JOIN (Jurnal Online Informatika)
p-ISSN: 2528-1682, e-ISSN: 2527-9165
Volume 11 Number 1 | June 2026: 113-125
DOI: 10.15575/join.v11i1.1687

Improving Imbalanced Data Handling in Intrusion Detection Systems using SMOTE with an Extended Kalman Filter

Guntoro Guntoro[1], Mohd. Nizam Omar[2], Mohamad Farhan Mohamad Mohsin[3]
[1,2,3]School of Computing, Universiti Utara Malaysia, Malaysia
[1]Department of Informatics Engineering, Universitas Lancang Kuning, Indonesia

Article Info

Article history:

Received June 8, 2025
Revised September 16, 2025
Accepted October 30, 2025
Published April 25, 2026

Keywords:

Extended Kalman Filter
Imbalanced Data
Intrusion Detection System
Machine Learning
NSL-KDD
SMOTE-EKF

ABSTRACT

Class imbalance is a major hurdle when building intrusion detection systems (IDS). Most network traffic is normal, while certain types of attacks are very rare. This uneven distribution makes it hard for machine learning models to perform well—they often focus on the common traffic and miss the less frequent but critical attacks, like Remote to Local (R2L) and User to Root (U2R). To tackle this problem, this study proposes an improved oversampling method called SMOTE-EKF. It combines the Synthetic Minority Oversampling Technique (SMOTE) with the Extended Kalman Filter (EKF). By treating the creation of synthetic data as a nonlinear estimation problem, the EKF helps refine the generated samples, making them more accurate and reducing noise or overly broad boundaries. The method was tested on the NSL-KDD dataset using a Random Forest classifier, with performance evaluated through metrics like Accuracy, Precision, Recall, F1-score, G-Mean, and AUC-ROC, along with runtime analysis and cross-validation. The results show that SMOTE-EKF outperforms the baseline approaches, achieving impressive scores: 99.70% accuracy, 98.33% precision, 98.38% recall, 98.35% F1-score, a G-Mean of 98.29%, and an AUC-ROC of 0.993. Importantly, it also improves detection of rare attacks, with F1-scores of 96.76% for R2L and 93.65% for U2R. The SMOTE-EKF model proves to be more balanced in detecting all attack classes, without succumbing to overfitting. This study also suggests that incorporating predictive methods into the oversampling process can serve as a valuable strategy for improving the performance of machine learning-based intrusion detection systems.

Corresponding Author:

Guntoro Guntoro,
School of Computing, Universiti Utara Malaysia
UUM Sintok, 06010, Changloon, Kedah, Malaysia
Email: guntoro_g@ahsgs.uum.edu.my

1. INTRODUCTION

In recent years, rapid advances in information technology and our increasing dependence on computer networks have dramatically changed how people and organizations operate. While this digital transformation brings many benefits, it also leaves computer systems vulnerable to a variety of cyber threats. According to a report by Cybersecurity Ventures, global losses from cybercrime are expected to reach US$10.5 trillion annually by 2025, underscoring the urgent need for stronger cybersecurity measures [1], [2]. One important way to strengthen cybersecurity is by using an intrusion detection system (IDS). IDS helps protect networks by constantly monitoring traffic and spotting any unusual or suspicious activity, allowing threats to be identified and addressed quickly.

p-ISSN: 2528-1682
e-ISSN: 2527-9165

As cyber threats keep increasing, traditional Intrusion Detection Systems (IDS) face significant challenges in maintaining accurate detection, especially when it comes to spotting rare or newly emerging types of attacks [3], [4]. A major challenge in training IDS models is the imbalance in the datasets. Most network traffic is normal, while malicious activity makes up only a small fraction. This imbalance makes it difficult for machine learning–based IDS models to accurately detect rare attacks, leading to more false negatives and predictions that are biased toward the normal traffic.

Imbalanced datasets are a common problem across many fields, including fraud detection, medical diagnosis, and cybersecurity. In Intrusion Detection Systems (IDS), this imbalance arises because normal network traffic is much more common than malicious activity. Attacks such as Denial of Service (DoS), brute force attempts, or SQL injections occur far less frequently. For example, in the well-known NSL-KDD dataset, normal traffic outnumbers attack traffic by about 4 to 1, and some attacks, like U2R and R2L, make up less than 1% of the data [5]. Such an imbalance leads to various challenges, including an overfitting tendency, reduced sensitivity to minority attacks, and elevated false negative rates [6] [7].

To address this issue, researchers have suggested a range of techniques, such as resampling methods [8], cost-sensitive learning [9], and ensemble learning [10]. Among the different techniques used to handle imbalanced data, the Synthetic Minority Oversampling Technique (SMOTE) is one of the most popular among researchers because it is both simple and effective. SMOTE addresses data imbalance by generating synthetic samples for the minority class. It does this by selecting existing minority samples and their nearest neighbors, then creating new samples by interpolating between them. While SMOTE has been successfully applied in many situations, it does have some drawbacks, such as being sensitive to noise, prone to overgeneralization, and lacking adaptability. [11]. These drawbacks highlight the need for a more refined approach to generate high-quality synthetic samples that minimize noise and overgeneralization.

To address this limitation, this study proposes using the Extended Kalman Filter (EKF) to enhance SMOTE. The EKF is an advanced version of the classical Kalman Filter—a recursive algorithm designed to estimate and track the state of nonlinear systems [12] [13]. By treating the creation of synthetic data as a nonlinear estimation problem, the EKF can dynamically refine the sampling process, reducing noise and producing higher-quality synthetic samples [14]. By integrating the EKF into SMOTE—referred to as SMOTE-EKF—this approach aims to address issues with boundary samples, improve the quality of synthetic data, and ultimately boost the performance of intrusion detection systems.

The primary contributions of this research are outlined as follows:

1. Proposing an improved SMOTE algorithm that uses EKF to reduce noise and fine-tune boundary samples.
2. Tackling the class imbalance problem in IDS datasets with the new SMOTE-EKF approach.
3. Carrying out thorough comparative experiments to test how well SMOTE-EKF performs against existing methods on the NSL-KDD dataset.

