Volume: 13 Issue: VIII Month of publication: August 2025
DOI: https://doi.org/10.22214/ijraset.2025.73783

International Journal for Research in Applied Science & Engineering Technology (IJRASET)
ISSN: 2321-9653; IC Value: 45.98; SJ Impact Factor: 7.538
Volume 13 Issue VIII Aug 2025- Available at www.ijraset.com

A Hybrid Deep Learning Approach for Network Intrusion Detection System in Software-Defined Networking with Hybrid Feature Selection

Ch. Divya[1], Vavilapalli Sulochana[2], Kaja Venkata Nitya Santhoshi[3], Indurthy Gokul Surya[4], Karna Veerendranadh[5], Jakkam Chaitanya Manikanta[6]
[1]Assistant Professor, [2,3,4,5,6]UnderGraduate, CSE-Data Science Department, St. Ann's College of Engineering & Technology, Chirala, Andhra Pradesh

Abstract: The increasing adoption of Software-Defined Networking (SDN) introduces flexible and programmable architectures but also creates new security challenges, including vulnerability to Distributed Denial-of-Service (DDoS), botnet, and probing attacks. Traditional intrusion detection systems often fall short in adapting to dynamic SDN environments d ue to high false alarm rates and limited generalization. This paper proposes a hybrid deep learning model—CNN-BiLSTM—designed to detect network intrusions in SDN infrastructures. The proposed approach leverages the spatial feature extraction capabilities of Convolutional Neural Networks (CNN) and the sequential learning power of Bidirectional Long Short-Term Memory (BiLSTM) networks. A hybrid feature selection technique combining Random Forest and Recursive Feature Elimination (RFE) is employed to enhance learning efficiency. Experiments conducted on benchmark datasets including NSL-KDD, UNSW-NB15, and InSDN demonstrate that the CNN-BiLSTM model achieves superior performance in both binary and multiclass classification tasks, outperforming baseline models in accuracy, F1-score, and detection rate. These results confirm the model's effectiveness in enhancing SDN security against evolving cyber threats.
Keywords: SDN, Network Intrusion Detection, CNN-BiLSTM, Hybrid Feature Selection, Deep Learning, Cybersecurity.

I. INTRODUCTION

The evolution of computer networks has led to the emergence of Software-Defined Networking (SDN), a transformative paradigm that decouples the control and data planes, offering centralized control, programmability, and dynamic adaptability. While SDN provides significant benefits for scalability, automation, and management, it also introduces new security vulnerabilities due to its centralized architecture. A compromised SDN controller can potentially jeopardize the entire network, making it a high-value target for adversaries. Intrusion Detection Systems (IDS) are essential in safeguarding SDN environments by identifying and mitigating unauthorized access or anomalous behavior in network traffic. Traditional IDS approaches—primarily based on rule sets or static signature matching—struggle to detect sophisticated, evolving threats such as zero-day attacks, polymorphic malware, and advanced persistent threats. These limitations are further amplified in SDN contexts, where the traffic is dynamic and distributed.

Deep learning-based IDS solutions have shown great promise in addressing these limitations by learning complex patterns from large-scale network data. Models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks have been utilized for their ability to extract spatial and temporal features from network traffic. However, single-architecture models often fail to generalize across diverse attack types or suffer from high false positive rates due to feature redundancy and class imbalance. To address these challenges, this paper proposes a hybrid deep learning model—CNN-BiLSTM—for intrusion detection in SDN environments. The CNN component captures spatial correlations within traffic features, while the BiLSTM network processes sequential dependencies in both forward and backward directions. To further enhance learning efficiency, a hybrid feature selection technique combining Random Forest and Recursive Feature Elimination (RFE) is applied to reduce dimensionality and focus on the most informative attributes.

The key contributions of this work include:

- Development of a robust hybrid CNN-BiLSTM model tailored for real-time SDN intrusion detection.
- Integration of hybrid feature selection to minimize redundancy and improve classification performance.
- Comprehensive evaluation using NSL-KDD, UNSW-NB15, and InSDN datasets across both binary and multiclass settings.
- Comparative analysis against baseline models (e.g., CNN-LSTM, LeNet-5, AlexNet), demonstrating superior accuracy and reduced false alarm rates.

