Hybrid Deep Learning Architecture for Network Intrusion Detection: A CNN–LSTM–Attention Approach

Maryam Khalid
Department of Data Science and AI
National University of Computer and Emerging Sciences
Islamabad, Pakistan
i221917@nu.edu.pk

Maryam Amjad
Department of Data Science and AI
National University of Computer and Emerging Sciences
Islamabad, Pakistan
i221924@nu.edu.pk

Azka Atiq
Faculty of Computing
National University of Computer and Emerging Sciences
Islamabad, Pakistan
azka.atiq@isb.nu.edu.pk

***Abstract*—Network intrusion detection has become increasingly challenging due to the volume, complexity, and dynamic nature of modern network traffic. Real-world data often contain noise, redundant features, and rapidly changing attack behaviours.**

In this work, a hybrid deep learning-based intrusion detection model is presented that combines convolutional neural networks (CNNs), long short-term memory (LSTM) networks, and an attention mechanism for multi-class classification. Feature dimensionality is reduced using mutual information-based feature selection. One-dimensional convolutional layers are used to extract local feature patterns, whereas LSTM layers are employed to capture temporal relationships within network flow sequences.

The proposed model is evaluated using the CICIDS2017 dataset, where network traffic is grouped into six semantically meaningful classes: Benign, DoS/DDoS, PortScan, Brute Force, Web Attack, and Botnet ARES. Data preprocessing includes removal of duplicate records, treatment of missing values, feature standardisation, class balancing through random under-sampling and SMOTE, and selection of the top 50 features. The model achieves an accuracy of 97.07

In addition to overall performance evaluation, class-wise results are analysed in detail, and an ablation study is conducted to examine the contribution of each model component. Experimental findings indicate that the strongest performance is achieved when convolutional and recurrent components are used together. In contrast, adding an attention layer results in only minor gains for this dataset. Performance drops most noticeably when the LSTM module is removed, with a smaller but still clear decline observed after removing the CNN component. This behaviour highlights the need to model both feature-level interactions and temporal dependencies in network traffic for effective intrusion detection.

Keywords: Intrusion detection systems, deep learning, CNN–LSTM architecture, attention mechanism, CICIDS2017.

I. Introduction

Computer networks today are exposed to constant security risks as cyberattacks continue to grow in both scale and complexity. With organizations depending heavily on digital systems for daily operations, communication, and essential services, even short-lived network disruptions can result in significant operational and financial consequences. As a result, intrusion detection systems (IDS) play a critical role in safeguarding networks by monitoring activity and identifying malicious behavior that may compromise data confidentiality, system integrity, or service availability.

Conventional IDS techniques are often signature-driven. While effective for recognized attack patterns, signature methods are inherently limited when adversaries modify payloads or introduce new exploit strategies. This has motivated anomaly-based approaches that learn normal behavior and flag deviations. Machine learning improved IDS automation by learning statistical boundaries from traffic features; however, classical ML models often require extensive feature engineering and may not generalize well to complex traffic distributions.

Deep learning has reduced the need for handcrafted features in intrusion detection and has shown better ability to model complex attack behaviour. However, many published IDS pipelines still suffer from practical shortcomings. Common problems include using very high-dimensional feature sets without a clear selection strategy, relying on simple fully connected architectures, training models for a limited number of epochs, and reporting results without sufficient analysis such as ablation studies. These limitations make it difficult to judge how well such systems would perform in real operational settings.

In this work, we design and evaluate a hybrid intrusion detection pipeline that aims to address these issues in a more systematic manner. The proposed workflow combines explicit feature selection with spatial and temporal representation learning, and places emphasis on controlled and transparent experimentation. All experiments are conducted using the

CICIDS2017 dataset, which provides a realistic mix of benign and malicious traffic collected over multiple days and includes a wide range of attack behaviours.

A. Key Contributions

We introduce a hybrid CNN–LSTM–Attention architecture for multi-class intrusion detection, along with a complete experimental workflow. Our pipeline uses mutual information for feature selection, reducing dimensionality while keeping the most useful features. We report detailed class-wise performance metrics and run ablation studies to understand how much each component (CNN, LSTM, attention) contributes.All experimental evaluations are conducted using the CICIDS2017 dataset, where network traffic is grouped into six classes to preserve the original attack meaning while allowing a fair and balanced assessment. In addition, the study examines the relationship between detection performance and computational cost by comparing the proposed hybrid model with simpler fully connected baseline architectures.

B. Paper Organization

The remainder of this paper is organised as follows. Previous research relevant to this work is discussed in Section II. Section III describes the dataset used in this study along with the preprocessing steps, model architecture, and training procedure. Experimental results, including class-level performance and the ablation analysis, are reported in Section IV. Finally, Section V summarises the main conclusions and outlines possible directions for future work.