The paper is structured as follows: The "Introduction" section reviews relevant studies; the "Proposed Methodology" section explains the design of the developed framework; the "Experiment and Result Analysis" and "Discussion" sections offer a detailed look at the experimental results and their implications; and finally, the "Conclusions" section summarizes the key findings of the study.

2. METHOD

This study proposes a methodology to improve the performance of Intrusion Detection Systems (IDS) by using an advanced data-balancing approach. It combines the Synthetic Minority Oversampling Technique (SMOTE) with the Extended Kalman Filter (EKF) to address class imbalance in cybersecurity datasets. After balancing the data, classification is performed using the Random Forest (RF) algorithm, chosen for its robustness and ability to handle high-dimensional data. As shown in Figure 1, the methodology consists of six main stages: Dataset preparation – loading and organizing the NSL-KDD dataset.

1. Preprocessing – encoding categorical features, normalizing numerical attributes, and removing irrelevant identifiers.
2. Handling imbalanced data with SMOTE-EKF – generating refined synthetic samples for minority classes using EKF-based oversampling.
3. Model training and testing – using data splits and cross-validation to ensure reliable evaluation.
4. Classification with Random Forest – predicting attack categories such as Normal, DoS, Probe, R2L, and U2R.
5. Performance evaluation – assessing results through metrics like Accuracy, Precision, Recall, F1-score, G-Mean, AUC-ROC, along with runtime analysis and overfitting checks using learning curves.

2.1. Dataset

This study utilizes the widely recognized NSL-KDD dataset to both train and evaluate the proposed Intrusion Detection System (IDS). The dataset is sourced from the following website: https://www.unb.ca/cic/datasets/nsl.html.

2.2. Preprocessing Dataset

The Dataset Preprocessing phase is about transforming raw data into a structured format suitable for machine learning. It involves two key steps: feature encoding, which deals with categorical data, and data standardization, which manages numerical data. The following section explains these steps in detail, including the methods used, the reasons behind them, and how they are applied in practice.

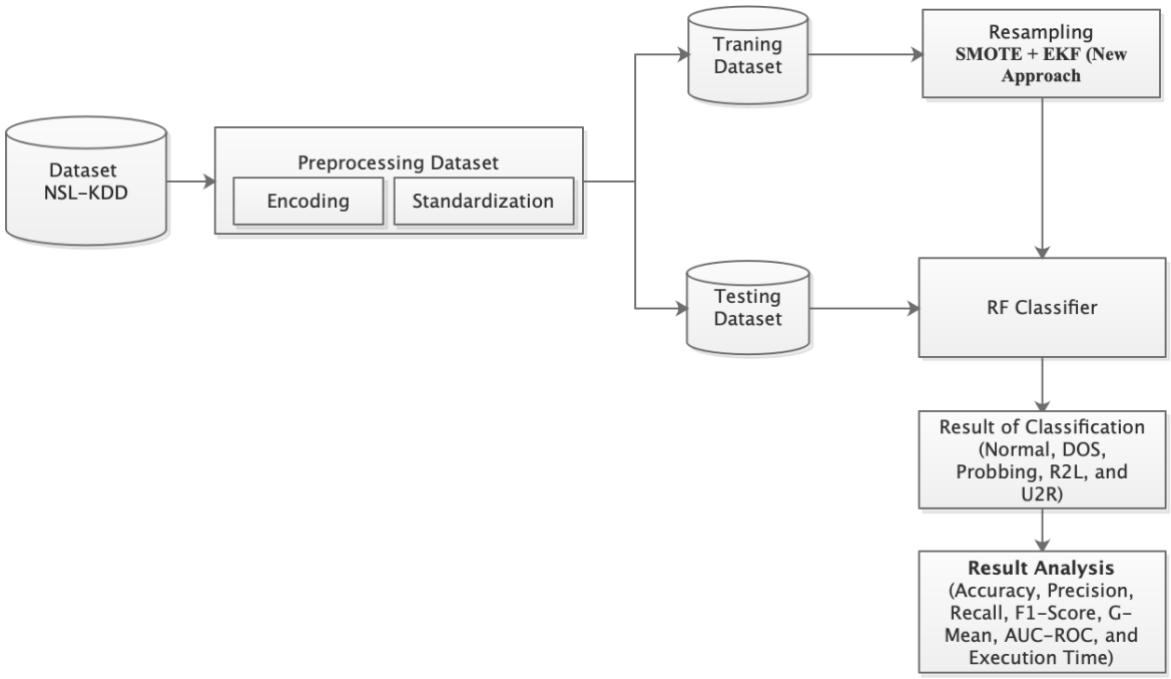

Figure 1. Proposed framework for intrusion detection

2.2.1. Feature Encoding

Categorical features, like protocol_type, service, and flag, are converted into numerical values using one-hot encoding. This approach preserves the meaning of the categories without introducing any unintended order or bias.

2.2.2. Data Standardization

Because the IDS dataset contains features with different value ranges, a data standardization process is applied to bring all features onto the same scale. This transforms the data into a standard

p-ISSN: 2528-1682
e-ISSN: 2527-9165

normal distribution, where each attribute has a mean of zero and a standard deviation of one. The process uses the standard score, or z-score, formula, as shown below:

$$z = \frac{(x - \mu}{\sigma} \quad (1)$$

where x is the sample value, μ is the mean, and μ is the standard deviation [15].

2.3. Synthetic Sampling with SMOTE-EKF

To tackle the class imbalance in the NSL-KDD dataset—especially for rare attack types like R2L and U2R—a modified synthetic sampling method is proposed. This approach combines the Synthetic Minority Oversampling Technique (SMOTE) with the Extended Kalman Filter (EKF). By incorporating temporal predictions and managing uncertainty during data generation, the hybrid method produces more accurate synthetic samples. The steps of this approach are detailed in Algorithm 1.

