Grenze International Journal of Engineering and Technology, June Issue

A Machine Learning-based Intrusion Detection Framework for Secure Communication in Next-Generation Computer Networks

Pushpavathi[1], Gnaneswari G[2] and Mariyan Richard A[3]
[1-3]Department of Computer Applications, Atria Institute of Technology, Autonomous Bangalore, Karnataka, India
Email: pushpavathip1999@gmail.com, gnaneswari.g@cmrit.ac.in, mariyanrich01@gmail.com

***Abstract*— The rapid expansion of computer networks and digital communication systems has increased the risk of cyber-attacks and network intrusions. Traditional intrusion detection systems rely mainly on predefined signatures to identify threats, which limits their ability to detect new and evolving attacks. Machine learning techniques provide an effective solution by enabling intelligent analysis of network traffic and identifying abnormal patterns. This paper presents a machine learning-based intrusion detection framework designed to enhance security in next-generation communication networks. The proposed system analyzes network traffic data and classifies activities as normal or malicious using supervised machine learning algorithms. The framework includes stages such as data preprocessing, feature extraction, model training, and attack detection. Experimental evaluation is conducted using benchmark network security datasets to assess the performance of different machine learning algorithms. The results indicate that machine learning models significantly improve the accuracy and efficiency of intrusion detection compared with traditional methods. The proposed framework can contribute to strengthening secure communication in modern network environments.**

***Keywords*— Intrusion Detection System, Machine Learning, Network Security, Communication Networks, Cyber Attack Detection.**

I. INTRODUCTION

These days, almost every digital task runs through computer networks. From buying things online to storing files far away, they make it possible. With more people depending on them, keeping connections safe matters more than ever. When hackers flood a system, it slows down or stops completely. Getting into systems without permission is another common problem. Bad software sneaks in, takes control, or steals details. Information meant to stay private often ends up exposed. Breaks like these hurt both individuals and organizations.Old-school security tools keep an eye on data moving through networks, spotting odd behavior. Because they rely on preset attack fingerprints saved in a library, they can catch familiar dangers fast. Yet when unfamiliar threats show up - ones never seen before - they usually slip right past.One way around these limits? Scientists turned to teaching machines how to spot digital threats. These smart systems study massive flows of network information, noticing habits tied to harmful actions. When odd behaviors pop up - patterns that feel out of place - the models flag them fast. Strange spikes, weird timing, silent gaps - each clue helps catch old tricks and brand-new ones alike.

Grenze ID: 01.GIJET.12.2.637_26

A fresh method pops up here - using machine learning to guard future networks better. Instead of old rule-based tricks, it leans on trained models that sort network flow by behavior. One after another, data points get checked, not just blocked on keywords. Supervised techniques take center stage, spotting odd patterns before harm spreads. Learning happens from labeled examples, turning past breaches into lessons. Traffic either looks normal or raises flags, based on what the system absorbed earlier.

II. Related Work

What started as a quiet corner of network safety soon drew serious attention. Back then, most tools hunted intruders using fixed rules or known patterns. Instead of learning, they matched clues already written down. Staying sharp meant constant refreshes, otherwise blind spots grew fast. New tricks in attacks often slipped past if not added to the list.

Now machines spot hackers better thanks to smarter software. Some clever code sorts through data, spotting odd behavior without prior examples. Instead of rules, patterns guide recognition - each clue builds a clearer picture. Earlier attempts struggled, yet newer methods adapt on their own. Code like forest models splits choices into tiny yes-or-no paths. Even brain-inspired networks mimic how humans learn from mistakes. Traffic flows constantly; hidden traps wait inside. Sharp tools catch what older filters missed completely. One step ahead means fewer surprises later. Learning never stops when threats keep changing shape.

Some research has tested machine learning for spotting intrusions using open data like KDD Cup 99, NSL-KDD, or CICIDS. While older techniques struggle to keep up, these tests show that algorithms often detect threats more accurately. Though not perfect, they adjust better over time when faced with new attack patterns. Results across multiple trials support this trend without depending on a single source. Each dataset brings different challenges, yet the models still perform consistently well. Because of varied test conditions, confidence in their effectiveness grows slightly with each study. Still, every approach shows limits under certain network behaviors. Even so, the edge over conventional systems remains noticeable in most cases. Over time, small improvements add up - without sudden breakthroughs. Despite differences in design, the outcome leans toward stronger performance overall.