II. Literature Review

Early research in network intrusion detection relied mainly on publicly available benchmark datasets. Among these, KDD Cup 99 and its improved version, NSL-KDD, were the most commonly used due to their accessibility and ease of comparison across studies. Many early intrusion detection models were evaluated using these datasets. However, these benchmarks are now considered outdated, as the traffic patterns and attack behaviours they contain do not reflect current network environments. Alladi et al. [2] evaluated a hybrid MLP–KMeans approach on the NSL-KDD dataset and reported acceptable detection results, but the relevance of their findings was limited by the dataset itself.

As the limitations of traditional machine learning techniques became clearer, researchers began exploring deep learning approaches for intrusion detection. Wei et al. [4] applied Deep Belief Networks and showed that deeper models can capture more complex relationships in network data compared to classical classifiers. Vinayakumar et al. [5] investigated the use of deep learning models trained at scale using Apache Spark. Their work demonstrated that distributed training is feasible for IDS applications, although their experimental evaluation still depended on older benchmark datasets.

The introduction of the CICIDS2017 dataset represented a step toward more realistic evaluation settings. This dataset includes benign background traffic as well as multiple attack types collected over several days. Muraleedharan and Janet [6] used CICIDS2017 to develop a flow-based deep neural network focused on detecting slow-rate denial-of-service attacks. While their approach showed promising results, it addressed only a limited subset of attack categories. Asad et al. [7] proposed the DeepDetect framework and reported strong multiclass detection performance on CICIDS2017. However, their use of a feedforward architecture limited the model's ability to capture temporal dependencies in network flows.

TABLE I: Comparison of Related Work in Network Intrusion Detection

Reference	Methodology	Dataset	Accuracy	Identified Gap
Alladi et al. [2]	MLP + KMeans	NSL-KDD	73.09%	Outdated traffic
Sapre et al. [3]	SVM/ANN/RF	NSL-KDD	78.51%	Limited deep modelling
Wei et al. [4]	Deep Belief Net	NSL-KDD	82.36%	Dataset limitations
Osa et al. [1]	6-layer DNN	CICIDS2017	99.68%	No feature selection
Muraleedharan et al. [6]	Flow DNN	CICIDS2017	99.61%	Narrow attack focus
Asad et al. [7]	Feedforward DNN	CICIDS2017	98.0%	Weak temporal modelling
Sahu et al. [8]	CNN–LSTM	IoT-23	96.0%	IoT-specific setting
Kan et al. [9]	Attention LSTM	CICIDS2017	97.5%	No CNN blocks

More recent studies have recognised that network traffic contains both feature-level patterns and time-dependent behaviour. This has led to increased interest in hybrid deep learning models. Sahu et al. [8] showed that CNN–LSTM architectures outperform single-model approaches in intrusion detection tasks, particularly in IoT-based environments. Kan et al. [9] incorporated attention mechanisms into LSTM-based models evaluated on CICIDS2017 and demonstrated that emphasising important time steps can improve detection performance.

Overall, intrusion detection research has moved away from shallow models evaluated on legacy datasets toward hybrid deep learning approaches tested on more representative traffic. Current studies continue to explore CNN–LSTM architectures and attention-based methods to improve robustness and handle multiple attack classes more effectively [10]–[15].

III. Methodology

A. Dataset

We use the Canadian Institute for Cybersecurity Intrusion Detection System 2017 (CICIDS2017) dataset [23]. This dataset includes realistic benign traffic and various attack scenarios collected over multiple days. It contains 2,830,743 labeled flows with 79 statistical features extracted using CICFlowMeter.

The original dataset has many attack categories. We consolidate these into six classes that preserve attack semantics: Benign, DoS/DDoS, PortScan, Brute Force, Web Attack, and Botnet ARES. This mapping keeps the attack meanings clear while making the classification task manageable.

TABLE II: CICIDS2017 Dataset Statistics After Preprocessing

Class	**Total Samples**	**Train Samples**	**Test Samples**
Benign	2,095,057	1,676,045	419,012
DoS/DDoS	321,759	257,407	64,352
PortScan	90,694	72,555	18,139
Brute Force	9,150	7,320	1,830
Web Attack	2,143	1,715	428
Botnet ARES	1,995	1,596	399

B. Workflow and Architecture

Our hybrid CNN–LSTM–Attention model combines convolutional feature extraction, temporal dependency modeling, and attention-based weighting. We reshape the 50-dimensional input vector into a $(50, 1)$ pseudo-sequence to enable consistent convolution and recurrent processing.

Two convolutional blocks are used to extract local patterns and feature interactions from the reshaped sequence. The convolution and pooling operations are defined in (1)–(4) [17], [24]:

$$h^{(1)} = \text{ReLU}(W^{(1)} * x + b^{(1)}) \tag{1}$$

$$h^{(1)}_{\text{pool}} = \text{MaxPool}(h^{(1)}) \tag{2}$$

$$h^{(2)} = \text{ReLU}(W^{(2)} * h^{(1)}_{\text{pool}} + b^{(2)}) \tag{3}$$

$$h^{(2)}_{\text{pool}} = \text{MaxPool}(h^{(2)}) \tag{4}$$

where $*$ is 1D convolution, and $W^{(1)}, W^{(2)}$ are learned kernels.

