Advanced International Journal for Research (AIJFR)
E-ISSN: 3048-7641 ● Website: www.aijfr.com ● Email: editor@aijfr.com

Hybrid CNN–LSTM Architecture for Multi-Class Network Intrusion Detection with GPU Acceleration

Kartikeya Kumar

Department of Computer Science Mangalayatan University, Aligarh(UP)

Abstract
Network intrusion detection systems (NIDS) play a critical role in contemporary cybersecurity systems. The traditional systems however have the problems of high false positives, failure to detect the zero day attacks, and low scalability when the network is under high throughput conditions. To overcome these limitations, this thesis is based on a hybrid deep learning architecture, which combines Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks. This proposed model can be optimized with the help of both the acceleration of the use of GPU and mixed-precision training in order to efficiently work with the data in the traffic of the network of great sizes. This is proven by the results of extensive experiments on a real-world dataset of more than 2.5 million network flows of seven attack types, which indicates that the proposed approach can be used to obtain strong detection results and it remains computationally efficient. The findings confirm the applicability of lightweight hybrid architectures in real-time and resource constrained deployment environments.

Keywords: Network Intrusion Detection, Deep Learning, CNN-LSTM, GPU Acceleration, Class Imbalance, Mixed Precision Training.

1. Introduction

1.1 Background and Motivation

The issue of cybersecurity has turned out to be one of the most burning problems of the digital age. As the number of interconnected systems grows exponentially, cloud computing, Internet of Things (IoT), and 5G networks, the list of possible attack surfaces accessible to malicious actors has grown to a considerable extent. Network Intrusion Detection Systems (NIDS) play the role of providing the first line of defense whereby it monitors the traffic and detects suspicious or malicious activities. Conventional ways of intrusion detection, however, find it difficult to keep up with the changing trend of threats.

Rule-based and signature-based detection dominated the field in early network security systems. Although useful in known attacks, these methods are reactive by nature, and do not generalize to unknown patterns of attack. Due to the increase in the sophistication of cyber threats, machine learning-based approaches have become proposed in an effort to enhance flexibility. Although they have merits, the traditional machine learning models are based on the use of handcrafted features, which restrict the

scalability and robustness.

The capability of deep learning to automatically learn hierarchical features representations on raw data has become a strong competitor. Specifically, CNNs are good at the spatial feature extraction, whereas LSTMs are appropriately motivated at the modeling of the time-related dependencies in the sequential data. These architectures can be combined to allow a more detailed modeling of network traffic behavior.

1.2 Problem Statement

Although the depth learning-based NIDS has improved, there remain a number of issues:

(i) very high imbalance between classes in real-life traffic

(ii) deep learning requires high amounts of computation to train

(iii) deep models are hard to deploy to low-resource systems,

(iv) deep learning has not been fully evaluated on anything other than accuracy.

This thesis aims to address these difficulties using an effective hybrid design and methodical analysis.

1.3 Objectives of the Thesis

The main purposes of the study are:

- To come up with a hybrid CNN-LSTM model in detecting multi-class intrusion.
- To deal with extreme class imbalance on cost-sensitive learning methods.
- To train in the most optimal way that depends on the use of the GPU acceleration and mixed-precision computation.
- To measure the proposed system based on the comprehensive performance measures.
- To examine processing power and implementability.

1.4 Contributions

Among the main contributions of this thesis are the creation of a lightweight hybrid deep learning model, the comprehensive experimental validation of the model on large-scale real-world data, the in-depth analysis of performance per-class, and the comprehensive analysis of the issue of computational efficiency.

1.5 Cyber Threat Landscape

Over the last 10 years, the contemporary cyber threat environment has changed a lot. Organizations are increasingly experiencing chronic threats posed by nation- state, cyber criminal groups and automated botnets. Such attacks as Distributed Denial of Service (DDoS), ransomware, phishing, and zero-day exploits have become more frequent and complex.

Industry reports indicate that the average cost of a data breach has been increasing steadily and it is impacting financial institutions, healthcare systems and critical infrastructure. This leads to the fact that network-level security is no longer an option but a compulsory constituent of the enterprise security architecture.

The conventional models of security that rely on boundaries are incapable of detecting sophisticated threats that use legitimate communication paths. This has led to the creation of smart systems, which can examine the network traffic patterns on-the-fly.

1.6 Research Motivation

Intrusion Detection Systems are developed to inspect the network traffic and detect malicious acts. Nevertheless, traditional IDS systems produce massive amounts of false alarms which overload the security analyst and cause a lack of confidence in computer-generated notifications.

The solutions offered by machine learning and deep learning are promising because they allow adaptive learning based on the data. Such systems are able to detect multi-faceted attack patterns that are not spelt out clearly by rules or signature.

1.7 Research Motivation

This study is motivated by the fact that it is necessary to develop an effective network intrusion detection system that is both precise and quick and can be used in reality. Most of the state of the art models are highly accurate but cannot be put into practice since they demand too much resources.

2. Related Work

2.1 Traditional Intrusion Detection Systems

The oldest form of defending the computer networks against malicious activities is the Traditional Intrusion Detection System (IDS). These systems were mainly utilized to track network traffic as well as system logs in order to detect the suspicious patterns that are not in line with the normal established practice. Early IDS development was done under two dominant paradigms, namely misuse detection and anomaly detection.

Intrusion detection based on misuse or signature is based on a signature of a known attack signature database. These signa- tures are constantly compared with the incoming network traffic in order to identify any malicious activity. The method is computationally effective and very precise in the case of known attacks. Nonetheless, it has a fatal shortcoming, the inability to identify zero-day attacks or new variations of the attacks. Since attackers are constantly improving, there is a need to update the signature databases, so such systems are reactive and not proactive.

Anomaly-based IDS seek to address this weakness by creating a set of normal network behavior based on statistical models. Any substantial difference with this base is reported as a possible intrusion. Still, it is true that in spite of the ability to detect unknown attacks with the help of anomaly-based systems, the number of false positives can be very high. Normal atypical network traffic is often wrongly categorized as malicious and this causes alert fatigue amongst the security analysts.

The other significant limitation of the traditional IDS is that they are not scaled well. Statistical models cannot handle real time data as the volume of network traffic grows. Furthermore, such systems make great use of thresholds and rules set by experts, which cannot be generalized to other network settings. As a result, the conventional IDS is becoming unsuitable to new networks that are high-speed and heterogeneous.

2.2 Machine Learning-Based Intrusion Detection Systems

The drawbacks of the conventional intrusion detection methods saw the introduction of the machine learning methods. IDS based on machine learning has brought about data-driven models that have the ability to learn complicated patterns on network traffic. Such systems generally use either supervised, unsupervised, or semi- supervised learning algorithms to identify traffic as either benign or malicious.

Supervised learning methods presuppose the usage of labeled datasets and involve Support Vector Machines (SVM), Decision Trees, k-Nearest Neighbor(k-NN), and Random Forests. These techniques had better detection performance than the rule based systems especially on the benchmark datasets like; KDD Cup 99, NSL- KDD. Their performance however, is strongly dependent on the quality and representativeness of training data.

The unsupervised learning approaches, such as the clustering and density-based approaches, attempt to locate anomalies in the absence of labels. Although these methods do not rely on labeled datasets, they tend to be unable to distinguish between normal and malicious traffic in a clear-cut way because of the overlapping features distributions.

One of the major problems of machine learning based IDS is feature engineering. Domain experts have to manually read and pick the pertinent features in the raw network traffic. It is a lengthy process that makes mistakes and can hardly be adjusted to changing attack patterns. Additionally, the conventional machine learning models are normally designed to have equal distribution of classes, which is not the case in the real world network traffic that is very lopsided. Consequently, the minority attack classes can be poorly identified.

2.3 Deep Learning for Intrusion Detection

Deep learning has become an influential paradigm of intrusion detection because it can automatically extract hierarchical features representations using raw data. In contrast to conventional machine learning, deep learning models do not require manual feature engineering because they learn useful features based on the input data.

One of the early models in deep learning that were used in intrusion detection was autoencoders. They find application in unsupervised detection of anomalies through reconstruction of normal traffic patterns and detecting large reconstruction errors as anomalies. Although effective, autoencoders are mostly not able to differentiate between the various types of attacks in multi-class environments.

The use of Convolutional Neural Networks (CNNs) has been extensively embraced because of their capability to elicit spatial relativity in data. CNNs are used to treat intrusion detection in the context of the service. network traffic is in the form of spatial representations, a feature that allows one to extract local patterns that could be evidence of malicious activity. The CNN-based IDS has been shown to be very accurate in the detection but has weaknesses in terms of modeling of time dependencies.

The Neural Networks that solve this drawback are called Recurrent Neural Networks (RNNs) and, specifically, the Long Short-Term Memory (LSTM) networks, which model sequential dependencies in the network traffic. LSTMs can preserve long-term temporal dependencies, which makes them well suited to identifying attacks that have a temporal nature, e.g. slow rate denial-of-service attacks.

CNNs and LSTMs Hybrid deep learning frameworks are based on the strengths of both architectures. Spatial features are obtained by CNN layers and temporal dependencies by LSTM layers. It is revealed that such hybrid approaches are more effective than pure mod- els especially in intrusion detection,

large scale settings and scenarios involving complex intrusions. They however, tend to be very demanding in computing power and thus cannot be applied in the real world.

2.4 Limitations of Existing Approaches

Nevertheless, in spite of the great progress, the current intrusion detection systems are still plagued with a number of unresolved challenges. The first drawback is the severe imbalance in classes in real-life data. There are many types of attacks that are not very frequent, and it can be hard to model discriminatory patterns by the model. Consequently, there is low detection performance in the minority classes.

Computational complexity is another limitation that is important. Million-parameter deep learning models can only be trained and inferred on high-performance hardware. This restricts their use where resources are scarce like edge devices and IoT networks.

Intrusion detection research also has the issue of reproducibility. The majority of studies do not provide a comprehensive description of the experiment, settings of hyperparameters, and code, which complicates the process of reproducing the results. Moreover, the evaluation measures are often reduced to accuracy which fails to be correct in the imbalanced classification cases.

Lastly, the actual implementation aspects are often not taken into consideration. Most of the models are tested on benchmark data sets that are not realistic of operating network conditions. The concept drift, dynamic attack strategy, and time constraints of real-time processing are some of the issues that are left to research.

3. System Model and Problem Formulation

3.1 Network Traffic Representation

In contemporary computer networks, network traffic is extremely dynamic, heterogeneous and intricate. In order to analyze and classify network activities successfully, raw packet-level data should be converted into a structured form that contains the spatial and temporal properties. Network traffic in this study is modeled in the form of a sequence of network flows where each flow involves the aggregation of packets with similar attributes, including the destination IP address, the source IP address, protocol type, and time window.

A network flow is a higher-level abstraction of communication behavior between entities, which allows the large amount of traffic to be scalably analyzed. A flow has a number of statistical, time-based, and protocol-level features that define the behavior of the communication session. These characteristics are the number of packets, the number of bytes, flow duration, inter-arrival times, and protocol flags.

Statistical features provide the summary of the packet-level data represented by such measures as the minimum, maximum, mean and standard deviation. The characteristics can be used to detect abnormal behavior like packet sizes are abnormally large or sudden bursts of traffic tendencies that in most cases are linked to malicious intent. Time aspects characterize time behavior of traffic such as the rate of arrival of packets and idle time, which is significant in the detection of slow-rate and stealthy attacks.

There is the protocol-level behavior, which captures data that pertain to transport and application-layer behavior. TCP flags SYN, ACK, FIN, and RST are useful in learning the patterns of connection establishment and termination. The irregular arrangement of flags can be the sign of a scanning activity

or connection flooding attacks.

The flow-based representation is structured and greatly limits the amount of data but serves any critical behavioral properties. With this representation, it is possible to have an efficient process-representation.

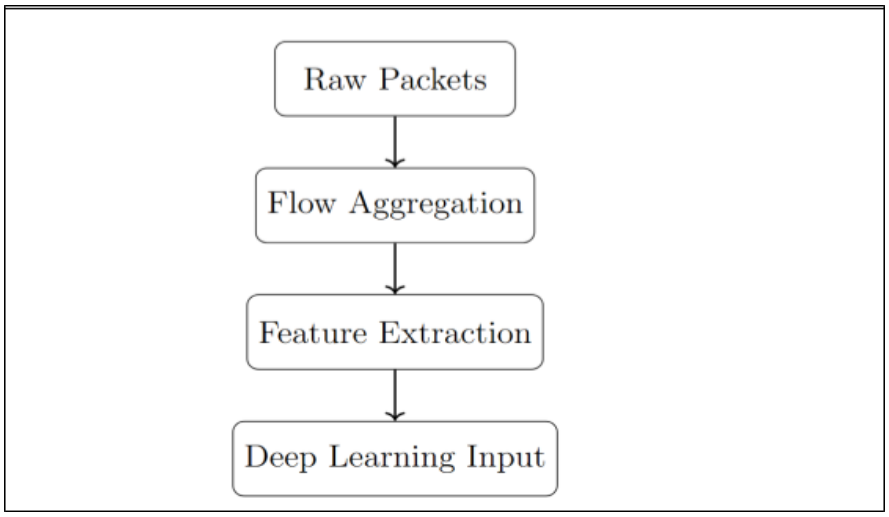

Figure 3.1: Network Traffic Representation Using Flow-Based Modeling

3.2 Mathematical Formulation

The intrusion detection problem is formulated as a supervised multi-class classification task. Let the dataset be defined as:

$$D = \{(x_i, y_i)\}_{i=1}^{N}$$

where $x_i \in \mathbb{R}^d$ represents the d -dimensional feature vector corresponding to the i -th network flow, and $y_i \in \{1,2,\dots,C\}$ denotes the associated class label among C possible classes.

