Svedberg Open
DISSEMINATION OF KNOWLEDGE

International Journal of Artificial Intelligence and Machine Learning

Publisher's Home Page: https://www.svedbergopen.com/



Research Paper Open Access

A Hybrid CNN-LSTM Deep Learning Model for Intrusion Detection in Network Security

Viral Kumar Gupta[1], Dr. Sunil Kumar[2]

[1]Research Scholar; Department of Computer Applications, School of Computer Science & Applications, IFTM University, Moradabad (UP) India, E-mail Id: viralkumargupta12@gmail.com , Orcid Id: https://orcid.org/0009-0003-6945-6057
[2]Assistant Professor; Department of Computer Applications, School of Computer Science & Applications, IFTM University, Moradabad (UP) India, E-mail Id: skiftm@gmail.com

ABSTRACT
As cyber threats keep growing in scale and sophistication, mostly because cloud computing got adopted fast, plus the Internet of Things, SDN, and 5G all being used more, it made the old signature and rule based Intrusion Detection Systems (IDS) show their limits. In other words, these systems can't really spot zero-day attacks well, they throw a lot of false alarms, and they don't adapt nicely to the constantly changing behavior of network traffic. This work puts forward a hybrid deep learning model, somehow mixing Convolutional Neural Networks (CNN) with Long Short-Term Memory (LSTM) networks, so it can learn both the spatial side and the temporal side at the same time, from network data. The CNN part does the heavy lifting for local spatial patterns, and also captures feature correlations from preprocessed traffic, whereas the LSTM part focuses on sequential, time dependent relationships, to uncover attack behaviors that evolve over time. For regularization, it uses L2 weight penalties along with multi stage dropout, so over fitting is reduced and generalization improves, bit better. The model is trained and tested using the benchmark NSL-KDD and KDDCup99 datasets, covering Normal traffic plus Denial-of-Service (DoS), Probe, Remote-to-Local (R2L), and User-to-Root (U2R) categories too. Experimental findings indicate that the proposed hybrid CNN-LSTM framework consistently beats classical machine learning classifiers like Logistic Regression, Naive Bayes, K-Nearest Neighbors, and Decision Tree, Ad a Boost, Random Forest, and SVM. It also beats deep neural network variants, and several earlier hybrid architectures (TFTC, MDPCA-DBNs, Siam-IDS, I-ELM, HFR-MLR). What really stands out is that it reaches a True Positive Rate of 89.45% on the historically difficult U2R class, while keeping the false positive rate around only 0.20% . This suggests the architecture improves minority class detection, in a pretty direct way, and without sacrificing too much reliability.

Keywords: Intrusion Detection System, Convolutional Neural Network, Long Short-Term Memory, Hybrid Deep Learning, Network Security, NSL-KDD, Class Imbalance, Regularization

1. INTRODUCTION

Network infrastructures kind of underpin nearly every critical service, from banking and healthcare to education, government, and defense. As digital transformation speeds up, organizations are leaning more and more on the confidentiality integrity, and availability of their networks, so cyber security ends up being a top operational priority. At the same time, cloud computing keeps expanding, the Internet of Things (IoT) is basically everywhere, Software-Defined Networking (SDN) is reshaping how routing actually happens, and 5G adds even more capacity. All of this has pushed an exponential jump in both the volume and the sheer complexity of network traffic. Sure these technologies improve connectivity and efficiency, but they also enlarge the attack surface , so networks are increasingly exposed to threats that feel more and more advanced, like malware, unauthorized access, phishing, ransom ware, and data breaches. Back then, Intrusion Detection Systems (IDS) leaned pretty heavily on rule based or signature driven techniques, where traffic gets flagged if it matches a known set of hostile behaviors from a database. Those methods can work well against threats that were already cataloged, but they often miss zero-day attacks, because no signature is available yet. They also need continual manual updates to stay useful, and they often have trouble keeping up with high-velocity traffic, in real time especially. Since these systems typically run on fixed thresholds, they don't really adapt to changing network behavior. The result is messy, either excessive false positives, meaning normal traffic gets misread as malicious, and or false negatives, meaning real attacks slip through because they're not detected. In both cases, the network security posture gets weakened. To overcome these limitations, intelligent and

adaptive detection methods, built with machine learning and deep learning, have started to become more prominent. Instead of leaning only on predefined rules, these models learn discriminative signals from the data itself, and it kind of shifts IDS design from a reactive to a proactive, more predictive style. Traditional machine learning methods like Support Vector Machines (SVM), Decision Trees (DT), K-Nearest Neighbors (K-NN), and Artificial Neural Networks (ANN) have improved the ability to detect known attack classes, however they're still touchy about high-dimensional feature spaces, they can be computationally expensive when scaled up, and they are especially vulnerable to the tough class imbalance issue that shows up in real-world traffic. In practice benign samples massively outnumber rare attack types such as Remote-to-Local (R2L) and User-to-Root (U2R) intrusions. Deep learning then offers another step forward, because it can automatically learn layered feature representations, which lowers the need for manual feature engineering in the first place. Convolutional Neural Networks (CNN) are especially useful for pulling out spatial correlations in structured traffic, while Long Short-Term Memory (LSTM) networks focus on temporal dependencies in sequential data. Since network traffic holds both spatial structure, meaning correlations within a connection, and temporal structure, meaning patterns unfold across connections over time, a combined CNN-LSTM setup feels like a natural match. CNN layers can extract the relevant spatial features, and the LSTM part learns how these features change over time, so the system can catch isolated anomalies, but also slow, stealthy attack campaigns, without missing them.

This paper makes the following contributions:

- It reviews existing machine learning and deep learning approaches to intrusion detection, identifying their strengths, weaknesses, and open challenges.
- It proposes a hybrid CNN-LSTM architecture that jointly exploits spatial and temporal traffic features, combined with L2 regularization and multi-stage dropout to control over fitting.
- It evaluates the proposed model on the NSL-KDD and KDDCup99 benchmark datasets across five traffic categories (Normal, DoS, Probe, R2L, U2R), with particular attention to minority-class detection, a persistent weakness of prior methods.
- It benchmarks the proposed model against thirteen baseline methods spanning classical machine learning, deep neural networks, and existing hybrid architectures.

The rest of this paper organized as follows, Section 2 reviews related work on machine learning and deep learning based intrusion detection. Then Section 3 gives background on CNN and LSTM architectures, plus also over fitting and regularization. After that Section 4 goes into the proposed hybrid CNN-LSTM methodology, with more details than before. Later Section 5 describes the experimental setup along with the dataset. Section 6 then presents and discusses results, basically what we found. Finally Section 7 concludes the paper and, outlines directions for future work too.

2. RELATED WORK

The evolution of Intrusion Detection Systems has sort of been shaped by the increasing complexity in network environments brought in by cloud computing, IoT, and SDN. In that situation, classic rule based systems don't really keep up, so researchers have looked into a whole bunch of intelligent models, hoping to boost detection accuracy, bring down the false alarms, and also manage high-dimensional traffic data that is imbalanced and kind of stubborn to model. The NSL-KDD and CIC-IDS2017 datasets are now basically the go-to benchmarks for testing these systems, letting studies compare results in a more standardized way. A lot of studies also point out that deep learning models tend to outperform conventional machine learning, not just in accuracy but also in detection rate and computational efficiency. One study combining ADASYN-based oversampling with a CNN architecture (AS-CNN) reported a 4.60% accuracy improvement over standard CNN and RNN baselines on NSL-KDD, together with a 10.27% rise in detection rate and reductions in false-alarm rate of 15.58% and 14.57% relative to CNN and RNN respectively, underscoring the importance of explicitly addressing class imbalance. Another line of work converts raw traffic vectors into image-like representations so that CNN architectures originally designed for computer vision can be applied directly to network data; experiments on KDD-CUP99 showed that this representation improves accuracy, detection speed, and false-alarm rate relative to conventional feature-vector approaches.

