International Journal of Computer Information Systems and Industrial Management Applications
ISSN 2150-7988 Volume 18(21s) 2026 pp. 793-807
https://doi.org/10.70917/ijcisim-2026-5344
Article

FLOW PATTERN CONSTRUCTION-BASED DECENTRALIZED INTRUSION DETECTION FRAMEWORK USING TL-IML3CSTM AND GAUSSIAN-Π FUZZY

Vineeta Shrivastava[1*], Sandeep Kumar Agrawal[2], Pravesh Kumar Dwivedi[3], Jyoti Pasi[4], Arvind Sharma[5], Yogita Verma[6]

[1]Department of Computer Science & Engineering, Lakshmi Narain College of Technology Excellence, Bhopal, Madhya Pradesh, India-462022
[2,5,6]RJIT BSF Academy, Gwalior,Madhya Pradesh, India-475001
[3]Parul Institute of Technology, Parul University,Vadodara,Gujarat,India-391760
[4]Galgotias College of Engineering & Technology, Greater Noida, Uttar Pradesh,India-201310
[1]shrivastavavinita21@gmail.com,[2]rjitsandeep@gmail.com,[3]mtpravesh@gmail.com,[4]jyotipasi100@gmail.com,[5]arvindsharma@rjit.ac.in,[6]dr.yogitaverma@rjit.ac.in
*Corresponding Author Name & Email : Vineeta Shrivastava & shrivastavavinita21@gmail.com

Abstract: A Decentralized Intrusion Detection System (DIDS) distributes detection tasks across multiple nodes and collaboratively identifies the network intrusions using data patterns. However, the traditional techniques failed to map the data flow patterns after the identification of normal and generic attacks, thus causing misclassifications or inaccuracies in the system. Hence, to overcome these issues, the HamKowski Ford-Fulkerson Algorithm (HKFFA) and Transfer Learning-based Inverse Mish Le-LayerCun Long Short-Term Memory (TL-IML3CSTM) are used in this framework. At first, the data is collected, preprocessed, and then balanced to reduce the training duration. After that, the behavioral patterns are identified using EGDBSCAN and flow patterns are constructed utilizing HKFFA to enhance the attack detection process. The normal and generic attacks are then categorized with high Attack Detection Accuracy (ADA) of 99.32% using TL-IML3CSTM. After that, the low-severe attacks and normal data are mapped with possible flow patterns. Here, the mapped data is preserved and stored in the cloud for future usage, while the non-mapped data is blocked for security purposes, thus outperforming the prevailing models.
Keywords: EntroGrid Density-Based Spatial Clustering of Applications with Noise (EGDBSCAN), Gaussian-Π Fuzzy (GΠ-Fuzzy), LMicro-Utility based K-Anonymity (LMU-KAnonymity), Generic Attack (GA), Transfer Learning (TL), Flow Pattern Construction (FPC), and Intrusion Detection (ID)

1. INTRODUCTION

In the realm of network security, DIDS represents a paradigm shift by distributing detection responsibilities across multiple nodes (Disha & Waheed, 2022). This approach not only enhances detection accuracy but also bolsters resilience against sophisticated attacks (Zoppi et al., 2024, Alotaibi & Rassam, 2023). Intrusions are identified through collaborative analysis of behavioral patterns, signature techniques (Ahmed et al., 2022), and anomaly detection algorithms deployed at each node (Rustam et al., 2024).

Prevailing Deep Learning (DL) techniques (Jiang et al., 2020) like Artificial Neural Network (ANN) and Long-Short Term Memory (LSTM) (Sworna et al., 2023) detected the intrusions by analyzing data patterns (Nabi & Zhou,

Copyright: © 2026 by the authors

2024). However, drawbacks like false positives and non-adaptation to new attack strategies (Samantaray et al., 2024) degraded the system's performance (Vitorino et al., 2023). Also, the prevailing methods failed to concentrate the flow patterns of the normal data, thus leading to inaccuracies in ID (Sharma et al. 2025). Hence, to overcome these drawbacks, the proposed work leveraged various techniques for effective ID in networks (Li et al,2025).

1.1 Problem Statement

The drawbacks in prevailing works are explained as follows,

- None of the works concentrated on the attacked flow patterns after the categorization of generic and normal data.
- None of the works identified behavioral patterns based on total entropy and jitter calculations, thus causing high false alarms.
- The Severity Level (SL) of the attacked data was not focused on prevailing works.
- In (Yu et al., 2023), the generic attacks were detected using imbalanced datasets that increased the duration of the training process.
- The unpreserved normal data (Park et al., 2023) was stored in the cloud for future usage, thus leading to security vulnerabilities.

The objectives of the proposed work are described below,

- The proposed work mapped the attained flow pattern with possible flow patterns for accurate attack detection.
- The behavioral patterns are identified using the EGDBSCAN algorithm.
- SL of the attacked data is predicted using GΠ-Fuzzy.
- Data balancing is done using the Synthetic Minority Over-sampling TEchnique (SMOTE).
- Privacy preservation for non-attacked data is done utilizing LMU-KAnonymity.

The rest of the paper is organized as: Section 2 discusses the related works, Section 3 describes the proposed methodology, Section 4 presents the results and discussion, and finally, Section 5 concludes the proposed work with future development.

2. LITERATURE SURVEY

(Yu et al., 2023) introduced a hybrid network classifier that combined the improved residual network blocks and Bidirectional Gated Recurrent Units. The feature dimensionality of network data in this framework was reduced using an improved autoencoder before attack categorization. However, the attack categorization in the imbalanced dataset caused inaccuracies that led to reduced performance.

(Park et al., 2023) developed an Artificial Intelligence (AI)-based ID framework for attack detection. This work used a Generative Adversarial Network (GAN) to focus on reconstruction error and Wasserstein distance for attack detection. However, the non-preservation of non-attacked data in this framework could lead to security vulnerabilities, thus degrading the overall performance.

(He et al., 2024) accomplished a DEformable VIsion Transformer (DE-VIT) method for network-ID. DE-VIT used flexible attention, deformable convolutions, sliding windows, and a layered focal loss function to improve the ADA. Nevertheless, the large amount of memory usage in this work hindered the entire work process.

