
Research on a Network Intrusion Detection Method Based on LSTM and Transformer

Sihao Liu*

International School, Beijing University of Posts and Telecommunications Beijing 100876, China

**Corresponding author: Sihao Liu.*

Abstract

Network intrusion detection is an important part of cybersecurity defense. Deep learning models have been widely used for this task, but it remains unclear whether the high performance reported in top-journal studies comes from model-structure innovation or from data-engineering methods. To address this problem, this study proposes a lightweight LSTM-Transformer fusion model. A bidirectional LSTM is used to extract local temporal features, and a single-layer Transformer encoder is used to model global dependencies. Experimental results on the NSL-KDD dataset show that, with fewer parameters and a shorter training time, the model achieves an F1 value of 0.7745, which is 2.97 percentage points higher than that of a pure Transformer. Further decomposition shows that data engineering accounts for about 68% of the performance improvement, while the model reaches only 32.90% accuracy on attack types unseen in the training set.

Keywords

network intrusion detection, deep learning, LSTM, Transformer, fusion model, NSL-KDD

1. Introduction

With the expansion of the Internet and the popularization of network applications, cybersecurity threats have become increasingly diverse and hidden. An intrusion detection system (IDS) identifies malicious behavior by monitoring network traffic or system audit records, and it is an important component of a cybersecurity defense system. Traditional intrusion detection methods mainly rely on expert rules and statistical models. In the face of constantly changing attack methods, they have prominent problems such as delayed rule updates and insufficient ability to detect unknown attacks.

After deep learning achieved breakthroughs in image recognition, natural language processing, and other fields, it has also been widely used in network intrusion detection. Recurrent neural networks (RNNs) and their variant, long short-term memory (LSTM) networks, are good at processing sequential data and can capture temporal dependencies in network traffic. Transformer is based on the self-attention mechanism, has strong parallel computing ability, and can also model long-range dependencies. However, systematic comparison of these two types of models in intrusion detection tasks is still insufficient. Most studies focus on a single model, and there is relatively little exploration of their fusion.

More importantly, existing studies generally report feature engineering, data resampling, multi-stage training, and model-structure improvement together, making it difficult to determine whether performance

improvement actually comes from structural innovation or data processing. The improved Transformer proposed by Liu and Wu [3] in Applied Sciences achieved 84.1% multi-class accuracy on NSL-KDD, but this result also used stacked autoencoder dimensionality reduction, KNN under sampling, Borderline-SMOTE oversampling, two-stage learning, and other data-engineering operations, so the pure structural contribution cannot be separated. This problem is not uncommon in the field: the high performance of top-journal studies is often attributed to new model structures, while the role of data engineering is rarely discussed separately.

To address this problem, this study limits the research to the pure model-structure level and does not introduce feature dimensionality reduction, data resampling, or multi-stage learning. Under unified preprocessing and training settings, this study carries out four tasks: (1) comparing the multi-class detection performance of LSTM and Transformer under the same experimental settings and giving clean baselines for the two basic models on NSL-KDD; (2) designing a lightweight LSTM-Transformer fusion model that uses a bidirectional LSTM to encode local temporal features and a single-layer Transformer to model global dependencies, obtaining a better F1 value while using fewer parameters and less training time than a standalone Transformer; (3) using the reported results of Liu and Wu [3] as an anchor to estimate the proportions of performance gain contributed by model structure and data engineering; and (4) pointing out a commonly missing part of top-journal evaluation systems through grouped evaluation of known and unknown attack types.

2. Related Work

Research on network intrusion detection based on machine learning has a long history. Early studies mainly used traditional methods such as support vector machines, decision trees, and random forests. After deep learning developed [15], various neural network models were gradually introduced into this field [8].

For recurrent neural networks, LSTM and gated recurrent units (GRUs) [9, 10] have been used to process the sequential characteristics of network traffic. These models can remember historical information and perform well on attack types such as DOS and Probe, but their detection ability for minority-class attacks such as R2L and U2R is limited.