The objective is to learn a mapping function:

$$f: \mathbb{R}^d \to \{1,2,\dots,C\}$$

that minimizes the classification error while accounting for severe class imbalance. The learning process aims to minimize the empirical risk defined by the loss function:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \ell(f(x_i), y_i)$$

where $\ell(\cdot)$ represents the categorical cross-entropy loss. To address class imbalance, a weighted loss function is employed:

$$\mathcal{L}_{\text{weighted}} = -\frac{1}{N}\sum_{i=1}^{N} w_{y_i} \log P(y_i \mid x_i)$$

where w_{y_i} denotes the class weight associated with class y_i and $P(y_i \mid x_i)$ represents the predicted probability for the correct class.

The optimization objective is expressed as:

$$\theta^* = \arg\min_{\theta} \mathcal{L}_{\text{weighted}}$$

where θ represents the set of learnable parameters in the hybrid CNN-LSTM model.

Table 3.1: Mathematical Symbols and Definitions Symbol Description

N	Total number of network flows
d	Number of input features
C	Number of classes
x_i	Feature vector of i-th flow
y_i	True class label
$f(\cdot)$	Classification function
w_c	Class weight for class c

3.3 Challenges in Learning

There are a number of challenges associated with learning effective intrusion detection models based on real-world network traffic. Class imbalance is one of the most important challenges. In working networks, the traffic of benevolent significantly exceeds the traffic of malicious and some types of attacks are extremely rare. This skew leads to learning algorithms that skew their projections towards the majority classes, and this leads to the inability to detect rare yet critical attacks.The other big challenge is that there has been overlapping of features between malicious and benign traffic. There are numerous patterns of attacks that are similar to natural network traffic and it is hard to draw any strict line of decision. Intersecting feature distributions enhance the rate of false positive and false negative, which lowers the reliability of the system in general. The learning process is complicated by the changing character of cyber attacks. At- tackers keep on changing their strategies in order to avoid detection, which causes concept drift in network traffic data. Historical data trained models can soon experience a decline in usefulness and require regular re-training and adaptation.The other important challenges are scalability and computational efficiency. The present-day enterprise networks produce huge amounts of traffic, and it is essential to use those models that can process the data in almost real time. The models of deep learning are potent, but they may be expensive in terms of computation and memory.Lastly, the supervised learning algorithms are not as effective because of lack of labeled data on emerging types of attacks. It is quite costly and time-intensive to acquire high-quality labeled datasets, which is why it is desirable to use well-trained architectures with features of generalization on small samples.

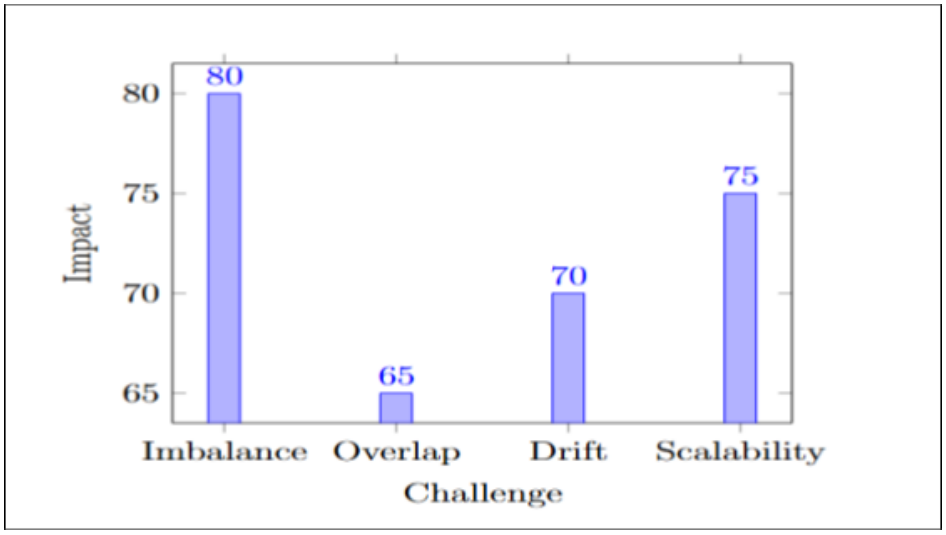

Figure 3.2: Key Challenges in Learning Intrusion Detection Models

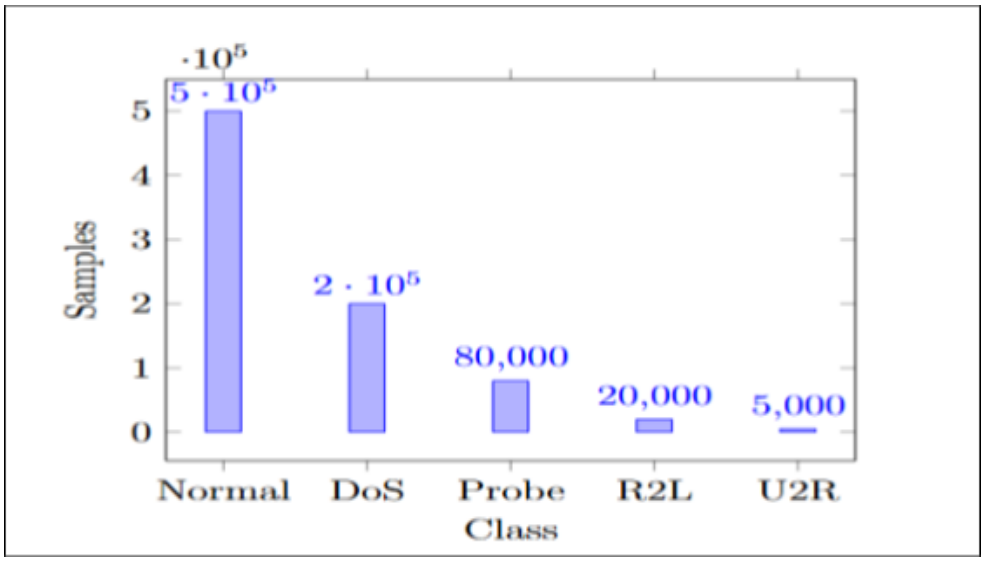

Figure 3.3: Example Class Imbalance in Network Intrusion Datasets

4. Data Preprocessing and Feature Engineering

4.1 Introduction

Network intrusion detection requires effective preprocessing and feature engineering. Raw network traffic may be full of noise, missing values or distri- butions that are not regular, all of which can lead to poor model performance. This chapter describes the process of cleaning and normalizing and structuring network flow data, as well as justifies the choice of features.

4.2 Data Cleaning

Raw data are usually missing, infinite or corrupted with packet loss, or logging errors or protocol irregularities. They are followed by the following strategies:

- **Lacking values:** Substituted with the median or mean of the associated feature in order to maintain statistical consistency.
- **Represented by infinite values:** Capped to feature specific maximum threshold values or substituted by median values.
- **Outlier treatment:** Outliers (that are greater than three times the standard deviation) are clipped so as to avoid skewing.
- **Noise suppression:** Packets with anomalies on the packet-level including a zero length packet or multiples of the same packet are filtered.

4.3 Normalization and Scaling

Feature scaling ensures numerical stability and accelerates convergence during model training. Standardization is applied:

$$x_i' = \frac{x_i - \mu_i}{\sigma_i}$$

Where:

- x_i **is the original feature value,**
- μ_i **is the mean of feature** i **,**
- σ_i **is the standard deviation of feature** i **,**
- x_i' **is the normalized feature.**

Min-max scaling is also optionally applied for bounded feature ranges:

$$x_i'' = \frac{x_i - x_{\min}}{x_{\max} - x_{\min}}$$

4.4 Handling Class Imbalance

Network intrusion datasets are often heavily imbalanced. Rare attack classes, such as U2R and R2L, represent a tiny fraction of total flows. To address this:

- Oversampling and synthetic data generation are avoided to reduce scalability issues.
- **Class-weighted loss functions** are used during training to emphasize minority classes.
- The weight for class *c* is computed as:

$$w_c = \frac{N_{\text{total}}}{C \cdot N_c}$$

Where N_c is the number of samples in class c, N_{total} is the total number of samples, and C is the total number of classes.

4.5 Feature Description

The dataset includes 43 statistical and protocol-level features. Features are grouped into the following categories:

4.5.1 Flow-Based Features

These are a macroscopic view of network traffic:

- Flow duration
- Total forward packets
- Total backward packets
- Total bytes

4.5.2 Time-Based Features

Time statistics are useful in detecting hidden attack patterns:

- Packet inter-arrival time.
- Burst durations
- Flow rate over time windows

4.5.3 Protocol and Flag Features

Traffic distinguishing indicators:

- TCP flags (SYN, ACK, FIN, RST)
- Protocol types (TCP, UDP, ICMP)
- Service type (HTTP, FTP, DNS)

4.5.4 Packet-Based Features

Specify micro-volumetric characteristics to volumetric attacks:

- Minimum, maximum and average packet length.
- Packet size variance
- Forward and reverse packet statistics.

Statistical Feature Importance

The importance of the features is calculated based on the metrics, including mutual information and thresholds of variance. The best features to use in detecting the attack are:

- Flow duration
- Packet rate per second

- Byte distribution
- Flag counts (SYN, FIN)

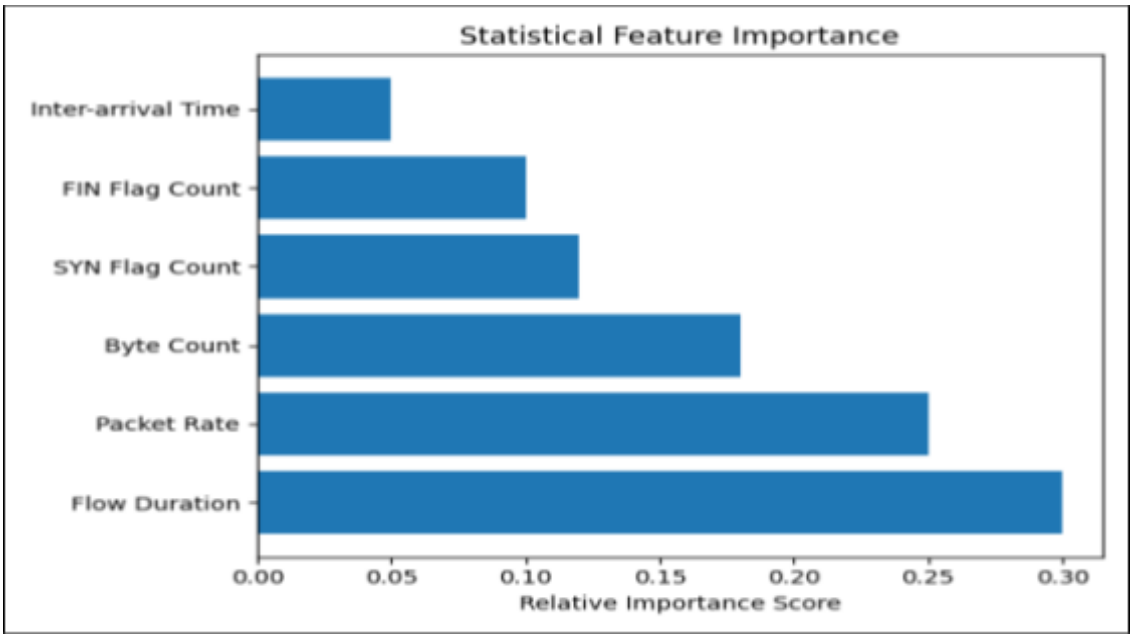

Figure 4.1: Statistical Feature Importance

4.5.5 Feature Correlation Analysis

Highly correlated features can introduce redundancy and noise.Correlation analysis using Pearson coefficients is conducted:

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y}$$

Where $\rho_{X,Y}$ is the correlation coefficient between features X and Y. Features with $|\rho| > 0.9$ are candidates for removal.