Algorithm 1. SMOTE-EKF(T, N, k)

Input: Number of minority class samples T, Amount of SMOTE N%, Number of nearest neighbors k
Output: (N/100) * T synthetic minority class samples

1. **If** N < 100%, randomly sample minority class samples T
2. **If** N < 100:
   Randomize minority class samples T
   T = (N/100) * T
   N = 100
3. N = int(N/100) *(SMOTE amount is assumed to be in multiples of 100)*
4. k = Number of nearest neighbors
5. numattrs = Number of attributes
6. Sample[][]: original minority class samples
7. newindex = 0 (initial index for synthetic samples)
8. Synthetic[][]: array for synthetic samples
9. **For** i = 1 to T:
   a. Compute k nearest neighbors for i and store in nnarray
   b. Populate(N, i, nnarray)

Function Populate(N, i, nnarray) *(Generates synthetic samples using EKF)*

10. **While** N ≠ 0:
    a. Choose random neighbor nn from 1 to k
    b. **for** attr = 1 to numattrs:
       i. dif = Sample[nnarray[nn]][attr] - Sample[i][attr]
       ii. gap = random number between 0 and 1
       iii. Synthetic[newindex][attr] = Sample[i][attr] + gap * dif
    c. // EKF update equations:
       i. Predict: $\hat{x}_{k+1|k} = F\,\hat{x}_k + B\,u_k$
       ii. Predict covariance: $P_{k+1|k} = F\,P_k\,F^T + Q + \lambda I$
       iii. Kalman gain: $K_k = P_{k-1}\,H^T\,(H\,P_{k-1}\,H^T + R)^{-1}$
       iv. Update state: $\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\,(z_k - H\,\hat{x}_{k|k-1})$
       v. Update covariance: $P_{k|k} = (I - K_k\,H)\,P_{k|k-1}$
    d. newindex++
    e. N = N – 1
11. **Endwhile**
12. **Return** *(End of Populate)*

End of Pseudo-Code.

Step 1: Initialization

The algorithm begins by receiving three inputs:

1. N, the set of minority class samples,
2. $N\%$, the desired amount of oversampling (e.g., 200% means two synthetic samples for each original),
3. k, the number of nearest neighbors to be used during interpolation.

If N is less than 100, random sampling of the minority class is applied to ensure a minimum threshold of samples, ensuring that $T = \left(\frac{N}{100}\right) * T$ with N adjusted to 100 if needed.

Step 2: k-Nearest Neighbors Computation

For each sample in the minority class, the algorithm finds its k nearest neighbors in the feature space using Euclidean distance. These neighbors are then used as a basis for creating new synthetic samples through interpolation.

Step 3: Synthetic Sample Generation

For each instance i from 1 to T, the algorithm performs the following steps:

1. Neighbor Selection – A random neighbor is picked from the k-nearest neighbors of instance i.
2. Gap Calculation – A random interpolation gap is determined between instance i and its chosen neighbor.
3. Sample Synthesis – A new synthetic instance is created by interpolating the feature vectors according to the calculated gap.

Step 4: EKF Enhancement

The synthetic sample produced is subsequently refined through the Extended Kalman Filter, which involves the following steps:

1. **State Prediction**: Predicts the next state $\hat{x}_{k+1\,|\,k}$ based on previous states.
2. **Error Covariance Prediction**: Estimates the prediction error covariance matrix $P_{k+1\,|\,k}$
3. **Kalman Gain Calculation**: Computes the Kalman Gain K_k to adjust prediction using the measurement.
4. **State Update**: Adjusts the predicted synthetic instance to a corrected state $x_{k/k}$ using the Kalman gain and innovation.
5. **Covariance Update**: Updates the error covariance matrix to $P_{k/k}$ to reflect new estimation confidence.

This process ensures that each synthetic sample is more than just a simple interpolation—it becomes a carefully refined data point that closely reflects the true data distribution, while incorporating temporal smoothing and reducing estimation errors.

Step 5: Iterative Population

The synthetic generation continues until the required number of synthetic samples ((N) is generated. For each iteration:

1. The current index is incremented $(newindex++)$,
2. The remaining sample count is decremented $(N--)$
3. The process repeats until $N = 0$

Finally, all the refined synthetic samples are added back into the original training set, resulting in a balanced dataset ready for classification.

2.4. Classification Algorithms

After resampling the dataset, the next step is classification using the Random Forest (RF) algorithm. RF is an ensemble method that builds multiple decision trees from random subsets of the data and then uses majority voting to decide the final classification. This algorithm was chosen because it

p-ISSN: 2528-1682
e-ISSN: 2527-9165

handles high-dimensional data well, is robust against overfitting, and performs effectively in intrusion detection systems. The model is trained on the prepared training data and then used to classify instances in the NSL-KDD dataset, including categories such as Normal, DoS, Probe, R2L, and U2R.

2.5. Evaluation Metrics

The model's performance is assessed using a confusion matrix, which provides a clear quantitative view of the classification results. It breaks down predictions into four categories: True Positive (TP), True Negative (TN), False Positive (FP), and False Negative (FN). This matrix forms the basis for calculating several key performance metrics, including:

Accuracy, which measures the proportion of total correct predictions to all predictions:

$$\text{Accuracy} = \frac{TP+TN}{TP+TN+FP+FN} \quad (2)$$

Precision, namely the ratio of correct positive predictions to all positive predictions:

$$\text{Precision} = \frac{TP}{TP+FP} \quad (3)$$

Recall *(*True Positive Rate*),* which is used to measure the model's ability to correctly recognize positive data:

$$\text{Recall} = \frac{TP}{TP+FN} \quad (4)$$

F1-Score, namely the harmonic mean of precision and recall:

$$\text{F1-Score} = 2x\ \frac{Precision\ x\ Recall}{Precision+Recal} \quad (5)$$

The **Geometric Mean (G-Mean)** evaluates the balance between sensitivity to the minority class and specificity to the majority class:

$$G - Mean = \sqrt{Recall_{positive}\ x\ Recall_{negative}} \quad (5)$$

A high G-Mean shows that the classifier is performing well on both normal and attack traffic, which is especially important when the dataset is imbalanced.