Since it was proposed, Transformer has been widely used in various sequence tasks. In the intrusion detection field, its self-attention mechanism can directly model dependencies between arbitrary positions and avoid the gradient vanishing problem of RNNs. Manocchio et al. proposed the FlowTransformer framework and systematically evaluated the performance of Transformer in flow-based intrusion detection. They found that the design of the classification head had a significant effect on performance, and that global average pooling performed poorly in NIDS tasks [1].

For hybrid models, models combining convolutional neural networks [11] and recurrent neural networks, such as CNN-LSTM and CNN-GRU, have been proposed. CNN is used to extract local features, and RNN is used to model temporal relationships. Zhang et al. proposed a Transformer-LSTM model for intrusion detection and verified the effectiveness of combining an attention mechanism with a recurrent network [2]. The improved Transformer of Liu and Wu used stacked autoencoder (SAE) dimensionality reduction, a combination of KNN under sampling and Borderline-SMOTE oversampling, improved positional encoding, and a two-stage learning strategy, achieving 84.1% multi-class accuracy and an F1 value of 83.8% on the official NSL-KDD test set [3]. Zhang et al. also proposed a model based on BiLSTM and a multi-head attention mechanism and reported relatively high accuracy on NSL-KDD [4]. The TBL model proposed by Gan and Kong combines a Transformer encoder with BiLSTM and DNN for intrusion detection.

Existing studies still have several problems. First, many studies report accuracy above 95% on NSL-KDD, but they often use the KDDTest-21 test subset, which removes hard-to-classify samples, or resample the test set, so the evaluation is not strict enough. Second, there are few direct comparisons between LSTM and Transformer under unified experimental settings. Third, many fusion models use simple concatenation, and their analysis of feature complementarity between the two types of models is not deep enough. Fourth, and most importantly for this study, top-journal studies generally report data-engineering methods and structural improvements together, so the sources of performance gain are not transparent, and almost no work separately reports model performance on attack types unseen in the training set. This study strictly uses the official NSL-KDD KDDTest+ test set, does not perform feature dimensionality reduction or data resampling, and provides pure structural performance baselines for LSTM, Transformer, and the fusion model. Taking the results of Liu

and Wu [3] as a reference, this study estimates the contribution proportion of data engineering and supplements grouped evaluation of unknown attack types.

3. Method

3.1 Problem Definition

Network intrusion detection can be formalized as a multi-class classification problem. Given a sequence of network connection records $X = (x_1, x_2, \dots, x_T)$, where each x_i is a d-dimensional feature vector, the model needs to predict the class $y \in \{\text{normal, DOS, Probe, R2L, U2R}\}$ to which the sequence belongs. Here, normal represents normal traffic, DOS represents denial-of-service attacks, Probe represents probing attacks, R2L represents remote-to-local attacks, and U2R represents user-to-root attacks.

3.2 Overall Framework

The overall process of this method includes four stages: data preprocessing, sequence construction, model training, and evaluation. First, one-hot encoding and normalization are performed on the original NSL-KDD data. Then, independent connection records are organized into sequence samples through a sliding window. Next, LSTM, Transformer, and the LSTM-Transformer fusion model are trained separately. Finally, the accuracy, precision, recall, and F1 value of each model are evaluated on the test set, and window-parameter comparison, known and unknown attack analysis, and horizontal comparison with existing methods are conducted.

3.3 Data Preprocessing

Each record in the NSL-KDD dataset [7] contains 41 features and 1 class label. The features are divided into discrete and continuous features. For the three discrete features, protocol_type, service, and flag, one-hot encoding is adopted, resulting in 84 dimensions after encoding. For the 38 continuous features, min-max normalization is used to map them to the interval [0, 1], as follows:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (1)$$

After processing, each sample is a 122-dimensional vector. The training set is divided into a training subset and a validation set by stratified sampling at an 8:2 ratio. To organize independent connection records into sequences, the sliding-window method is used: for a window size w , the window slides with a step size of 1, and consecutive w records are taken as one sequence sample. The label is the class of the last record in the window. The window size w is determined through comparative experiments.