Hybrid architectures combining CNN with Bi-directional LSTM (BiLSTM) have also been investigated, using CNN to capture spatial structure and BiLSTM to model temporal dependencies in both forward and backward directions; such models achieved accuracies between 77.16% and 83.58% on NSL-KDD and UNSW-NB15, moderate relative to more specialized architectures but demonstrating the value of joint spatio-temporal learning. Graph-based transformations of traffic data have similarly been explored to simplify downstream processing and improve CNN performance, yielding gains in accuracy, false-alarm rate, and computational efficiency on NSL-KDD. Feature-selection techniques based on quantifying each feature's contribution to detection performance have also been proposed, followed by reconstruction of feature correlations to guide hybrid CNN-LSTM models in extracting temporally and spatially relevant patterns; the addition of attention mechanisms has been shown to further improve feature representation. A Hybrid Neural Network (HNN) approach reportedly gave accuracy gains over the baseline deep learning models, around 3.78% on UNSW-

NB15, 1.31% on AWID, 0.21% on CICIDS 2017, and 1.13% on CICIDS 2018, which basically backs up the idea that using feature selection together with temporal-spatial deep learning works in a fairly consistent way across datasets even when the data shifts a bit. Still, class imbalance stays one of the stubbornest problems: benign traffic and the majority attack stream (like DoS) heavily outnumber rare categories such as R2L and U2R, so the model often leans toward the dominant classes. This can look good in overall accuracy, but it quietly hides the fact that critical rare attacks are missed. Adaptive Synthetic Sampling (ADASYN) has been suggested to counter this, by creating synthetic minority-class examples so the training distribution becomes more balanced, and then CNN-based classification accuracy can improve. Another line of work, Few-Shot Learning (FSL), tries to solve it differently by learning from extremely limited labeled samples. For instance, a particular FSL model trained using under 1% of NSL-KDD reached 92.34% accuracy on KDD-Test+ and 85.75% on KDD-Test-21, and it pushed U2R detection from 13% to 81.50% while also lifting R2L detection from 44.41% to 75.93%. Results like that suggest that data-efficient learning can noticeably reduce imbalance-driven blind spots, and not just inflate metrics. Going beyond classic IT networks, deep learning based IDS methods have also been used for industrial control systems (ICS), where a security failure brings serious operational danger. In evaluations on the Safe Water Treatment (SWaT) dataset, models are reported to reach an average accuracy of 0.99, a recall of 0.85, and an F1-score of 0.91, plus the false positive rate is lower, which makes them more suitable for real-world industrial deployments, not just lab testing. Despite this progress, deep learning-based IDS face recurring limitations: high computational cost for training and deployment, dependence on large volumes of labeled data that are often unavailable or privacy-restricted, and limited interpretability due to the black-box nature of deep architectures — a significant concern in security-critical applications. Collectively, the literature indicates that hybrid CNN-LSTM models, augmented with feature selection, attention mechanisms, and imbalance-handling strategies, represent a promising and still-developing direction for building accurate, generalizable, and dependable intrusion detection systems, motivating the approach proposed in this paper.

3. BACKGROUND

Convolutional Neural Networks (CNN)

Convolutional Neural Networks are a kind of supervised deep learning models that were first made for computer vision jobs, like image classification and object detection, and now they are also used a lot for structured data issues, including network intrusion detection. One big reason CNNs often win versus fully connected neural networks is something called weight sharing, so the same filter (kernel) gets used across different parts of the input. Because of that the trainable parameters drop a lot, and the processing stays pretty efficient even when the data is high-dimensional. A typical CNN layout usually comes together from three main layer types, each doing its own part for feature extraction, dimensionality shrinking, and the final decision stage:

• Convolutional Layer — this layer moves learnable filters over the input to catch local cues such as edges, textures, and small-scale feature correlations, and it outputs feature maps that basically show where specific patterns show up. The early stages tend to catch plain, low-level patterns, while the later stages find more and more abstract representations. After each convolution there's usually a non-linear activation, commonly the Rectified Linear Unit (ReLU), applied so the model can represent more involved relationships, instead of staying linear.

$$z_{i,j}^{(k)} = \sum_{m=0}^{K-1}\sum_{n=0}^{K-1} w_{m,n}^{(k)} x_{i+m,j+n} + b^{(k)} \quad (1)$$

$$a_{i,j}^{(k)} = \max\left(0, z_{i,j}^{(k)}\right) \quad (2)$$

• Pooling Layer — it down-samples the feature maps (often by max pooling) to shrink the spatial size, cut the computation cost, and still keep the most relevant signals. This step can also help with over fitting, which is the usual annoyance in deep models.

$$p_{i,j}^{(k)} = \max(a_{2i,2j}, a_{2i,2j+1}, a_{2i+1,2j}, a_{2i+1,2j+1}) \quad (3)$$

• Fully Connected Layer — at the end, it takes the output from the previous convolutional and pooling layers, flattens it into a one-dimensional vector, and then it maps that vector to class probabilities. In practice it works sort of like a standard neural network classifier, just after the convolutional blocks have already done the heavy feature lifting.

In operation, a CNN starts by pushing the input data through a chain of convolution plus pooling layers to tease out discriminative clues, then those outputs get flattened and sent into a fully connected layer, for the actual classification step. Since CNNs learn their own features straight from the data, rather than depending on hand-crafted feature engineering stuff, they tend to work really well for use cases like intrusion detection,

where tiny and even previously unseen patterns have to be noticed inside huge volumes of network traffic. Also, CNNs usually have efficient parameter usage, automatic feature extraction, solid pattern-recognition accuracy, and they scale fairly well when datasets get large; their main downsides are that they really lean on large labeled datasets, they can be computationally expensive, they have limited built-in capability for sequential dependencies, so people often hybridize them with recurrent architectures such as LSTM, and in deeper setups they can be less interpretable.

Long Short-Term Memory (LSTM) Networks

Long Short-Term Memory, or LSTM, networks are sort of a specialized kind of Recurrent Neural Network (RNN) made to deal with sequential, time related data. Typical RNNs compute each layer output using the current input together with information taken from earlier time steps, so they can somehow capture temporal dependencies. But the conventional RNNs run into the vanishing gradient issue, because when gradients get propagated backward through many time steps during training, and they're updated like weight = weight − learning rate × gradient, they gradually shrink until they become almost meaningless, which then blocks the model from learning long range dependencies. LSTM networks kind of fix this limitation by using a memory cell that's ruled by three different gate mechanisms, which makes things more stable over time:

$$w \leftarrow w - \eta \frac{\partial L}{\partial w} \quad (4)$$

• Forget Gate it decides what part of the old cell state should be thrown away.
• Input Gate it regulates how much fresh information gets inserted into the memory cell.
• Output Gate it figures out which slice of the memory cell should be revealed to the next layer or even the next time step.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \quad (5)$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (6)$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$

$$\tilde{c}_t = \tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (8)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \quad (9)$$