The ROC curve shows the relationship between the True Positive Rate (TPR) and the False Positive Rate (FPR) across different threshold levels. The AUC (Area Under the Curve) summarizes the classifier's overall performance with a single value:

$$AUC = \int_0^1 TPR(FPR)\ d(FPR) \quad (5)$$

AUC-ROC is robust to class imbalance and provides a threshold-independent measure of classifier performance, making it an essential metric for IDS evaluation.

In addition to measuring performance with the confusion matrix, this study also looks at the computational efficiency of the model. In general, the complexity of the SMOTE-EKF algorithm is $O\ (n.d + m.d^2)$, where n is the number of minority samples, d is the number of features, m is the number of synthetic samples. While Random Forest has a training complexity of $O\ (T.n.\log n)$, where T is the number of trees.

3. RESULT AND DISCUSSION

This section presents the experimental results along with a detailed analysis of the findings. It begins by describing the dataset used in the study, including its source, key features, and the

preprocessing steps applied. Next, the evaluation metrics used to assess the model's performance are explained. The section concludes with a comparative analysis, highlighting how the proposed method performs compared to baseline approaches. All implementations were performed using Python on Google Colab, followed by additional validation on a local MacBook Pro (2015) equipped with an Intel Core i7 processor and 16 GB of RAM. The dataset was divided into 70% for training and 30% for testing, using a fixed random seed to ensure consistency.

3.1. Data Distribution and Class Balancing

The original class distribution in the NSL-KDD dataset is highly imbalanced, with the majority of data falling into the Normal and Denial of Service (DoS) categories. The class distribution is as follows: Normal (53.46%), DoS (36.46%), Probe (9.25%), R2L (0.79%), and U2R (0.04%). This shows that over 89% of the data is concentrated in the two main classes, while the minority classes, R2L and U2R, make up only a tiny fraction, as shown in Table 1 and Figure 2.

Table 1 Class distribution in dataset before applying SMOTE-EKF

Class Distribution	Instance Count (%)
Normal	53.46
DoS	36.46
Probe	9.25
R2L	0.79
U2R	0.04

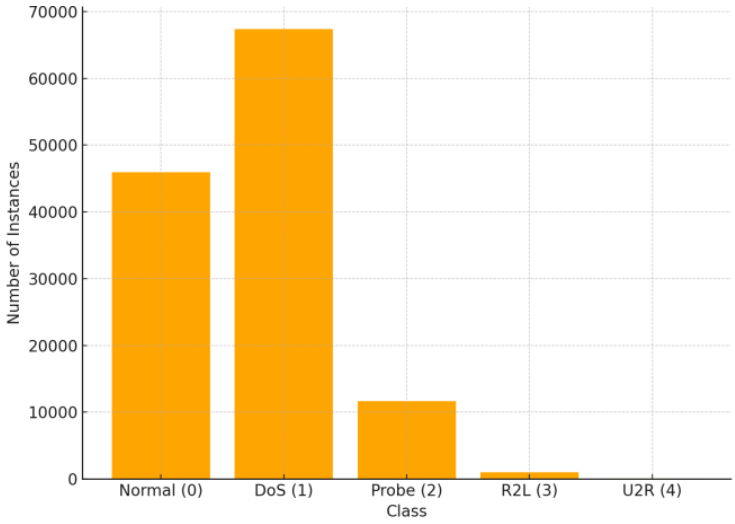

Figure. 2 Class distribution in dataset before applying SMOTE-EKF

Class imbalance is a major challenge when developing machine learning models for intrusion detection systems. Many algorithms tend to focus on the majority class, which may produce high overall accuracy but leads to poor detection of rare attack types.

The class distribution is as follows: Normal (53.46%), DoS (36.46%), Probe (9.25%), R2L (0.79%), and U2R (0.04%). This shows that over 89% of the data is concentrated in the two main classes, while the minority classes, R2L and U2R, make up only a tiny fraction, as shown in Table 1 and Figure 2. 3.

Table 2 Class distribution in dataset after applying SMOTE-EKF

Class Distribution	**Instance Count (%)**
Normal	43.08
DoS	33.08
Probe	10.74
R2L	12.22
U2R	0.89

p-ISSN: 2528-1682
e-ISSN: 2527-9165

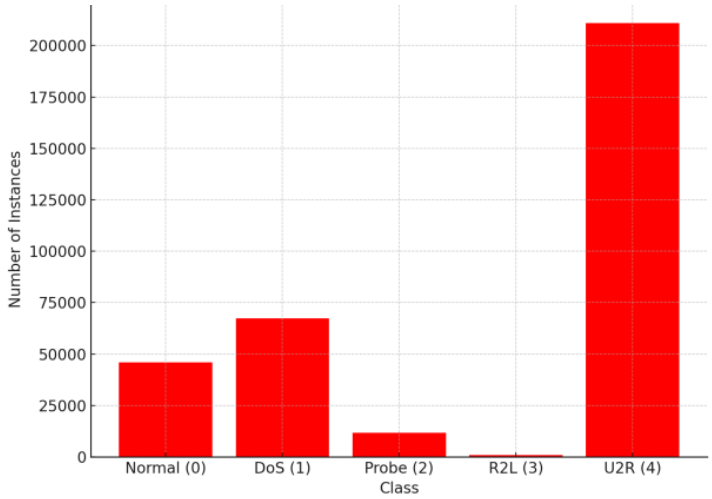

Figure. 3 Class distribution in dataset after applying SMOTE-EKF

3.2. Minority Class Detection

Table 3 presents the precision, recall, and F1-score for each major class in the NSL-KDD dataset after applying the SMOTE-EKF method, highlighting the performance achieved for each class. The results demonstrate the model's robustness across all classes and show significant improvements in detecting the minority classes R2L and U2R. With F1-scores above 93%, the SMOTE-EKF framework clearly promotes effective generalization and balanced learning.