NSL-KDD has a severe class imbalance problem. The numbers of samples in each class in the training set are as follows: 53,874 normal samples, 36,741 DOS samples, 9,325 Probe samples, 796 R2L samples, and 42 U2R samples. To alleviate this problem, a weighted cross-entropy loss function is used, and the weight of each class is inversely proportional to the number of samples.

3.4 LSTM Model

LSTM controls information flow through the input gate, forget gate, and output gate, and can effectively alleviate the gradient vanishing problem of ordinary RNNs [5]. This study uses a single-layer LSTM with a hidden dimension of 128 and a fully connected classification layer. The hidden state of the last time step is taken and transformed linearly to output logits for 5 classes. The number of model parameters is 129,669.

3.5 Transformer Model

Transformer is based on the self-attention mechanism and can model dependencies between arbitrary positions in a sequence in parallel [6]. The Transformer used in this study first projects the 122-dimensional input to 128 dimensions through a linear layer and adds learnable positional encoding. It then passes through 2 Transformer encoder layers, each with 4 attention heads, a feed-forward network dimension of 256, and

dropout of 0.1. The output of the last time step is taken for classification through a fully connected layer. The number of model parameters is 345,349. The self-attention calculation formula is:

$$Attention(Q, K, V) = \text{soft max} \left(\frac{QK^T}{\sqrt{d_k}} \right) V \quad (2)$$

3.6 LSTM-Transformer Fusion Model

The LSTM-Transformer fusion model proposed in this study is intended to combine the local temporal modeling ability of LSTM with the global dependency modeling ability of Transformer. The model structure mainly includes the following modules:

(1) Bidirectional LSTM feature extraction layer: the input sequence first passes through a bidirectional LSTM layer, in which the forward and backward hidden dimensions are both 64, to extract local temporal features at each time step. The bidirectional structure enables the model to use both past and future contextual information, and the output dimension is 128.

(2) Linear projection layer: a linear layer projects the 128-dimensional output of the bidirectional LSTM to the d_{model} dimension of the Transformer (128 dimensions) to achieve dimensional matching.

(3) Positional encoding and Transformer encoder: after learnable positional encoding is added, the features pass through 1 Transformer encoder layer with 4 attention heads, a feed-forward network dimension of 256, and dropout of 0.1. The self-attention mechanism is used to model global dependencies in the sequence.

(4) Classification layer: the output of the last time step from the Transformer encoder passes through dropout (0.3) [14], and then a linear layer outputs logits for 5 classes.

The fusion model has 309,893 parameters, about 10.3% fewer than the standalone Transformer (345,349), and uses only 1 Transformer encoder layer. The design idea is to let the bidirectional LSTM first perform temporal encoding on the original features and capture local patterns between adjacent connections, thereby reducing the modeling burden of the Transformer. On this basis, the Transformer models long-range dependencies through self-attention, and the two types of features complement each other. This design seeks competitive performance with fewer parameters and a shorter training time, rather than exchanging a larger model scale for better metrics.

4. Experiments and Results

4.1 Experimental Environment and Settings

The experimental environment is shown in Table 1. The experiments are completed in a CPU environment using the PyTorch deep learning framework. Training uses the Adam optimizer [13], with a learning rate of 0.001, a batch size of 128, a maximum of 30 training epochs, and an early-stopping patience value of 8. Training is stopped if the validation loss does not decrease for 8 consecutive epochs. All experiments fix the random seed at 42 to ensure reproducibility.

Table 1: Experimental Environment

Item	Configuration
Operating system	Windows
Python version	3.x
Deep learning framework	PyTorch 2.13.0 (CPU)
Main dependencies	Pandas, scikit-learn, NumPy
Hardware	CPU

4.2 Evaluation Metrics

Accuracy, weighted precision, weighted recall, and weighted F1 value are used as evaluation metrics. The F1 value considers both precision and recall and is a more reliable metric in class-imbalanced scenarios. In addition, the F1 value of each attack class is reported to analyze differences in model performance across attack types.