By leveraging the synergy of spatial-temporal learning and intelligent feature selection, the proposed system provides an effective and scalable solution for enhancing SDN security.

II. LITERATURE SURVEY

The increasing sophistication of network-based cyber threats, particularly in programmable and virtualized infrastructures like Software-Defined Networking (SDN), has prompted significant research into intelligent Intrusion Detection Systems (IDS). This section explores traditional and modern approaches to intrusion detection, emphasizing the evolution toward deep learning-based hybrid models that combine spatial and temporal analysis.

A. Intrusion Detection Systems in SDN

Intrusion Detection Systems (IDS) serve as the first line of defense against malicious activity in networks by analyzing traffic patterns and flagging suspicious behaviors. In SDN, the centralized controller becomes a critical component, making it an attractive target for attackers. Network-based IDS (NIDS) must therefore operate in real time, adapt to rapidly changing traffic flows, and detect both known and unknown threats.

Traditional IDS techniques, such as signature-based detection and rule-driven engines, are effective for identifying well-documented attacks but fail against zero-day exploits and polymorphic malware. This has led to the adoption of machine learning (ML) and deep learning (DL) methods capable of learning latent patterns in high-dimensional network data.

B. Traditional vs. Deep Learning-Based IDS

Classical machine learning models—such as Decision Trees, Random Forests, and Support Vector Machines—have been widely used in early IDS research. While these models offer fast inference and interpretability, their performance is limited by manual feature engineering and poor scalability.

In contrast, deep learning models, including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTM) architectures, have demonstrated superior capabilities in automatically extracting high-level features and learning temporal dependencies. However, standalone deep models often suffer from overfitting, require large datasets, and exhibit high computational costs during training.

C. Hybrid Architectures: CNN-BiLSTM Models

Recent literature has shown that hybrid deep learning models, particularly CNN-BiLSTM, outperform standalone architectures by leveraging the strengths of both CNN and BiLSTM networks. CNN layers effectively extract spatial features from structured traffic data, while BiLSTM layers capture bidirectional temporal patterns, allowing for comprehensive modeling of sequential behaviors in multi-stage attacks.

Studies such as [1] and [3] highlight the effectiveness of CNN-BiLSTM in reducing false positives and improving generalization in intrusion detection tasks. These models have been particularly useful in SDN environments where attack signatures may evolve over time or exhibit subtle behavioral shifts.

D. Feature Selection for IDS

Feature redundancy and noise can significantly hinder the performance of deep learning models. Hybrid feature selection techniques—combining algorithms like Random Forest and Recursive Feature Elimination (RFE)—help isolate the most relevant features, reduce computational overhead, and enhance classification accuracy. Tools like SHAP (Shapley Additive Explanations) further aid in understanding the contribution of each feature to the final prediction, improving model interpretability.

E. Research Gaps and Challenges

Despite promising results, several challenges persist in current research:

1) Many models focus on spatial or temporal patterns independently, missing cross-domain interactions.
2) Datasets are often imbalanced, leading to biased learning and poor generalization for rare attack types.
3) Few studies incorporate SDN-specific traffic patterns, limiting real-world applicability.
4) Computational inefficiencies hinder deployment in resource-constrained environments.

F. Related Work

Several notable studies have explored deep learning-based IDS:

1) ElSayed et al. [1] proposed a CNN-LSTM approach but encountered overfitting due to unbalanced training data.
2) Jiang et al. [3] combined CNN-BiLSTM with SMOTE, achieving strong results but lacking focus on SDN environments.
3) Khan et al. [3] presented a two-stage model using autoencoders and SoftMax, though it was limited to the UNSW-NB15 dataset.
4) Tang et al. [4] introduced a DNN for SDN intrusion detection, but it required extensive computational resources and offered modest accuracy (~75%).

These works collectively demonstrate the growing importance of hybrid deep learning models and the need for SDN-specific solutions to enhance network security.

III.PROPOSED METHODOLOGY

This research proposes a hybrid deep learning architecture—CNN-BiLSTM—for robust intrusion detection in Software-Defined Networking (SDN) environments. The model aims to leverage CNN's strength in spatial feature extraction and BiLSTM's ability to capture bidirectional temporal dependencies. To address challenges such as feature redundancy and class imbalance, the framework integrates a hybrid feature selection strategy and data balancing techniques. The complete pipeline involves data preprocessing, feature engineering, model training, and evaluation across multiple benchmark datasets.