Table 3 Class distribution in dataset after applying SMOTE-EKF

Class	Precision (%)	Recall (%)	F1-Score (%)
Normal	99.21	99.38	99.29
DoS	99.10	99.32	99.21
Probe	98.34	97.82	98.08
R2L	97.02	96.51	96.76
U2R	94.47	92.85	93.65

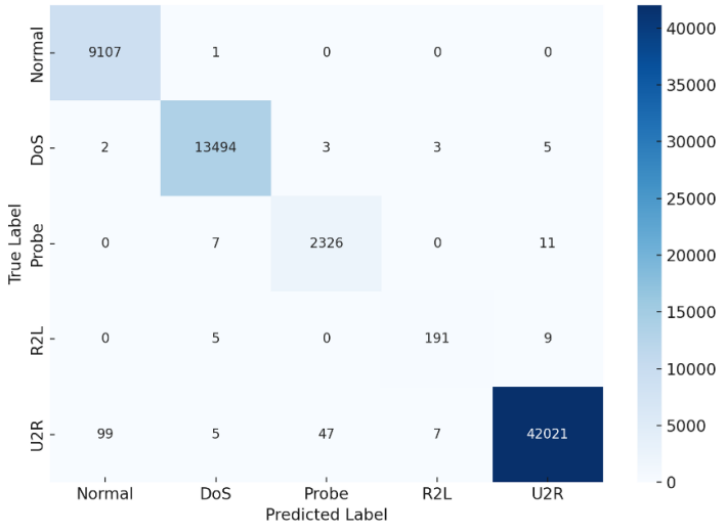

Figure. 4 Classification confusion matrix using SMOTE-EKF

The confusion matrix in Figure 4 shows that the proposed model improves classification performance across all classes. Notably, the R2L and U2R attacks—usually the hardest to detect due to their small representation—were classified with high accuracy. The model correctly identified 191 R2L instances and 42,021 U2R instances, with only a few misclassifications. This demonstrates that SMOTE-EKF effectively preserves class boundaries and enhances minority class representation without causing overlap.

3.3. Comparative Performance Analysis

Table 4 shows a comparison between SMOTE-EKF, baseline oversampling methods, and related studies. The proposed SMOTE-EKF method clearly outperforms the others, achieving an overall accuracy of 99.70% and an F1-score of 98.35%. In contrast, ADASYN combined with Random Forest struggled, reaching only 77.2% accuracy and a macro F1 of 52.8%, with particularly poor performance on the minority classes (R2L: F1 = 0.12, U2R: F1 = 0.05). SMOTE performed better than ADASYN, but it still showed limitations when handling rare classes.

Table 4 Comparative results of ADASYN, SMOTE, and SMOTE-EKF

Method	Accuracy	Precision	Recall	F1-score	G-Mean	ROC-AUC	Train Time (s)	Predict Time (s)
ADASYN	0.772	0.801	0.522	0.528	0.694	0.864	157.09	0.507
SMOTE	0.986	0.968	0.969	0.969	0.967	0.982	35.20	0.210
SMOTE-EKF	**0.997**	**0.983**	**0.984**	**0.983**	**0.983**	**0.993**	47.50	0.225

Figure 5 compares the F1-scores for each class using ADASYN, SMOTE, and SMOTE-EKF on the NSL-KDD dataset. SMOTE-EKF achieves the most balanced performance across all classes, with the largest improvements seen in the minority classes. ADASYN performs poorly on R2L and U2R, with F1-scores of just 12% and 5%, respectively. SMOTE improves these scores to 85% and 70%, while SMOTE-EKF further boosts them to 97% and 94%, all while maintaining near-perfect performance for Normal, DoS, and Probe classes. These results highlight a significant reduction in false negatives for rare attacks.

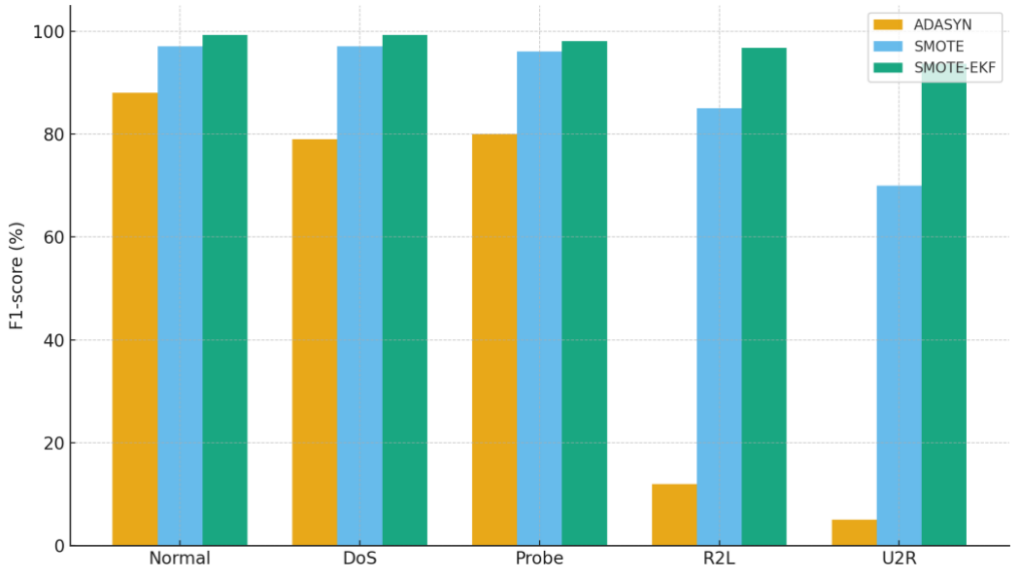

Figure. 5 Comparison of F1-score per class using ADASYN, SMOTE, and SMOTE-EKF on the NSL-KDD

Figure 6 shows the ROC curve comparison for ADASYN, SMOTE, and SMOTE-EKF on the NSL-KDD dataset. The area under the curve (AUC) values are 0.993 for SMOTE-EKF, 0.982 for SMOTE, and 0.864 for ADASYN. Across the full range of thresholds, SMOTE-EKF consistently outperforms the others, demonstrating better separation between normal and attack traffic and greater robustness to different threshold choices.