(Halbouni et al., 2022) established a hybrid ID system for generic attack and normal data identification. This work combined the Convolutional Neural Networks for spatial feature extraction and LSTM networks for temporal feature extraction. However, this framework had a high false alarm rate, thereby lowering the attack detection rate.

(Hnamte et al., 2023) presented a two-stage DL technique by combining LSTM and Auto-Encoders for ID. This work optimized network parameters and enhanced the network's ADA. However, the prolonged training duration and complexity of large datasets caused scalability issues in detecting intrusions.

3. PROPOSED METHODOLOGY FOR NETWORK INTRUSION DETECTION

The structural diagram of the proposed work using TL-IML3CSTM and GΠ-Fuzzy is shown in Figure 1,

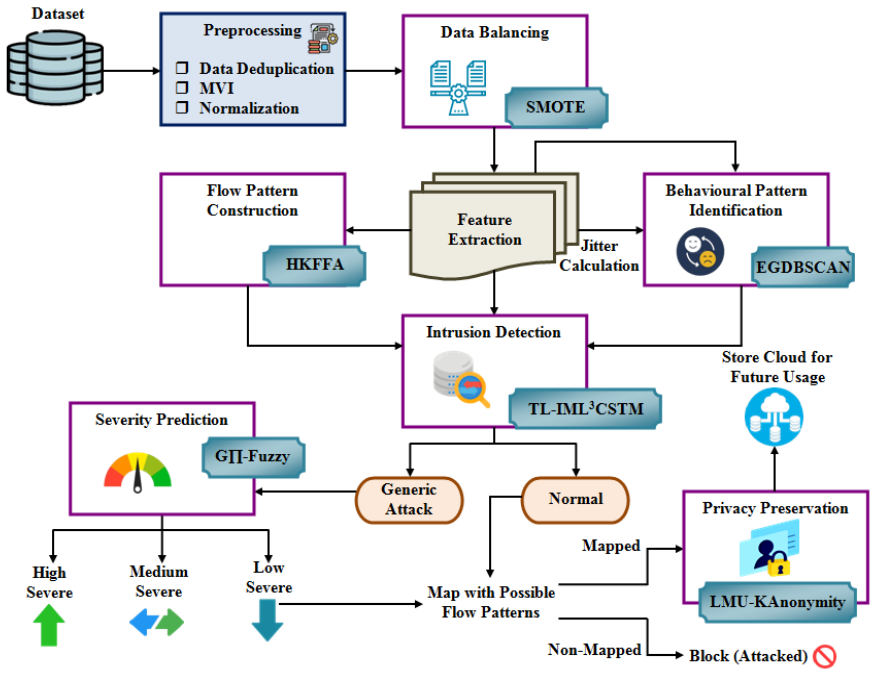


Figure 1: Structural Diagram of the Proposed Work

3.1 Data Collection

The proposed work starts by collecting the data from two datasets, namely Network Flow Dataset (NFD) and Network Intrusion Detection Dataset (NIDD), which are expressed as,

$$C = C^1, C^2, \dots\dots, C^{\hat{d}} \quad (1)$$

Here, $(\hat{d})$represents the total number of collected data (C).

3.2 Preprocessing

From (C), the preprocessing is performed using Data Deduplication (DD), Missing Value Imputation (MVI), and Normalization and is described below,

- The data is initially reduced (R^{data}) using original data size(C) and deduplicated data size (D^{dup}) as shown below,

$$R^{data} = \frac{(C - D^{dup})}{C} \quad (2)$$

- MVI then fills absent data (A^d) that are then normalized (D^{nor}) using minimum and maximum values as shown below,

$$D^{nor} = \frac{A^d - A^d_{min}}{A^{d^d_{min}}_{max}} \quad (3)$$

Here, $\left(A^{d^d_{min}}_{max}\right)$ represent the maximum and minimum values $(max, min())$ of (A^d), respectively. Hence, the preprocessed data is denoted as (P^{data}).

3.3 Data Balancing and Feature Extraction

(P^{data}) is then balanced using SMOTE, which helps to prevent models from being biased towards the majority class as shown below (Govindarajan et al.,2025),

→ SMOTE selects the minority class instance (M^{cls}) from (P^{data}) and also selects the subset of instance $\left(m^i\right)$ from (M^{cls}).

→ The distance (L^{ed}) is then calculated between $\left(m^i\right)$ and all other instances in minority class (m) as equated below,

$$L^{ed} = \sqrt{(m^i - m)^2} \quad (4)$$

→ A synthetic value (s_v) is then generated for each attribute using the random values [0,1], which repeats until the maximum number of iterations is reached.

→ After that, the original $(M^{cls})$is combined with a newly generated (s_v) and is provided as,

$$D^{balanced} = M^{cls} \cup s_v \quad (5)$$

Here, $(D^{balanced})$ represents the balanced data. After that, the features, such as src_ip, src_port, dest_ip, dest_port, protocol, total_entropy, duration, sttl, dttl, sloss, dloss, label, and so on are extracted and are represented as (E^{fea}).

3.4 Behavioural Pattern Identification

Then, based on arrival and departure time, the jitter is calculated as,

$$C^{jit} = (A_z - A_{z-1}) - (D_z - D_{z-1}) \quad (6)$$

Here, $\left(C^{jit}\right)$ represents the calculated jitter value, and $(AandD)$ represent the arrival and departure time of packets $(z, z-1)$, respectively.

After that, based on calculated jitter, flow_durations, and total entropy, the behavioral patterns are identified using Density-Based Spatial Clustering of Applications with Noise (DBSCAN). DBSCAN groups closely packed data points into clusters and distinguishes them from noise (Khan et al. 2025). However, MinPts in DBSCAN can be too small (including noise) or too large (missing clusters). Therefore, Entropy-Grid Holdout (EGH) tuning, which optimizes MinPts by evaluating clustering metrics across values by ensuring robust noise reduction and effective cluster capture, is introduced.