4.3 Comparative Experimental Results

To analyze the sensitivity of different models to sequence length, two settings with window sizes of 10 and 20 are selected for comparison. The results are shown in Table 2. Window 10 is the setting used in the subsequent main experiments, and window 20 is the setting under which LSTM performs best.

Table 2: Performance Comparison of Each Model Under Different Window Sizes

Window	LSTM Acc.	LSTM F1	Trans. Acc.	Trans. F1	Fusion Acc.	Fusion F1
10	0.7676	0.7461	0.7755	0.7448	0.7909	0.7745
20	0.7864	0.7748	0.7737	0.7444	0.7645	0.7447

Several points in Table 2 are worth noting. LSTM has a higher F1 value at window 20 (0.7748 vs. 0.7461), indicating that it needs a longer context to exert its temporal modeling ability. Transformer performs similarly under the two windows and is not very sensitive to sequence length. The fusion model reaches an F1 value of 0.7745 at window 10, which is clearly better than the two standalone models under the same window. However, its performance decreases at window 20, indicating that the effect of the fusion strategy depends on a suitable sequence length and that the feature complementarity of the two models is strongest under a medium window. Therefore, window 10 is used in all subsequent main experiments.

In the NSL-KDD training set, there are only 42 U2R samples and 796 R2L samples, whereas there are 53,874 normal samples, showing an extremely imbalanced class distribution. This study uses weighted cross-entropy loss, with each class weight inversely proportional to the number of samples of that class in the training set, so that the model pays more attention to minority classes during training. This study does not oversample or under sample the training data, in order to maintain the original data distribution and avoid noise introduced by synthetic over-sampling techniques such as SMOTE [12]. From the subsequent results, weighted loss improves the recognition effect of minority classes to some extent, but the F1 values of R2L and U2R are still low. This is related to the large number of attack subtypes in the test set that were unseen during training.

Under the condition that the window size is 10, the overall performance of the three models on the test set is shown in Table 3.

Table 3: Overall Performance Comparison of Each Model (window = 10)

Model	Acc.	Prec.	Recall	F1	Time
LSTM	0.7676	0.7914	0.7676	0.7461	76s
Transformer	0.7755	0.8137	0.7755	0.7448	261s
LSTM-Transformer	0.7909	0.8145	0.7909	0.7745	233s

Table 3 shows that the fusion model outperforms the two standalone models on all metrics. Compared with LSTM, accuracy increases by 2.33 percentage points and F1 increases by 2.84 percentage points. Compared with Transformer, accuracy increases by 1.54 percentage points and F1 increases by 2.97 percentage points. More importantly, these three advantages hold at the same time: the parameter count is 10.3% lower than that of the pure Transformer (309,893 vs. 345,349), the training time is 10.7% shorter (233 s vs. 261 s), and F1 is instead 2.97 percentage points higher. The reason is that the fusion model uses only 1 Transformer encoder layer, while the bidirectional LSTM undertakes part of the temporal modeling work. The two are functionally complementary, rather than relying on stacking model scale to exchange for performance. The F1 values of each model on different attack classes are shown in Table 4.

Table 4: F1 Values of Each Model by Attack Class

Model	normal	DOS	Probe	R2L	U2R
LSTM	0.7874	0.8620	0.7467	0.3162	0.3646
Transformer	0.8015	0.8782	0.7504	0.2125	0.4309
LSTM-Transformer	0.8225	0.8845	0.7453	0.3656	0.2402

All three models perform best on DOS attacks, with all F1 values above 0.86. This is related to the large number of DOS samples and the clear feature patterns. The fusion model has the highest F1 values for the normal, DOS, and R2L classes. In particular, R2L increases from 0.3162 for LSTM and 0.2125 for Transformer to 0.3656, indicating that the local features of bidirectional LSTM and the global features of self-attention are complementary and helpful for minority-class detection. However, the fusion model has an F1 value of only 0.2402 on U2R, lower than the two standalone models. Since the U2R training set has only 42

samples and the fusion model has a relatively large number of parameters, the model may overfit or underfit this class. The three models perform similarly on Probe, and the fusion model does not show an advantage.