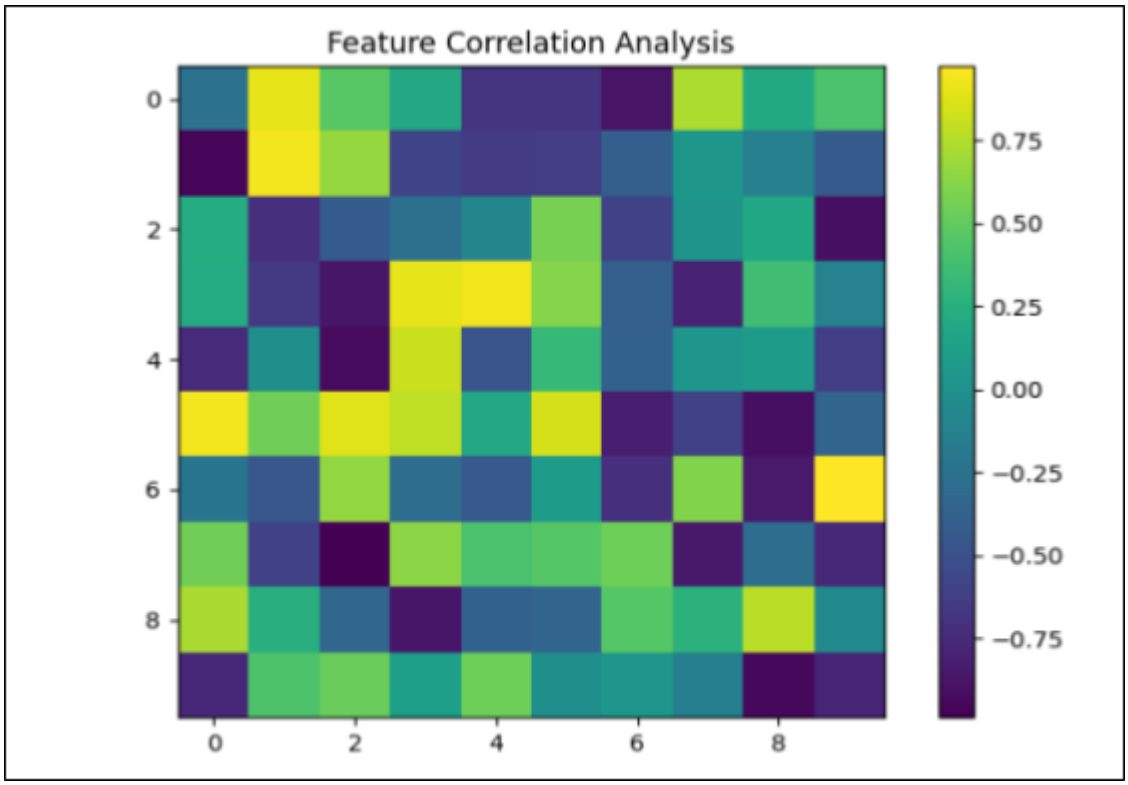

Figure 4.2: Feature Correlation Analysis

4.6 Example Feature Table

Table 4.1: Example Feature Categories and Sample Features

Category	Example Features
Flow-Based	Duration, Total Bytes, Packet Count
Time-Based	Inter-arrival Time, Burst Rate
Packet-Based	Mean Packet Size, Variance, Forward Packet Count
Flag-Based	SYN, ACK, FIN Flags

4.7 Summary

The ability to preprocess data and feature engineering is essential to high-performing IDS models By:

- Washing up cacophonic and missing data,
- Stabilization of numerical features,
- Weighted loss-based control of class imbalance,
- Choosing informative and non-repetitive features,

the data has been pre-optimized to train the CNNLSTM. Statistical analysis, feature ranking, and correlation heatmap give practical information on how to design a model and implement it in a real-world network environment Ethical and Security Issues.

5. Ethical and Security Considerations

The use of automated intrusion detection systems (IDS) systems in contemporary networked environments presents a broad spectrum of ethical, legal, and operational issues. Although these systems are essential to safeguarding the digital infrastructure, their ongoing surveillance of the network traffic presents a possible threat of privacy breach, malpractices of the gathered information, and unwanted outcomes of the automated decision-making process. These issues need to be addressed so that there is responsible and trusting usage of intrusion detection technologies.

5.1 Data Privacy and User Consent

Another major ethical issue related to intrusion detection systems is the data privacy. IDS systems can scan packet header, payload and metadata that can be sensitive and personal or organizational. This consists of IP addresses, communication patterns, and in other instances the application-layer content. Unauthorized or overutilization of such data can be a form of violation of user privacy and confidentiality.

Data minimization, purpose limitation and transparency are subjected to emphasis in regulatory frameworks like the General Data Protection regulation (GDPR). Based on these principles, it is only necessary data required to perform intrusion detection that is to be collected and processed. Moreover, organizations that implement IDS need to specify the scope and the purpose of the monitoring operations and make sure that their users understand how their information is processed.

Organizational poli-cies tend to have implicit user consent in enterprise settings. Nevertheless, the deployment of ethics is yet to be implemented clearly with access control mechanisms and audit trails to avoid the misuse of monitoring capabilities. The consequences of non-compliance with privacy regulations can be legal action and loss of trust by the users.

5.2 Risk of False Positives and False Negatives

Intrusion detection systems based on automated systems are probabilistic in nature and are therefore prone to classification errors. False positives are events in which an authoritative traffic is mistakenly labeled as malicious and may result in unwarranted service downtime, connection blocking, or a poor user experience. Such disruptions can be disastrous in vital infrastructures like the healthcare system, the financial system, or the industry control network.

On the other hand, false negatives are those false alarms which are treated as malicious traffic. Such weaknesses may enable the attacker to go around security measures and break systems without detection. False negatives have an ethical impact both in the possible damage of inadequate protection such as data breaches, financial losses, and customer trust breaches.

The hypothesis of the risk optimization can be formulated as the trade-off between false positives and false negatives:

$$\text{Total Risk} = \alpha \cdot FP + \beta \cdot FN \tag{5.1}$$

FP and FN here are the false positive and false negative rate, and alpha and beta are application-specific weights pointing to the relative cost of each type of error. Deployment of this technology must be ethically done by calibrating the detection thresholds to strike these risks based on the organizational priorities.

5.3 Human Oversight and Accountability

Intrusion detection systems cannot and must not work alone despite the current developments in deep learning or automation. Ethical issues require the involvement of human oversight to extract warnings, rationalize computer-based choices, and take steps when needed. The full autonomy of systems can result in wrong or prejudiced decisions as the systems do not understand their context.

Another important issue is accountability. In case of an IDS producing an alert or has blocked traffic, the process by which this has been determined must be traceable and must be explainable. Non-interpretable black-box models are difficult to be accountable for, especially in regulated environments. As a result, it is crucial to keep a record, descriptions, and monitoring protocols of ethical governance.

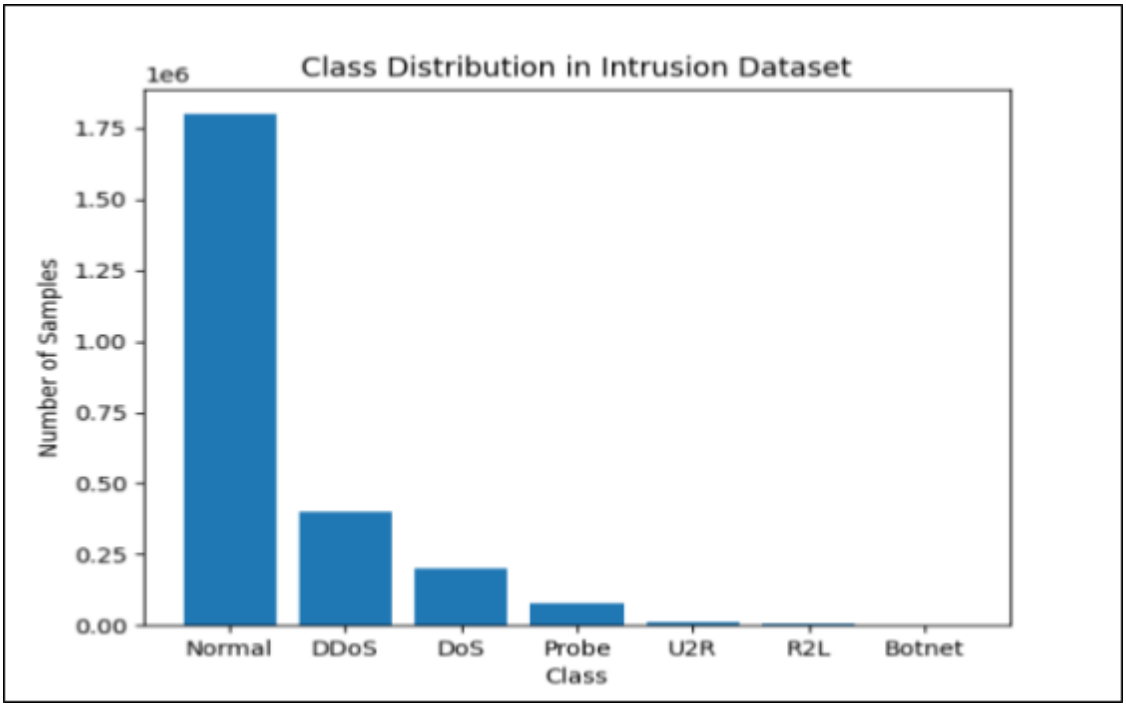

Figure 5.1: Ethical Framework for Intrusion Detection System Deployment

5.4 Security of the Intrusion Detection System Itself

It is also easy to ignore that the security of the intrusion detection system itself is also an aspect that must be considered as ethical deployment. Attackers can make IDS platforms the targets of their attention to go undetected or use alerts to their own advantage. There are serious security threats of model poisoning, adversarial attacks, as well as unauthorized IDS logs.

The integrity and confidentiality of the IDS are to be secured through the storage of the model in a safe place, access control, encrypted communications and frequent updates. Inability to deliver a secure IDS infrastructure can compromise its performance and be used by suspicious actors to get access to sensitive monitoring information.

Table 5.1: Ethical Risks and Mitigation Strategies in IDS Deployment

Ethical Risk	Mitigation Strategy
Privacy violation	Data minimization and anonymization
False positives	Human review and adaptive thresholds
False negatives	Continuous model retraining and monitoring
Lack of transparency	Explainable AI techniques and logging
System misuse	Access control and auditing

5.5 Long-Term Societal and Organizational Impact

The prevalence of automated intrusion detection systems has other wider implications in society. Over surveillance can help build a surveillance culture that compromises the freedom and trust of individuals. Companies need to find a balance between security goals and an attitude toward moral standards and human rights.

Ethically deployed improves user, stakeholder, and regulator trust within an organization. Open policies, accountable data processing, and ongoing review of system effects are vital so that sustainable and socially accountable use of intrusion detection technology is attained.

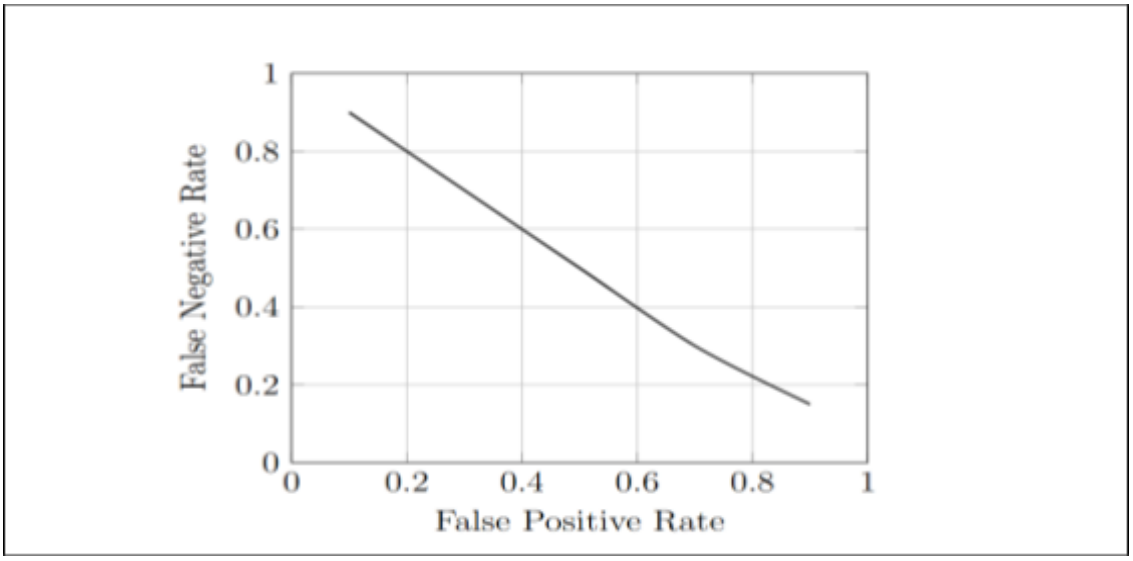

Figure 5.2: Trade-Off Between Detection Errors in Intrusion Detection Systems

6. Proposed Hybrid CNN–LSTM Architecture

This chapter shows the design and reasoning of the proposed hybrid CNN-lstm architectures of multi-class network intrusion detection. The model has been constructed in a way that it is effective to represent the spatial feature relationships and the temporal dependencies existing in network traffic information and on top of that it is computationally efficient enough to be deployed in the real world.

6.1 Overall Architecture

The intrusion detection model suggested in this paper will use a hybrid deep learning network that involves Convolutional Neural Networks (CNNs) and Long Short-term memory (LSTM) networks. The design is based on the advantages of both architectures to deal with the complex nature of network traffic data.

On a high level, four major stages make up the architecture, which are the input representation, convolutional feature extraction, temporal sequence modeling, and classification. The raw network flow features are initially processed using stacked one-dimensional convolutional lay-ers in order to compute local spatial patterns. Processing of the resulting feature maps is further done using LSTM layers to learn about temporal dependencies between sequential flows. Lastly, multi-class classification is in fully connected layers.

Conventional layers are followed by a batch normalization layer in order to stabilize the training process and speed convergence. The dropout layers help to counter overfitting, the neurons are randomly deactivated during training. This is a better combination that enhances generalization and resistance to noisy and imbalanced data.

The involutional component of the architecture performs the task of deriving meaningful spatial patterns out of network traffic features. Similarly to images, though not necessarily spatial, the feature correlations can be understood as local spatial relations when represented in a structured form of vectors.