A. Data Collection and Preprocessing

The model is trained and validated on three widely used intrusion detection datasets:

- InSDN: An SDN-specific dataset containing attack types like DDoS, Web, Probe, and Botnet.
- NSL-KDD: A benchmark dataset derived from KDD Cup 1999, suitable for binary and multiclass intrusion detection tasks.
- UNSW-NB15: A modern dataset reflecting realistic network traffic patterns and advanced threats.

Each dataset undergoes preprocessing to enhance data quality and consistency. Missing values are handled via row removal or forward-fill imputation, while inconsistent entries across datasets are standardized using volume-weighted averaging.

B. Handling Missing Data and Outliers

To ensure robustness, the following techniques are applied:

- Z-Score and IQR Methods: Applied for outlier detection and removal to reduce noise.
- Forward Fill: Used to address missing values in time-sequenced records.
- Row Removal: Incomplete or corrupted entries are discarded during data cleaning.

These steps help in mitigating data quality issues and improve downstream classification performance.

C. Data Transformation and Normalization

The datasets are normalized and reshaped to ensure compatibility with deep learning models:

- Min-Max Scaling: Scales numerical features to a 0–1 range.
- Z-Score Standardization: Applied to features with high variance to stabilize learning.
- Reshaping for CNN Input: Features are transformed into 2D matrices for convolutional processing.

An 80:20 train-test split is used, with 5-fold cross-validation applied to enhance generalization.

D. Feature Selection Strategy

A hybrid feature selection pipeline is employed to eliminate redundant and irrelevant features:

- Random Forest Classifier: Ranks features based on importance scores derived from decision trees.
- Recursive Feature Elimination (RFE): Iteratively removes the least impactful features.
- SHAP (Shapley Additive Explanations): Interprets the contribution of each feature to the model output.

This strategy improves interpretability and reduces the risk of overfitting while optimizing training time.

E. Model Architecture: CNN-BiLSTM

The proposed model integrates two powerful components:

1) Convolutional Neural Network (CNN)
    - Extracts local spatial dependencies across input features.
    - Consists of convolutional, pooling, and batch normalization layers to enhance learning stability.

2) Bidirectional LSTM (BiLSTM)
- o Processes feature sequences in both forward and backward directions.
- o Captures long-range dependencies in network traffic for improved anomaly detection.

F. Hyperparameter Optimization
Model performance is tuned via systematic hyperparameter tuning. Table I lists the key parameters:

Table I: Hyperparameters Used

Parameter	Description	Affected Components
Learning Rate	Controls weight update step size	CNN, BiLSTM
Dropout Rate	Prevents overfitting via neuron dropout	CNN, BiLSTM
Number of Layers	Determines network depth	CNN, BiLSTM
Neurons per Layer	Controls representational power	BiLSTM
Batch Size	Controls samples per training iteration	All Models

Optimization was performed using grid search and empirical tuning, resulting in faster convergence and improved generalization.

G. Model Training and Evaluation
The CNN-BiLSTM model is trained using categorical cross-entropy loss and the Adam optimizer. Evaluation metrics include:
- Accuracy
- Precision
- Recall
- F1-score
- False Alarm Rate (FAR)

The model is benchmarked against other architectures including CNN-only, CNN-LSTM, LeNet-5, and AlexNet across binary and multiclass tasks using the InSDN dataset.

IV. IMPLEMENTATION
The proposed CNN-BiLSTM-based intrusion detection system is implemented using Python and a suite of machine learning and deep learning libraries. The system includes modules for data ingestion, preprocessing, feature engineering, model training, evaluation, and prediction. The entire pipeline is optimized for real-time detection and is compatible with both cloud and edge-based SDN deployments.

A. Software Requirements
The system was developed using the following tools and libraries:
- Python 3.10.11: Core programming environment
- Anaconda Navigator: Environment and package manager
- Jupyter Notebook: Interactive development interface
- TensorFlow / Keras: For designing and training deep learning models
- scikit-learn: For feature selection, evaluation metrics, and preprocessing
- OpenCV: Optional integration for packet-level feature visualization
- Matplotlib / Plotly: Used for performance analysis and result visualization

All components are cross-platform and tested on Windows 10 and Ubuntu 22.04 LTS.