Even with progress, spotting real threats without mistakes still trips up systems at big scales. To tackle this, the new setup uses a smart machine method that learns patterns to catch intrusions faster.

III. Proposed Intrusion Detection Framework

Whatever sneaky moves happen online, this system keeps an eye on data flows to catch them. Not just watching, it digs into patterns using smart algorithms that learn over time. Instead of guessing, it sorts activity by spotting what fits regular use versus what stands out. When something unusual shows up, the method flags it without needing exact rules ahead of time. Learning from past examples helps tell harmless quirks apart from real dangers.

Starting off, the setup includes multiple core steps - gathering data comes first, followed by cleaning it up before turning raw inputs into meaningful traits. After that, a model learns patterns using those refined features, ending with spotting intrusions effectively. A full view of how these pieces fit together appears in Fig. 1, showing the structure behind the suggested approach.

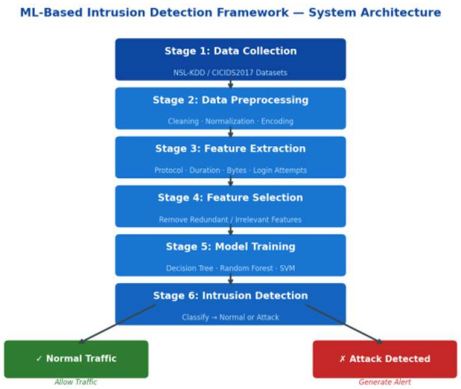

Fig. 1. System architecture of the proposed ML-based intrusion detection framework.

A. Data Collection

Gathering information kicks off the process - and it matters more than anything else when building a solid intrusion detection setup. Machine learning works better when fed rich, varied examples, nothing less. From this angle, traffic details come straight from well-known public collections, the kind researchers often turn to for spotting intrusions.

A widely adopted test collection, the NSL-KDD data serves as a core reference point. Addressing flaws found in the earlier KDD Cup 99 version - like repeated entries and uneven attack representation - it offers improved balance.

Featuring various kinds of digital threats, it groups incidents into distinct classes

A flood hits system limits when denial of service strikes. Resources choke under pressure from overloaded traffic. Capacity vanishes fast during these surges. Performance drops as flow becomes too much.

- Probe attacks - scan networks for vulnerabilities.
- Remote to Local (R2L) - unauthorized remote access.

Getting from user level to root means gaining full control. One wrong step can expose the system. Access shifts when limits are bypassed. Power changes hands without warning. Normal barriers fail under pressure. A quiet breach leads upward. Control spreads where it should not.

Used too is the CICIDS2017 dataset, which holds actual recent attack examples like brute-force attempts alongside DDoS incidents plus various web exploits. While not perfect, it covers a broad range of hostile actions seen today across networks.

B. Data Preprocessing

Cleaning up network traffic data happens first, preparing it for machine learning by turning messy inputs into something organized.

The full steps taken are shown in Fig. 2, walking through how the system handles incoming information before any analysis begins.

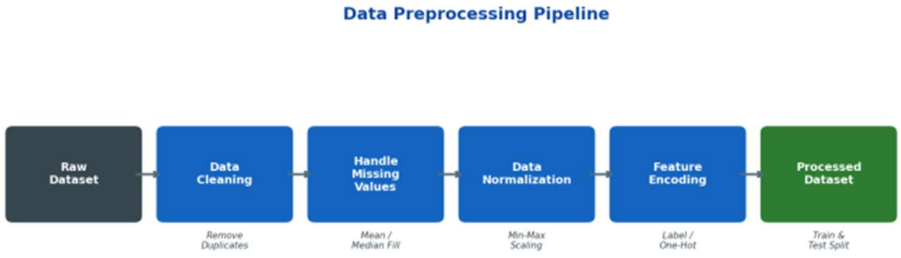

Fig. 2. Data preprocessing pipeline showing the sequence of transformation stages.