$$h_t = o_t \odot \tanh(c_t) \quad (10)$$

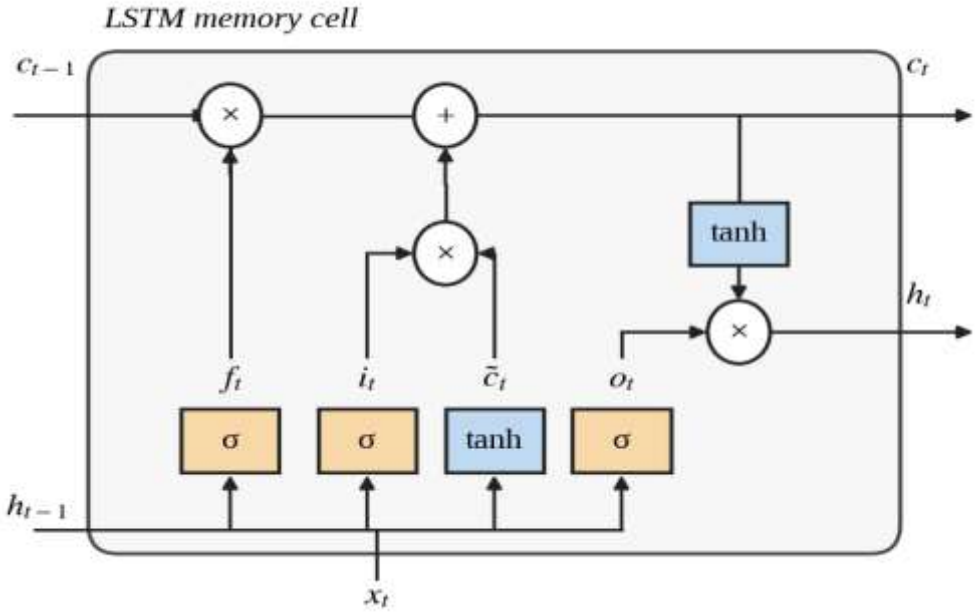


Figure 1: Structure of an LSTM memory cell with forget, input and output gates.

Altogether, these gates help the model keep the useful longer-term context, while letting go of the irrelevant stuff sort of. In practice it deals with the vanishing gradient issue, more or less, at least enough to make training smoother. Also since network traffic is pretty sequential and very timing dependent, LSTM is a strong fit for learning evolving relationships, the kind a purely spatial method like a CNN generally can't represent that well. So when you merge the CNN and the LSTM into one hybrid setup, you end up with a joint extraction

of spatial cues and temporal clues straight from raw traffic data, and that tends to improve spotting complicated attack patterns that keep shifting over time.

Over fitting and Regularization

A main challenge when training deep neural networks is over fitting, where the model ends up doing great on training data but then it sort of falls apart on new unseen inputs. The idea is that it has learned the noise and those dataset specific quirks rather than real transferable cues. This becomes particularly annoying for intrusion detection systems, because they have to keep working when the traffic is new, messy, and kind of unpredictable. You can reduce over fitting by making the training set bigger and more diverse, but full labeled network traffic datasets are expensive to get, and they are often limited by privacy constraints. So, regularization isn't just optional—it ends up being a essential partner strategy. Two commonly used regularization methods are L1 (Lasso) and L2 (Ridge) regularization, both of which tack on an extra penalty term to the loss so the model doesn't grow weights too large. With L1 you penalize the sum of the absolute weight values, and that can push some weights all the way to zero. In a sense, this is like built-in feature selection, even if you didn't ask for it directly. With L2 you penalize the sum of squared weights, which tends to produce a tighter, steadier weight spread; it usually doesn't force any particular weight to become exactly zero. In both cases, model complexity is curbed, and generalization improves. Dropout is also a widely used regularization approach, where a randomly chosen fraction of neurons is turned off during each training step, based on a set probability. This kind of nudges the network to form redundant, spread-out internal representations, so it doesn't rely on any one neuron too heavily. As a result, robustness against unseen data tends to improve. During testing though, all neurons are active again, and the outputs are adjusted with the right scaling so the behavior stays consistent. Additional strategies like early stopping (where training gets halted as soon as validation performance starts to degrade), data augmentation and batch normalization are often used in parallel with L1/L2 regularization and dropout to keep over fitting under control. In real setups, combining more than one regularization technique helps people tune the model complexity versus generalization performance, and that sort of trade off sits at the core of the design for the model that I describe in the next section.

$$\Omega_{L1} = \lambda \sum_{i=1}^{n} |w_i| \quad (11)$$

$$\Omega_{L2} = \lambda \sum_{i=1}^{n} w_i^2 \quad (12)$$

$$\tilde{h} = m \odot h \quad (13)$$

4. PROPOSED METHODOLOGY

The proposed system is a hybrid deep learning model that jointly studies and classifies network traffic using both spatial and temporal pattern recognition. It combines a CNN module for spatial feature extraction with an LSTM module for temporal feature learning and classification, structured as a two-stage pipeline: the first stage extracts spatial features from pre-processed traffic data, and the second stage learns temporal dependencies among those features before producing a final classification.

System Architecture

The end-to-end pipeline is basically six stages, shown in a sort of conceptual way like this

• Input Layer — raw network traffic data, with both packet-level plus flow-based features, is gathered. This raw material ends up being high dimensional and mixes useful signals with irrelevant ones, so it needs more processing before it's actually usable for model training.

• Data Pre-processing — first there is cleaning, so noise gets removed, missing values are handled, and inconsistencies are fixed. Then comes normalization where features are scaled into a consistent range, and finally feature selection which shrinks the dimensionality by spotting the most relevant attributes. The outcome is structured data, ready for deep learning.

• CNN Module (Spatial Feature Extraction) — convolutional filters sort of slide over the pre-processed input to find traffic anomalies, packet-structure oddities, and these less obvious hidden correlations between features.

• LSTM Module (Temporal Feature Learning) — the features coming out of the CNN module are then fed into an LSTM layer. That layer learns sequential dependencies, helping identify longer-term connections, plus attack behaviors that evolve step by step over time.

• Fully Connected Layer — after the spatial and temporal representations are merged, they go through a fully connected layer, which does the final feature integration before the model decides on a label.

• Classification Layer (Output) — a Soft max output layer returns class probabilities that add up to one. That gives the final classification of the traffic as Normal or Attack.