Under CPU training conditions, each LSTM epoch takes about 4-5 seconds, Transformer takes about 17 seconds, and the fusion model takes about 15 seconds. LSTM stops early at epoch 17, while Transformer and the fusion model both stop early at epoch 15. The total training time of the fusion model is 233 seconds, about 10.7% shorter than that of the standalone Transformer (261 seconds). The reason is that it uses only 1 Transformer encoder layer, and the bidirectional LSTM shares part of the temporal modeling. Together with the better F1 of the fusion model (0.7745 vs. 0.7448), this result reflects the trade-off between efficiency and performance in the fusion architecture.

The KDDTest+ test set of NSL-KDD contains 17 attack subtypes that do not appear in the training set, totaling 3,748 samples and accounting for 16.6% of the test set. These zero-day attack samples have never been seen by the model during training. Model performance on these unseen attacks is a key indicator for measuring the practical deployment value of an intrusion detection system. However, the literature review found that top-journal studies, including Liu and Wu [3], generally report only overall accuracy and F1 values and do not separately disclose model performance on attack types unseen in the training set. This study divides the test set into two groups according to whether the attack subtype appears in the training set and evaluates the fusion model separately. The results are shown in Table 5.

Table 5: Grouped Performance of the Fusion Model on Known and Unknown Attacks

Range	Sample size	Acc.	F1
All test samples	22535	0.7828	0.7706
Known attack types	18787	0.8734	0.8645
Unknown attack types	3748	0.3290	-

For known attack types, the fusion model has an accuracy of 87.34% and an F1 value of 0.8645, with DOS reaching an F1 value as high as 0.9628, normal reaching 0.8977, and Probe reaching 0.8288. However, for unknown attack types, the accuracy is only 32.90%, indicating that the model has limited generalization ability for attack patterns unseen during training. Behind the overall metric of about 79%, there is actually a large structural difference: the model performs well on the 83.4% of known samples, but the 16.6% of unknown samples lowers the overall result by about 8 percentage points. This also means that the high overall accuracy reported by existing top-journal studies, such as 84.1%, may conceal deficiencies on zero-day attacks even when KDDTest+ is strictly used. Overall metrics cannot distinguish between “seen attacks are classified well” and “unseen attacks are classified poorly.” This study suggests that future studies report grouped accuracy for unknown attacks as one of the standard evaluation items in addition to overall metrics.

To evaluate the relative level of this method in the field and quantify the sources of performance gain, the three models in this study are compared with multiple deep learning methods reported by Liu and Wu [3] on the official NSL-KDD KDDTest+ test set. The results are shown in Table 6.

Table 6: Performance Comparison with Existing Methods on NSL-KDD

Method	Data method	Acc. (%)	F1 (%)
CNN [3]	Basic preprocessing	78.8	78.2
CNN-LSTM [3]	Basic preprocessing	79.7	79.1
CNN-GRU [3]	Basic preprocessing	80.0	79.5
ViT [3]	Basic preprocessing	82.8	82.4
CNN-Transformer [3]	Basic preprocessing	83.1	82.7
RTIDS (Transformer) [3]	Basic preprocessing	83.5	83.0
Improved Transformer [3]	SAE dimensionality reduction + hybrid sampling + two-stage learning	84.1	83.8
LSTM (this study)	Normalization + weighted loss only	76.8	74.6
Transformer (this study)	Normalization + weighted loss only	77.6	74.5
LSTM-Transformer (this study)	Normalization + weighted loss only	79.1	77.5