→ At first, the core points (C_1, C_2) needed for grouping are selected based on two parameters, namely Minpts(υ) and Epsilon(ε). Here, (ε) is selected based on neighboring points (n^{pts}). But, (υ) is tuned using EGH for an accurate (C_1, C_2) determination as shown below,

$$\upsilon = \left(\sum_{\theta=1}^{g} p_g \, log_2(p_g)\right)^{H^s} \quad (7)$$

Here, $(\theta = 1tog)$ represents the total grid values for each (n^{pts}), $(log())$ represents the logarithmic function, (p_g) represents the probability of (g), and (H^s) represents the holdout validation that splits the data for training and testing.

→ After that, based on (υ) and min-max epsilon(εmax_{min}), (C_1, C_2) are calculated as,

$$C^1 = \varepsilon_{max} \quad (8)$$

$$C^2 = \varepsilon_{min} \quad (9)$$

Then, the noise points are removed, and behavioral patterns are identified that are represented as (I^{BP}).

3.5 Flow Pattern Construction

Simultaneously, from (E^{fea}), the data flow patterns are constructed as trees using the Ford-Fulkerson Algorithm (FFA). However, FFA may not always choose the most efficient paths due to its focus on residual capacities. Therefore, Hamkowski Distance (HD) that combines Hamming Distance (prioritizes shortest paths) and Minkowski (to consider all paths) for improved path selection is introduced in FFA.

- ❖ Firstly, the flow is initialized based on source (s^c), sink (s^y), and capacity (c^z) for tree formation.
- ❖ Then, the augmenting paths are accurately identified (Y^{ham}) using HD and are shown below,

$$Y^{ham} = \sum \times \omega(s^c, s^y, c^z) \times |s^c, s^y, c^z|^{\tau} \quad (10)$$

Here, (ω) represents the Kronecker delta function in HD, and (τ) represents the controlling parameter. Then, based on bottleneck capacity in each node, the possible flow patterns (trees) are constructed and are denoted as (F^{con}).

3.6 Intrusion Detection

Next, the constructed patterns, extracted features, and identified patterns are inputted to LSTM for generic attack and normal data categorization. But, LSTM suffers from overfitting and slow learning efficiency. Hence, to overcome this issue, Transfer Learning (TL) with Inverse-Mish Square Root Unit Activation (IMSRUA) and Le-LayerCun Sequential Unit Variance (2L-CSUV) weight initialization technique, which ensures the balanced cell state updates and reduces the Training Time (TT) for enhanced performance, are introduced.

TL speeds up LSTM training by using previously learned patterns and knowledge. This helps the LSTM learn faster and perform better on new tasks without needing as much data or time for training. Therefore, it makes the model more efficient and adaptable for various applications. The TL-IML3CSTM classifier is shown in Figure 2,

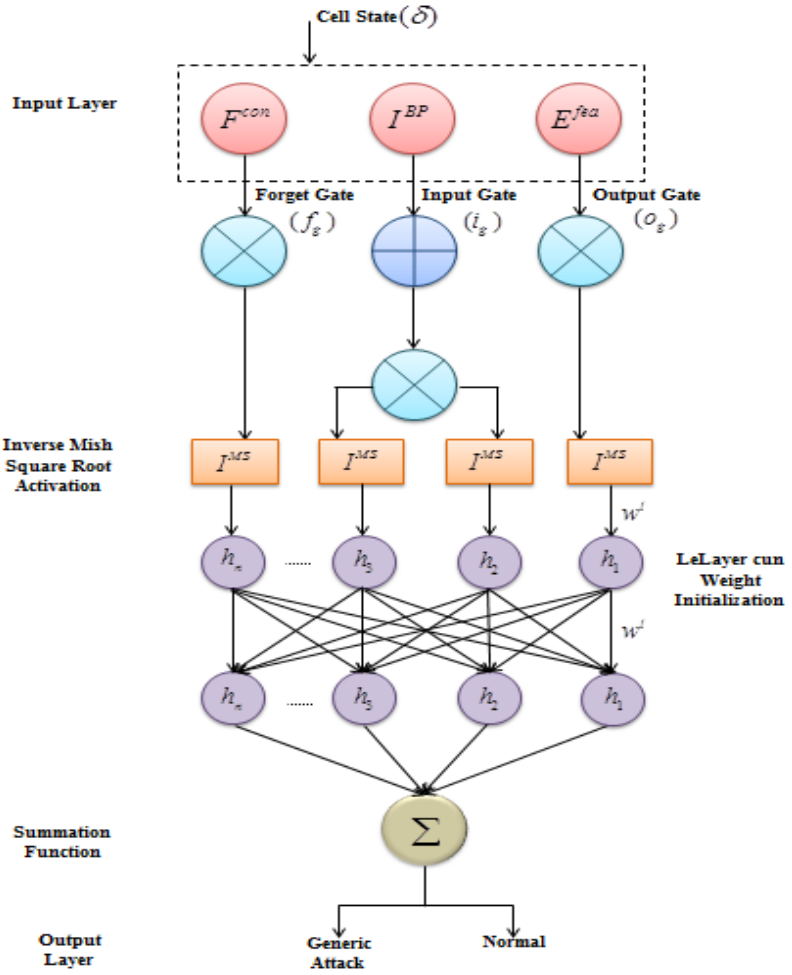


Figure 2: TL-IML3CSTM classifier

Here, the TL-IML3CSTM uses gating mechanisms, such as an input gate$\left(i_g\right)$, forget gate$\left(f_g\right)$, and output gate $\left(o_g\right)$ to control the flow of information throughout the network. Here, the weights are initialized (w^l) using the Le-LayerCun technique as,

$$w^l = \frac{U}{\sigma^2}\left(-\sqrt{\frac{1}{I_{com}}}, \sqrt{\frac{1}{I_{com}}}\right) \tag{11}$$

Here, $(Uand\sigma)$ represent uniform distribution and standard deviation for combined input (I_{com}), respectively.