The LSTM component models longer-range dependencies in the extracted representation. The gating operations in (5)–(10) follow the standard formulation [18]:

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

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

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{7}$$

$$C_t = f_t \odot C_{t-1} + i_t \odot \tilde{C}_t \tag{8}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{9}$$

$$h_t = o_t \odot \tanh(C_t) \tag{10}$$

An attention layer assigns weights to time steps before classification. The attention scoring and normalization are defined in (11)–(13) [19]:

$$e_t = \tanh(W_a h_t + b_a) \tag{11}$$

$$\alpha_t = \frac{\exp(e_t)}{\sum_{k=1}^{T} \exp(e_k)} \tag{12}$$

$$c = \sum_{t=1}^{T} \alpha_t h_t \tag{13}$$

C. Training Algorithm Overview

Algorithm 1 outlines the training process for our hybrid model. Each mini-batch goes through convolution blocks to capture local feature interactions, then through LSTM layers for longer-range dependencies, followed by attention weighting using (11)–(13). A softmax layer produces the final class probabilities. We monitor validation loss and stop early when it stops improving, preventing overfitting and saving computation time.

Algorithm 1 Hybrid CNN–LSTM–Attention Training Algorithm

```
Require: Training data $D_{\text{train}}$, validation data $D_{\text{val}}$, hyperparameters
Ensure: Trained model parameters $\Theta^*$
  Initialize model parameters $\Theta_0$ randomly
  Initialize Adam optimizer with learning rate $\eta = 0.001$
  $best\_val\_loss \leftarrow \infty$, $patience \leftarrow 0$
  for epoch $= 1$ to $15$ do
      Shuffle $D_{\text{train}}$
      for each mini-batch $(x, y)$ in $D_{\text{train}}$ do
          $x' \leftarrow$ Reshape$(x)$                    ▷ $50 \times 1$ sequence
          $h_{\text{CNN}} \leftarrow$ CNN_Forward$(x')$
          $h_{\text{LSTM}} \leftarrow$ LSTM_Forward$(h_{\text{CNN}})$
          $\alpha \leftarrow$ Attention_Weights$(h_{\text{LSTM}})$
          $c \leftarrow \sum \alpha h^{(t)}_{\text{LSTM}}$
          $\hat{y} \leftarrow$ Softmax(Dense$(c)$)
          $loss \leftarrow$ SparseCategoricalCrossentropy$(y, \hat{y})$
          $\nabla\Theta \leftarrow$ BackwardPass$(loss)$
          $\nabla\Theta \leftarrow$ Clip$(\nabla\Theta, 1.0)$
          $\Theta \leftarrow$ AdamUpdate$(\Theta, \nabla\Theta)$
      end for
      $val\_loss \leftarrow$ Evaluate$(D_{\text{val}}, \Theta)$
      if $val\_loss < best\_val\_loss$ then
          $best\_val\_loss \leftarrow val\_loss$
          $\Theta^* \leftarrow \Theta$
          $patience \leftarrow 0$
      else
          $patience \leftarrow patience + 1$
          if $patience \geq 5$ then
              break
          end if
      end if
  end for
  return $\Theta^*$
```

D. Data Preprocessing

We cleaned the data, normalized features, selected relevant attributes, and balanced the classes. First, we standardized column names for consistent processing. We removed duplicate flows to prevent bias from repeated samples inflating performance. Missing values were filled using mean imputation.

Some rate-based features like `flow_bytes/s` and `flow_packets/s` had infinite values from division-by-zero cases. We replaced these with feature-wise medians to keep training stable. We also dropped low-variance features (standard deviation below 0.01), which reduced the feature count from 79 to 68.

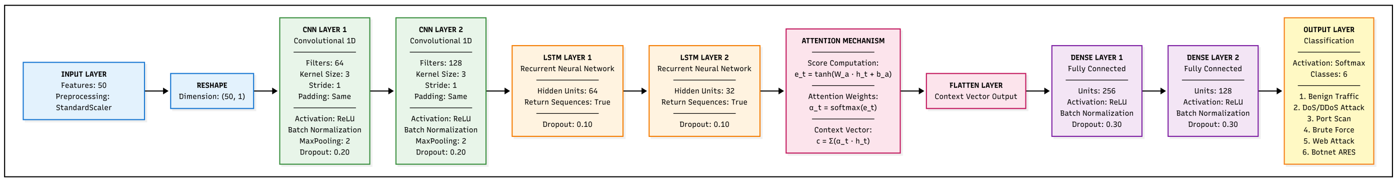


Fig. 1: Overview of the proposed CNN–LSTM–Attention model: reshaping, convolutional extraction, LSTM sequence modelling, attention weighting, and final softmax classification.