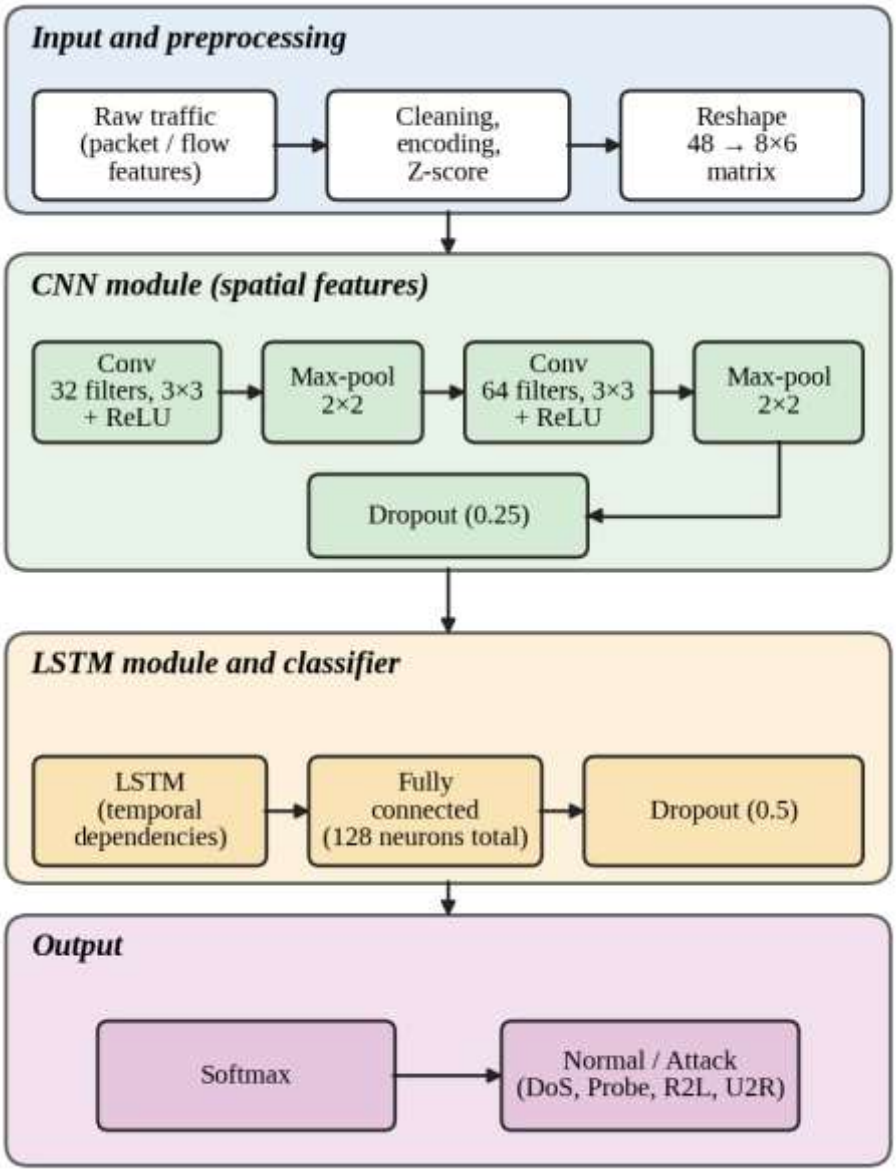


Figure 2: Architecture of the proposed hybrid CNN-LSTM intrusion detection model.

Phase 1: CNN-Based Spatial Feature Extraction

That CNN part is made up of two convolution layers, first with 32 output filters, and then with 64 output filters. Each layer uses a 3×3 kernel, so it can pull out local patterns and some feature ties from the preprocessed traffic data. After every convolution, there's a 2×2 max pooling stage, which shrinks the dimensionality and cuts computational cost too. At the same time it keeps the more informative features, so you get a kind of regularizing effect and it helps restrain over fitting.

Phase 2: LSTM-Based Temporal Learning and Classification

The spatial features that come out of the CNN are fed into an LSTM layer, where it sort a models the temporal dependencies found in the traffic traces, and then it goes through a fully connected layer. The LSTM plus the fully connected layers together add up to 128 neurons so the network can learn those richer, high level kinds of representations. After that, the fully connected layer takes the LSTM output and sort of prepares it for the classification step, and then the last Soft max layer spits out normalized class probabilities.

Here in this study, the system does binary classification, so it's basically separating:

Normal traffic, and

Attack traffic, which includes DoS, Probe, R2L, and U2R categories.

$$\hat{y}_{c} = \frac{e^{z_c}}{\sum_{k=1}^{K} e^{z_k}} \quad (14)$$

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i)] \quad (15)$$

Regularization Strategy

To get better generalization, and also try to stop over fitting, the model uses a few regularization techniques that kind of keep it in check:

L2 Regularization used with a coefficient 0.1, and it basically punishes big weight values so the model ends up learning more straightforward, more general sable representations.

Dropout placed at two points across the network: a dropout rate 0.25 comes after the L2-regularised layer, then a dropout rate 0.5 after the fully connected layer.

That stepwise dropout schedule, where the rate is smaller earlier and larger later, helps the model not lean too hard on a few specific neurons. At the same time, it doesn't cause too much information loss in the earlier part, so it can still hold onto the important signal, but it keeps gaining the regularization benefits. For training, the Adam optimizer is used, it tunes the model parameters in an adaptive way, which improves convergence speed and keeps things more stable.

$$L_{total} = L_{CE} + \lambda \sum_{i=1}^{n} w_i^2 \quad (16)$$

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t \quad (17)$$

$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \quad (18)$$

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \varepsilon} \quad (19)$$

5. EXPERIMENTAL SETUP

System Configuration

All experiments were conducted using the hardware and software configuration summarized in Table 1.

Table 1: Experimental Environment and System Configuration

Sr. No.	Component	Specification
1	Processor (CPU)	16-Core Processor
2	Graphics Processing Unit	NVIDIA RTX 4000 GPU
3	System Memory (RAM)	64 GB
4	Storage Capacity	1 TB
5	Operating System	Linux (Ubuntu)
6	Deep Learning Framework	TensorFlow 2.10
7	High-Level API	Keras 2.10

5.2 Dataset Description

The model that was proposed gets evaluated using two pretty standard benchmark datasets, KDDCup99 and NSL-KDD, and in both cases the traffic is grouped into Normal, and then four other kinds of attacks… like DoS, Probe, R2L, and U2R. In Table 2 we show how the samples are distributed per class for each dataset, so you can see the imbalance in a clearer way, and yeah it's not just one summary number.

Table 2: Description of Datasets Used (KDDCup99 and NSL-KDD)

Attack Category	Description	KDDCup99 (Train)	KDDCup99 (Test)	NSL-KDD (Train)	NSL-KDD (Test)
Normal	Normal connection records	99,548	55,321	67,343	9,710
DoS	Attacks that make network resources unavailable	265,471	132,458	45,927	7,458
Probe	Scanning and gathering network/system information	5,214	1,365	11,656	2,422
R2L	Unauthorized access from a remote machine	12,365	2,684	995	2,887
U2R	Gaining root or super-user privileges	658	305	52	67
Total Records	Total number of samples	385,444	192,486	128,979	23,505

Both datasets show a pretty strong class imbalance, like DoS Probe and Normal traffic end up taking over most of the sample distribution, while R2L and, especially U2R instances, are just really scarce. Because of that skew models that are trained only to chase overall accuracy tend to lean toward the majority classes, and they can report high aggregate accuracy but still do badly on the rare yet security critical kinds of attacks. So in this study we don't lean on a single overall accuracy number only, instead we report per class True Positive Rate (TPR) , False Positive Rate (FPR) , and Accuracy (Acc), to make those differences actually visible.