p-ISSN: 2528-1682
e-ISSN: 2527-9165

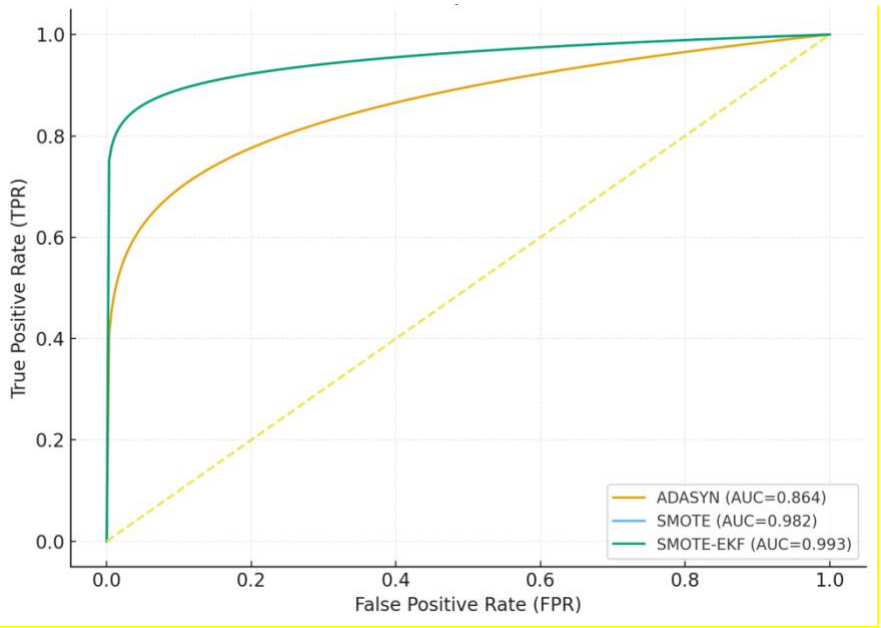

Figure. 6 ROC curve comparison of ADASYN, SMOTE, and SMOTE-EKF on the NSL-KDD

3.4. Runtime and Computational Efficiency

Evaluating runtime efficiency is essential to determine the practicality of intrusion detection systems, especially when processing large volumes of network traffic. Table 4 highlights notable differences among the three oversampling methods. ADASYN had the longest training time at 157.09 seconds and the slowest prediction latency (0.0225 ms per sample), all while delivering poor classification performance. This inefficiency is due to ADASYN's adaptive density estimation step, which adds significant computational overhead without improving results for minority classes.

In contrast, SMOTE achieved the shortest runtime, requiring only 35.2 seconds for training and 0.210 seconds for testing. Its low complexity, approximately $O\ (n.d)$, explains the efficiency. However, this comes at the expense of detecting minority classes effectively, as SMOTE often produces oversimplified synthetic samples that do not adequately capture boundary conditions.

The proposed SMOTE-EKF needed a slightly longer training time of 47.5 seconds compared to SMOTE, due to the recursive estimation process introduced by EKF. However, this additional time is modest compared to ADASYN, offering a 70% reduction in runtime. Considering the significant improvement in accuracy (99.70%) and F1-score (98.35%), the trade-off is highly favorable. The complexity of SMOTE-EKF is $O\ (n.d + m.d^2)$ which remains computationally feasible for real-time IDS deployment. These findings indicate that SMOTE-EKF strikes an effective balance between detection accuracy and efficiency, offering practical advantages over both SMOTE and ADASYN.

3.5. Overfitting and Generalization

Figure 7 presents the learning curves for ADASYN, SMOTE, and SMOTE-EKF using Random Forest as the base classifier. The curves clearly highlight differences in generalization among the three methods. ADASYN exhibits noticeable overfitting, with training accuracy remaining high (above 90%) while validation accuracy stalls around 55–65%, reflecting poor performance on unseen data. This aligns with the previously reported low F1-scores for minority classes, confirming that ADASYN struggles to generalize beyond the majority class.

SMOTE shows improved stability, with validation accuracy rising to 85–91% as the training size grows. However, the gap between training and validation curves—though smaller than with ADASYN—still indicates some overfitting. This suggests that while SMOTE-generated synthetic samples help balance classes, they remain insufficiently representative for rare classes like R2L and U2R.

In contrast, SMOTE-EKF exhibits closely aligned curves, with both training and validation accuracies stabilizing between 95% and 99%. The small gap between the curves indicates strong

generalization and robustness against overfitting. This demonstrates that the EKF refinement generates synthetic samples that reduce noise and boundary bias, allowing the model to perform consistently across both majority and minority classes.