All remaining numeric features were standardized using the StandardScaler transformation as shown in (14):

$$\tilde{x}_i = \frac{x_i - \mu_i}{\sigma_i} \tag{14}$$

where x_i denotes the raw input value, μ_i represents the mean of the feature, and σ_i is its standard deviation.

For feature selection, mutual information was employed as a relevance measure following the formulation in [22], as defined in (15):

$$\mathrm{MI}(X;Y) = \sum_{x \in X} \sum_{y \in Y} p(x,y) \log \frac{p(x,y)}{p(x)p(y)} \tag{15}$$

where X represents the feature variable and Y denotes the class label. The top 50 features based on mutual information scores were selected, reducing dimensionality while preserving discriminative power.

The dataset had severe class imbalance, so we used a two-stage approach combining random under-sampling and SMOTE [21]. The final balanced training set had 2,400,000 samples with equal representation across all six classes.

E. Model Training

Our hybrid model has 262,726 trainable parameters. We trained it using Adam [20] with a learning rate of 0.001, batch size 256, gradient clipping at norm 1.0, and sparse categorical cross-entropy loss. We used early stopping with patience 5, and a learning rate scheduler that reduced the rate by 0.5 when validation loss plateaued.

F. Evaluation Metrics

Accuracy, precision, recall, and F1-score were computed using TP, TN, FP, and FN:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}} \tag{1}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{3}$$

$$\text{F1 Score} = \frac{2 \cdot \text{TP}}{2 \cdot \text{TP} + \text{FP} + \text{FN}} \tag{4}$$

Training time was also reported to quantify computational cost.

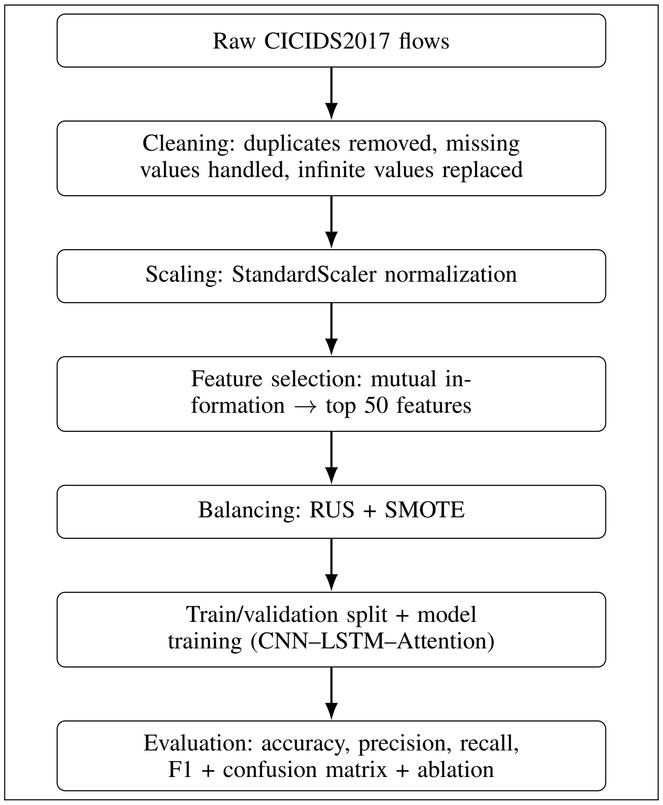

Raw CICIDS2017 flows
Cleaning: duplicates removed, missing values handled, infinite values replaced
Scaling: StandardScaler normalization
Feature selection: mutual information → top 50 features
Balancing: RUS + SMOTE
Train/validation split + model training (CNN–LSTM–Attention)
Evaluation: accuracy, precision, recall, F1 + confusion matrix + ablation

Fig. 2: Complete experimental pipeline showing data preprocessing, feature engineering, model training, and evaluation stages.

IV. Results and Discussion

A. Performance Comparison

To provide a fair benchmark, the baseline model proposed by Osa et al. [1] was reproduced and evaluated under the same experimental settings. The baseline achieved an accuracy of 97.00%, a precision of 98.44%, and an F1-score of 97.56% on the test data. Using the same dataset and evaluation protocol, the proposed hybrid model obtained an accuracy of 97.07%, with a precision of 98.24% and an F1-score of 97.50%. These results indicate a small but consistent improvement in overall accuracy and recall when compared to the baseline approach.