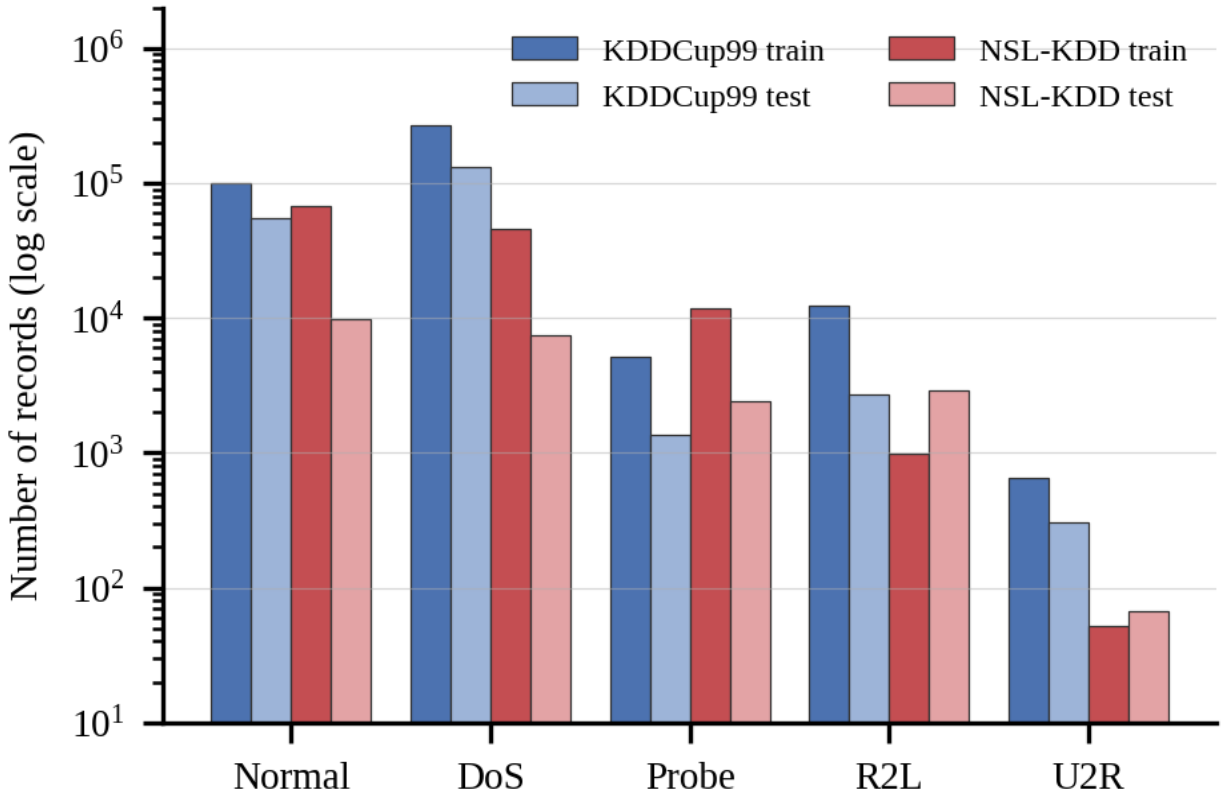


Figure 3: Class distribution of the KDDCup99 and NSL-KDD training and test sets (logarithmic scale).

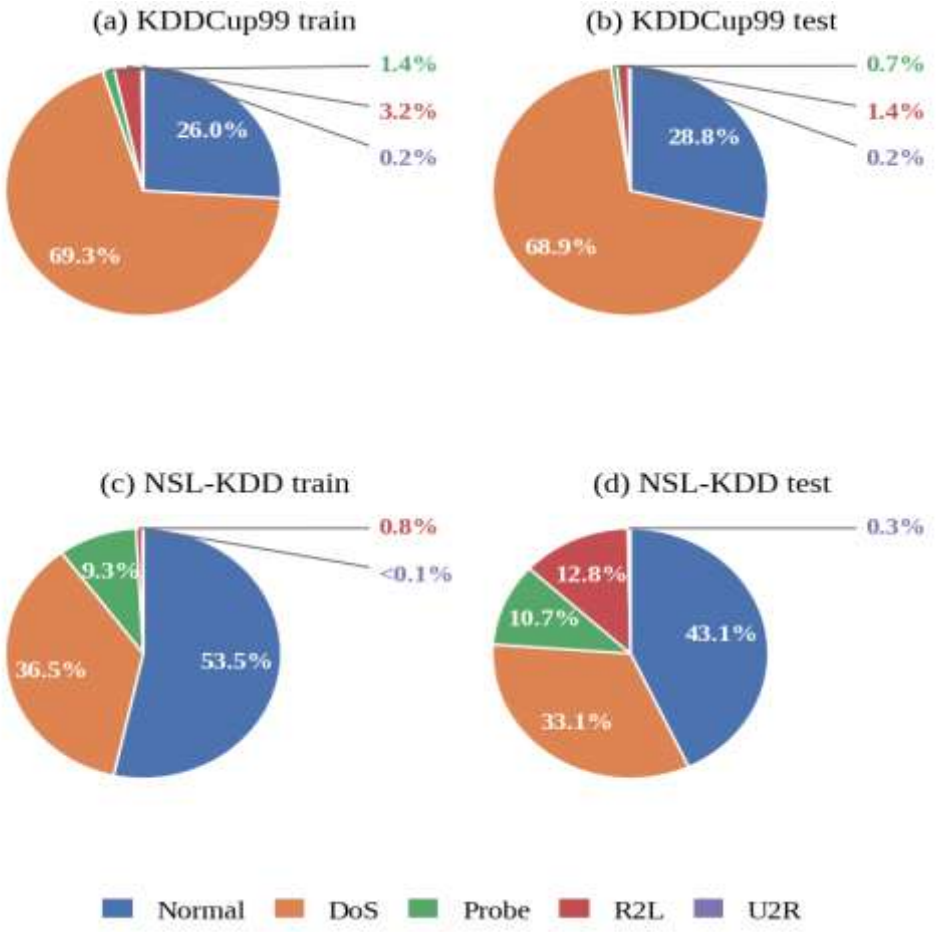


Figure 4: Share of each traffic class in the (a) KDDCup99 training, (b) KDDCup99 test, (c) NSL-KDD training and (d) NSL-KDD test sets.

Data Preprocessing

Data Normalization: the feature values are standardized into a similar, uniform scale using Z-score normalization, relying on the mean and standard deviation for each feature. This helps keep every feature contributing the same way during training, and it reduces the chance of bias toward those with larger numeric ranges

$$x' = \frac{x - \mu}{\sigma} \quad (20)$$

Data Transformation for CNN Input: to benefit from the parameter-sharing efficiency you get with 2D convolutional layers, every one dimensional record holding 48 traffic features is reshaped into a 2D matrix sized 8 × 6. so the CNN can infer spatial patterns even if the traffic data itself has no real visual structure

$$X = \text{reshape}(x, 8, 6) \quad (21)$$

Encoding and Label Conversion: Symbolic, meaning categorical, attributes are turned into numeric representations. while the original datasets include several attack types, this work uses a binary labeling scheme 0 for normal traffic and 1 for any attack traffic, so the model is concentrated on separating malicious from benign behavior, and then in Section 6 the results are further teased apart based on the original attack category.

$$\text{TPR} = \frac{TP}{TP + FN} \quad (22)$$

$$\text{FPR} = \frac{FP}{FP + TN} \quad (23)$$

$$\text{Acc} = \frac{TP + TN}{TP + TN + FP + FN} \quad (24)$$

6. RESULTS AND DISCUSSION

The proposed hybrid CNN-LSTM model is sort of evaluated on the NSL-KDD dataset, against thirteen baseline approaches, and it spans a bunch of classical machine learning methods — for example Logistic Regression (LR), Naive Bayes (NB), K-Nearest Neighbors (KNN), Decision Tree (DT), Ad a Boost (AB), Random Forest

(RF), and SVM with an RBF kernel (SVM-rbf) — plus deep neural network variants, with one to five hidden layers (DNN1–DNN5), and then there are also existing hybrid or ensemble structures such as TFTC, MDPCA-DBNs, Siam-IDS, I-ELM, HFR-MLR. For results they report True Positive Rate (TPR), False Positive Rate (FPR), and Accuracy (Acc) per traffic class, because reporting just aggregate accuracy might hide that minority classes get handled poorly, or at least it can make the behavior look nicer than it actually is.