B. Hardware Requirements
The implementation was tested on mid-range computing hardware. Recommended specifications include:
- Processor: Intel i5 / Ryzen 5 or higher

- RAM: 8–16 GB (minimum)
- GPU (Optional): NVIDIA GPU for faster model training (e.g., GTX 1650 or better)
- Storage: 20 GB (for datasets, logs, and models)
- Network: Standard Ethernet/Wi-Fi for data collection (SDN simulation)

The model can be trained on CPU, but significant acceleration is observed on GPU-supported systems.

C. Data Preparation

Data preparation includes collection, cleaning, encoding, and transformation:

1) Data Cleaning:
   - Null values removed or filled using forward-fill.
   - Irrelevant features (e.g., ID fields) dropped.

2) Encoding:
   - Categorical attributes (e.g., protocol types) are converted using label encoding.
   - Target labels (attack categories) are mapped to integers for classification.

3) Normalization:
   - Min-Max scaling is applied to all numeric features to bring values between 0 and 1.
   - Z-score normalization is used for high-variance features.

4) Reshaping:
   - Data is reshaped to 2D matrices (for CNN) and sequences (for BiLSTM) as needed.

D. Feature Selection

A hybrid approach is employed to reduce the dimensionality and improve model interpretability:

- Random Forest ranks features by importance using Gini index.
- Recursive Feature Elimination (RFE) iteratively removes the least important features.
- Selected Features include: packet length, flow duration, protocol, source/destination ports, byte counts, and flags.

This ensures high signal-to-noise ratio and reduces training time.

E. Model Selection and Configuration

The final CNN-BiLSTM model was chosen for its superior performance:

1) CNN Layers:
   - 3 convolutional layers with ReLU activation and kernel sizes (5, 3, 5)
   - Followed by max-pooling and batch normalization

2) BiLSTM Layer
   - Bidirectional LSTM with 10 hidden units
   - Captures temporal dependencies across traffic windows

3) Fully Connected Layers
   - Dense layers with dropout (0.5) for regularization
   - Final softmax or sigmoid activation for binary/multiclass classification

Model is compiled using the Adam optimizer and trained with categorical cross-entropy loss.

F. Training and Evaluation

1) Train-Test Split: 80% for training, 20% for testing
2) Cross-Validation: 5-fold for generalization
3) Batch Size: 256; Epochs: 20
4) Evaluation Metrics: Accuracy, Precision, Recall, F1-score, R^2, and Confusion Matrix

During training, learning curves (accuracy/loss vs. epochs) are monitored to detect overfitting or underfitting.

G. Code Implementation Summary

A sample snippet from the CNN-BiLSTM pipeline is shown below:

```
python
model = Sequential([
    Conv1D(128, kernel_size=5, activation='relu', padding='same', input_shape=(time_steps, num_features)),
    MaxPooling1D(pool_size=2), BatchNormalization(),
    Conv1D(64, kernel_size=3, activation='relu', padding='same'),
    MaxPooling1D(pool_size=2), BatchNormalization(),
    Conv1D(32, kernel_size=5, activation='relu', padding='same'),
    MaxPooling1D(pool_size=2), BatchNormalization(),
    Flatten(), Dense(128, activation='relu'), Dropout(0.5),
    Dense(64, activation='relu'), Dropout(0.5),
    Dense(15, activation='relu'), Reshape((15, 1)),
    Bidirectional(LSTM(10)), Dense(num_classes, activation='softmax')])
```

The model is trained using model.fit() and evaluated using standard scikit-learn metrics.

V. RESULTS

The proposed CNN-BiLSTM model was empirically evaluated using three benchmark datasets: NSL-KDD, UNSW-NB15, and InSDN. The experiments were designed to measure the model's performance on both binary and multiclass intrusion detection tasks. Results were compared against baseline models, including CNN-only, CNN-LSTM, LeNet-5, and AlexNet, using standard classification metrics.