This increase reflects the added computational cost associated with integrating convolutional, recurrent, and attention-based components within a single model. A direct comparison of the two approaches is provided in Figure 3 and Table

TABLE III: Performance Comparison: Base vs Proposed Model

Metric	Base Model [1]	Proposed Model	Change
Accuracy (%)	97.00	97.07	+0.07
Precision (%)	98.44	98.24	-0.20
Recall (%)	97.00	97.07	+0.07
F1 Score (%)	97.56	97.50	-0.06
Training Time (s)	136.97	722.92	+585.95

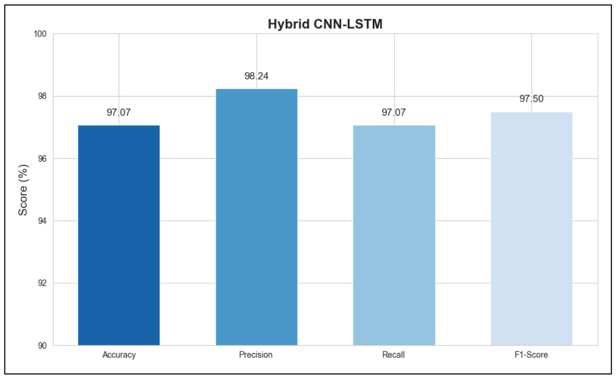


Fig. 3: Performance metrics comparison between base and proposed models.

III, which summarise their performance across the primary evaluation metrics.

The training curves shown in Figure 4 suggest that both models converge reliably during training. Although the hybrid model is trained for a greater number of epochs, it reaches a lower final training loss than the baseline. Validation accuracy remains stable over time, with less than a 2

B. Per-Class Performance Analysis

Both models achieve consistently high recall across all attack categories, suggesting that malicious traffic is rarely missed once it occurs. Differences become more apparent when precision is examined, especially for minority classes, where false positives remain difficult to control despite the use of balancing methods.

Relative to the baseline model [1], the hybrid architecture performs better for some underrepresented attack types. In particular, improvements are observed in the precision of the Botnet ARES class, indicating improved discrimination for this minority category. Specifically, precision for the Botnet ARES class increases from 0.08 to 0.13, while Brute Force precision improves from 0.78 to 0.86. These results suggest that combining convolutional feature extraction with sequential modelling enables better discrimination for specific attack patterns. A detailed breakdown of class-wise performance metrics is provided in Table IV.

Table IV shows that although both models consistently achieve high recall for all classes, precision differs notably for less frequent attack categories.

TABLE IV: Per-Class Classification Performance

	Base Model [1]			Proposed Model		
Class	Prec.	Rec.	F1	Prec.	Rec.	F1
Benign	1.00	0.96	0.98	1.00	0.97	0.98
Botnet ARES	0.08	0.99	0.15	0.13	1.00	0.22
Brute Force	0.78	0.98	0.87	0.86	1.00	0.92
DoS/DDoS	0.97	1.00	0.98	0.95	1.00	0.98
PortScan	0.74	1.00	0.85	0.73	1.00	0.84
Web Attack	0.20	0.99	0.33	0.19	0.99	0.31

TABLE V: Ablation Study: Component Contribution

Model Variant	Accuracy (%)
Full Model	97.07
Without Attention	97.15
Without LSTM	94.58
Without CNN	94.65
Dense Only	94.83

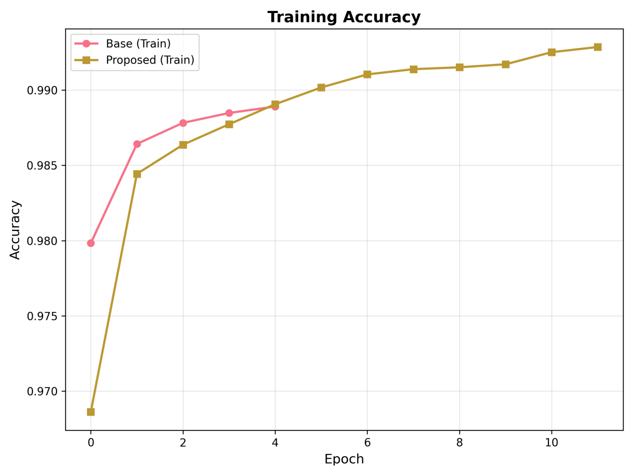
The confusion matrices presented in Figure 5 provide insight into the classification behaviour of both models. In each case, predictions are largely concentrated along the diagonal, indicating a high rate of correct classifications, while most errors are associated with minority classes. This behaviour highlights the difficulty of identifying infrequent attacks within predominantly benign traffic. In both models, minority attack samples are most often misclassified as Benign, which is consistent with the similarity in feature distributions between certain attack behaviours and normal network activity. Compared to the baseline, the proposed hybrid model shows slight improvements in distinguishing Botnet ARES and Brute Force traffic.