Forget Gate

The non-used information in the previous cell state $(\delta^{t-1})$with time(t) is removed in $\left(f_g\right)$ using IMSRUA (I^{MS})as,

$$f_g = \{w^l * (I_{com}, h^{t-1}) + b\} \times I^{MS} \tag{12}$$

Here, (h^{t-1}) represents the previous hidden state, and(b) represents the bias value. (I^{MS}) is represented as,

$$I^{MS} = \frac{I_{com} \cdot \left(e^{I_{com}} - e^{-I_{com}}\right)}{\sqrt{1+\alpha I_{com}^2} \cdot \left(e^{I_{com}} + e^{-I_{com}}\right)} \times \left(ln(1 + e^{I_{com}})\right) \quad (13)$$

Here, (α) represents the parameter of (I_{com}), and $(ln\, a\, nde)$ represent the logarithmic and exponential functions, respectively.

Input Gate

Then, the useful information is added to (δ)using $\left(i_g\right)$ as,

$$i_g = f_g \otimes \{(I_{com}, h^{t+1}) + l\} \times I^{MS} \quad (14)$$

Output gate

The $\left(o_g\right)$ determines what information that LSTM passes on from (δ) to (h) and is given as,

$$o_g = \left[f_g \oplus i_g - \delta^t\right] \times I_{com} \quad (15)$$

$$o_g \rightarrow \{G^{attack}, G^{non-attack}\} \quad (16)$$

Thus, the classified results, namely generic attacked data $\left(G^{attack}\right)$ and normal data$\left(G^{non-attack}\right)$ are obtained.

Pseudo code of TL-IML3CSTM

Input: Combined input, (I_{com})

Output: Generic attacked data and normal data, $\{G^{attack}, G^{non-attack}\}$

Begin

Initialize iterations $(t, t^{max}())$

While $(t < t^{max}())$

Initialize (I_{com}) with weights,

Compute $\left(f_g\right)$, and (I^{MS})

Evaluate $i_g = f_g \otimes \{(I_{com}, h^{t+1}) + l\} \times I^{MS}$

Calculate $\left(o_g\right)$,

End while

Return$\{G^{attack}, G^{non-attack}\}$

End

After that, from (G^{attack}), SL is predicted as described in the below sections.

3.7 Severity Prediction

SL is then predicted using Fuzzy for easier understanding. However, Fuzzy has difficulty tuning membership functions to control rules. Therefore, the Gaussian-Π (GΠ) membership function, which offers a data-driven approach to automatically determine parameters, is introduced. Thus, it enhances accuracy in predicting the severity of the attacked data.

i. At first, the rules $(\Re)$ are set based on the if-then condition as,

$$\Re = \begin{cases} if C^{jit} \leq 0.3s, then L^{severe} \\ if C^{jit} < 0.3sto > 0.6s, then M^{severe} \\ if C^{jit} > 0.6s, then H^{severe} \end{cases} \quad (17)$$

Here, the condition states that if $\left(C^{jit}\right)$ is $(\leq 0.3s, < 0.3sto > 0.6sor > 0.6s)$, then low, medium, and high attack severity $(L^{severe}, M^{severe}, H^{severe})$ are predicted.

ii. Then, the GΠ membership function (Π^G) is assigned for fuzzy using (c_1, c_2) as Gaussian centers to overcome the tuning difficulties as,

$$\Pi^G = \frac{e^{-\left(\frac{(\Re - c_1)^2}{2\sigma^2}\right)}}{1+\left(\frac{\Re - c_1}{\sigma_1}\right)^2} \cdot \frac{e^{-\left(\frac{(\Re - c_2)^2}{2\sigma^2}\right)}}{1+\left(\frac{\Re - c_2}{\sigma_2}\right)^2} \quad (18)$$

iii. After that, the final decision (F^{dec}) is obtained using a fuzzy relationship(ϑ) as,

$$F^{dec} = \frac{\sum G^{attack} \times \vartheta(\Re)}{\Pi^G} \quad (19)$$

iv. Finally, from (F^{dec}), the crisp output $\left(Q^{crisp}\right)$ is obtained as,

$$Q^{crisp} = \frac{\sum F^{dec} \times \Pi^G}{\sum \Pi^G} \quad (20)$$

Hence, based on rules, the attack severities $(L^{severe}, M^{severe}, and H^{severe})$ are categorized.

3.8 Privacy Preservation

After prediction, (L^{severe}) and $(G^{non-attack})$ are mapped with (F^{con}) to check whether the data flow (path) is attacked or not based on the below condition $(\Im)$ as,

$$\Im = \begin{cases} if \psi^{map} then B \\ f \psi^{map*} then B * \end{cases} \quad (21)$$

Here, the condition states that if the data path is mapped (ψ^{map}), then it is not blocked (B) and used for further analysis, and if the data path is not mapped (ψ^{map*}), then it is blocked $(B *)$, which is considered as an attacked data path.

Then, (ψ^{map}) privacy is preserved using K-Anonymity based on suppression and generalization techniques. However, K-Anonymity can cause significant information loss through over or under-generalization. Hence, the LMicro-Utility (LMU) technique, which optimizes the generalization levels accurately for detailed analysis, is introduced.

(ψ^{map}) is now suppressed using the suppressing function(Θ) and is expressed as,

$$P^{sup(\psi^{map})} \quad (22)$$

Here, $(p^{sup}())$ represents the suppressed data.

The $(p^{sup}())$ is then generalized (Ψ) using LMU technique as shown below,

$$\Psi = \frac{\Omega}{P^{sup}{}_x^{P^{sup}{}_x^{P^{sup}}}} \quad (23)$$

Here, (Ω) represents the diverse sensitive information, and $\left(w_x^{P^{sup}{}_x^{P^{sup}}}\right)$ represent the weight and utility score of each $(p^{sup}())$, respectively.

Pseudo code of LMU-KAnonymity

Input: Mapped data, (ψ^{map})

Output: Privacy preserved data, $\left(D^{pre-privacy}\right)$

Begin

Initialize $(t, t^{max}())$

While $(t < t^{max}())$

Initialize (ψ^{map})

Suppress (ψ^{map}),$P^{sup(\psi^{map})}$

Generalize privacy information using LMU

End while

Return(D^{**})

End

Therefore, the privacy-preserved data is represented as(D^{**}), which is then stored in the cloud for future usage. The performance assessment is thus described in the below sections.

4. RESULTS AND DISCUSSION

This section compares the performance assessment of the proposed technique with prevailing techniques and related works. The entire work is implemented in the PYTHON platform.