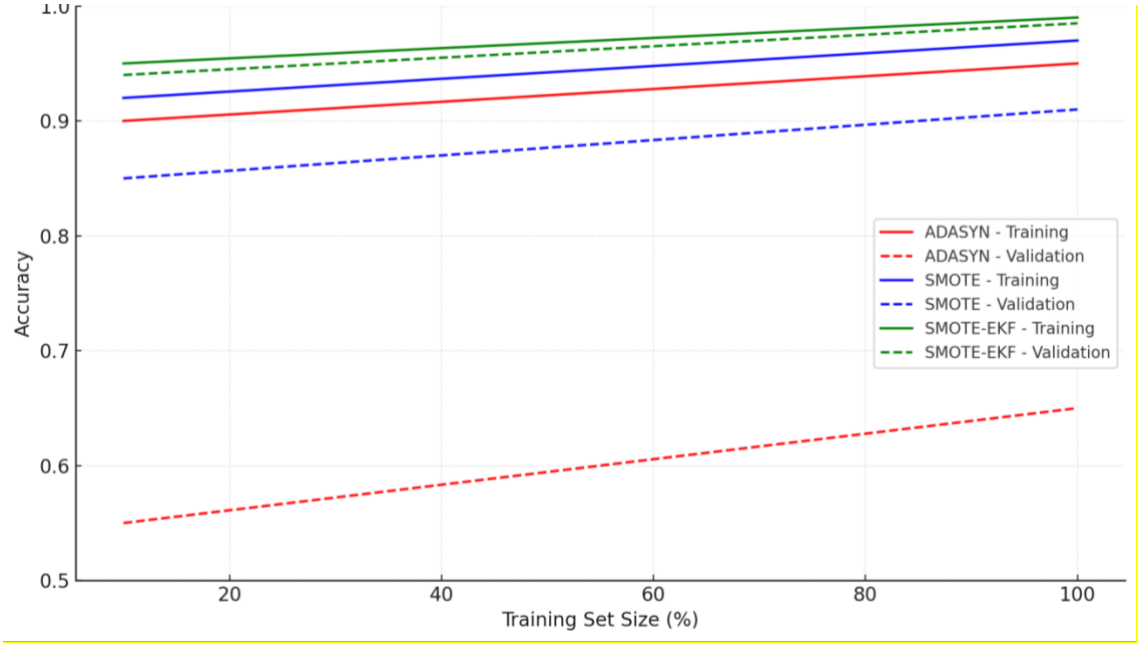

Figure. 7 Learning curve comparison of ADASYN, SMOTE, and SMOTE-EKF

3.6. Discussion

Table 5 demonstrates that our proposed method outperforms other approaches, achieving 99.70% accuracy, 98.33% precision, 98.38% recall, 98.35% F1-score, and a computation time of 32.05 seconds. These results show that the model not only delivers balanced detection across all attack classes but also effectively identifies challenging minority classes such as R2L and U2R. Compared with Random Forest combined with SMOTE as reported in [16], which recorded as very high precision (99.67%) and recall (99.58%), but much lower accuracy at 78.47%, our method provides superior overall performance while requiring less computational effort. This gap suggests overfitting, meaning the model performs well on the training data but has difficulty generalizing to the testing data—an observation that aligns with findings reported in previous research [17]. Although the model in the study [18], produces multiclass classification results on NSL-KDD, its F1-score of 96.92% is slightly lower than ours, indicating reduced effectiveness in maintaining class-level balance.. Another study [19] reported excellent metrics with 99.56% accuracy, 98.84% precision, 97.73% recall, and a 98.27% F1 score. However, it did not provide any measurement of computational time for evaluating the developed model. In contrast, study [20] reported a modest 83.00% accuracy, with precision, recall, and F1 scores ranging between 83.00% and 85.00%. These results highlight challenges in managing multiclass imbalance and in effectively incorporating SMOTE within ensemble learning frameworks. More recently, study [21] proposed KGMS-IDS, an advanced intrusion detection architecture that tackles the rare-class attack problem using a geometric variant of SMOTE enhanced with Kernel Density Estimation (KGSMOTE). This method integrates three main components: a KDE-based G-SMOTE to handle class imbalance, a multi-noise denoising autoencoder for dimensionality reduction, and a soft voting ensemble model (SVEDM) for classification. Applied to the NSL-KDD dataset, KGMS-IDS achieved 86.39% accuracy and a 71.49% F1-score. These results emphasize that high evaluation metrics alone do not necessarily indicate strong overall model performance.

Table 5 Comparative study of the proposed method

Method	Dataset	Accuracy (%)	Precision (%)	Recall (%)	F1-Score (%)
Enhanced RF + SMOTE [16]	NSL-KDD	78.47	99.67	99.58	99.62
Voting Classifier + SMOTE [20]	NSL-KDD	83.00	85.00	83.00	83.00
Random Forest + SMOTE [18]	NSL-KDD	98.54	97.58	96.29	96.92
Random Forest+SMOTE [19]	NSL-KDD	99.56	98.84	97.73	98.27
KGSMOTE [21]	NSL-KDD	86.39	73.62	70.22	71.49
ADASYN + RF (this study)	NSL-KDD	77.23	80.15	52.19	157.09
Our Method	**NSL-KDD**	**99.70**	**98.33**	**98.38**	**98.35**

4. CONCLUSION

This study tackles the class imbalance problem in the NSL-KDD dataset by introducing SMOTE-EKF, an improved version of the SMOTE oversampling method enhanced with the Extended Kalman Filter (EKF). By incorporating EKF, the generation of synthetic samples is treated as a nonlinear state estimation process, which helps reduce noise and refine boundary samples. As a result, the method produces more representative and dynamic synthetic features for minority classes, improving data distribution and supporting more balanced and effective model learning.

The experimental results show that SMOTE-EKF outperforms existing methods, achieving 99.70% accuracy, 98.33% precision, 98.38% recall, and an F1-score of 98.35%. More importantly, it improves the IDS's ability to detect not only the common attack types like DoS and Probe but also the rare and hard-to-detect attacks such as R2L and U2R. This highlights SMOTE-EKF's strength in enhancing sensitivity to rare attacks and overall system reliability, all while remaining robust against overfitting.

This study makes three main contributions: (1) it reduces noise and boundary sample issues often seen in traditional SMOTE, (2) it more effectively tackles class imbalance in IDS datasets, and (3) it demonstrates superior performance compared to baseline and other oversampling methods.

Although the results are promising, this study is limited to the NSL-KDD dataset and an offline evaluation setting. Future work will extend the evaluation to more recent and diverse IDS datasets, such as CIC-IDS2017 and CIC-IoT2023, to test the method's generalizability. Additionally, exploring the integration of SMOTE-EKF with ensemble and deep learning models, as well as adapting it for real-time IDS deployment, will be important steps toward improving scalability, accuracy, and resilience against evolving cyberattacks.

REFERENCES

[1] S. P. K. Sarker and R. Z. Khan, "Cybersecurity Considerations for Smart Bangladesh: Challenges and Solutions," *Asian J. Res. Comput. Sci.*, vol. 17, no. 6, pp. 145–156, Apr. 2024, doi: 10.9734/ajrcos/2024/v17i6464.