Performance on Normal, DoS, and Probe Classes

Table 3: Detailed Test Results for Normal, DoS, and Probe Classes (NSL-KDD)

Method	Normal TPR	Normal FPR	Normal Acc	DoS TPR	DoS FPR	DoS Acc	Probe TPR	Probe FPR	Probe Acc
LR	92.34	51.84	69.42	64.85	32.98	79.65	25.36	12.62	91.21
NB	4.68	8.56	59.85	88.34	45.62	42.65	32.54	13.85	87.54
KNN	98.64	42.75	79.25	77.85	89.37	89.84	62.35	13.78	89.32
DT	98.24	38.64	79.14	79.82	5.32	92.84	66.95	12.65	91.24
AB	68.42	28.95	71.36	87.30	5.62	76.39	35.62	12.41	91.35
RF	98.48	42.85	77.58	78.94	32.45	91.25	66.32	13.52	90.25
SVM-rbf	99.87	53.46	74.84	66.87	11.65	89.25	52.36	12.65	90.35
DNN 1-layer	97.70	32.84	91.41	79.85	2.60	92.65	63.25	5.21	90.74
DNN 2-layer	97.60	33.45	91.65	78.54	1.80	92.35	74.25	6.95	92.36
DNN 3-layer	97.90	32.41	91.21	77.64	1.40	93.45	66.32	4.25	92.45
DNN 4-layer	97.86	33.56	87.68	77.98	1.60	93.65	65.25	6.32	92.85
DNN 5-layer	97.88	31.25	88.56	77.85	2.50	94.01	66.85	5.62	92.74
TFTC	98.54	36.56	91.45	88.62	6.30	92.52	66.32	6.30	91.36
MDPCA-DBNs	97.85	35.15	90.15	89.41	4.20	94.68	68.45	5.30	92.50
Siam-IDS	96.35	38.45	89.48	84.35	3.50	93.27	69.15	4.50	93.56
I-ELM	98.45	34.52	91.63	82.65	6.50	93.54	68.26	4.80	94.75
HFR-MLR	96.35	36.25	93.45	81.45	4.50	94.35	64.12	5.30	93.65
Proposed	99.56	21.01	98.45	98.45	1.20	99.45	71.25	4.15	96.48

Performance on R2L and U2R (Minority) Classes

Minority-class detection is the most demanding aspect of multi-class intrusion detection, given the extreme scarcity of R2L and U2R samples in the training data. Table 4 reports results for these classes.

Table 4: Detailed Test Results for R2L and U2R Classes (NSL-KDD)

Method	R2L TPR	R2L FPR	R2L Acc	U2R TPR	U2R FPR	U2R Acc
LR	5.62	8.40	97.25	5.80	9.10	88.24
NB	18.65	6.40	97.14	6.30	4.60	89.62
KNN	14.62	4.30	97.52	4.50	8.70	91.25
DT	45.62	9.50	97.65	9.60	5.60	92.48
AB	46.21	10.50	96.25	21.54	4.60	93.48
RF	56.29	6.10	96.34	12.65	9.50	91.21
SVM-rbf	9.65	50.20	97.15	9.48	11.00	91.45
DNN 1-layer	49.71	0.10	99.45	11.35	24.68	97.38
DNN 2-layer	27.20	0.10	99.98	0.20	9.24	99.48
DNN 3-layer	0.30	0.10	99.98	0.70	0.10	99.85
DNN 4-layer	0.20	0.10	99.98	0.60	0.10	99.12
DNN 5-layer	5.30	0.10	99.98	14.65	0.30	98.25
TFTC	4.90	0.10	98.62	13.85	0.20	98.14
MDPCA-DBNs	9.70	0.10	98.24	15.36	0.20	98.35
Siam-IDS	8.50	0.10	97.85	14.85	0.30	98.61
I-ELM	6.90	0.10	97.36	14.29	0.20	98.75
HFR-MLR	5.30	0.10	98.25	13.91	0.20	97.06
Proposed	36.45	0.10	99.85	89.45	0.20	99.95

DISCUSSION

Majority Classes (Normal and DoS)

Most baseline models do pretty well on Normal and DoS traffic, mostly because those cases get plenty of training samples; Random Forest and SVM for instance manage a Normal-class TPR over 98% even so, they are often linked with a high FPR, so normal traffic gets misread as attacks more than you would like. The model I proposed reaches a Normal-class TPR of 99.56% with a much lower FPR of 21.01%, and for DoS it gets a DoS-class TPR of 98.45% while keeping the FPR down at just 1.20%. Overall it beats nearly all baselines, and it suggests the model can spot the large scale attacks properly without pushing too many false alarms.

Probe Attacks

Most base line models do pretty well on Normal and DoS traffic, mostly because those cases get plenty of training samples, like Random Forest and SVM for instance manage a Normal-class TPR over 98% even then they are often tangled with a high FPR so normal traffic gets sort of misread as attacks more than you would really want. The model I proposed ends up with a Normal-class TPR of 99.56% with a much lower FPR of 21.01%, and for DoS it reaches a DoS-class TPR of 98.45% while holding the FPR at just 1.20%. Overall, it beats nearly all the baselines and it hints that the model can detect the large scale attacks properly without producing too many false alarms.

Minority Classes (R2L and U2R)

R2L and U2R attacks are, kind of the hardest to detect mainly because they are so extremely underrepresented in the training data. With the traditional stuff like LR and KNN, the TPR values are often below 10% for these classes, and even when people use ensemble methods such as Random Forest the improvement is just limited. The DNN variations can be a bit all over the place depending on depth sometimes

they trade TPR for FPR without a clear trend that you could actually rely on. In contrast, the model we proposed reaches an R2L TPR of 36.45% and, perhaps more importantly, a U2R TPR of 89.45% — which is a big step up compared to every baseline, none of which go beyond about 21.5% on U2R. Also, this improvement comes while keeping false positive rates low at 0.10% (R2L) and 0.20% (U2R). So the minority-class boost is not just "buying" performance with extra alarms.

False Positive Rate Analysis

False alerts that are too frequent can overload security analysts, and after a while that erodes trust in automated detection systems. So the FPR is as important as detection rate if you want something that survives real deployment. Many traditional models show high FPR, especially for Normal and DoS classes. Some approaches even look good in accuracy but only because they produce more false positives, which is, honestly not ideal. Our proposed model keeps FPR low across all five classes, and it is especially low for the minority classes too. That mix suggests it is accurate, but also dependable for real-world network environments that keep running continuously.

Comparison with Deep Learning and Hybrid Baselines

Multi-layer DNNs generally out do the traditional methods since they can learn layered almost nested, feature representations, but the performance does not always grow smoothly when the depth gets bigger. In fact, in this study, some deeper setups show weird over fitting, or they turn unstable, kind of like they lose their footing. There are also existing hybrid models, TFTC, MDPCA-DBNs, Siam-IDS, and I-ELM, they still cope with complicated traffic patterns fairly well. Yet they keep stumbling when it comes to catching the minority class, and the false positive rate only goes up a little, but it does go up. The proposed model does better than all of that, because it keeps a strong TPR for every class, it manages to bring down the FPR overall, and it stays balanced without any signs of over fitting, at least in the observed results.

Impact of Class Imbalance

Class imbalance shows up as one of the most significant hurdles in this study, and the dataset is kind of lopsided: there are much more DoS and Normal examples than R2L and U2R samples. That situation quietly steers learning toward the majority classes, it pushes overall accuracy upward even as minority-class detection stays weak, plus it can give you a misleading view if accuracy is treated on its own. In line with that, the baseline models here often go past 95% overall accuracy, yet they still struggle noticeably with R2L and U2R. The proposed model tackles that gap head on, it boosts the identification of those minority attacks a lot while still keeping solid results for the majority categories.