4.1 DATASET DESCRIPTION

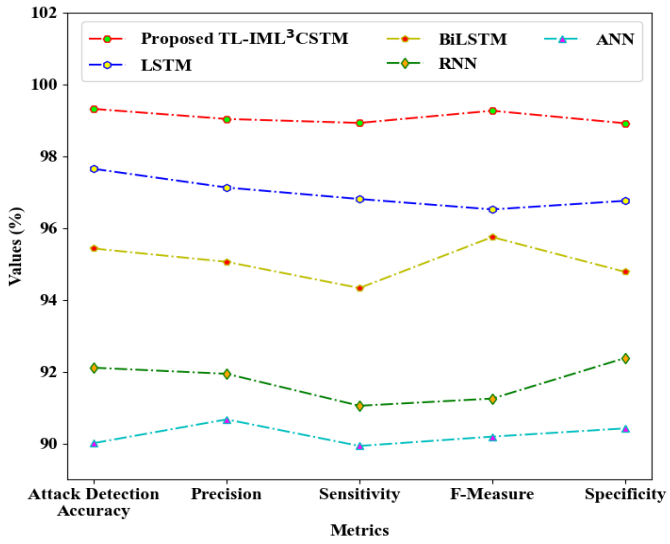
The proposed work used two datasets, namely NFD and NIDS, which are gathered from publicly available sources. From the datasets, the proposed work used 80% and 20% of data for training and testing, respectively, and is shown in Table1,

Table-1: Dataset Description

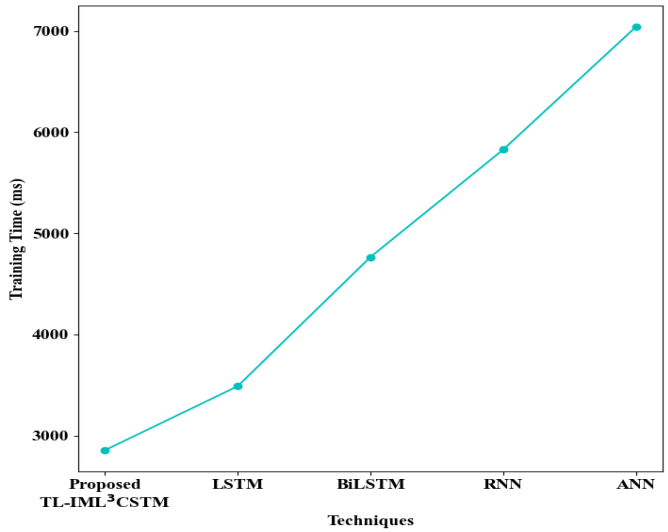
Datasets	Training	Testing
NFD	80000	20000
NIDD	175341	82332
Total	255341	102332

4.2 Performance Analysis of the Proposed Work

In this section, the performance of TL-IML3CSTM, GΠ-Fuzzy, EGDBSCAN, HKFFA, and LMU-KAnonymity is compared with the prevailing techniques. The performance assessment of the proposed TL-IML3CSTM is depicted in Figure 3 (a) and (b),



(a)

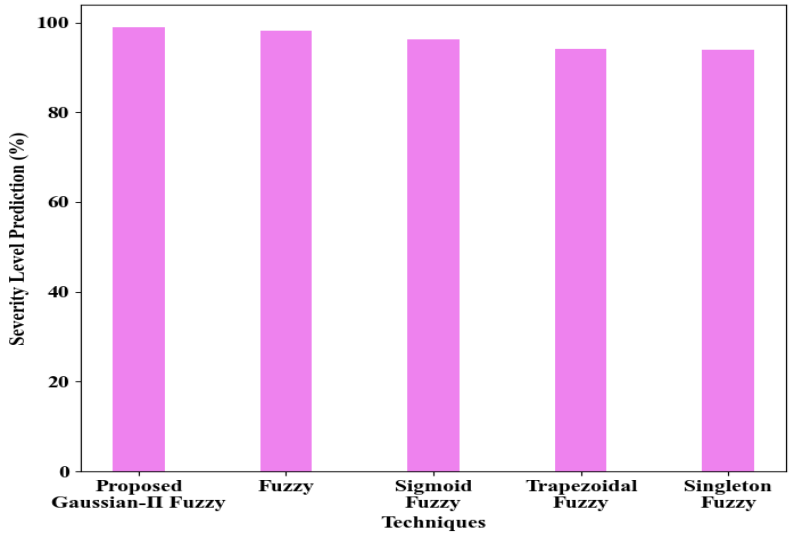


(b)

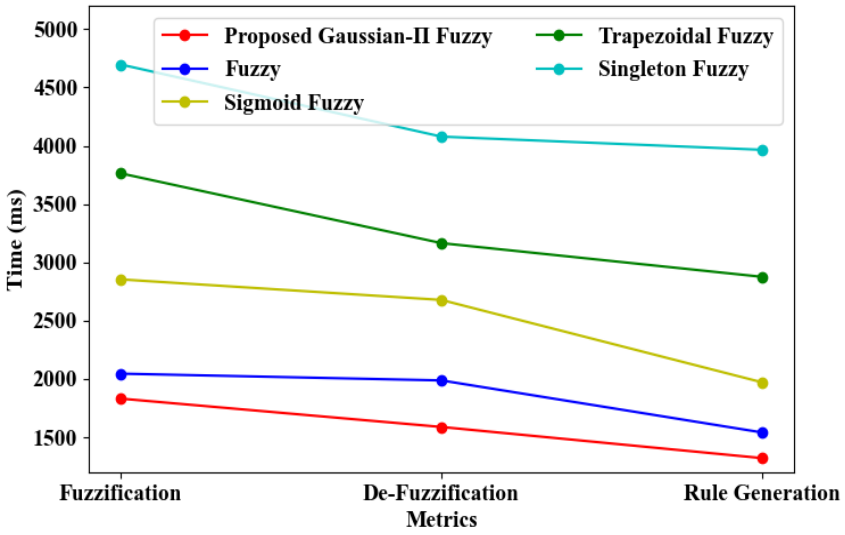
Figure 3: (a) Performance assessment and (b) Training Time of the proposed work

The performance assessment and TT of the proposed TL-IML3CSTM are compared with traditional LSTM, Bidirectional-LSTM, Recurrent Neural Network (RNN), and Artificial Neural Network (ANN) techniques as shown in Figure 3 (a) and (b). When compared to prevailing techniques, the proposed work attained high ADA (99.32%), precision (99.04%), sensitivity (98.93%), f-measure (99.27%), and specificity (98.92%) with minimum TT (2856ms).