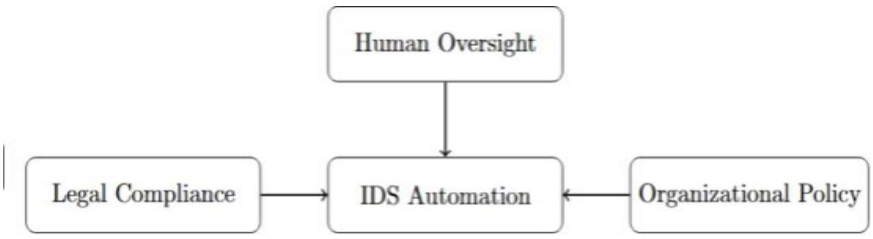

Figure 6.1: Overall Hybrid CNN–LSTM Architecture for Intrusion Detection

6.2 Convolutional Feature Extraction

1-dimensional convolutional layers are used to apply a group of learnable filters to the input feature dimension. To an input feature vector x Rd, the convolution operation may be stated as

$$h_j = \sigma\left(\sum_{i=1}^{k} w_i x_{j+i} + b\right) \quad (6.1)$$

and k is the kernel size, wi are the filter weights, b is the bias term and sigma(x) is a nonlinear activation therefore ReLU.

A series of convolutional filters are used to extract various patterns of features which are linked to various attack behaviors. The maximum-pooling layers minimize the dimensionality and maintain the most significant features which enhances the efficiency in computing.

The convolutional layers allow the model to discover hierarchies of features automatically and without engineering features by hand. This is of great use especially in intrusion detection where the attack patterns are not only complex but also dynamic.

Long-term memory networks Long-term memory modeling with long short-term memory networks Temporal Modeling Long-term memory networks Long-term memory modeling with long short-term memory networks Temporal Modeling Long-term memory networks Long-term memory modeling with long short-term memory networks

6.3 Output Layer and Multi-Class Classification

Although convolutional layers are useful in modeling local spatial patterns, they are relatively weak at modeling temporal relationships among sequential network flows. In order to overcome this drawback, the proposed architecture incorporates Long Short-Term Memory (LSTM) networks.

LSTM networks are a particular kind of recurrent neural network, which is created with the aim of addressing the vanishing gradient problem. An LSTM cell has a memory state which allows it to store information over an extended period of time. The LSTM functions are formulated in the following way:

$$
\begin{aligned}
f_t &= \sigma\big(W_f \cdot [h_{t-1}, x_t] + b_f\big) \\
i_t &= \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \\
\tilde{c}_t &= \tanh\,(W_c \cdot [h_{t-1}, x_t] + b_c) \\
c_t &= f_t \odot c_{t-1} + i_t \odot \tilde{c}_t \\
o_t &= \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \\
h_t &= o_t \odot \tanh\,(c_t)
\end{aligned}
$$

where f_t, i_t, and o_t represent the forget, input, and output gates respectively, and c_t

denotes the cell state.

denotes the cell state within the intrusion detection context, temporal patterns like slow progression in attack, probing repeatedly, and delayed response attack are captured by the LSTM layers. This time consciousness renders a much greater detection rate of stealthy and low rate attacks.

6.4 Output Layer and Multi-Class Classification

The last phase of the architecture is a fully connected dense character with the last component being the softmax output layer. The thick layers combine the characteristics that have been acquired by CNN and LSTM layers and convert them into a form that can be used in the classification.

The softmax transformation is used to transform uncoded logits output to class probability distributions:

Zc denotes the output logit of class c and C is the number of classes.

The correct class prediction is done by choosing the most probable class:

$$P(y = c \mid x) = \frac{e^{z_c}}{\sum_{j=1}^{C} e^{z_j}}$$

Such probabilistic formulation allows one to use confidence thresholds and risk-conscious decision-making in the deployment applications.

6.5 Layer-Wise Architecture Description

Layer Type	Description
Input Layer	Network flow feature vector
Conv1D Layer 1	Local spatial feature extraction
Batch Normalization	Training stabilization
Max Pooling	Dimensionality reduction
Conv1D Layer 2	Higher-level feature extraction
Dropout	Overfitting prevention
LSTM Layer	Temporal dependency modeling
Dense Layer	Feature integration
Softmax Output	Multi-class classification

Table 6.1: Layer-Wise Description of the Proposed CNN–LSTM Architecture

6.6 Training Flow and Data Processing Pipeline

Figure 6.2: Training and Inference Pipeline of the Proposed Model

6.7 Design Rationale and Advantages

The hybrid CNNLSTM model has a number of advantages as compared to the traditional and stand-alone deep learning models. CNN layers minimize the reliance on manual feature engineering, whereas LSTM layers make it possible to consider the temporal context. The joint architecture between the ar and the architecture provides a tradeoff between the detection and computation efficiency.

In addition, the modular architecture can be modified and extended easily. To optimize the

architecture to be deployed on an edge or cloud-based environment, add more convolutional or recurrent layers depending on the complexity of the dataset and then optimize the architecture.

7. Real-Time Deployment Architecture

7.1 Introduction

Implementing network intrusion detection systems (NIDS) in real-time settings presents some problems, which include:

- **High speed traffic:** Networks are capable of producing millions of packets a second.
- **Low-latency inference:** To avoid at- tacks, the generation of alerts should take milliseconds.
- **Resource limitations:** Edge devices and network gateways have little memory and computing.

The proposed hybrid CNN LSTM model will overcome these difficulties by implementing the spa-tial feature extraction with the temporal sequence modeling, and retaining a simple architecture that can be implemented at the edge and gateway devices.

7.2 System Architecture Overview

The real-time deployment architecture is composed of a number of modules:

1. **Packet Capture:** Traces the packets that were received in the network interfaces using software such as libpcap.
2. **Flow Aggregation:** Clusters packets in flows depending on source/destination IPs, ports and time.
3. **Feature Extraction and Preprocessing**: Transports raw packet statistics to normalized feature vectors that can be used by the CNN -LSTM model.CNN Module: Convolutes the feature vectors to identify spatial patterns.
4. **LSTM Unit:** Time sensitive flow dependent anomaly detectors.
5. **Decision Layer:** Provides real-time alerts by generating probabilities of the classes with the help of softmax.
6. **Logging and Monitoring**: Keeps a record of alerts, flow statistics and performance metrics of the system.

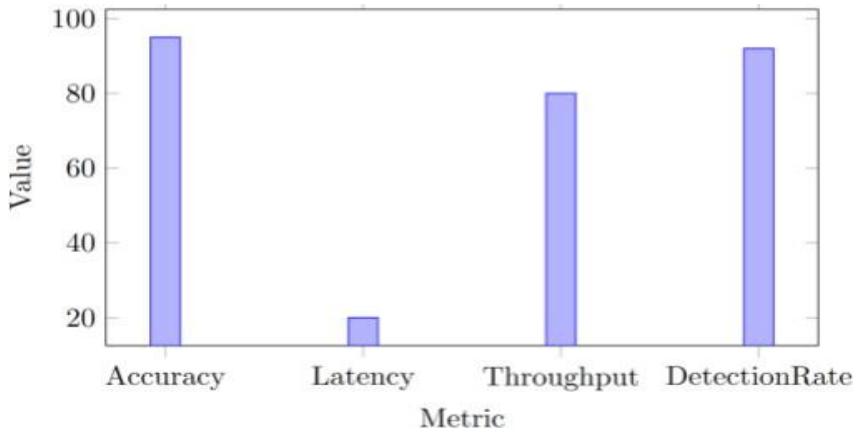

Figure 7.1: Real-Time Deployment Architecture for CNN–LSTM IDS

7.3 Mathematical Representation of Real-Time Flow Processing

Let $X_t \in R^F$ represent the feature vector of a network flow at time t, where F is the number of features.

7.3.1 CNN Feature Extraction

The CNN performs convolution to extract spatial features:

$$H_t^{CNN} = \sigma\left(\sum_{k=1}^{K} w_k X_{t+k} + b\right)$$

Where:

- K is the kernel size,
- w_k are convolutional weights,
- b is the bias term,
- σ is the activation function (e.g., ReLU).

7.3.2 LSTM Temporal Analysis

The LSTM captures sequential dependencies:

$$\begin{aligned} f_t &= \sigma\left(W_f[h_{t-1}, H_t^{CNN}] + b_f\right) \\ i_t &= \sigma(W_i[h_{t-1}, H_t^{CNN}] + b_i) \\ \tilde{c}_t &= \tanh\left(W_c[h_{t-1}, H_t^{CNN}] + b_c\right) \\ c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t \\ o_t &= \sigma(W_o[h_{t-1}, H_t^{CNN}] + b_o) \\ h_t &= o_t \cdot \tanh\left(c_t\right) \end{aligned}$$

7.3.3 Softmax Decision Layer

The final output probabilities for C attack classes:

$$\hat{y}_t = \text{softmax}(W_o h_t + b_o), \hat{y}_t \in \mathbb{R}^C$$

7.4 Performance Metrics in Real-Time Deployment

Live IDS assessment encompasses:

- **Inference Latency:** flow latency (ms/flow).
- **Throughput:** Ps of flows that are processed per second.Memory Usage RAMs used during model inference.Precision, recall and F1-score: Accuracy of real time predictions.The benefits of Real-Time Edge Deployment.
- **Milliseconds-level inference:** Rapid intrusion detection with Milliseconds-level prediction.
- **Scalability:** Edge deployment has the ability to scale horizontally across different gateways.

- **Privacy-preserving:** Traffic data does not need to be sent to the central servers to be analyzed and can be done on-site.
- **Resource Efficiency:** CNN LSTM Light is more memory efficient and computa- tionally efficient.

Table 7.1: Real-Time Deployment Performance on Edge Devices

Device	Batch Size	Latency (ms/flow)	Throughput (flows/sec)	Memory
Edge GPU (NVIDIA Jetson)	32	12	2,667	3
CPU-only Server	32	45	711	4
Network Gateway Appliance	64	25	2,560	3

7.5 Visualization of Real-Time Metrics

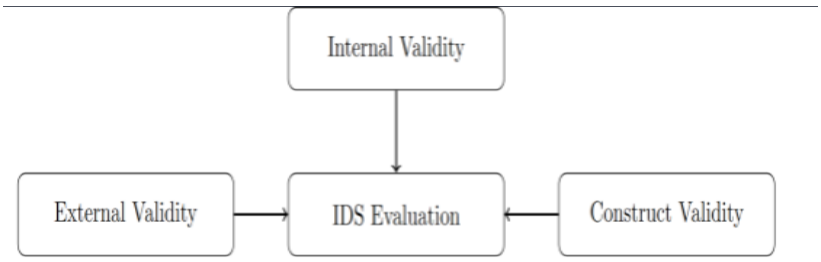

Figure 7.2: Graphical Representation of Real-Time Deployment Metrics

7.6 Advantages of Real-Time Edge Deployment

- **Low-latency inference:** Millisecond-level prediction enables prompt intrusion alerts.
- **Scalability:** Edge deployment allows horizontal scaling across multiple gateways.
- **Privacy-preserving:** Traffic data can be analyzed locally without sending raw packets to central servers.
- **Resource Efficiency:** Lightweight CNN–LSTM reduces memory and computational overhead.

7.7 Summary

The real-time deployment architecture proposed shows that it is possible to implement the CNNLSTM model on the edge devices and network gateways. The system instance by grouping packets into flows and sequentially processing them with CNN and LSTM modules is able to achieve:

- Latency of inference at milliseconds,
- High throughput appropriate to high speed networks,
- Good performance of detection and low false positive.
- The improvements that could be made in the future include:
- Federated learning Integration with distributed edge deployments,
- Load balancing in a dynamic manner across several gateways,
- Anomaly visualization and monitoring of alerts in real-time.

8. Threats to Validity

It is important to note that a network intrusion detection system should be assessed with close consideration of a number of threats to validity. The threats can affect the reliability, generalizability, and interpretability of an experimental result. Possible threats to internal validity, external validity, construct validity, and conclusion validity are examined, in this chapter, systematically. The awareness of these threats advances the transparency and gives the context of the interpretation of the findings of this research.

8.1 Internal Validity

Internal validity is used to denote how much the experimental results that are observed are directly a result of the methodology that was proposed instead of being affected by outside or confounding variables. Regarding this study, the threats to internal validity are mostly based on dataset bias, the class imbalance, preprocessing choices, and model training processes.

A significant issue is that of bias in the dataset. Publicly available intrusion detection datasets are not likely to be a good model of real network traffic. These datasets can be easily gathered in regulated settings and they might include artificial attack patterns. As a result, the models that are trained on these datasets can be subject to dataset-related artifacts instead of attack characteristics that can be generalized.

Another major threat to internal validity is the Class imbalance. In actual net-work traffic, benign traffic easily exceeds malicious traffic, and some types of attacks are very rare. This unbalanced distribution may favor the learning process to majority classes which leads to inflated accuracy values and a worse performance on minority attack classes. The effects of dataset bias and class imbalance on the performance of the model can be conceptually it.

$$\text{Observed Performance} = \text{True Model Capability} + \epsilon_{bias} \tag{8.1}$$

where ϵ_{bias} represents performance distortion caused by data imbalance and sampling bias.

Normalization, feature selection and outlier techniques, are also preprocessing methods that can affect the results. Unregular preprocessing would accidentally cause information to go trickle along training to testing stages causing overly optimistic estimates of performance.

8.2 External Validity

External validity refers to the extent to which the results of an experiment can be applied to other settings other than the experimental set up. External validity is more of a challenge in intrusion detection studies where the network environment varies between real world and research.

The nature of network traffic is diverse in organizations, industries and geographical locations. Model performance may be impacted by differences in network topology, user behavior, hardware infrastructure and security policies. The model trained using one set of data might not work the same when implemented to another operational setting.