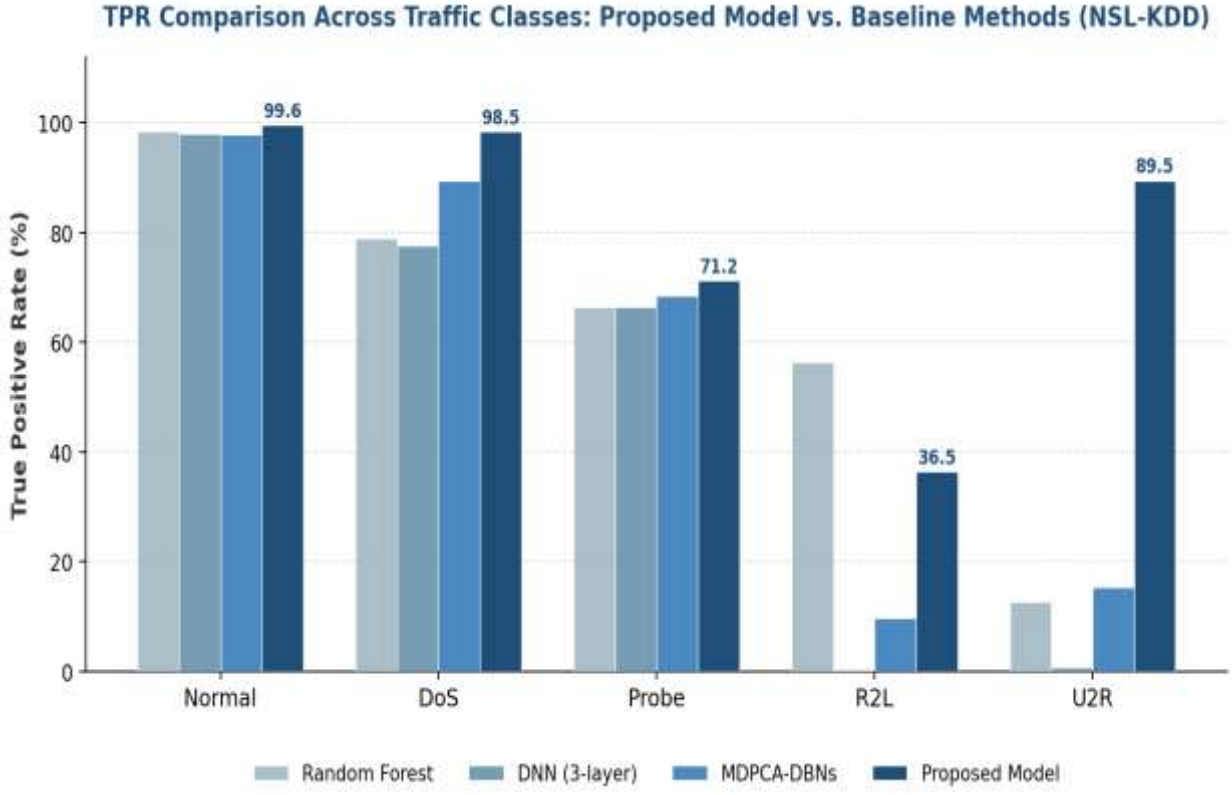


Figure 5: True Positive Rate Comparison Across Traffic Classes — Proposed Model vs. Representative Baselines (NSL-KDD)

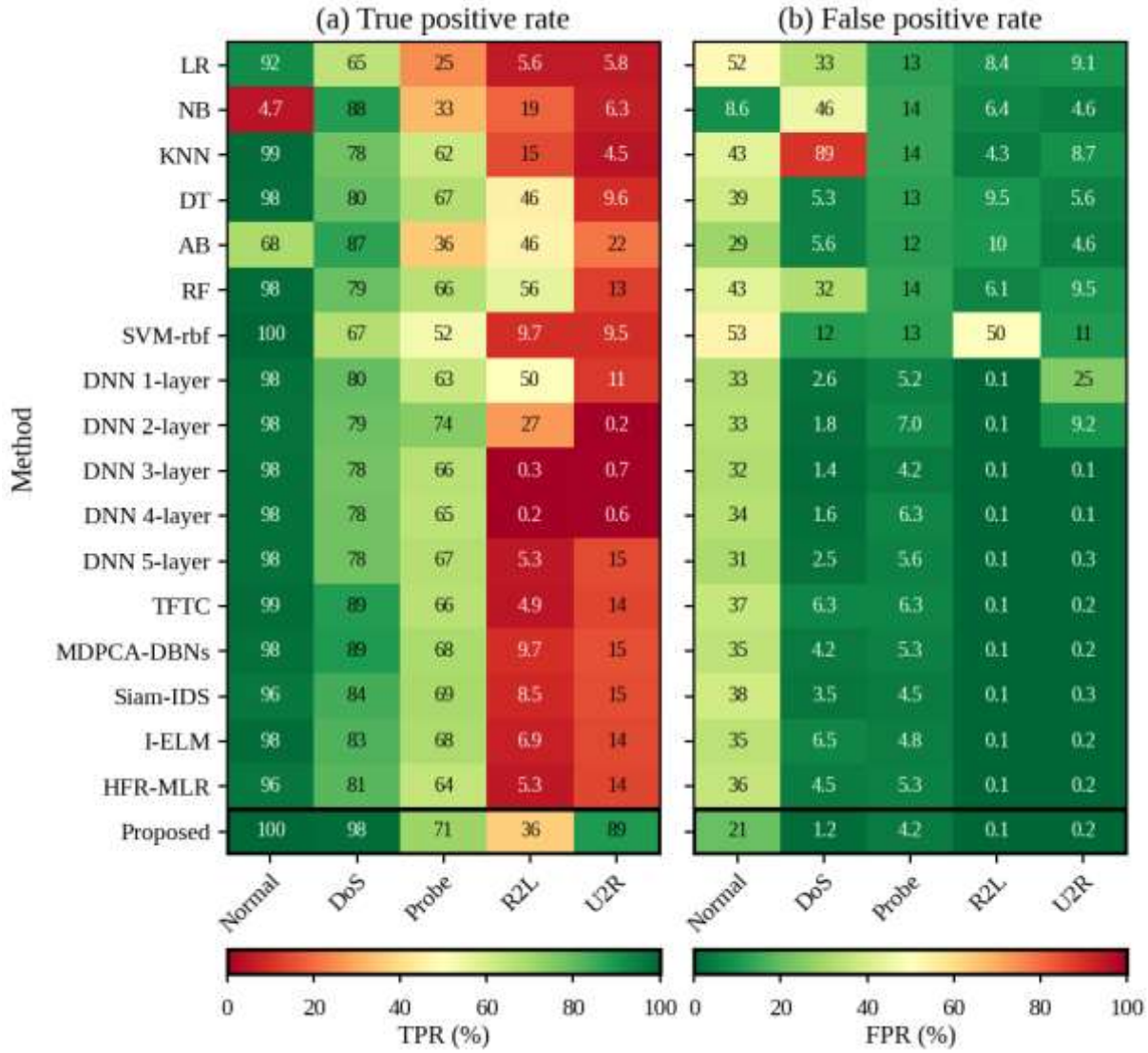


Figure 6: Heat maps of (a) true positive rate and (b) false positive rate per method and traffic class on NSL-KDD (Tables 3 and 4).