This is because, TL-IMSRUA and 2L-CSUV ensure the balanced cell state updates to speed up the training process and to improve the LSTM's performance, thus outperforming the prevailing techniques.



(a)



(b)

Figure 4: Comparative Analysis based on (a) Severity Level Prediction and (b) Performance validation of the proposed GΠ-Fuzzy

Figure 4 (a) and (b) describe the enhanced performance of the proposed GΠ-Fuzzy when compared with traditional Fuzzy, Sigmoid Fuzzy, Trapezoidal Fuzzy, and Singleton Fuzzy techniques. This enhanced performance is due to GΠ involvement, which automatically determines membership parameters by reducing biases from subjective design. Hence, the proposed GΠ-Fuzzy had a high SL-prediction of 99.03% with minimum Fuzzification Time (FT= 1832ms), Defuzzification Time (DFT= 1589ms), and Rule Generation Time (RGT= 1322ms). However, the prevailing

techniques had a low average SL prediction of 95.68% with lower FT, DFT, and RGT values, thus degrading the entire work performance.

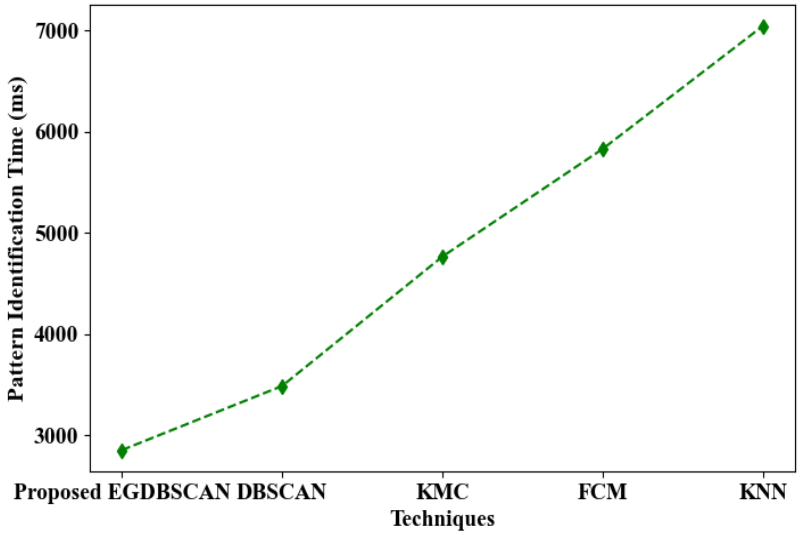


Figure 5: Pattern Identification Time

Pattern Identification Time (PIT) of the proposed EGDBSCAN is compared with traditional DBSCAN, K-Means Clustering (KMC), Fuzzy C Means (FCM), and K-Nearest Neighbor (KNN) techniques as depicted in Figure 5. Here, EG optimizes the DBSCAN's MinPts by systematically evaluating the cluster metrics for efficient noise reduction. Thus, the proposed EGDBSCAN had minimum PIT (984ms), while the prevailing DBSCAN, KMC, FCM, and KNN had maximum PIT values of 1053ms, 1286ms, 1371ms, and 1592ms, respectively, thereby slowing down the overall performance.

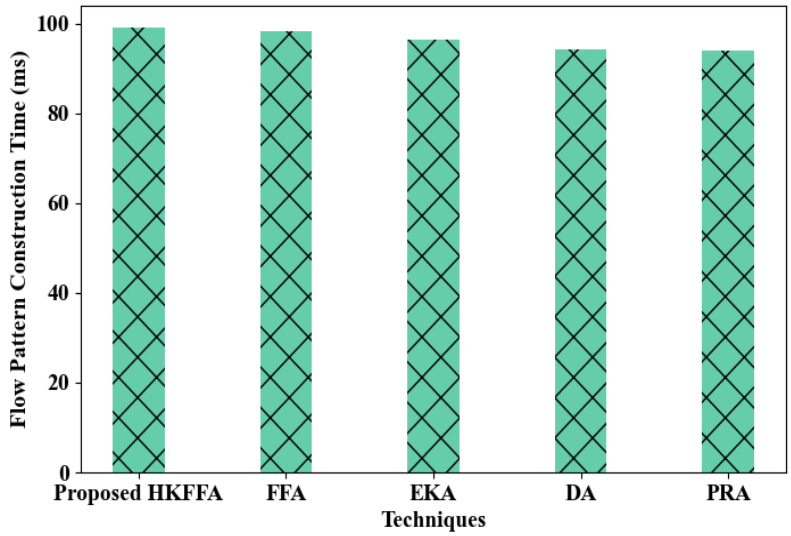
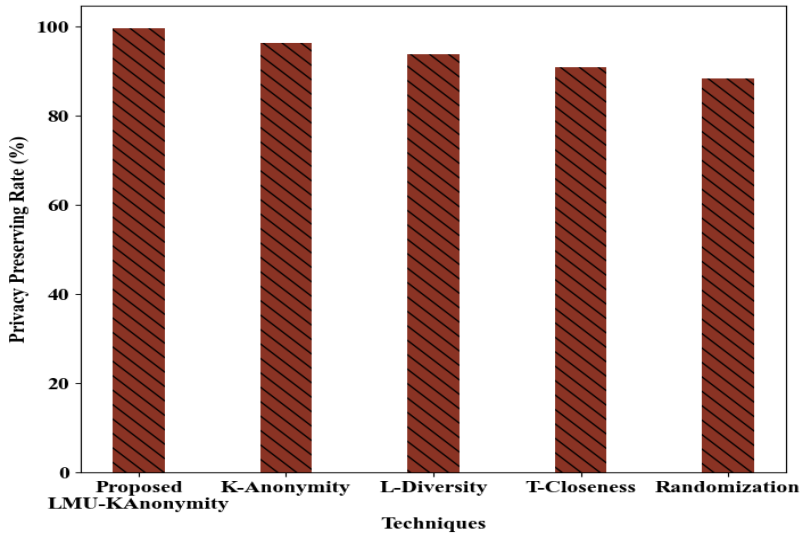
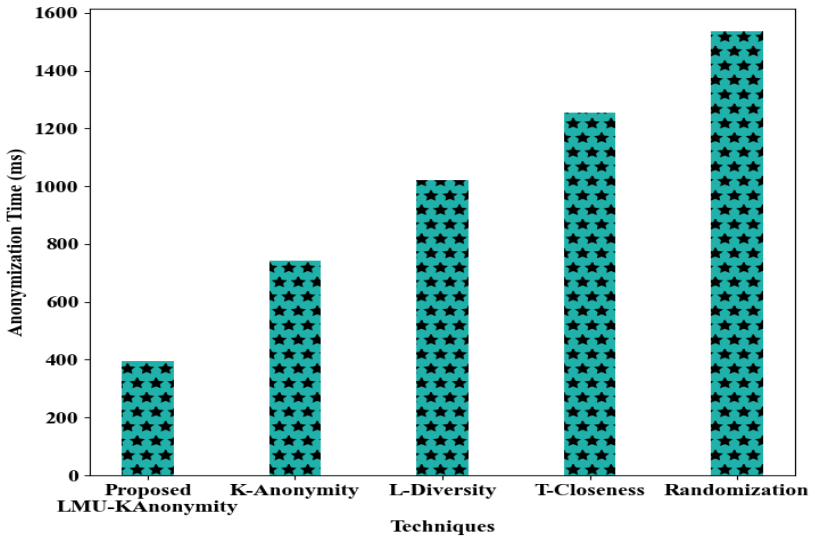