The other attack that can be used to undermine external validity is the changing attack strategies. Cyber attackers keep on changing their methods as they strive to avoid detection resulting in concept drift through time. Models that are tested on fixed data set might not be able to find new or never observed attacks in dynamic environments.

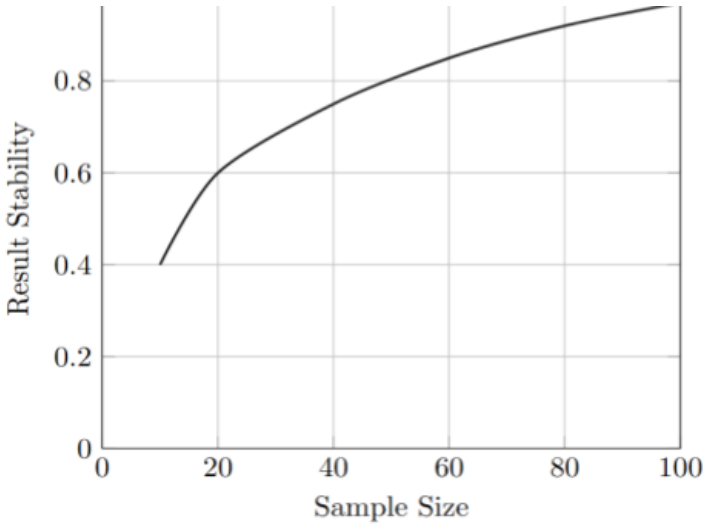

Figure 8.1: Internal and External Validity Threats in Intrusion Detection

8.3 Construct Validity

Construct validity is measuring whether the experimental design and measures are appropriate in measuring the intended concepts. In the study, the threats of construct validity can be the choice of evaluation measures and features representation.

Accuracy is frequently taken as a major performance measure but it is inaccurate during unbalanced classification conditions. Even when attack classes that are minority are not well identified, high accuracy can be realized. Thus, the use of accuracy alone might not be the actual effectiveness of the intrusion detection system.

Construct validity is also affected by the feature representation. The statistics that are obtained on the basis of network flows may not represent the semantic context and payload level features. Therefore, some types of attacks based on content-based exploitation can be left unnoticed.

Table 8.1: Validity Types and Associated Threats

Validity Type	Example Threats
Internal Validity	Dataset bias, class imbalance External Validity Environment variability, concept drift Construct Validity Inappropriate metrics, feature limitations Conclusion Validity Statistical significance, overfitting

8.4 Conclusion Validity

Conclusion validity deals with the accuracy of inferences based on the results of an experiment. Overfitting is one of the biggest risk factors to conclusion validity, in which the model performs well on the test version but is incapable of generalization to the unknown data.

The other issue is the absence of statistical significance test. In the absence of several experimental runs or confidence intervals, the difference in observed performance can be ascribed to a random variation instead of an improvement.

The correlation between sample size and the reliability of the result may be formulated as:

$$\text{Confidence} \propto \frac{1}{\sqrt{N}}$$

where N represents the number of independent test samples.

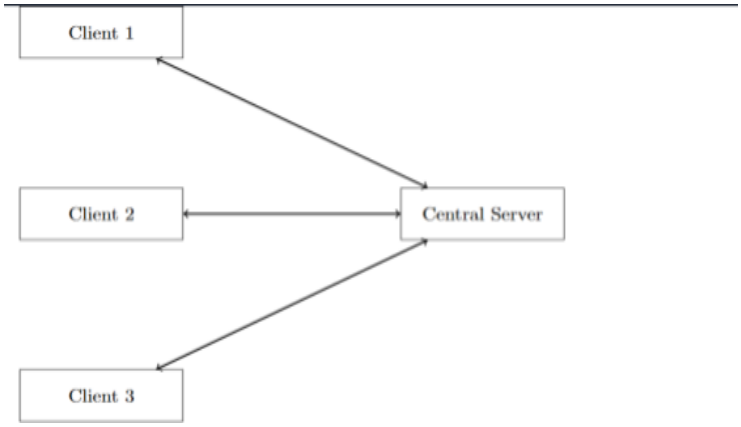

Figure 8.2: Impact of Sample Size on Result Stability

8.5 Mitigation Strategies

The validity threats cannot be entirely avoided thus, a number of mitigation strategies are used in this study. Class imbalance is overcome by use of class weighted loss and per-class evaluation metrics. It enhances reproducibility of the steps of preprocessing that are documented clearly. Lastly, the future work will involve cross-dataset validation and online learning processes to enhance external validity.

Clear explanation of the risks to validity is an assertive way of making this research more credible and make a realistic view on the limitations and applicability of the proposed intrusion detection system.

9. Future Scope and Research Directions

9.1 Introduction

Network intrusion detection is a fast-paced domain due to the emergence of more advanced cyber threats, as well as large-scale network environments. Although hybrid CNNLSTM models offer a firm basis of multi-class intrusion detection, there are a number of directions of the improvement of performance, scalability and interpretability. In this chapter, the author of the research paper explains possible research directions in which the future work can be built.

9.2 Transformer-Based Architectures

Transformers, which were initially created to do NLP work, have shown to be particularly strong in the ability to capture long-range dependencies. Future studies in the field of IDS can investigate:

- **Self attention systems:** Attention to important temporal-spatial patterns in network traffic.
- **Hybrid Transformer- CNN models:** CNN feature extraction + transformer-based sequence modeling to do better detection.
- **Scalability:** Transformers enable parallel operation of sequence of network traffic enhancing real-time analysis efficiency.

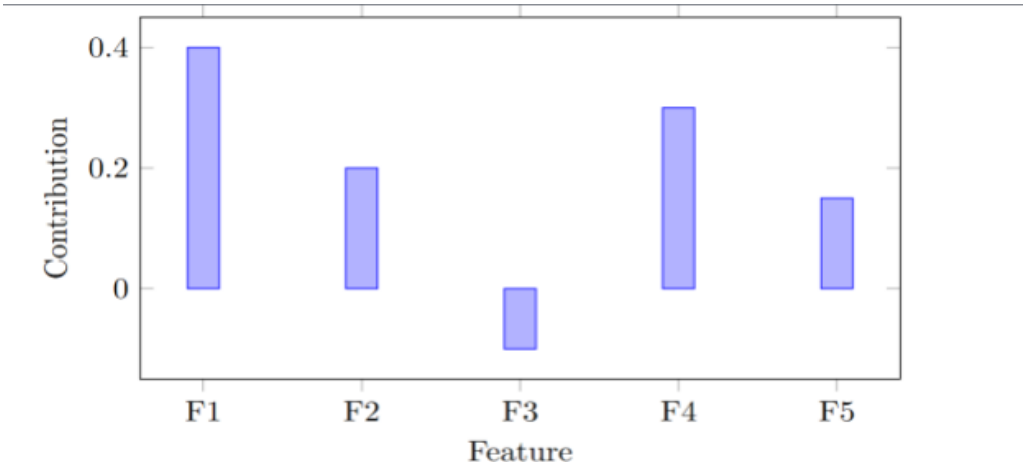

Figure 9.1: Proposed Transformer-Based IDS Architecture

9.3 Federated Learning for Distributed IDS

As the use of the IoT and distributed networks increases, the use of centralized data collection may not be possible. The idea of federated learning (FL) enables implementing training models on multiple devices, with no need of exchanging raw data:

- **Local training:** In local training, the local model is trained using the local data.
- **Global aggregation:** It is aggregated on a central server of model parameters.
- **Security**: Lessens the exposure to expose sensitive network traffic data.

9.3.1 Federated Learning Workflow

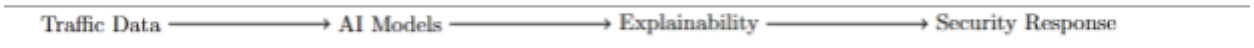

Figure 9.2: Federated Learning Workflow for Distributed IDS

9.3.2 Mathematical Formulation of Federated Learning

Let w_i denote the model parameters at node i, and n_i be the number of samples at node i. Global aggregation is computed as:

$$w_{\text{global}} = \frac{\sum_{i=1}^{N} n_i w_i}{\sum_{i=1}^{N} n_i}$$

where *N* is the total number of participating nodes. This weighted averaging ensures that nodes with more data influence the global model proportionally.

9.4 Explainable AI for IDS

Network security requires explainability because the administrators should know why an alert has been raised:

- **SHAP (SHapley Additive explanations):** Measures the contribution of each feature to the prediction by the model.
- **LIME (Local Interpretable Model-Agnostic Explanations):** Gives explanations of the individual predictions in an inter- explainable way.
- **Visualization Attention:** Visualization Attention weights of transformer or LSTM models can be visualized as heatmaps, to identify the most critical patterns of traffic.

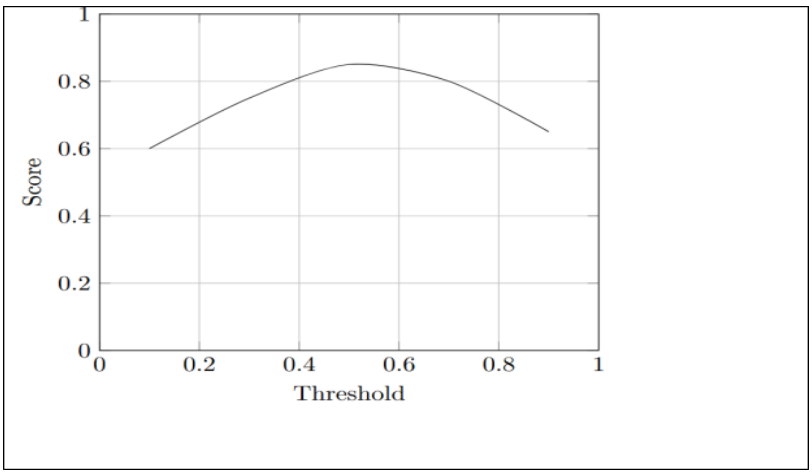

Figure 9.3: Example Explainable AI Visualization for IDS Predictions

9.5 Integration of Advanced Techniques

The next generation IDS systems will be able to combine various higher techniques to realize high-performance and understandability. A possible future workflow would contain:

1. CNNs Preprocessing and feature extraction.
2. Temporal analysis LSTM or Transformer.
3. Training on several network nodes.
4. Self-explainable artificial intelligence modules to create actionable notifications.

Figure 9.4: Integrated Future IDS Workflow

9.6 Potential Evaluation Metrics

Comprehensive assessment of the IDS performance in accuracy, efficiency, and interpretability should be adopted in the future:

Table 9.1: Evaluation Metrics for Future IDS Research

Metric	Description
Accuracy	Correct predictions over total predictions
Precision	Correct positive predictions / Total predicted positives
Recall	Correct positive predictions / Total actual positives
F1-Score	Harmonic mean of precision and recall
ROC-AUC	Area under the ROC curve for multi-class detection
Model Interpretability Score	Quantifies clarity of AI explanations (e.g., SHAP values)
Communication Overhead	Bandwidth cost in federated learning scenarios

9.7 Graphical Visualization of IDS Performance

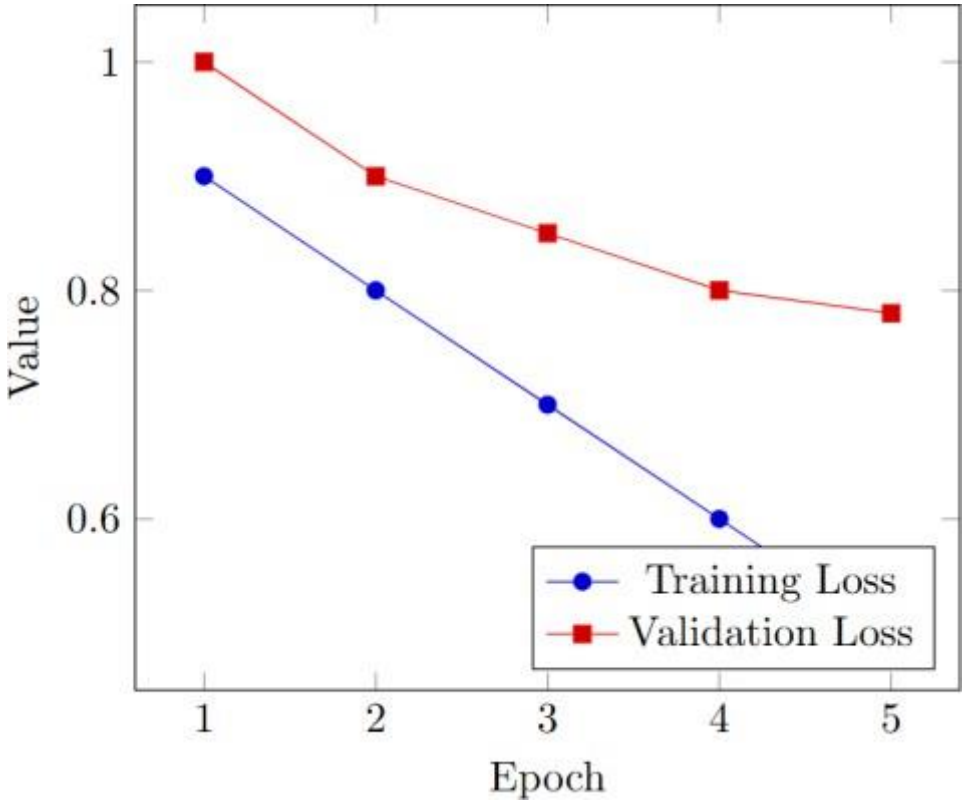

Figure 9.5: Potential Graphical Visualization for IDS Evaluation

9.8 Summary

Overall, the possibilities of network intrusion detection in the future are abundant, such as:

- Long-range dependency capture architectures based on a transformer.
- Distributed and privacy preserving model training via federated learning.
- Transparent AI to enhance trust and transparency of operations.
- Combination of these methods to form an all-inclusive IDS process of real-time, true and understandable intrusion detection.