Outliers get tossed out so models aren't misled. Unneeded columns vanish, cutting down processing load without slowing insight. Noise fades where clarity helps more.

When data points go missing, they get filled in by borrowing typical values - like averages or most common entries - from the rest of that column. Sometimes it uses a middle number instead, depending on how things are spread out across the set.

Starting from zero, min-max scaling adjusts every feature so values fit neatly within a span from 0 to 1. Because of this shift, no single feature dominates simply by size. Each one pulls its weight during training. Through rescaling, balance emerges across inputs. That balance helps the model learn more evenly. Without it, larger numbers might shout too loud. Here, everything speaks at the same volume.

Turning names like protocol or service type into numbers happens through methods such as labeling each category or creating separate flags for every option. Numbers stand in for words so machines can process them without confusion later on. Each unique name gets its own code instead of staying as text. This step changes how data appears but keeps meaning intact underneath. Labeling takes less space while flag-style splits offer more detail by default.

C. Feature Extraction and Selection

Picking out key details starts by pulling useful traits from basic network activity records. Some of these details matter more than others when spotting threats. Unneeded or repetitive ones might confuse the system instead of helping it. The method used here appears in Fig. 3, showing how certain aspects get chosen over others.

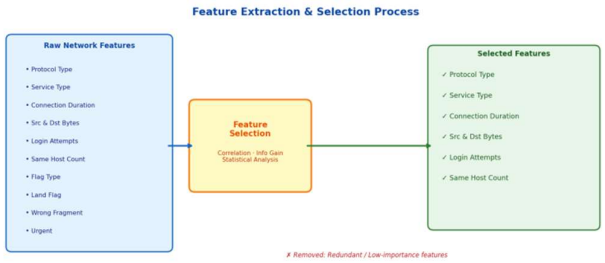

Fig. 3. Feature extraction and selection process showing raw features filtered to the most informative subset.

Commonly extracted network traffic features include:
Choose between TCP, yet also consider UDP or perhaps ICMP instead.
HTTP shows up first. Following that comes FTP. Then there is SMTP tagging along behind.
Lasted how long the device stayed linked. Session span measured in minutes or hours. Kept track of active connection time. Ran until user disconnected. Timed from start to finish. Measured intervals between log-in and log-out.
Bytes moving out, those arriving - measure how much travels. Data flow tracked by what leaves, matched against what lands.
Count how often login tries fail.
How many times linked to one host.
Not every feature stays - some get dropped once tested through numbers, links between variables, or how much each one adds to understanding. Picking which ones remain depends on what they reveal when checked by these approaches. Each method acts like a filter, keeping only those that matter most. What makes the cut is based less on guesswork, more on measured impact. The weak contributors fade out, leaving behind just the strong signals.

D. Model Training

Once the data gets cleaned and key features picked, it splits - eighty percent goes to training, twenty to testing. Not one, not two, but three supervised machine learning models step in to learn from the training chunk
A branching path of choices shapes how data gets sorted. One split leads to another based on specific traits. Rules guide movement from top to bottom. Every stop along the way checks one attribute. The journey ends at a final label. Outcomes appear only after all steps finish.
A forest of separate tree models works together when outcomes vote for a result. Each one grows on its own, shaped by different data slices. When predictions come due, most votes win. This setup tends to hold up better under noise. Mistakes balance out across branches.
A line drawn where space between groups is widest - that's what SVM finds. Where dimensions grow tall, it still works well because kernels handle twists in data shapes. Instead of straight splits, these tools bend the rules to fit messy patterns.

E. Intrusion Detection and Classification

Right off the bat, network data flows through a preprocessing step during detection. Following that, feature extraction happens - identical to what was used while training. Afterward, predictions roll out: normal behaviour or signs of intrusion, thanks to the already-trained model making the call.
Alerts pop up the moment something sneaks into the system, notifying whoever manages the network. To judge how well each setup performs, results get sorted through a grid that counts hits, correct rejections, false alarms, and missed detections.