A. Experimental Setup

The model was trained and tested using an 80:20 train-test split with 5-fold cross-validation to ensure generalizability. All experiments were conducted on a system with the following specifications:

- Processor: Intel Core i7 @ 2.60 GHz
- RAM: 16 GB
- GPU: NVIDIA GTX 1650 (4 GB)
- Software: Python 3.10.11, TensorFlow 2.x, scikit-learn, Jupyter Notebook

B. Evaluation Metrics

Model performance was evaluated using the following metrics:

- Accuracy (ACC): Correct predictions over total predictions
- Precision (P): True positives over predicted positives
- Recall (R): True positives over actual positives
- F1-Score: Harmonic mean of precision and recall
- False Alarm Rate (FAR): Incorrectly classified normal traffic

These metrics provide a comprehensive view of the classifier's robustness, especially in imbalanced intrusion detection tasks.

C. Binary Classification Results

For binary classification (attack vs. normal), the CNN-BiLSTM model achieved near-perfect results on the InSDN dataset.

Table I: Binary Classification Results on InSDN Dataset

Model	Accuracy	Precision	Recall	F1-Score	FAR
LSTM	95.18%	97.91%	95.18%	96.53%	4.8%
CNN-LSTM	83.69%	95.17%	83.69%	88.99%	16.3%
CNN-BiLSTM	99.97%	99.97%	99.97%	99.97%	0.03%

As shown in Fig. 7.3, the proposed hybrid model consistently outperformed traditional architectures in both precision and recall, indicating its ability to reduce false positives and identify complex patterns of intrusion.

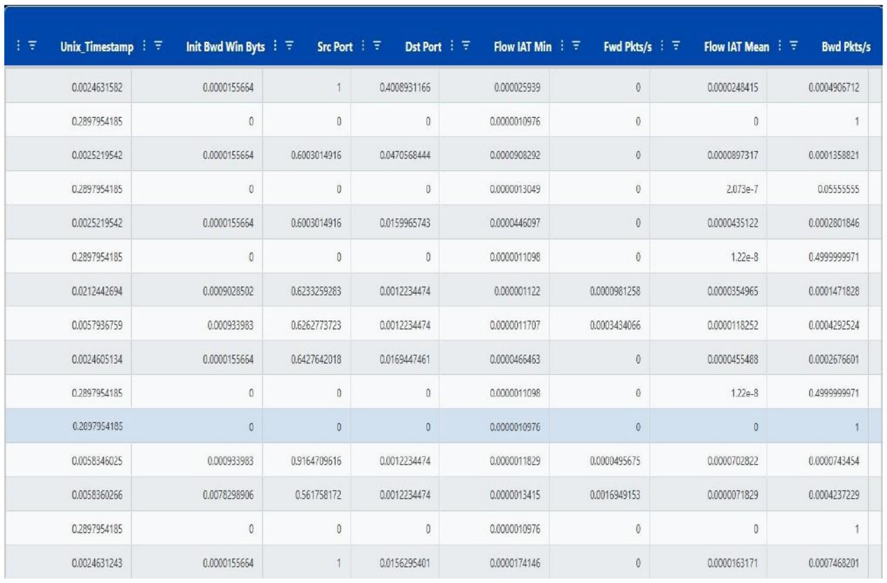
D. Multiclass Classification Results

The CNN-BiLSTM model was also evaluated on multiclass detection tasks involving various types of attacks such as DDoS, Probe, Web, and Botnet.

- Train Accuracy: 99.99%
- Test Accuracy: 99.97%
- Macro-Averaged F1-Score: 97.36%
- Confusion Matrix: Demonstrated balanced prediction across minority and majority classes

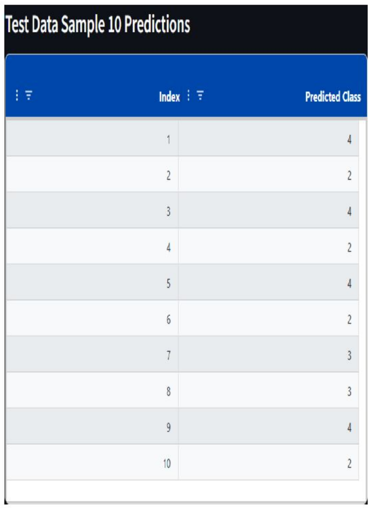
E. Visualization of Model Performance

Several visual outputs were generated to interpret the model's behavior:



Unix_Timestamp	Init Bwd Win Byts	Src Port	Dst Port	Flow IAT Min	Fwd Pkts/s	Flow IAT Mean	Bwd Pkts/s
0.0024631582	0.0000155664	1	0.4008931166	0.000025939	0	0.0000248415	0.0004906712
0.2897954185	0	0	0	0.0000010976	0	0	1
0.0025219542	0.0000155664	0.6003014916	0.0470568444	0.0000908292	0	0.0000897317	0.0001358821
0.2897954185	0	0	0	0.0000013049	0	2.073e-7	0.05555555
0.0025219542	0.0000155664	0.6003014916	0.0159965743	0.0000446097	0	0.0000435122	0.0002801846
0.2897954185	0	0	0	0.0000011098	0	1.22e-8	0.4999999971
0.0212442694	0.0009028502	0.6233259283	0.0012234474	0.000001122	0.0000981258	0.0000354965	0.0001471828
0.0057936759	0.000933983	0.6262773723	0.0012234474	0.0000011707	0.0003434066	0.0000118252	0.0004292524
0.0024605134	0.0000155664	0.6427642018	0.0169447461	0.0000466463	0	0.0000455488	0.0002676601
0.2897954185	0	0	0	0.0000011098	0	1.22e-8	0.4999999971
0.2897954185	0	0	0	0.0000010976	0	0	1
0.0058346025	0.000933983	0.9164709616	0.0012234474	0.0000011829	0.0000495675	0.0000702822	0.0000743454
0.0058360266	0.0078298906	0.561758172	0.0012234474	0.0000013415	0.0016949153	0.0000071829	0.0004237229
0.2897954185	0	0	0	0.0000010976	0	0	1
0.0024631243	0.0000155664	1	0.0156295401	0.0000174146	0	0.0000163171	0.0007468201

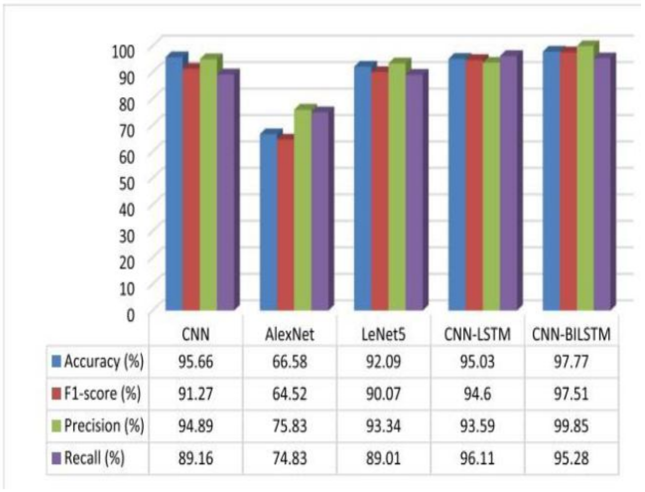
Figure 7.1: Sample input data from the test set



Test Data Sample 10 Predictions

Index	Predicted Class
1	4
2	2
3	4
4	2
5	4
6	2
7	3
8	3
9	4
10	2

Figure 7.2: Predicted intrusion categories for 10 random samples



	CNN	AlexNet	LeNet5	CNN-LSTM	CNN-BILSTM
🟦 Accuracy (%)	95.66	66.58	92.09	95.03	97.77
🟥 F1-score (%)	91.27	64.52	90.07	94.6	97.51
🟩 Precision (%)	94.89	75.83	93.34	93.59	99.85
🟪 Recall (%)	89.16	74.83	89.01	96.11	95.28