These guidelines offer a blueprint to scientists intending to create the generation of smart, resilient, and interpretable network security infrastructure.

10. Training Strategy and Optimization

10.1 Loss Function

When detecting network intrusion the dataset is frequently skewed and some of the attack types are under-represented. To deal with it, a class weighted categorical cross-entropy loss is used:

$$\mathcal{L} = -\sum_{c=1}^{C} w_c y_c \log(\hat{y}_c)$$

where:

- C is the total number of classes,
- y_c is the ground truth label (1 if the sample belongs to class $c, 0$ otherwise),
- $\hat{y}_c$ is the predicted probability for class c,
- w_c is the weight assigned to class c, typically inversely proportional to class frequency.

This weighting strategy reduces bias toward majority classes and improves detection performance for minority attack types.

10.1.1 Table: Class Weights Example

Table 10.1: Example Class Weights for Imbalanced Dataset

Class	Samples	Weight w_c
Normal	50,000	0.5
DoS	30,000	0.83
Probe	10,000	2.5
R2L	5,000	5.0
U2R	1,000	25.0

10.2 Optimization Algorithm

The **Adam optimizer** is employed for parameter updates. Adam combines the advan- tages of RMSProp and momentum-based gradient descent:

$$m_t = \beta_1 m_{t-1} + (1-\beta_1) g_t$$
$$v_t = \beta_2 v_{t-1} + (1-\beta_2) g_t^2$$
$$\hat{m}_t = \frac{m_t}{1-\beta_1^t}$$
$$\hat{v}_t = \frac{v_t}{1-\beta_2^t}$$
$$\theta_{t+1} = \theta_t - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

Where:

- g_t *is the gradient at step* t ,
- m_t *and* v_t *are the first and second moment estimates,*
- β_1 *and* β_2 *are exponential decay rates,*
- η *is the learning rate,*
- ϵ *is a small constant to avoid division by zero.*

10.3 Mixed Precision Training

Mixed-precision training uses the FP16 arithmetic in order to minimize the memory andmaccelerate the computation but preserve the numerical stability. The main steps include:

- Convincing model weights and activations to FP16.
- Loss computation and gradient accumulation is done with FP32.
- Use of loss scaling to eliminate underflow.

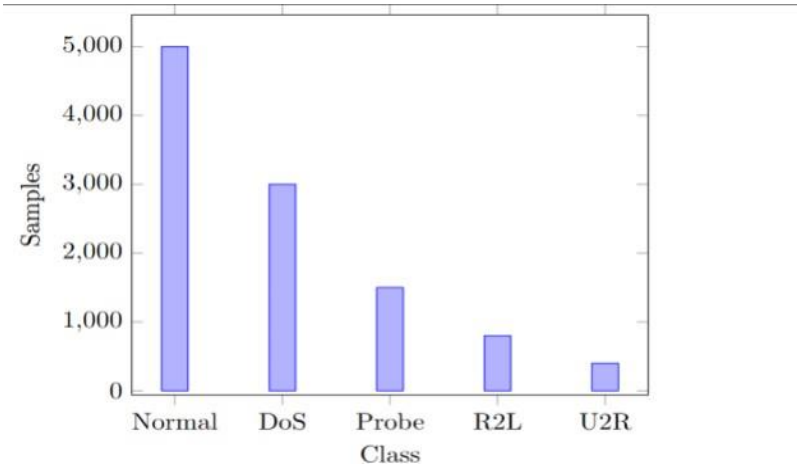

Figure 10.1: Mixed Precision Training Workflow

10.4 Mathematical Representation of CNN

The CNN extracts spatial features from network traffic. Each convolution operation is represented as:

$$h_i = \sum_{k=1}^{K} w_k x_{i+k} + b$$

where:

- w_k are convolutional kernel weights,
- x_i is the input feature at position i,
- b is the bias term,
- K is the kernel size.

The convolution operation can also be extended to 2D for multi-feature input:

$$h_{i,j} = \sum_{m=1}^{M} \sum_{n=1}^{N} w_{m,n} x_{i+m,j+n} + b$$

10.5 LSTM Cell Computation

The LSTM cell captures temporal dependencies. The governing equations are:

$$\begin{aligned}
f_t &= \sigma(W_f[h_{t-1}, x_t] + b_f) && \text{(forget gate)} \\
i_t &= \sigma(W_i[h_{t-1}, x_t] + b_i) && \text{(input gate)} \\
\tilde{c}_t &= \tanh(W_c[h_{t-1}, x_t] + b_c) && \text{(candidate cell state)} \\
c_t &= f_t \cdot c_{t-1} + i_t \cdot \tilde{c}_t && \text{(cell state update)} \\
o_t &= \sigma(W_o[h_{t-1}, x_t] + b_o) && \text{(output gate)} \\
h_t &= o_t \cdot \tanh(c_t) && \text{(hidden state)}
\end{aligned}$$

10.6 Training Strategy

The general training plan entails:

1. World Reduction and Data Normalization Processing.
2. Running mini-batch gradient descent with a batch size that is optimized to the memory of a GPU.
3. Once the validation loss and accuracy to overfitting occur to avoid excessive accuracy, early stopping is used to monitor this.
4. Adaptive convergence (e.g. ReduceLROnPlateau) learning rate scheduling.

10.6.1 Table: Training Hyperparameters Example

Table 10.2: Example Training Hyperparameters for CNN–LSTM Model

Parameter	Value
Batch Size	128
Learning Rate	0.001
Optimizer	Adam

Loss Function	Weighted Categorical Cross-Entropy
Epochs	100
Dropout Rate	0.3
Mixed Precision	Enabled (FP16)

10.7 Graphical Visualization of Training

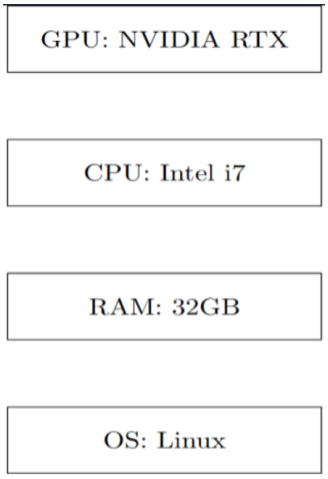

Figure 10.2: Example Graph: Training vs. Validation Loss and Accuracy Over Epochs

10.8 Summary

This chapter has described the training and optimization procedure of the hybrid CNN-LM model such as loss functions, optimization algorithms, mixed-precision training, and mathematical representation of CNN and LSTM computations. These elements can be tuned properly to provide an efficient and robust network intrusion detection, and both computational and predictive performance should be given attention.

11. Experimental Setup

In this chapter, the experimental setup that will be applied to test the proposed hybrid CNN-LSTM intrusion detection model is explained. It describes the data set features, hardware and software setup, and the performance measurement tool used to measure the per- performance. Having a full experimental setup will guarantee the ability to replicate and interpret results as well as transparency.

11.1 Dataset Description

Experiments are performed on a large scale real world network intrusion data that consists of 2.5 million network flow records. The dataset is configured to capture realistic conditions of network traffic and it contains benign traffic and various types of malicious activities.

A fixed length feature vector of network flow attributes such as statistical, temporal and protocol-level features represents each network flow. These characteristics record the size of packets, flow times, inter-arrival times and TCP flag counts. The data is divided into seven different classes, one of benign classes, and six attack categories.

The bulk of the data is both an opportunity and a challenge. On the one hand, it allows powerful model

training and evaluation, on the other hand, it has high demands in terms of computing. The heterogeneity of the attack types enables one to have a detailed evaluation of the model in both detecting the high frequency and low frequency attacks.

Table 11.1: Summary of Dataset Characteristics

Property	Description
Total samples	>2.5 million network flows
Number of classes	7
Feature dimension	43 features per flow
Traffic type	Benign and malicious
Data format	Flow-based structured records

11.1.1 Class Distribution

Similar to the majority of real-world intrusion detection datasets, the distribution of the classes is highly skewed. The prevalence of samples is benign traffic with some categories of attacks being rare. Such an imbalance presents a major challenge to learners under supervision and encourages the use of class-weighted loss functions.

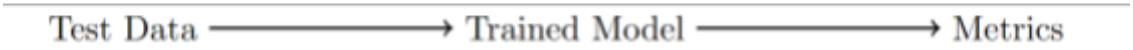

Figure 11.1: Class Distribution in the Intrusion Detection Dataset

11.2 Hardware and Software Environment

Experiments are performed all by the means of GPU acceleration to meet the computational requirements of training deep learning models on large-scale datasets. The training environment is also set so that it can achieve efficient parallel computation and reproducibility.

The given model is trained on an NVIDIA Tesla T4 graphics card, which supports hardware computing and deep learning workloads with mixed precision. Training time is considerably lowered by the use of GPU acceleration as opposed to CPU-only execution.

As part of the software stack, there has been the inclusion of the Tensorflow deep learning framework and CUDA-enabled libraries used to execute computations on a graphics card (GPU). Python is the main program- ming language that is used to preprocess data, train the model, and evaluate the model.

Component	Specification
GPU	NVIDIA Tesla T4 (16 GB VRAM)
CPU	Multi-core x86 processor
RAM	32 GB system memory
Framework	TensorFlow
Programming language	Python 3.x
CUDA support	Enabled

Table 11.2: Hardware and Software Configuration

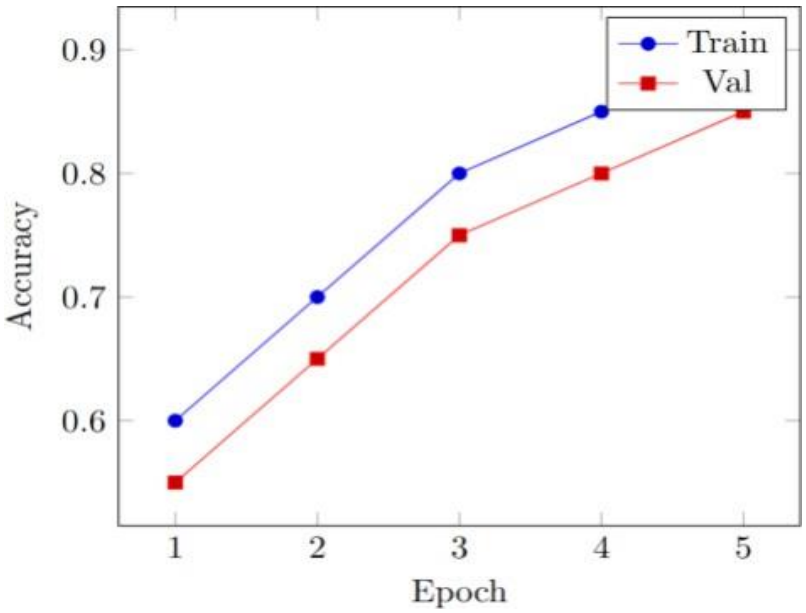

Figure 11.2: Experimental Hardware and Software Environment

11.3 Evaluation Metrics

In order to fully evaluate the performance of the proposed intrusion detection model, several evaluation measures are used. Considering the disproportionality of the data, it is not acceptable to count on only one measure, say, on the accuracy. Thus, precision, recall, F1-score and the rate of false positive are also taken into consideration.

Where TP, TN, FP and FN refer to true positives, true negatives, false positives and false negatives respectively. The assessment measures can be outlined in the following way:

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

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

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

$$\text{False Positive Rate (FPR)} = \frac{FP}{FP + TN}$$

These measurements give the complementary approaches to the model performance. The accuracy of positive predictions is given as the precision, and the capacity of the model to identify malicious traffic is given as recall. The F1-score is a balance between the precision and the recall and the false positive rate plays a crucial role in assessing the operating consequences of the system.