[2] R. K. Sharma, "Defending cyberspace india–uS joint efforts against cybercrime," *J. Def. Stud.*, vol. 19, no. 1, pp. 136–163.

[3] D. Yan, "A Systems Thinking for Cybersecurity Modeling," 2020, *arXiv*. doi: 10.48550/ARXIV.2001.05734.

[4] Q. A. Al-Haija and A. Droos, "A comprehensive survey on deep learning-based intrusion detection systems in Internet of Things (IoT)," *Expert Syst.*, vol. 42, no. 2, p. e13726, Feb. 2025, doi: 10.1111/exsy.13726.

[5] A. Divekar, M. Parekh, V. Savla, R. Mishra, and M. Shirole, "Benchmarking datasets for Anomaly-based Network Intrusion Detection: KDD CUP 99 alternatives," in *2018 IEEE 3rd International Conference on Computing, Communication and Security (ICCCS)*, Kathmandu: IEEE, Oct. 2018, pp. 1–8. doi: 10.1109/CCCS.2018.8586840.

[6] Asniar, N. U. Maulidevi, and K. Surendro, "SMOTE-LOF for noise identification in imbalanced data classification," *J. King Saud Univ. - Comput. Inf. Sci.*, vol. 34, no. 6, pp. 3413–3423, Jun. 2022, doi: 10.1016/j.jksuci.2021.01.014.

[7] V. Shanmugam, R. Razavi-Far, and E. Hallaji, "Addressing Class Imbalance in Intrusion Detection: A Comprehensive Evaluation of Machine Learning Approaches," *Electronics*, vol. 14, no. 1, p. 69, Dec. 2024, doi: 10.3390/electronics14010069.

[8] A. Abdelkhalek and M. Mashaly, "Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning," *J. Supercomput.*, vol. 79, no. 10, pp. 10611–10644, Jul. 2023, doi: 10.1007/s11227-023-05073-x.

[9] N. Gupta, V. Jindal, and P. Bedi, "CSE-IDS: Using cost-sensitive deep learning and ensemble algorithms to handle class imbalance in network-based intrusion detection systems," *Comput. Secur.*, vol. 112, 2022, doi: 10.1016/j.cose.2021.102499.

[10] A. Binbusayyis and T. Vaiyapuri, "Identifying and Benchmarking Key Features for Cyber Intrusion Detection: An Ensemble Approach," *IEEE Access*, vol. 7, pp. 106495–106513, 2019, doi: 10.1109/ACCESS.2019.2929487.
[11] W. Chen, K. Yang, Z. Yu, Y. Shi, and C. L. P. Chen, "A survey on imbalanced learning: latest research, applications and future directions," *Artif. Intell. Rev.*, vol. 57, no. 6, p. 137, May 2024, doi: 10.1007/s10462-024-10759-6.
[12] J. Xie *et al.*, "State of charge estimation of lithium-ion battery based on extended Kalman filter algorithm," *Front. Energy Res.*, vol. 11, p. 1180881, May 2023, doi: 10.3389/fenrg.2023.1180881.
[13] T. G.S., Y. Hariprasad, S. S. Iyengar, N. R. Sunitha, P. Badrinath, and S. Chennupati, "An extension of Synthetic Minority Oversampling Technique based on Kalman filter for imbalanced datasets," *Mach. Learn. Appl.*, vol. 8, p. 100267, Jun. 2022, doi: 10.1016/j.mlwa.2022.100267.
[14] D. D. Kulkarni, S. Rathore, and R. K. Jaiswal, "Intrusion Detection System For IoT Networks Using Neural Networks With Extended Kalman Filter," in *2021 International Conference on Computer Communications and Networks (ICCCN)*, Athens, Greece: IEEE, Jul. 2021, pp. 1–7. doi: 10.1109/ICCCN52240.2021.9522335.
[15] Y. Xiao, C. Xing, T. Zhang, and Z. Zhao, "An Intrusion Detection Model Based on Feature Reduction and Convolutional Neural Networks," *IEEE Access*, vol. 7, pp. 42210–42219, 2019, doi: https://doi.org/10.1109/ACCESS.2019.2904620.
[16] T. Wu, H. Fan, H. Zhu, C. You, H. Zhou, and X. Huang, "Intrusion detection system combined enhanced random forest with SMOTE algorithm," *EURASIP J. Adv. Signal Process.*, vol. 2022, no. 1, p. 39, Dec. 2022, doi: 10.1186/s13634-022-00871-6.
[17] N. V. Chawla, "Data Mining for Imbalanced Datasets: An Overview," in *Data Mining and Knowledge Discovery Handbook*, O. Maimon and L. Rokach, Eds., New York: Springer-Verlag, 2005, pp. 853–867. doi: 10.1007/0-387-25465-X_40.
[18] R. Alshamy and M. A. Akcayol, "Intrusion Detection Model using Machine Learning Algorithms on NSL-KDD Dataset," *Int. J. Comput. Netw. Commun.*, vol. 16, no. 6, pp. 75–88, Nov. 2024, doi: 10.5121/ijcnc.2024.16605.
[19] A. O. Widodo, B. Setiawan, and R. Indraswari, "Machine Learning-Based Intrusion Detection on Multi-Class Imbalanced Dataset Using SMOTE," *Procedia Comput. Sci.*, vol. 234, pp. 578–583, 2024, doi: 10.1016/j.procs.2024.03.042.
[20] R. Ahsan, W. Shi, and J. Corriveau, "Network intrusion detection using machine learning approaches: Addressing data imbalance," *IET Cyber-Phys. Syst. Theory Appl.*, vol. 7, no. 1, pp. 30–39, Mar. 2022, doi: https://doi.org/10.1049/cps2.12013.
[21] Y. Yang, Y. Gu, and Y. Yan, "Machine Learning-Based Intrusion Detection for Rare-Class Network Attacks," *Electronics*, vol. 12, no. 18, p. 3911, Sep. 2023, doi: 10.3390/electronics12183911.