Based on Table 6, the sources of performance gain are decomposed as follows. It should be noted in advance that the following proportions are estimates based on public cross-paper results, and that different studies differ in preprocessing details and hyperparameter tuning. The exact proportions need to be verified through ablation

experiments under a unified code framework. Taking the LSTM in this study ($F1 = 74.6\%$) as the pure structural baseline, the fusion model in this study improves $F1$ to 77.5% through structural fusion, an increase of 2.9 percentage points. On this basis, the improved Transformer of Liu and Wu further increase $F1$ to 83.8% through data-engineering methods such as SAE dimensionality reduction (from 122 dimensions to 35 dimensions), KNN under sampling and Borderline-SMOTE oversampling, and two-stage learning, an increase of 6.3 percentage points. By $F1$, structural fusion contributes about 32%, and data engineering contributes about 68%. By accuracy, structural improvement contributes 2.3 percentage points ($76.8\% \rightarrow 79.1\%$), and data engineering contributes 5.0 percentage points ($79.1\% \rightarrow 84.1\%$), with the proportions also about 32% and 68%. This shows that, on the NSL-KDD multi-class task, the contribution of data engineering to final performance is about twice that of model-structure improvement. In addition, pure structural models such as RTIDS reported by Liu and Wu ($F1 = 83.0\%$) are higher than the results of this study, possibly due to preprocessing details, hyperparameter tuning, or implementation differences. However, the 0.8 percentage-point increase of the improved Transformer over RTIDS can be clearly attributed to data-engineering methods. This study does not pursue the highest absolute performance. Instead, under the condition that data-engineering variables are controlled, it provides a performance baseline for pure model structures, so that subsequent studies can distinguish “how much structural innovation is worth” from “how much data engineering is worth.”

4.4 Result Analysis

Based on the above experimental results, the following conclusions are obtained:

(1) Under a unified plain setting, namely only normalization and weighted loss, without feature dimensionality reduction or data resampling, the LSTM-Transformer fusion model has the highest $F1$ value at window 10 (0.7745), and both its parameter count and training time are lower than those of the standalone Transformer. This indicates that this fusion architecture is feasible as an efficiency-performance trade-off scheme.

(2) Different models have different sensitivities to window length. LSTM performs better under a longer window (20), Transformer is not sensitive to window length, and the fusion model is optimal at window 10. Under a medium sequence length, the complementarity of the two types of models is strongest.

(3) Class imbalance remains a performance bottleneck. Weighted loss is helpful, but the $F1$ values of R2L and U2R remain low. There are only 42 U2R samples in the training set, and the fusion model is even inferior to the standalone models on this class, indicating that model complexity needs to match sample size.

(4) Unknown attack types are a key factor limiting overall performance and are also a blind spot in top-journal evaluation systems. The fusion model reaches 87.34% accuracy on known attacks, but only 32.90% on 17 unseen attacks. Top-journal studies such as Liu and Wu [3] have not separately reported this metric.

(5) The decomposition of performance contribution shows that, taking the LSTM in this study as the baseline, structural fusion contributes about 2.9 percentage points of $F1$ gain, while data-engineering methods contribute about 6.3 percentage points, accounting for about 68% of the total gain. On the NSL-KDD task, the contribution of data engineering to performance is clearly greater than that of model-structure innovation, and the high accuracy of top-journal studies mainly benefits from it.

5. Discussion

Regarding the strengths and weaknesses of LSTM and Transformer in intrusion detection, the two models have very similar performance under the same settings in this study, with $F1$ values of 0.7461 and 0.7448, respectively. Transformer does not show an obvious advantage. This differs from the situation in natural language processing (NLP), where Transformer comprehensively surpasses RNN. A possible reason is that each NSL-KDD record consists of independent connection statistical features, and the temporal dependency between adjacent records is weaker than the word-order relationship in text. When the data volume is limited or the temporal pattern is not obvious, the global modeling advantage of self-attention is difficult to exert. Manocchio et al. also pointed out that the design of the classification head has a significant effect on Transformer performance, and the method used in this study, taking the last time step, may not be the optimal choice [1].