Figure 6: Flow Pattern Construction Time

Figure 6 shows the Flow Pattern Construction Time (FPCT) of the proposed HKFFA and the traditional FFA, Edmonds-Karp Algorithm (EKA), Dinic's Algorithm (DA), and Push-Relabel Algorithm (PRA) techniques. Here, HD optimizes FFA by prioritizing the shortest path and considering all paths within the minimum FPCT of 1021ms. But, when compared to the proposed technique, the prevailing FFA, EKS, DA, and PRA attained minimum FPCT of 1576ms, 2343ms, 2685ms, and 2931ms, respectively. Thus, the prevailing techniques had reduced ADA.



(a)



(b)

Figure 7: (a) Privacy Preservation Rate and (b) Anonymization Time

Table 2: Information Loss validation

Techniques	Information Loss

Proposed LMU KAnonymity	96
K-Anonymity	363
L-Diversity	797
T-Closeness	1052
Randomization	1232

Privacy Preservation Rate (PPR), Anonymization Time (AT), and Information Loss (IL) of the proposed LMU-KAnonymity are compared with the existing K-Anonymity, L-Diversity, T-Closeness, and Randomization techniques. As shown in Figure 7 (a) and (b), the proposed LMU-KAnonymity attained a high PPR (99.68%) with a minimum AT (395ms). This enhancement is due to the usage of the LMU technique that minimizes IL to 96 as shown in Table 2 by optimizing generalization levels to ensure privacy and maintain data utility. However, the prevailing techniques had low PPR, high IL, and maximum AT values, thus causing inefficiencies in data security.

Table 3: Comparative Analysis with Related Works

Study	Techniques	ADA (%)
Proposed Work	TL-IML3CSTM	99.32
(Rao & Babu, 2023)	GAN	98
(Zhu & Liu, 2024)	LR	97.05
(Rahman et al., 2024)	GAN	90
(Moore et al., 2023)	MLP	73.44
(Siddiqi & Pak, 2022)	CNN	97.75

Table 3 depicts the performance of the proposed and related works. The proposed TL-IML3CSTM achieved a high ADA of 99.32% by using TL-based IMSRUA and 2L-CSUV to speed up LSTM training and enhance efficiency with initialized weights. In contrast, (Rao & Babu, 2023) and (Rahman et al., 2024) used GAN techniques with a lower average ADA of 94%, while (Zhu & Liu, 2024), (Moore et al., 2023), and (Siddiqi & Pak, 2022) used Logistic Regression (LR), MultiLayer Perceptron (MLP), and CNN techniques with low average ADA of 89.41%. This is because the prevailing techniques had scalability and overfitting issues, thus degrading the overall performance.

5. CONCLUSION

This research effectively categorized the generic attacks and normal data for enhanced ID in the network. At first, the data was collected, preprocessed, and then balanced for further accurate analysis. The behavioral patterns and flow patterns were then identified and constructed within 984ms and 1021ms, respectively. After that, the generic attacks were detected with 99.34% ADA. Then, from the attacked data, the SL was predicted based on rules within 1322ms RGT. Finally, the required flow patterns were mapped with possible flow patterns for accurate attack detection in networks.

Future Recommendation

Despite accurate attack prediction, various techniques will be implemented in the future to categorize the generic attack types for more enhanced performance.

REFERENCES

Dataset Links:

1. https://www.kaggle.com/datasets/saidasalloum/network-flows
2. https://www.kaggle.com/datasets/mrwellsdavid/unsw-nb15?select=NUSW-NB15_features.csv
3. Ahmed, N., Ngadi, A. Bin, Sharif, J. M., Hussain, S., Uddin, M., Rathore, M. S., Iqbal, J., Abdelhaq, M., Alsaqour, R., Ullah, S. S., & Zuhra, F. T. (2022). Network Threat Detection Using Machine/Deep Learning in SDN-Based Platforms:

A Comprehensive Analysis of State-of-the-Art Solutions, Discussion, Challenges, and Future Research Direction. *Sensors (Basel, Switzerland)*, *22*(20), 1–34. https://doi.org/10.3390/s22207896