C. Ablation Study Results

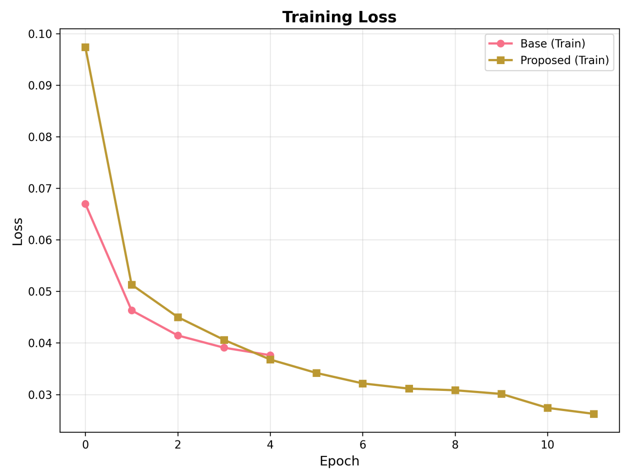
Our ablation study measures how much each component contributes. Removing LSTM dropped accuracy to 94.58% (the biggest drop), while removing CNN reduced it to 94.65%. Removing attention actually slightly improved accuracy, suggesting that for CICIDS2017, LSTM and CNN provide most of the benefit while attention adds little value. The complete ablation results are presented in Table V and Figure 6.

The ablation study results in Table V quantify the contribution of each architectural component, revealing that both CNN and LSTM are essential for optimal performance.

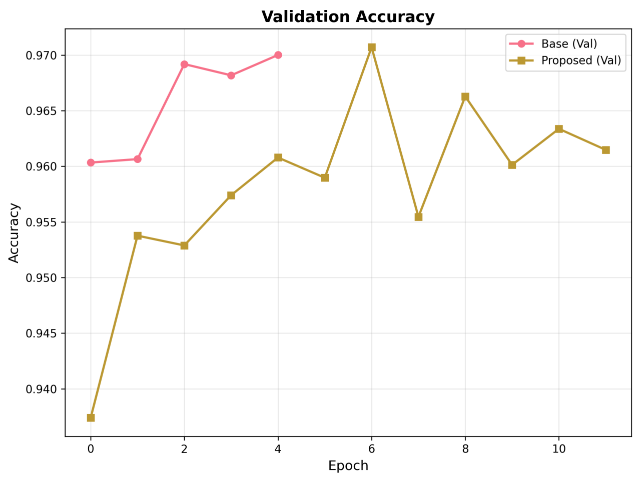
Figure 6 provides a visual comparison of the ablation study results, clearly showing the performance impact of removing each component from the hybrid architecture. The accuracy comparison shows that LSTM removal causes the largest performance drop (–2.49%), followed by CNN removal (–2.42%) and the dense-only configuration (–2.24%), whereas attention removal has a minimal impact (+0.08%). The F1 score comparison reveals a similar pattern, indicating that LSTM and CNN components are critical for balanced clas-