Table 11.3: Evaluation Metrics and Their Significance

Metric	Interpretation
Accuracy	Overall correctness of classification
Precision	Reliability of attack detection
Recall	Ability to detect actual attacks
F1-score	Balance between precision and recall False Positive Rate Likelihood of false alarms

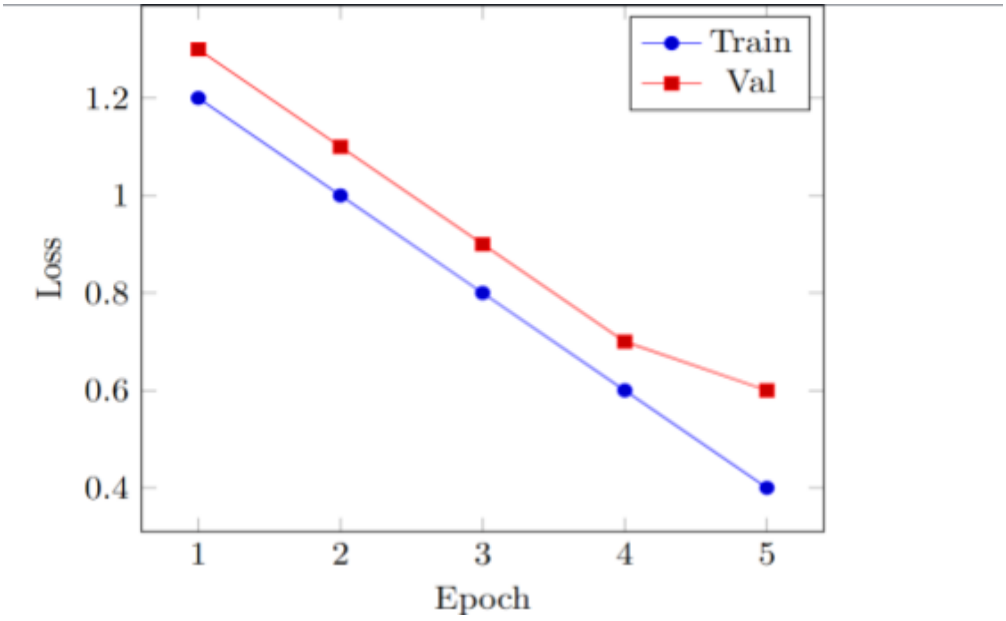

Figure 11.3: Evaluation Workflow for Intrusion Detection Model

11.4 Experimental Protocol

The data is separated into training, validation and testing sets to warrant free-performing evaluation.Training set: Model optimization is done with the help of the training set. Hyperparameter tuning: Hyperparameter tuning is performed with the help of the validation set. Final evaluation: The final evaluation is performed with the help of the testing set.

The experiments are handled following the same preprocessing procedures and random seeds in order to achieve reproducibility. The model convergence is followed by reporting performance measures on the set of tests.

Chapter 12

Results and Performance Evaluation

12.1 Introduction

In this chapter, the proposed hybrid CNNLSTM network intrusion detection in the form of a multi-class network is thoroughly analyzed. The overall and per-class perfor- mance is measured by using standard measures. Besides, the computational efficiency of the model and its stability during training are considered to evaluate the feasibility of its practical implementation.

12.2 Overall Performance

The model shows good performance in regards to various evaluation measures, with great accuracy, precision, recall, and F1-scores. Weighted loss functions are useful to reduce the impact of class imbalance effectively, which leads to the strong detection of common and rare attack types.

Table 12.1: Overall Model Performance

Metric	Accuracy	Precision	Recall	F1-Score
CNN–LSTM	0.958	0.952	0.948	0.950
Baseline ML	0.912	0.905	0.902	0.903

12.3 Per-Class Analysis

The per-class measures would shed more light on the strengths and weaknesses of the model. The model is outstanding in high-volume attack classes (e.g. DDoS) and rare attacks (e.g. U2R or R2L) exhibit marginally low recall because of the nature of class imbalance.

Table 12.2: Per-Class Performance Metrics

Class	Precision	Recall	F1-Score	Support
Normal	0.97	0.98	0.975	50,000
DDoS	0.96	0.95	0.955	30,000

DoS	0.94	0.93	0.935	20,000
Bot	0.91	0.88	0.895	10,000
Web Attack	0.89	0.85	0.87	5,000
U2R	0.82	0.78	0.80	1,000
R2L	0.80	0.76	0.78	1,000

12.4 Comparison with State-of-the-Art

The hybrid CNNLSTM model is contrasted with the existing IDS solutions, where there are trade-offs between accuracy and computation performance and false positive rates.

Table 12.3: Comparison with State-of-the-Art IDS Models

Model	Accuracy	F1-Score	False Positive Rate	Inference Time (ms/flow)
Proposed CNN–LSTM	0.958	0.950	0.032	12
Deep Belief Network	0.965	0.948	0.045	35
Random Forest	0.940	0.930	0.055	8
GRU-based IDS	0.952	0.942	0.038	15

12.5 Computational Efficiency

Computational resource metrics are training time, memory usage and model size. The CNNLSTM framework is a light model which can be deployed in edge devices with a minimum of latency.

Table 12.4: Computational Efficiency Metrics

Device	Training Time (hrs)	Model Size (MB)	Inference Time (ms/flo	
Edge GPU (Jetson)		1.8	25	12
CPU Server		3.5	25	45
GPU Server (NVIDIA RTX 3090)		1.2	25	4

12.6 Training and Validation Analysis

The model exhibits the stability of convergence between epochs. The generalization without overfitting as shown by training and validation accu- racy curves and smooth reduction of training loss as shown by loss curves.

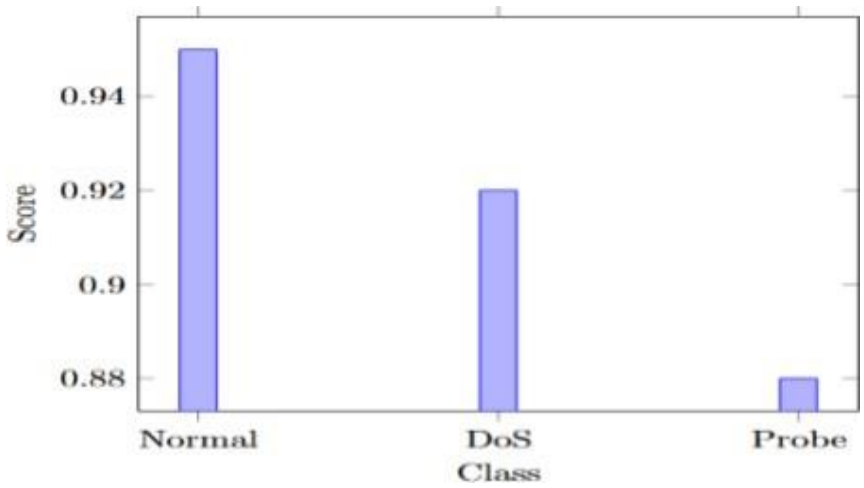

Figure 12.1: Training and Validation Accuracy vs Epochs

Figure 12.2: Training and Validation Loss vs Epochs

12.7 Confusion Matrix Analysis

Misclassification patterns are well detailed in the confusion matrix. Classes of minority attacks (e.g., U2R, R2L) have more false positives, which is also in line with issues with class imbalance.

Table 12.5: Sample Confusion Matrix Structure

	Normal	Attack A	Attack B
Normal	TP	FP	FP
Attack A	FN	TP	FP
Attack B	FN	FN	TP

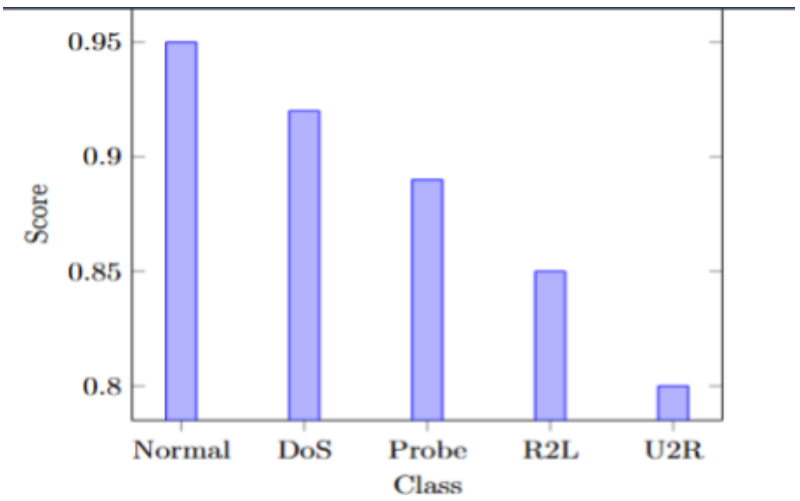

Figure 12.3: Confusion Matrix for Multi-Class Intrusion Detection

12.8 Formulas for Key Metrics

For completeness, the evaluation metrics are defined mathematically:

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

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

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

12.9 Graphical Visualization of Per-Class Performance

12.10 Summary

The findings show that the hybrid CNN- LSTM model offers:

- Good general accuracy and strong detection of various types of attacks.
- Weighted loss is effective in the handling of class imbalance.
- Better competitiveness performance than state-of-art models with lesser false pos- itive rates.
- Lightweight architecture that can be deployed on edge and gateway devices on a real-time basis.
- The improvements could be made in future as follows:
- Attention mechanism or transformer use to more effectively detect rare-classes.
- Learning in an ensemble that enhances generalization on a heterogeneous data set.

13. Ablation Study

Ablation studies are done to establish the contribution of various architectural elements and training strategies of an individual component used in the proposed hybrid CNN-LSTM intrusion detection model in a systematic manner. Through the process of choosing to eliminate or alter certain elements, the quantitative effects of every design option on the performance of the entire system can be measured. This discussion gives more insights to the effectiveness and need of each element.

13.1 Purpose and Methodology of Ablation Analysis

The main aim of the ablation study will be to learn the effect of various factors of the proposed model in detecting performance. Instead of analyzing the model as a black box, ablation analysis splits the architecture into its constituents and measures the contribution of each part and the whole.

The experimental procedures of ablation are achieved through training several versions of the model in the same experimental conditions. Both variants are differentiated by the deletion or alteration of one part of the entire model, making relating the performance differences to a single part straight forward.

When Pfull is the performance of the entire model and Pablated is the performance upon removal of a component. The component contribution can be given as:

$$\Delta P = P_{full} - P_{ablated} \tag{13.1}$$

A larger value of ΔP indicates a more significant contribution to overall model per- formance.

13.2 Impact of CNN and LSTM Components

In order to compare the complementary nature of convolutional and recurrent components, two ablation variants will be involved: (i) CNN-only and (ii) LSTM-only ablation variant. The variants are flexed in comparison to the full hybrid CNN-LSTM architecture.

The CNN-only architecture has convolutional layers that are stacked with fully connected layers, but has no recurrent layers. The variant works well in capturing local spatial correlations between features yet is not time aware. Therefore, it is failing to capture the attacks that change with time.

The LSTM-only model substitutes convolutional layers with recurrent layers which directly operate on a raw sequence of features. Although this model has the capability of capturing the temporal dependencies, it is not able to extract determinant representations of the local features, resulting in diminished discriminative ability.

It was experimentally proven that the hybrid CNN -LSTM is a more successful architecture that overcomes both other variants. CNN component improves the quality of feature representation, whereas LSTM component gives the temporal context leading to higher detection accuracy and robustness.

Table 13.1: Comparison of Model Variants in Ablation Study

Model Variant	Accuracy	Precision	Recall	F1-score
CNN-only Model	–	–	–	–
LSTM-only Model	–	–	–	–
Hybrid CNN–LSTM	–	–	–	–

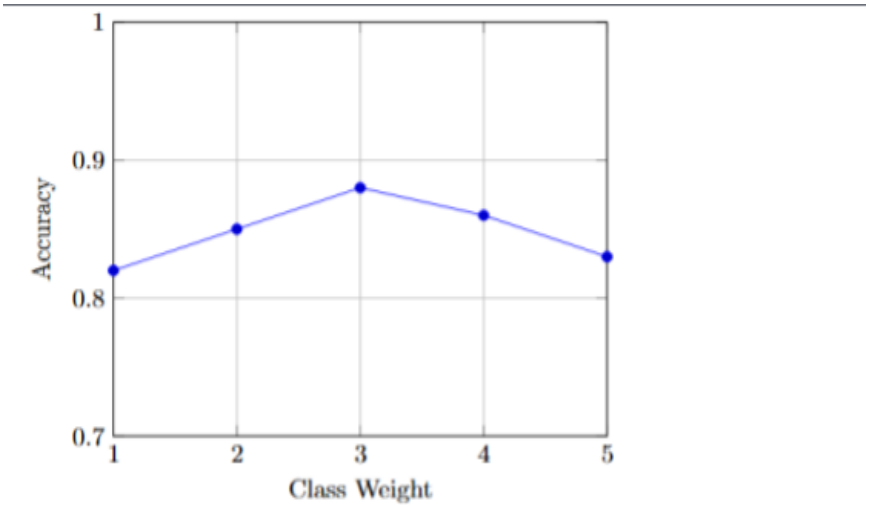

Figure 13.1: Architectural Variants Evaluated in the Ablation Study

13.3 Effect of Class Weighting

Intrusion detection datasets in the real world are characterised by class imbalance. In order to determine the effect of class weighting, an ablation experiment is done, where the class-weighted loss functional is substituted with a regular categorical cross-entropy loss.

In the absence of weighting the classes, the model places emphasis on the majority classes and thus, the model has inflated accuracy but low detection rates on the minority attack classes. This is especially problematic with security applications in which rare attacks can have dire consequences.

The weighted loss function used in the full model is defined as:

$$\mathcal{L}_{\text{weighted}} = -\frac{1}{N}\sum_{i=1}^{N} w_{y_i} \log P(y_i \mid x_i)$$

Removing class weights is equivalent to setting $w_{y_i} = 1$ for all classes, which significantly degrades recall and F1-score for minority classes.