The significance of the fusion model as an efficiency-performance trade-off scheme is also worth discussing. BiLSTM → single-layer Transformer has the highest F1 value at window 10 (0.7745), 10.3% fewer parameters (309,893) than the standalone Transformer (345,349), and 10.7% shorter training time. The significance of this result does not lie in a large improvement in absolute performance, but in proving that, through a reasonable functional division, namely bidirectional LSTM for local temporal encoding and single-layer Transformer for global dependency modeling, performance gain can be obtained while reducing model scale. For deployment scenarios with limited computing resources, such as edge network devices, this lightweight fusion idea is more practical than simply stacking Transformer layers. Of course, the fusion model is not better than the standalone models under all windows. Its performance decreases at window 20, indicating that the effect of the fusion strategy depends on a suitable sequence length.

Decomposing the sources of performance gain is the core finding of this study. Taking the LSTM in this study as a pure structural baseline ($F1 = 74.6\%$), structural fusion brings a gain of 2.9 percentage points. Liu and Wu [3] further obtain 6.3 percentage points on this basis through data-engineering methods such as SAE dimensionality reduction, hybrid sampling, and two-stage learning, and data engineering accounts for about 68% of the total gain. This quantitative result shows that the high accuracy of top-journal studies does not mainly come from structural innovation, but depends more on data-engineering methods such as feature dimensionality reduction, class balancing, and training strategies. This provides a direct reference for subsequent studies: when reporting the performance of new models, data-engineering variables should be controlled, and structural contribution should be clearly distinguished from data contribution; otherwise, it is easy to misattribute the gains from data engineering to structural innovation. It should be stated again that the decomposition in this study is based on cross-paper comparison. Different studies have differences in preprocessing details and hyperparameter tuning, and the exact proportion still needs to be verified through ablation experiments under a unified framework.

The evaluation blind spot also cannot be ignored. After grouping by whether the attack subtype appears in the training set, the fusion model has an accuracy of 87.34% on known attacks, but only 32.90% on 17 unseen attacks. Liu and Wu [3] and the multiple comparison methods they cited all report only overall metrics and do not separately disclose performance on unknown attacks. Overall metrics mix together “seen attacks are classified well” and “unseen attacks are classified poorly,” which may conceal model deficiencies in zero-day attack scenarios. The practical value of an intrusion detection system largely depends on its ability to discover unknown threats. Therefore, this study suggests that grouped accuracy for unknown attacks be included as one of the standard reporting items for NSL-KDD evaluation.

This study also has obvious limitations. It evaluates only a single dataset, NSL-KDD, and the generality of the conclusions needs to be verified on newer datasets such as UNSW-NB15 and CIC-IDS2017. The decomposition of performance contribution is based on cross-paper comparison rather than ablation experiments under a unified framework, so the exact proportion may contain errors. The hyperparameters of the fusion model have not been systematically optimized. This study also does not compare the model with other methods such as ensemble learning and graph neural networks.

6. Conclusion and Future Work

This study limits the research to the pure model-structure level. Without using feature dimensionality reduction or data resampling on NSL-KDD, it compares LSTM, Transformer, and their lightweight fusion model. The main conclusions are as follows: (1) the LSTM-Transformer fusion model uses bidirectional LSTM to encode local temporal features and a single-layer Transformer to model global dependencies. It has 10.3% fewer parameters and 10.7% shorter training time than the standalone Transformer, and its F1 reaches 0.7745, making it a lightweight efficiency-performance trade-off scheme; (2) taking the LSTM in this study as the baseline, structural fusion contributes about 2.9 percentage points of F1 gain, while the data-engineering methods adopted by Liu and Wu [3] contribute about 6.3 percentage points, accounting for about 68% of the total gain. This indicates that the performance improvement of top-journal studies mainly comes from data engineering rather than structural innovation; (3) the fusion model has an accuracy of 87.34% on known attacks, but only 32.90% on 17 attacks unseen in the training set, and existing top-journal studies generally do not report this metric; (4) different models have different sensitivities to window length, and the fusion model has the strongest complementarity under a medium window (10).