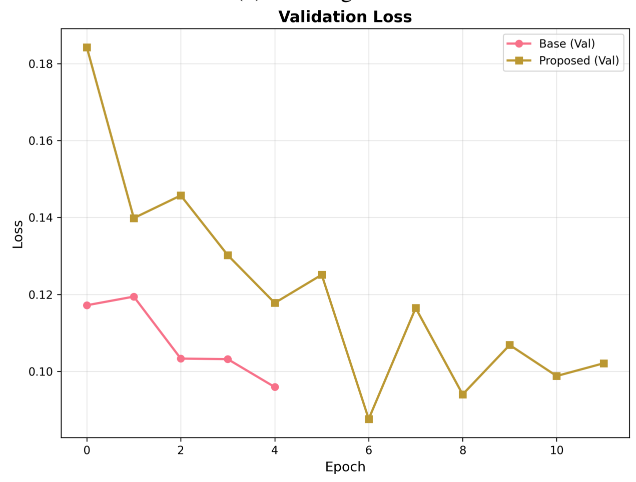
(a) Training Accuracy



(b) Training Loss

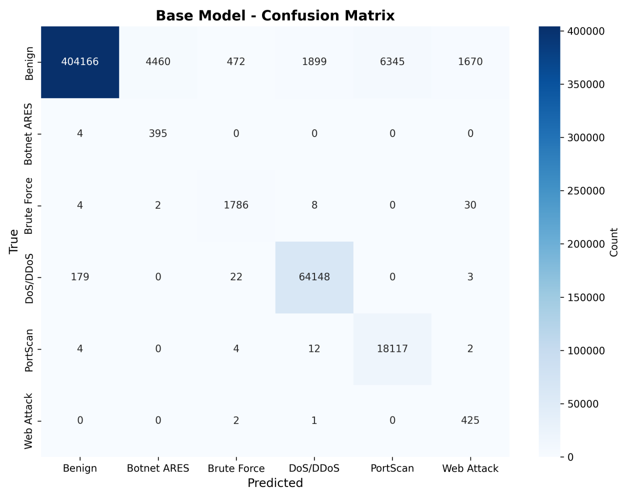


(c) Validation Accuracy



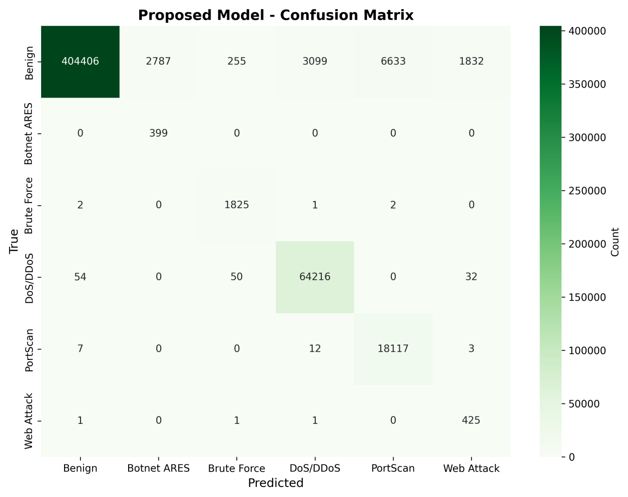
(d) Validation Loss

Fig. 4: Training/validation curves for base and proposed models.



	Benign	Botnet ARES	Brute Force	DoS/DDoS	PortScan	Web Attack
Benign	404166	4460	472	1899	6345	1670
Botnet ARES	4	395	0	0	0	0
Brute Force	4	2	1786	8	0	30
DoS/DDoS	179	0	22	64148	0	3
PortScan	4	0	4	12	18117	2
Web Attack	0	0	2	1	0	425

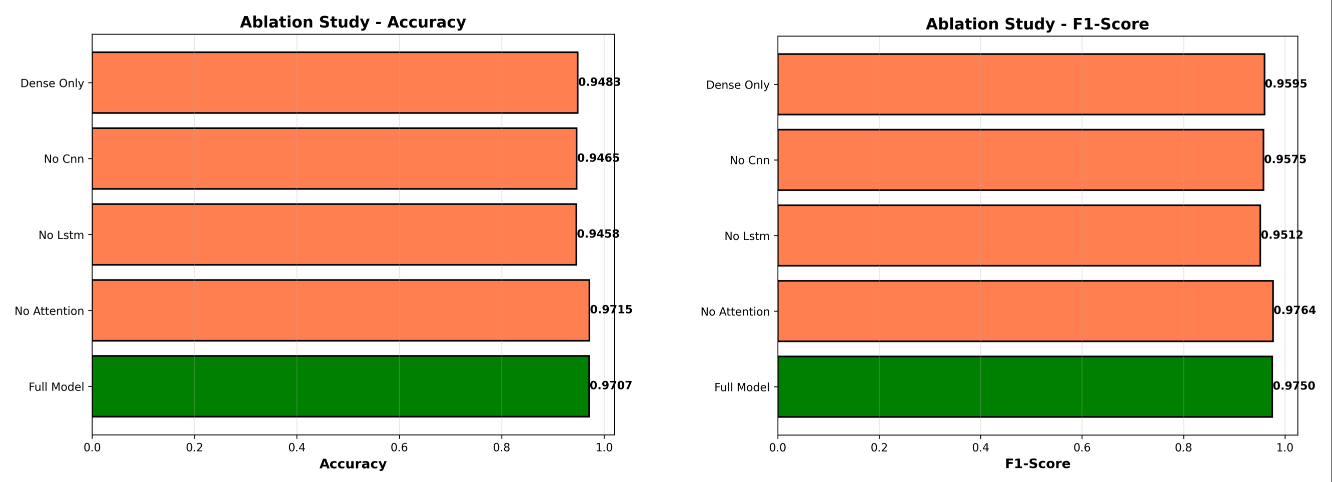
(a) Base Model [1]



	Benign	Botnet ARES	Brute Force	DoS/DDoS	PortScan	Web Attack
Benign	404406	2787	255	3099	6633	1832
Botnet ARES	0	399	0	0	0	0
Brute Force	2	0	1825	1	2	0
DoS/DDoS	54	0	50	64216	0	32
PortScan	7	0	0	12	18117	3
Web Attack	1	0	1	1	0	425

(b) Proposed Model

Fig. 5: Confusion matrices for base and proposed models.



(a) Accuracy Comparison

(b) F1 Score Comparison

Fig. 6: Ablation study plots.

sification performance, while attention brings marginal benefit in this particular application.

D. Discussion

Our results show diminishing returns from adding complexity beyond a solid baseline, particularly with engineered flow features. CNN and LSTM blocks consistently improve representation quality, but attention doesn't help much on this dataset. Minority-class precision stays challenging, suggesting that real deployment might need threshold tuning, cost-sensitive learning, or other calibration methods to cut down false positives.

The feature selection process reduced the dimensionality from 68 to 50 features effectively while sustaining the classification performance. The most informative features chosen according to mutual information, packet size statistics, backward flow metrics, and port information are well matched with domain knowledge in network security, hence justifying the selection methodology.

The ablation study provided additional insight regarding architectural components. The very modest gain from the attention mechanism is inconsistent with its more significant reported benefits in other domains and may indicate that the network flow features do not exhibit strong positional dependencies on which attention mechanisms typically rely. The significant value contributed by both CNN and LSTM components confirms that the spatial and temporal patterns of network traffic are important to capture, even when abstracted via tabular feature vectors.