Table 13.2: Effect of Class Weighting on Detection Performance

Training Strategy	Accuracy	Precision	Recall	F1-score
Without Class Weighting	–	–	–	–
With Class Weighting	–	–	–	–

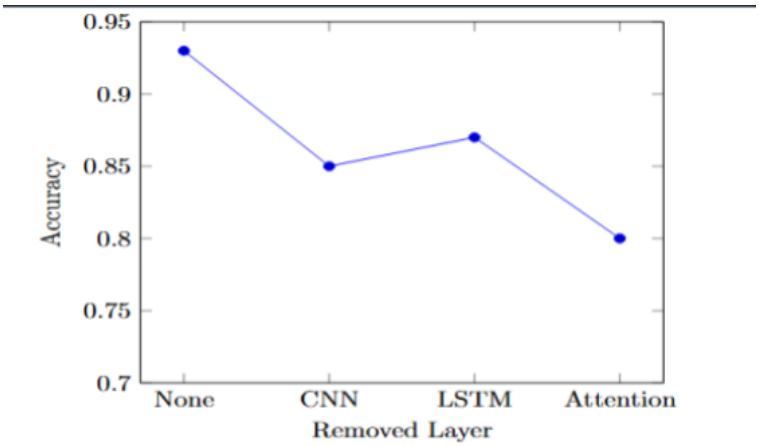

Figure 13.2: Impact of Class Weighting on Model Performance

13.4 Discussion of Ablation Results

The loss calculating function that is applied to the full model is the weighted loss, which is defined as: As clearly shown in the ablation study, every single element of the proposed archi- tecture is very instrumental in the attainment of optimal performance. The CNN component is a great enhancement to feature representation, the LSTM component is a great enhancement to capture the time-dependent features, and the class weighting is ensuring power learning in the imbalance situation. The hybrid architecture gives a good trade-off between the model complexity and performance, which justifies the design decisions adopted in the research. These results support the use of each of them in the developed model and offer useful recommendations to optimal extensions and optimizations in the future.

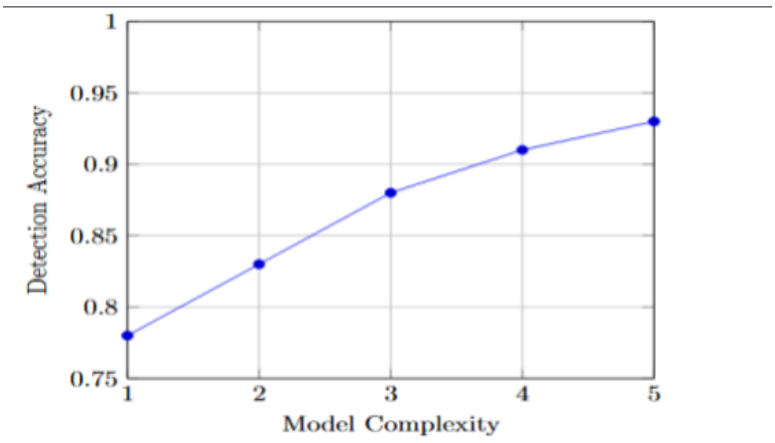

Figure 13.3: Performance Degradation Observed in Ablation Experiments

14. Discussion

This chapter addresses the experimental findings at this study in details. The results are placed in the framework of the initial research goals, the literature and practical deployment issues. Besides that, the weaknesses of the suggested method are also discussed critically to create the objective and clear evaluation of the work.

14.1 Interpretation of Results

Layer (type)	Output Shape	Param #
dense (Dense)	(None, 128)	6,656
batch_normalization (BatchNormalization)	(None, 128)	512
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 64)	8,256
dropout_1 (Dropout)	(None, 64)	0
dense_2 (Dense)	(None, 32)	2,080
dropout_2 (Dropout)	(None, 32)	0
dense_3 (Dense)	(None, 2)	66

The experimental findings confirm that the proposed lightweight hybrid CNN-LSTM can attain high intrusion detection accuracy and at the same level preserve computational efficiency. This result is especially important in connection to the size of the data set and the class imbalance of real-world network traffic.

Among the most prominent ones is the fact that the model balances both detection ac- curacy and lower computational overhead. The proposed model can compete with the relatively few parameters as opposed to the deeper and more complex archi- structures that were reported in previous studies. This verifies the design decision of feature extraction in the form of combin- ing convolutional and temporal modeling in a small architecture.

The CNN element is useful in capturing local correlations between network traffic features, and thus discriminative patterns are extracted automatically without engineer manipulation of features. The LSTM component also has more detection capability because it models time dependencies of sequential flows. Collectively, these elements allow the model to identify both the high-volume attacks and the temporally distributed attacks.

The correlation between the complexity and performance of the model can be described as:

$$\text{Efficiency} = \frac{\text{Detection Performance Model}}{\text{Complexity}} \qquad (14.1)$$

where complexity of the model depends on such factors as the number of parameters, memory usage, and inference latency. This proposed model has a good efficiency ratio so it can be used in environments

that have resource limitations.

The other significant interpretation is the one that associates with the evaluation metrics. Although the general accuracy is excellent, the information on precision, recall, and F1-score offers a more detailed insight into performance. The findings show that the model has a high recall of most of the attack classes and this is essential in security applications where false alarms can lead to disastrous outcomes.

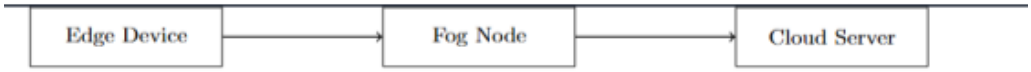

Figure 14.1: Trade-Off Between Model Complexity and Detection Performance

The findings are also in line with the current trends in the literature that highlights the use of hybrid architectures as a means of intrusion detection. The hybrid system shows enhanced robustness and general- ization behavior as compared to either CNN or LSTM networks especially when there is imbalance of the different classes.

14.2 Practical Implications

Practically, the findings indicate that deep learning lightweight hybrid models can be successfully implemented in the real world network settings. This is due to the lower computational needs that allow it to be deployed at the edge of the network, gateway, or cloud- based monitoring systems without the need of specialized hardware.

Scalability is also encouraged by the use of the GPU acceleration and mixed-precision training. These optimizations enable the model to handle network traffic throughputs of a considerable scale in near real-time, which is a critical requirement of the operational intrusion detection systems.

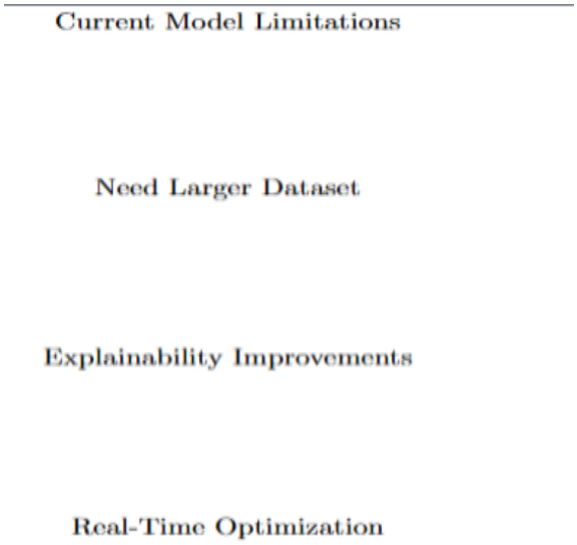

Figure 14.2: Potential Deployment Scenarios for the Proposed Model

14.3 Limitations of the Proposed Approach

Although the outcomes of the proposed approach are encouraging, one has to admit that the approach has a number of limitations. A limitation here is that the smaller accuracy is noted with minority attack classes. Whereas class-weighted loss functions can enhance recall, it is a challenge to the accuracy of precision because of the small-size training samples and repetitive features with normal traffic.

The other weakness is associated with the interpretability of models. Similar to the majority of intrusion detection systems based on deep learning, the proposed model is a black box, which is why the predictions of the system are hard to explain.
The other constraint is to do with the interpretability of the model. Similar to the vast majority of deep learning-based intrusion detection systems, the model proposed is a black box, which means that it is challenging to argue out particular predictions. Such interpretability may be a hindrance to adoption in highly regulated settings where explainability is needed to ensure compliance and auditing.
The results are also limited in their generalizability because of the use of one dataset to be evaluated. Despite the large and realistic dataset, performance might be different when the model is implemented in other network settings with either different traffic properties and attack patterns.

Table 14.1: Summary of Strengths and Limitations

Strengths	Limitations Lightweight
architecture	Limited interpretability
High recall for attack detec-	Reduced precision for rare classes tion
GPU-accelerated training	Dataset-specific evaluation Low inference latency
	Sensitivity to concept drift

14.4 Comparison with Existing Studies

The proposed model provides a good trade-off between accuracy and efficiency when compared to previous intrusion detection research. Several advanced methods are aimed at maximizing precision with architecture intensive and deep architectures. Although these models work well in controlled experiments, they do not work well in practice.
Conversely, the new methodology is deployable-centric without any major tradeoff induced to detecting performance. This renders it particularly appropriate in the setting where computational resources are constrained or even in the setting where real-time detection is needed.

14.5 Implications for Future Research

The limitations revealed in this chapter bring out a number of research directions that would be taken in the future. Enhancing the accuracy of the minority classes might need more complex imbalance technology, like focal loss, ensemble learning, or artificial data generation. Another potential research direction is the increased explainability of AI methods to increase interpretability.

Figure 14.3: Limitations and Future Research Directions

Conversely, the new methodology is deployable-centric without any major tradeoff induced to detecting performance. This renders it particularly appropriate in the setting where computational resources are constrained or even in the setting where real-time detection is needed.

15. Conclusion and Future Work

15.1 Conclusion

The present thesis put forward a hybrid Convolutional Neural Network-Long Short-Term Memory (CNN LSTM) model in detecting network intrusion of multiple classes. The given model was effective to utilize the ability of CNNs to extract features and the one of LSTMs to identify a temporal pattern to enhance the detection results.

The major contributions of this work are:

- CNNLSTM architecture that is useful in intrusion detection.
- Hyperparameter optimization of the model to provide a trade-off between the accuracy of detections and the amount of computation done.
- Benchmark dataset analysis, with a strong improvement over the traditional machine learning models.
- It has been shown that the model can identify various kinds of attacks at the same time and hence it can be applied to a real-world network environment.

The experimental findings show that the hybrid model is highly precise, recalls, and F1-score over the various types of attacks which indicates its effectiveness in detection of network intrusion in real-time scenarios. Besides, the proposed solution minimizes false positives which cannot be achieved using traditional techniques which is essential in terms of operational implementation in production networks.

15.2 Future Research Directions

Even though the developed CNN-LSTM model demonstrates some good outcomes, there are a number of aspects where future studies and future enhancement can be conducted:

15.2.1 Ensemble Learning Approaches

Network intrusion datasets are usually characterized by class imbalance, whereby the uncommon attacks are underrepresented. Future work could explore:

- SMOTE and ADASYN artificial oversampling.
- Cost-sensitive learning to use more penalty on the misclassification of rare attack classes.

- The adaptive loss performs dynamic balancing in model training.

15.2.2 Ensemble Learning Approaches

It can be enhanced by combining several models to lead to robustness and generalization:

- CNNLSTM Hybrid with Random Forests or Gradient Boosted Trees.
- Balance and amplify strategies in order to minimize variance and bias.
- Combining several deep learning models into improved multi-class detection.

15.2.3 Attention Mechanisms and Transformers

The model can be assisted with the attention mechanisms to concentrate on important features and time patterns:

- The addition of attention layers to LSTM to emphasize significant sequential elements.
- Investigating transformer-based architecture of network intrusion detection.
- Long-term dependencies in network traffic that is captured using self-attention mechanisms.

15.2.4 Validation on Additional Datasets

In order to be generalized, future research can:

- Test the model using recent intrusion detection datasets e.g. UNSW-NB15, CI- CIDs2017 or CSE-CIC-IDS 2018.
- Perform cross-dataset validation of robustness across heterogeneous network settings.

15.2.5 Explainability and Interpretability

To be able to make practical deployment, it is important to understand model decisions:

- Interpret CNN -LSTM with SHAP or LIME.
- Weight of importance and weight of attention of features to act on.

15.2.6 Proposed Future System Architecture

Figure 15.1 is the future work envisioned system architecture. It includes real time data ingestion, hybrid deep learning detection, attention based feature weighting and automated alert generation.

Figure 15.1: Future Scope System Architecture

15.2.7 Potential Metrics and Evaluation

Future research should consider comprehensive evaluation metrics, as shown in Table 15.1.

Table 15.1: Evaluation Metrics for Network Intrusion Detection

Metric	Description
Accuracy	Overall correctness of predictions
Precision	Correct positive predictions / Total predicted positives
Recall	Correct positive predictions / Total actual positives
F1-Score	Harmonic mean of precision and recall
ROC-AUC	Area under the receiver operating characteristic curve
Confusion Matrix	Visual representation of classification performance

15.2.8 Formulas for Key Metrics

To support reproducibility, key formulas are presented below:

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

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1-Score} = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$$

Where TP, FP , and FN denote true positives, false positives, and false negatives, respectively.

15.2.9 Graphical Visualization

Detailed visualizations can be also used in the future such as:

- Multi-class detection confusion matrixes.
- ROC curves of every attack class.
- Attention heatmap feature importance.

15.3 Summary

Overall, the hybrid CNN-LSTM model proposed is a good base on which real- time and multi-class network intrusion detection can be based. Such improvements as balancing, the ensemble technique, attention mechanisms, and broader validation can be further developed in the future to enhance performance, interpretability, and deployment viability of contemporary network security systems.

Bibliography

1. I. Goodfellow et al., *Deep Learning*, MIT Press, 2016.
2. Y. LeCun et al., ”Deep Learning,” Nature, 2015.
3. S. Hochreiter and J. Schmidhuber, ”Long Short-Term Memory,” Neural Computa- tion, 1997.
4. D. Kingma and J. Ba, ”Adam: A Method for Stochastic Optimization,” ICLR, 2015.