Future work can be carried out from several directions: verifying the exact proportion in the contribution decomposition through strict ablation experiments under a unified code framework; introducing autoencoders for feature dimensionality reduction and representation learning, and quantifying the independent contribution of data engineering while controlling structural variables; designing open-set recognition methods for zero-day attacks and promoting grouped accuracy for unknown attacks as part of standard evaluation; verifying the generality of the conclusions on more datasets such as UNSW-NB15 and CIC-IDS2017; and exploring more efficient fusion architectures, such as cross-attention and adaptive feature fusion, to further improve performance while maintaining lightweight design.

References

- [1] Manocchio, L. D., Layeghy, S., Lo, W. W., Kulatilleke, G. K., Sarhan, M., & Portmann, M. (2024). Flowtransformer: A transformer framework for flow-based network intrusion detection systems. *Expert Systems with Applications*, 241, 122564. <https://doi.org/10.1016/j.eswa.2023.122564>
- [2] Zhang, Z., Si, X., Li, L., Gao, Y., Li, X., Yuan, J., & Xing, G. (2023). An intrusion detection method based on Transformer-LSTM model. In *2023 3rd International Conference on Neural Networks, Information and Communication Engineering (NNICE)* (pp. 352-355). IEEE. <https://doi.org/10.1109/NNICE58320.2023.10105733>
- [3] Liu, Y., & Wu, L. (2023). Intrusion detection model based on improved Transformer. *Applied Sciences*, 13(10), 6251. <https://doi.org/10.3390/app13106251>
- [4] Zhang, J., Zhang, X., Liu, Z., Fu, F., Jiao, Y., & Xu, F. (2023). A network intrusion detection model based on BiLSTM with multi-head attention mechanism. *Electronics*, 12(19), 4170. <https://doi.org/10.3390/electronics12194170>
- [5] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, 9(8), 1735-1780. <https://doi.org/10.1162/neco.1997.9.8.1735>
- [6] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.
- [7] Tavallaei, M., Bagheri, E., Lu, W., & Ghorbani, A. A. (2009). A detailed analysis of the KDD CUP 99 data set. In *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications* (pp. 1-6). IEEE. <https://doi.org/10.1109/CISDA.2009.5356528>
- [8] Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., & Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7, 41525-41550. <https://doi.org/10.1109/ACCESS.2019.2895334>
- [9] Xu, C., Shen, J., Du, X., & Zhang, F. (2018). An intrusion detection system using a deep neural network with gated recurrent units. *IEEE Access*, 6, 48697-48707. <https://doi.org/10.1109/ACCESS.2018.2867564>
- [10] Yin, C., Zhu, Y., Fei, J., & He, X. (2017). A deep learning approach for intrusion detection using recurrent neural networks. *IEEE Access*, 5, 21954-21961. <https://doi.org/10.1109/ACCESS.2017.2762418>
- [11] Wu, K., Chen, Z., & Li, W. (2018). A novel intrusion detection model for a massive network using convolutional neural networks. *IEEE Access*, 6, 50850-50859. <https://doi.org/10.1109/ACCESS.2018.2868993>
- [12] Chawla, N. V., Bowyer, K. W., Hall, L. O., & Kegelmeyer, W. P. (2002). SMOTE: Synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, 16, 321-357. <https://doi.org/10.1613/jair.953>
- [13] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- [14] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1), 1929-1958.

- [15] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). Deep learning (Vol. 1, No. 2, pp. 1-800). Cambridge: MIT press.

Funding

This research received no external funding.

Conflicts of Interest

The authors declare no conflict of interest.

Acknowledgment

This paper is an output of the science project.

Copyrights

Copyright for this article is retained by the author (s), with first publication rights granted to the journal. This is an open-access article distributed under the terms and conditions of the Creative Commons Attribution license (<http://creativecommons.org/licenses/by/4.0/>).