The continued precision–recall imbalance of the minority classes, even under balanced training, speaks to deep-seated challenges in learning from highly skewed distributions. This suggests that threshold tuning, cost-sensitive learning, or alternative loss functions could be necessary techniques for deployment in a production environment where false positive rates have significant operational costs.

V. Conclusion

This paper evaluated a hybrid CNN–LSTM–Attention IDS pipeline on CICIDS2017 using a six-class mapping that preserves attack semantics. The proposed model achieved 97.07% accuracy with 98.24% precision and 97.50% F1-score, demonstrating modest improvements over the baseline. The ablation study shows that the convolutional and recurrent components are the primary contributors to detection performance, whereas the attention module adds only marginal value for the dataset used. From a practical perspective, the main drawback of the proposed approach lies in its computational cost, as training the hybrid model takes considerably longer than training simpler dense architectures. Future work could explore alternative sequence models such as transformer encoders, evaluate performance on newer datasets including CICIDS2018, and apply improved techniques for handling class imbalance, such as focal loss and calibrated decision thresholds, to better manage false positives in minority classes.

References

[1] E. Osa, P. E. Orukpe, and U. Iruansi, "Design and implementation of a deep neural network approach for intrusion detection systems," *e-Prime - Advances in Electrical Engineering, Electronics and Energy*, vol. 7, p. 100434, 2024.

[2] S. K. Alladi, "Effectively improving the efficiency and performance of an intrusion detection system using hybrid machine learning models," M.Sc. thesis, National College of Ireland, 2020.

[3] S. Sapre, P. Ahmadi, and K. Islam, "A robust comparison of the KDD-Cup99 and NSL-KDD IoT network intrusion detection datasets through various machine learning algorithms," arXiv preprint arXiv:1912.13204, 2019.

[4] P. Wei, Y. Li, Z. Zhang, T. Hu, Z. Li, and D. Liu, "An optimization method for intrusion detection classification model based on deep belief network," *IEEE Access*, vol. 7, pp. 87593–87605, 2019.

[5] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019.

[6] N. Muraleedharan and B. Janet, "A deep learning based HTTP slow DoS classification approach using flow data," *ICT Express*, 2020.

[7] M. Asad, M. Asim, T. Javed, M. O. Beg, H. Mujtaba, and S. Abbas, "Deep-detect: detection of distributed denial of service attacks using deep learning," *The Computer Journal*, vol. 63, pp. 983–994, 2020.

[8] A. K. Sahu, S. Sharma, M. Tanveer, and R. Raja, "Internet of things attack detection using hybrid deep learning model," *Computer Communications*, vol. 176, pp. 146–154, 2021.

[9] X. Kan, Y. Fan, Z. Fang, L. Cao, N. N. Xiong, D. Yang, and X. Li, "A novel IoT network intrusion detection approach based on adaptive particle swarm optimization convolutional neural network," *Information Sciences*, vol. 568, pp. 147–162, 2021.

[10] P. Sun, P. Liu, Q. Li, C. Liu, X. Lu, R. Hao, and J. Chen, "DL-IDS: Extracting features using CNN-LSTM hybrid network for intrusion detection system," *Security and Communication Networks*, 2020.

[11] M. Sajid, K. R. Malik, A. Almogren, T. S. Malik, A. H. Khan, J. Tanveer, and A. U. Rehman, "Enhancing intrusion detection: a hybrid machine and deep learning approach," *Journal of Cloud Computing*, vol. 13, art. 123, 2024.

[12] N. Biyouki, A. Biyouki, and A. Bagheri, "A deep learning framework for network intrusion detection," *Scientific Reports*, 2025.

[13] A. Almadhor *et al.*, "A deep learning model for detecting data leaks and intrusion detection system," *Scientific Reports*, 2025.

[14] A. M. Alashjaee, "Transformer models for anomaly detection in network traffic: A review," 2025.

[15] M. I. Mahmood and R. A. Javed, "Hybrid deep learning models for network intrusion detection in cloud environments," *Journal of Theoretical and Applied Information Technology*, 2025.

[16] "Deep learning vs. machine learning for intrusion detection in computer networks: A comparative study," *Applied Sciences*, 2025.

[17] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[18] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[19] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," arXiv preprint arXiv:1409.0473, 2014.

[20] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.

[21] N. V. Chawla, K. W. Bowyer, L. O. Hall, and W. P. Kegelmeyer, "SMOTE: Synthetic minority over-sampling technique," *Journal of Artificial Intelligence Research*, vol. 16, pp. 321–357, 2002.

[22] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: criteria of max-dependency, max-relevance, and min-redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005.

[23] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," in *Proc. International Conference on Information Systems Security and Privacy (ICISSP)*, 2018, pp. 108–116.

[24] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.