Figure 7.3: Bar chart comparing model performance in binary classification

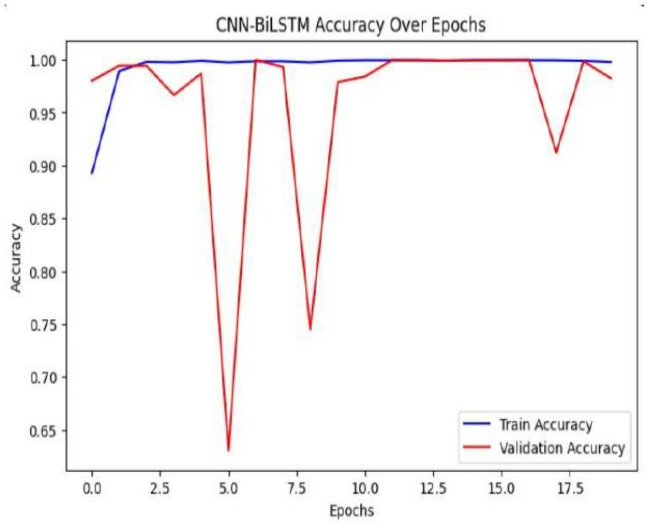


Figure 7.4: Accuracy vs. Epochs plot showing stable training and minor overfitting in validation

Although validation accuracy fluctuated slightly, the overall learning curve remained stable across all experiments, confirming the model's reliability.

F. Comparison with Existing Studies

Compared to models proposed in [1]–[4], the CNN-BiLSTM hybrid model achieves:

- Lower false alarm rate
- Higher F1-scores in minority classes
- Faster convergence and training times with optimized hyperparameters
- Robustness across both general-purpose and SDN-specific datasets

These results validate the proposed model as a competitive and scalable solution for securing SDN environments.

VI. CONCLUSION

This study presents a hybrid deep learning-based intrusion detection system that integrates Convolutional Neural Networks (CNN) with Bidirectional Long Short-Term Memory (BiLSTM) to enhance the detection of cyber threats in Software-Defined Networking (SDN) environments. The proposed CNN-BiLSTM architecture leverages CNN's capability for spatial feature extraction and BiLSTM's strength in modeling bidirectional temporal dependencies, enabling comprehensive detection of both known and evolving intrusion patterns.

To address feature redundancy and class imbalance—common challenges in intrusion detection—a hybrid feature selection mechanism combining Random Forest and Recursive Feature Elimination (RFE) was employed. Additionally, normalization and data augmentation techniques were incorporated to ensure robust model generalization.

Experimental evaluations on NSL-KDD, UNSW-NB15, and InSDN datasets demonstrate that the CNN-BiLSTM model significantly outperforms existing architectures such as LSTM, CNN-LSTM, and other baseline models. The model achieved a binary classification accuracy of 99.97% with a macro-averaged F1-score of 97.36% for multiclass scenarios, while maintaining a low false alarm rate. The system also exhibited stable convergence and resilience to overfitting under cross-validation.

The findings confirm that hybrid deep learning approaches, when combined with proper feature selection and preprocessing strategies, can serve as effective tools in modern SDN security frameworks. The proposed architecture provides a scalable, accurate, and real-time solution for protecting programmable networks from a wide range of threats.

REFERENCES

[1] L. Zhang, J. Huang, Y. Zhang, and G. Zhang, "Intrusion detection model of CNN-BiLSTM algorithm based on mean control," in Proc. IEEE 11th Int. Conf. Software Engineering and Service Sciences (ICSESS), Oct. 2020, pp. 22–27.

[2] S. Revathi and A. Malathi, "A detailed analysis on NSL-KDD dataset using various machine learning techniques for intrusion detection," Int. J. Eng. Res. Technol., vol. 2, no. 12, pp. 1848–1853, 2013.

[3] F. A. Khan, A. Gumaei, A. Derhab, and A. Hussain, "A novel two-stage deep learning model for efficient network intrusion detection," IEEE Access, vol. 7, pp. 30373–30385, 2019.

[4] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghosh, "Deep learning approach for network intrusion detection in software-defined networking," in Proc. Int. Conf. Wireless Networks and Mobile Communications (WINCOM), Oct. 2016, pp. 258–263.

[5] S. Boukria and M. Guerroumi, "Intrusion detection system for SDN network using deep learning approach," in Proc. Int. Conf. Theoretical and Applicative Aspects of Computer Science (ICTAACS), vol. 1, Dec. 2019, pp. 1–6.

[6] T. A. Tang, L. Mhamdi, D. McLernon, S. A. R. Zaidi, and M. Ghogho, "Deep convolutional neural network for intrusion detection in SDN-based networks," in Proc. 4th IEEE Conf. Network Softwarization Workshops (NetSoft), Jun. 2018, pp. 202–206.