IV. Evaluation Metrics

From the confusion matrix, standard metrics help measure how well the new intrusion detection system works. Performance shows up through numbers pulled straight out of that table. Each score reflects real outcomes without extra assumptions. What comes out depends on those initial comparisons made during testing. Results take shape only after every match and mismatch gets counted.
What you get is how many guesses were right out of all tries. A correct call means the prediction matched reality every single time it counted
Accuracy equals true positives plus true negatives divided by sum of all outcomes

Finding attacks without mistakes shows what you get every time the system says there's a threat. What matters most? The hits versus the false alarms, each call counted straight up
Precision Equals True Positives Divided By True Positives Plus False Positives
Finding real threats is what this metric shows - how often true attacks get spotted. It tracks success in catching genuine incidents, nothing more
Recall Equals True Positives Divided by the Sum of True Positives and False Negatives
F1-Score mixes Precision with Recall through a kind of average that treats both equally.
F1 equals two multiplied by precision times recall, divided by precision added to recall
False Positive Rate counts good traffic marked bad
FPR Equals False Positives Divided by Sum of False Positives and True Negatives

V. Experimental Results and Performance Analysis

Tests ran on standard cybersecurity data, split so four-fifths helped models learn while one-fifth checked their accuracy. One after another, three smart prediction methods took turns: a branching logic system, a boundary-drawing technique, then a crowd of tiny decision trees voting together.

A. Confusion Matrix Analysis

Few mistakes show up when spotting break-in tries - most get caught without triggering unnecessary alerts. That outcome comes from using the top method tested so far: Random Forest.

Table I. Confusion Matrix — Random Forest

Actual / Predicted	Attack	Normal
Attack	950 (TP)	25 (FN)
Normal	30 (FP)	995 (TN)

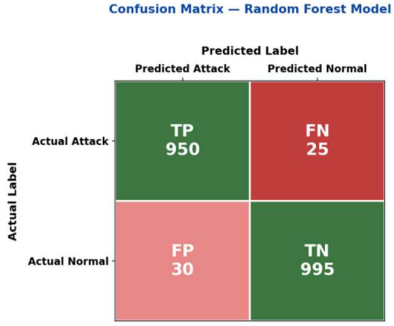

Fig. 4. Visual confusion matrix for the Random Forest classifier.

B. Performance Comparison

Table II summarizes the classification performance of all three algorithms across the four key metrics. The Random Forest algorithm achieved the highest performance, demonstrating the advantage of ensemble learning methods.

Table II. Performance Comparison of ML Algorithms

Algorithm	Accuracy	Precision	Recall	F1 Score
Decision Tree	91.2%	90.5%	89.8%	90.1%
Support Vector Machine	93.6%	92.8%	92.3%	92.5%
Random Forest	96.4%	95.7%	95.2%	95.4%

The results confirm that Random Forest (96.4% accuracy) outperforms SVM (93.6%) and Decision Tree (91.2%). The ensemble mechanism aggregates multiple decision trees, reducing variance and improving generalization.

C. Performance Comparison Graph

Fig. 5 provides a visual comparison of all four-performance metrics across the three algorithms, clearly illustrating the superiority of the Random Forest model.

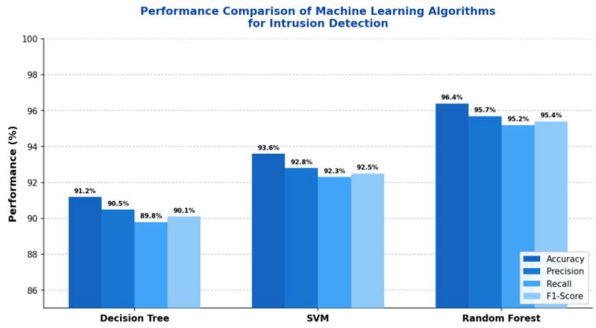

Fig. 5. Bar chart comparing Accuracy, Precision, Recall, and F1-Score of the three ML algorithms.

VI. Discussion

When tests were done, they showed machines could get much better at spotting network intrusions. Because these systems study how data moves, unusual actions stand out more clearly. With enough examples, learning algorithms adjust themselves to catch digital dangers faster than older methods did.

Not far behind others in testing, one model stood out - Random Forest - with the sharpest results. Its strength comes from stacking many small decisions instead of relying on just one path. Because it pulls together varied outcomes, errors balance out across the group. That mix leads to steadier guesses when facing new data.