Practical Implications

- **Improved Security —** we get better detection of unusual attacks like U2R and R2L , and that lowers the exposure to tricky low-frequency intrusions which are often kind of a precursor to larger, more serious breaches
- **Reduced False Alarms —** the FPR stays reliably low, so security analysts don't end up being flooded by unnecessary warnings. This helps incident response stay more efficient
- **Balanced, Deployable Performance —** the model's combo of strong detection rates plus few false alarms across both the majority and the minority classes means it can be put in production. it works in fast changing real-world network situations , where both precision and trustworthiness matter

7. CONCLUSION AND FUTURE WORK

This paper presented a kind of hybrid CNN-LSTM deep learning model for network intrusion detection, meant to somehow capture both spatial and temporal traits of network traffic. In other words, it mixes CNN based spatial feature extraction with LSTM based temporal modeling, and it adds L2 regularization plus multi-stage dropout, so the model wont over fit too easily. Overall, the proposed architecture directly targets two stubborn weaknesses we've seen in earlier approaches: bad performance on minority-class samples, and the annoying balance issues between detection rate and false alarms. When evaluated against thirteen baseline methods, covering classical machine learning, DNN variants, and some existing hybrid architectures, on the NSL-KDD dataset, the proposed model was either the best or basically tied for best across all five traffic categories. It kept a Normal-class TPR of 99.56% and a DoS-class TPR of 98.45%, and at the same time the false positive rates stayed low. More importantly, the minority-class detection improved a lot, and the standout part was U2R: it reached a U2R TPR of 89.45% at an FPR of only 0.20%, which is well beyond every

baseline method that was tested. So these results really support the idea that hybrid patio-temporal deep learning gives a more balanced, dependable foundation for intrusion detection than purely spatial (CNN only) or purely sequential (LSTM only) approaches, and also better than classical machine learning methods that don't really have the capacity to model both dimensions together. Future work is going to extend this study in a few directions, kind a at the same time, maybe not exactly in that order: for one, we plan to evaluate the proposed architecture on extra and more recent benchmark datasets like CIC-IDS2017 and UNSW-NB15, so we can check generalize ability, beyond NSL-KDD. Also, we will try adding attention mechanisms to improve the feature representation even further and more importantly, to boost minority class sensitivity in a steadier way. Another line of work is to look into few-shot learning and synthetic sampling strategies (for example ADASYN) together with the hybrid architecture, with the goal of strengthening detection for rare attacks. In parallel, we want to test whether real-time deployment is actually feasible, and what the computational overhead looks like in production network settings. Finally, we will investigate explain ability approaches, since the black-box issue in deep learning based IDS is still one of the major obstacles for adoption in security critical scenarios, where people need some clarity, not only accuracy.

References

1. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
2. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from over fitting. Journal of Machine Learning Research, 15(1), 1929–1958.
3. Kingma, D. P., & Ba, J. (2015). Adam: A method for stochastic optimization. In Proceedings of the International Conference on Learning Representations (ICLR).
4. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning. MIT Press.
5. Hinton, G. E., Osindero, S., & Teh, Y. W. (2006). A fast learning algorithm for deep belief nets. Neural Computation, 18(7), 1527–1554.
6. Tavallaee, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In Proceedings of the IEEE Symposium on Computational Intelligence for Security and Defense Applications (CISDA).
7. He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In Proceedings of the IEEE International Joint Conference on Neural Networks (IJCNN).
8. Kim, J., Kim, H., Kim, T., & Kim, H. (2019). Hybrid CNN-LSTM network in network intrusion detection. In Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), 942–944.
9. Katkuri, A., et al. A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system. GitHub research repository, evaluated on the NSL-KDD dataset with feature selection and model optimization.
10. Jain, R., et al. (2026). Optimizing intrusion detection using a hybrid CNN-LSTM deep learning framework on the NSL-KDD dataset. Discover Computing, 29(194).
11. A hybrid CNN-LSTM approach for intelligent cyber intrusion detection system (2024). Computers & Security. Evaluates ANN, LSTM, BiLSTM, CNN-LSTM, GRU, and BiGRU with Recursive Feature Elimination on NSL-KDD.
12. Intrusion X: A hybrid convolutional-LSTM deep learning framework with Squirrel Search optimization for network intrusion detection (2025). arXiv:2510.00572.
13. The problem of improving network intrusion detection using a hybrid CNN-LSTM deep learning model: A cyber security perspective (2026). In Proceedings volume, Springer Nature.
14. Deep learning for network security: An Attention-CNN-LSTM model for accurate intrusion detection (2025). Scientific Reports, evaluated on NSL-KDD and Bot-IoT.
15. A hybrid CNN+LSTM-based intrusion detection system for industrial IoT networks (2023). Ad Hoc Networks / Science Direct, evaluated across Kyoto, CICIDS2017, KDD-Cup99, NSL-KDD, UNSW-NB15, and WSN-DS datasets.
16. Bagui, S., & Li, K. (2021). Resampling imbalanced data for network intrusion detection datasets. Journal of Big Data, 8(1), 6.
17. Fu, Y., Du, Y., Cao, Z., Li, Q., & Xiang, W. (2022). A deep learning model for network intrusion detection with imbalanced data. Electronics, 11(6), 898.
18. Abdelkhalek, A., & Mashaly, M. (2023). Addressing the class imbalance problem in network intrusion detection systems using data resampling and deep learning. The Journal of Supercomputing, 79(10), 10611–10644.
19. Impact of resampling techniques in deep learning based intrusion detection: A comparative study

on NSL-KDD and UNSW-NB15 (2025). Procedia Computer Science / Science Direct. Evaluates a CNN-BiLSTM model with ADASYN, RUS, Tomek Links, and OSS.
20. Overcoming class imbalance in network intrusion detection: A Gaussian Mixture Model and ADASYN augmented deep learning framework (2024). In Proceedings of the International Conference on Internet of Things and Machine Learning (IoTML). Proposes the C3BANet CNN-BiLSTM model with channel attention.
21. Addressing imbalanced data in network intrusion detection (2024). International Journal of Advanced Computer Science and Applications (IJACSA), 15(2).
22. Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. IEEE Access, 7, 41525–41550.
23. An intrusion detection method using Few-Shot Learning (2020). IEEE Access. Achieves 92.34% accuracy on KDD-Test+ and 85.75% on KDD-Test-21 using under 1% of NSL-KDD training data.
24. Few-shot network intrusion detection using discriminative representation learning with supervised autoencoder (2022). Applied Sciences, 12(5), 2351.
25. A few-shot network intrusion detection method based on mutual centralized learning (2025). Scientific Reports.
26. Intrusion detection system for NSL-KDD dataset based on deep learning and recursive feature elimination (2021). Evaluates a recurrent neural network for multi-class classification of DoS, Normal, Probe, R2L, and U2R traffic.
27. Advanced cloud intrusion detection framework using graph-based features, transformers, and contrastive learning (2025). PMC / journal article, evaluated on NSL-KDD and CIC-IDS2018.
28. Benchmarking datasets for anomaly-based network intrusion detection: KDD CUP 99 alternatives (2018). arXiv:1811.05372. Analyzes minority-class (R2L, U2R) performance limitations on KDD-99 and NSL-KDD.
29. Goh, J., Adepu, S., Junejo, K. N., & Mathur, A. (2016). A dataset to support research in the design of secure water treatment systems (SWaT). In Proceedings of the International Conference on Critical Information Infrastructures Security (CRITIS).
30. Moustafa, N., & Slay, J. (2015). UNSW-NB15: A comprehensive data set for network intrusion detection systems. In Proceedings of the Military Communications and Information Systems Conference (MilCIS).