4. Alotaibi, A., & Rassam, M. A. (2023). Adversarial Machine Learning Attacks against Intrusion Detection Systems: A Survey on Strategies and Defense. *Future Internet*, *15*(2), 1–34. https://doi.org/10.3390/fi15020062
5. Disha, R. A., & Waheed, S. (2022). Performance analysis of machine learning models for intrusion detection system using Gini Impurity-based Weighted Random Forest (GIWRF) feature selection technique. *Cybersecurity*, *5*(1), 1–22. https://doi.org/10.1186/s42400-021-00103-8
6. Govindarajan, Vijay, and Junaid Hussain Muzamal. "Advanced cloud intrusion detection framework using graph based features transformers and contrastive learning." *Scientific Reports* 15.1 (2025): 20511.
7. Khan, Hasim, et al. "A secure and efficient deep learning-based intrusion detection framework for the internet of vehicles." *Scientific Reports* 15.1 (2025): 12236.
8. Halbouni, A., Gunawan, T. S., Habaebi, M. H., Halbouni, M., Kartiwi, M., & Ahmad, R. (2022). CNN-LSTM: Hybrid Deep Neural Network for Network Intrusion Detection System. *IEEE Access*, *10*, 99837–99849. https://doi.org/10.1109/ACCESS.2022.3206425
9. He, K., Zhang, W., Zong, X., & Lian, L. (2024). Network Intrusion Detection Based on Feature Image and Deformable Vision Transformer Classification. *IEEE Access*, *12*, 44335–44350. https://doi.org/10.1109/ACCESS.2024.3376434
10. Hnamte, V., Nhung-Nguyen, H., Hussain, J., & Hwa-Kim, Y. (2023). A Novel Two-Stage Deep Learning Model for Network Intrusion Detection: LSTM-AE. *IEEE Access*, *11*, 37131–37148. https://doi.org/10.1109/ACCESS.2023.3266979
11. Jiang, K., Wang, W., Wang, A., & Wu, H. (2020). Network Intrusion Detection Combined Hybrid Sampling with Deep Hierarchical Network. *IEEE Access*, *8*(3), 32464–32476. https://doi.org/10.1109/ACCESS.2020.2973730
12. Moore, S. J., Cruciani, F., Nugent, C. D., Zhang, S., Cleland, I., & Sani, S. (2023). Deep learning for network intrusion: A hierarchical approach to reduce false alarms. *Intelligent Systems with Applications*, *18*, 1–13. https://doi.org/10.1016/j.iswa.2023.200215
13. Nabi, F., & Zhou, X. (2024). Enhancing intrusion detection systems through dimensionality reduction: A comparative study of machine learning techniques for cyber security. *Cyber Security and Applications*, *2*, 1–8. https://doi.org/10.1016/j.csa.2023.100033
14. Li, Min, Yuansong Qiao, and Brian Lee. "Multi-view intrusion detection framework using deep learning and knowledge graphs." *Information* 16.5 (2025): 377.
15. Park, C., Lee, J., Kim, Y., Park, J. G., Kim, H., & Hong, D. (2023). An Enhanced AI-Based Network Intrusion Detection System Using Generative Adversarial Networks. *IEEE Internet of Things Journal*, *10*(3), 2330–2345. https://doi.org/10.1109/JIOT.2022.3211346
16. Rahman, S., Pal, S., Mittal, S., Chawla, T., & Karmakar, C. (2024). SYN-GAN: A robust intrusion detection system using GAN-based synthetic data for IoT security. *Internet of Things (Netherlands)*, *26*, 1–13. https://doi.org/10.1016/j.iot.2024.101212
17. Rao, Y. N., & Babu, K. S. (2023). An Imbalanced Generative Adversarial Network-Based Approach for Network Intrusion Detection in an Imbalanced Dataset. *Sensors (Basel, Switzerland)*, *23*(1), 1–26. https://doi.org/10.3390/s23010550
18. Rustam, F., Aljedaani, W., Elsayed, M. S., & Jurcut, A. D. (2024). FAMTDS: A novel MFO-based fully automated malicious traffic detection system for multi-environment networks. *Computer Networks*, *251*, 1–31. https://doi.org/10.1016/j.comnet.2024.110603
19. Samantaray, M., Barik, R. C., & Biswal, A. K. (2024). A comparative assessment of machine learning algorithms in the IoT-based network intrusion detection systems. *Decision Analytics Journal*, *11*, 1–16. https://doi.org/10.1016/j.dajour.2024.100478
20. Siddiqi, M. A., & Pak, W. (2022). Tier-Based Optimization for Synthesized Network Intrusion Detection System. *IEEE Access*, *10*, 108530–108544. https://doi.org/10.1109/ACCESS.2022.3213937
21. Sworna, Z. T., Mousavi, Z., & Babar, M. A. (2023). NLP methods in host-based intrusion detection systems: A systematic review and future directions. *Journal of Network and Computer Applications*, *220*, 1–29. https://doi.org/10.1016/j.jnca.2023.103761
22. Sharma, Nikhil, and Prashant Giridhar Shambharkar. "Multi-attention DeepCRNN: an efficient and explainable intrusion detection framework for Internet of Medical Things environments: N. Sharma, PG Shambharkar." *Knowledge and Information Systems* 67.7 (2025): 5783-5849.
23. Vitorino, J., Praca, I., & Maia, E. (2023). SoK: Realistic adversarial attacks and defenses for intelligent network intrusion detection. *Computers and Security*, *134*, 1–10. https://doi.org/10.1016/j.cose.2023.103433
24. Yu, H., Kang, C., Xiao, Y., & Yang, Y. (2023). Network Intrusion Detection Method Based on Hybrid Improved Residual Network Blocks and Bidirectional Gated Recurrent Units. *IEEE Access*, *11*, 68961–68971. https://doi.org/10.1109/ACCESS.2023.3271866
25. Zhu, J., & Liu, X. (2024). An integrated intrusion detection framework based on subspace clustering and ensemble learning. *Computers and Electrical Engineering*, *115*, 1–22. https://doi.org/10.1016/j.compeleceng.2024.109113
26. Zoppi, T., Gazzini, S., & Ceccarelli, A. (2024). Anomaly-Based Error and Intrusion Detection in Tabular Data: No DNN Outperforms Tree-based Classifiers. *Future Generation Computer Systems*, 1–21. https://doi.org/10.1016/j.future.2024.06.051