Starting off, the SVM method showed solid results in sorting data points correctly. It really shines when working with many features at once, along with tricky separation zones. That said, things can slow down if the dataset becomes too massive to handle easily.

Despite weaker results, the Decision Tree method stays popular because it trains quickly, keeps things clear, while remaining straightforward to follow - helpful whenever tracing choices matters most.

Even so, keeping up with shifting threats still trips systems up. What sticks out is how much data floods in each second, overwhelming current tools. Speed matters more every day, yet most models lag behind live attacks. Fewer mistakes would help - that part cannot be ignored. Work continues, mainly because errors pile up when alerts fire too often.

VII. Conclusion and Future Work

A fresh look at safeguarding future networks came by way of a machine learning approach for spotting intrusions. Through collected traffic information, the system moves step by step - first gathering inputs, then cleaning them up before pulling out key traits. Following that, patterns take shape during training phases where the model learns what to flag. Detection kicks in once the setup recognizes odd behavior similar to past threats. Each stage connects not just sequentially but with purpose, shaping how alerts emerge later on.

From the start, three supervised models - Decision Tree, SVM, along with Random Forest - took shape through testing on standard data collections. It turned out that Random Forest stood apart, hitting a precision mark of 96.4%. This outcome underlined how combining multiple models can sharpen threat identification.

What stands out is how crucial it becomes to weave machine learning into tools that guard networks. Instead of relying only on fixed rules, these smart systems study behavior from live data flows - spotting familiar threats along with new ones others might miss. A shift like this changes what detection means entirely.

One path ahead dives into deep learning, using tools like CNNs alongside RNNs to catch tricky attack behaviors more effectively. Live traffic analysis could become possible through systems built for real-time threat spotting. Another angle mixes different machine learning methods into blended models. These combined approaches might link up with cloud platforms or edge devices. Scaling up across vast networks becomes easier when these pieces connect.

References

[1] IEEE. (2024). Deep learning-based network traffic classification for intelligent communication systems. IEEE Access, 12, 21534–21547.

[2] IEEE. (2024). Machine learning approaches for intelligent network traffic prediction in next-generation networks. IEEE Transactions on Network and Service Management, 21(1), 456–468.

[3] IEEE. (2024). Encrypted network traffic classification using deep learning techniques. IEEE Communications Letters, 28(3), 612–616.
[4] IEEE. (2023). Intelligent anomaly detection in network traffic using hybrid machine learning models. IEEE Access, 11, 103245–103258.
[5] IEEE. (2023). Machine learning-based network traffic analysis for software-defined networks. IEEE Transactions on Network and Service Management, 20(3), 2541–2553.
[6] IEEE. (2023). Network traffic classification using convolutional neural networks for communication networks. IEEE Communications Letters, 27(9), 2391–2395.
[7] IEEE. (2023). Artificial intelligence-based network traffic prediction for communication systems. IEEE Systems Journal, 17(4), 6125–6136.
[8] IEEE. (2022). A deep learning framework for intelligent network traffic analysis. IEEE Access, 10, 84521–84533.
[9] IEEE. (2022). Machine learning techniques for network anomaly detection in communication networks. IEEE Transactions on Information Forensics and Security, 17, 3148–3160.
[10] IEEE. (2022). Intelligent network traffic classification using ensemble machine learning models. IEEE Communications Letters, 26(8), 1875–1879.
[11] IEEE. (2022). Traffic analysis and prediction in communication networks using artificial intelligence. IEEE Systems Journal, 16(2), 2764–2775.
[12] IEEE. (2021). Network traffic classification using machine learning for cybersecurity applications. IEEE Access, 9, 112789–112801.
[13] IEEE. (2021). Intelligent intrusion detection using machine learning in network environments. IEEE Transactions on Network and Service Management, 18(3), 2687–2699.
[14] IEEE. (2020). Machine learning-based anomaly detection for network traffic monitoring. IEEE Communications Surveys & Tutorials, 22(3), 1763–1785.
[15] IEEE. (2019). A survey on machine learning techniques for network traffic classification. IEEE Communications Surveys & Tutorials, 21(4), 3483–3519.