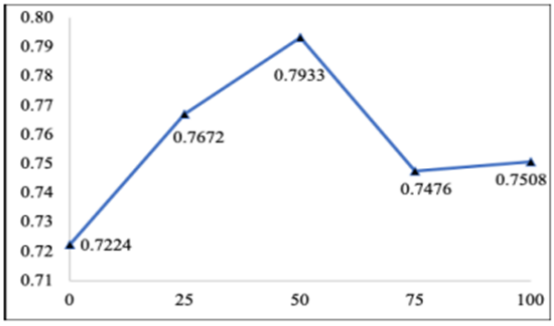



(b) CSE-CIC-IDS2018

## V. EXPERIMENTAL RESULTS

In our experimentation, we initially delved into assessing the classifier's performance on the training set while employing various deflation factors. Within the proposed DSSTE algorithm, there exists a parameter denoted as the scaling factor 'K'. As 'K' undergoes increments within a specific range, the count of difficult samples tends to rise. However, once 'K' surpasses this range, the count of difficult samples stabilizes. Nevertheless, both majority compression and minority augmentation within the difficult samples increase with variations in 'K'. Consequently, to ensure the effectiveness of data sampling without introducing excessive noise and to attain optimal sampling outcomes through the DSSTE algorithm, we conducted experiments involving different scaling factors.

We applied these experiments to process the training sets within NSL-KDD and CSE-CIC-IDS2018 datasets while varying the scaling factor 'K'. Subsequently, these experiments were executed across six proposed classifiers. To evaluate the performance, we utilized the average F1-Score for each classifier as the metric of assessment.




(a) NSL-KDD KDDTest+

In our experimental analysis, we explored the performance of classifiers on the training sets using various deflation factors, particularly focusing on the scaling factor 'K' within the DSSTE algorithm. As 'K' increases within a certain range, the count of difficult samples rises, but beyond this range, the count stabilizes. To optimize data sampling without excessive noise, we conducted experiments with different 'K' values on the NSL-KDD and CSE-CIC-IDS2018 datasets.

In NSL-KDD, 'K = 50' demonstrated outstanding classifier performance by compressing difficult samples in Normal, DoS, and Probe categories while augmenting difficult samples in R2L and U2R. Conversely, in CSE-CIC-IDS2018, 'K = 10' yielded exceptional performance, similar to the NSL-KDD treatment for difficult samples. This resulted in a refined new training set.Our evaluation comparison between DSSTE and other sampling methods across NSL-KDD and CSE-CIC-IDS2018 indicated the superior performance of DSSTE.

For NSL-KDD, different classifiers showed varying performance improvements after employing various sampling methods. Notably, AleNet achieved the highest accuracy of 82.84% and the highest recall of 81.66% with DSSTE sampling, outperforming other methods. In CSE-CIC-IDS2018, although random forest initially scored the highest accuracy and F1-Score in the unprocessed training set, DSSTE sampling led to significant enhancements. MiniVGGNet achieved the highest accuracy of 96.99% and the highest recall of 97.04% after DSSTE sampling.

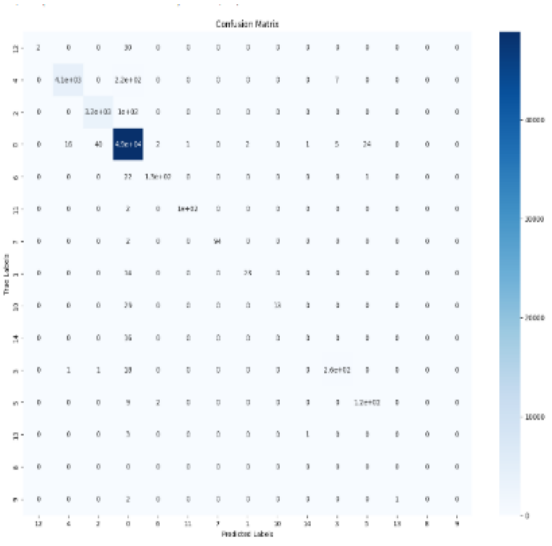


Figure 4 depicted average accuracy and F1-Score for each sampling method. Notably, DSSTE exhibited substantial improvements, enhancing the average accuracy by 4.75% and the average F1-Score by 7.1% in NSL-KDD. Conversely, in CSE-CIC-IDS2018, after DSSTE sampling, the average accuracy improved by 2.54%, and the average F1-Score improved by 3.13%.In summary, traditional methods address data imbalance but might not faithfully replicate the real data distribution.


RUS may result in information loss, ROS in redundancy and overfitting, and SMOTE in noise and data overlap. In contrast, DSSTE, by compressing and augmenting difficult data, significantly enhances the classifier's understanding of data distribution and improves classification performance.

## VI. FUTURE WORK

- Semi-Supervised and Unsupervised Learning: Investigate semi-supervised and unsupervised learning approaches that can handle imbalanced data, including anomaly detection and clustering methods. These techniques can leverage the unlabelled data effectively to enhance detection accuracy..
- Real-Time and Scalable Solutions: Focus on developing real-time intrusion detection systems that can handle high-speed network traffic efficiently and effectively scale to large-scale network infrastructures.
- Robust Preprocessing Techniques: Further refine and develop preprocessing methods specifically designed for imbalanced network traffic data. This includes handling missing values, noise reduction, and applying resampling techniques effectively.

## VII.CONCLUSION

As the landscape of network intrusion continually evolves, the challenges confronting network intrusion detection systems intensify. The issue of imbalanced network traffic compounds these difficulties, impeding intrusion detection systems' ability to anticipate the distribution of malicious attacks and posing a substantial threat to cyberspace security. This research introduces a novel Difficult Set Sampling Technique (DSSTE) algorithm designed to enhance the learning process for imbalanced network data within classification models. By strategically increasing the number of minority samples earmarked for learning, this approach aims to mitigate network traffic imbalances and bolster the learning capacity for minority samples, consequently enhancing classification accuracy. Leveraging six classical classification methods in both machine learning and deep learning, this technique is combined with various sampling approaches. Experimental results demonstrate the algorithm's efficacy in accurately identifying and expanding crucial samples within imbalanced network traffic, thereby significantly improving attack recognition.In our experiments, we observed superior performance of deep learning compared to machine learning when utilizing the DSSTE algorithm to sample imbalanced training sets. However, deep learning's effectiveness is somewhat limited by the pre-extracted features in publicly available datasets. These preprocessed features restrict deep learning's capacity for automatic feature extraction, undermining its potential advantages. Consequently, our future plans involve directly employing deep learning models for feature extraction and training on original network traffic data. This approach aims to capitalize on deep learning's capabilities in feature extraction, reducing the impact of imbalanced data and achieving more precise clssification outcomes.

## REFERENCES

[1] "Intrusion Detection Systems" by Richard Bejtlich
[2] "Applied Network Security Monitoring" by Chris

Sanders and Jason Smith
[3] Garcia-Teodoro, P., Diaz-Verdejo, J., Macia-Fernandez, G., & Vazquez, E. (2009). "Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges." Computers & Security, 28(1-2), 18-28.
[4] Patcha, A., & Park, J. M. (2007). "An Overview of Anomaly Detection Techniques: Existing Solutions and Latest Technological Trends." Computer Networks, 51(12), 3448-3470.
[5] Kumar, S., & Reddy, P. (2016). "A Comprehensive Review on Imbalanced Data Problem." International Journal of Computer Applications, 139(11), 6-11.