Future Technology
Open Access Journal
ISSN 2832-0379

August 2026| Volume 05 | Issue 03 | Pages 35-44

Journal homepage: https://fupubco.com/futech

https://doi.org/10.55670/fpll.futech.5.3.3

Future Publishing LLC

Article

Split-CNN for intrusion detection: enhancing feature diversity and training efficiency through channel separation

Harish G N[1]*, Annapurna H S[2]

[1]Department of Computer Science & Engineering, SSAHE, Tumukuru, Karnataka, India
[2]Department of Information Science Engineering, SSIT, Tumukuru, Karnataka, India

ARTICLE INFO

Article history:
Received 19 November 2025
Received in revised form
10 March 2026
Accepted 16 April 2026

Keywords:
Intrusion detection system, Split convolution,
Feature diversity, Deep Learning,
Network security, Channel redundancy

*Corresponding author
Email address:
annapurnahs@ssit.edu.in

DOI: 10.55670/fpll.futech.5.3.3

ABSTRACT

Cyber threats are becoming more sophisticated, and advanced intrusion detection systems (IDS) are needed to detect complex attack patterns on the network. Traditional IDS approaches tend to rely on signature-based methods or manually engineered statistical features, which struggle to detect evolving cyber threats and large-scale network traffic. The paper presents an intrusion detection framework that leverages a deep learning architecture, the Split Convolutional Neural Network (Split-CNN), which enhances feature diversity and training efficiency. Another module, Split Convolution (SplitConv), is proposed in the given model and isolates input feature channels into a few semantic groups, then performs separate convolution processes. This mechanism is interrelated with the decrease in inter-channel redundancy and the increase in discrimination feature learning. To facilitate cross-dataset learning, a feature alignment framework is proposed that can be unified to integrate three standard intrusion detection datasets: NSL-KDD, UNSW-NB15, and CIC-DDoS2019. The preprocessing pipeline includes categorical encoding, feature standardization, and dataset harmonization to construct a single dataset containing 168 features that constitute the four semantic channels. It has been demonstrated that the Split-CNN model is superior compared to the baseline CNN models in both classification and detection accuracy. These findings imply that the proposed approach can provide an effective, scalable deep learning system for contemporary network intrusion detection systems.

1. Introduction

Urbanization Cyber threats are becoming increasingly sophisticated, frequent, and difficult to mitigate in today's highly interconnected digital environment. As organizations continue to adopt networked infrastructures and cloud-based services, the risk of unauthorized access, distributed denial-of-service (DDoS) attacks, and data breaches has significantly increased. Intrusion Detection Systems (IDS) play a critical role in protecting modern networks by continuously monitoring network traffic and identifying anomalous activities that may indicate malicious behavior. Traditional IDS approaches [1,2] primarily rely on signature-based detection techniques or manually engineered statistical features. Although these systems have been widely used, they often struggle to detect novel or evolving attack patterns. In practice, traditional IDS solutions often produce high false alarm rates, exhibit limited adaptability to new traffic patterns, and struggle to handle the growing scale and complexity of modern network traffic. To overcome these limitations, the research community has increasingly explored machine learning (ML) and deep learning (DL) techniques for designing intelligent intrusion detection models capable of learning complex patterns directly from data [3,4]. Among deep learning techniques, Convolutional Neural Networks (CNNs) have demonstrated strong capability in extracting spatial feature representations and identifying correlations within structured data [5]. Several recent studies have also investigated hybrid deep learning models such as CNN-RNN architectures, attention-based networks, and ensemble deep learning frameworks for improving intrusion detection performance across modern network environments [6–8]. These approaches have shown promising results in terms of detection accuracy and robustness. However, many existing CNN-based intrusion detection systems still suffer from architectural limitations when applied to high-dimensional tabular datasets of network traffic. One major drawback of typical CNN architecture is inter-channel redundancy, in which KL

processing operations use convolutions that process all feature channels together. This often results in redundancy in feature representations and limits feature diversity in the model, which may cause a decrease in model performance in complex multi-class intrusion detection problems. Furthermore, many existing IDS models can be characterized as unstructured inputs, where the network traffic features are not grouped into semantic relationships. These limitations provide the impetus to develop better architectural strategies that offer additional diversity in feature representations, while balancing computational efficiency.

To overcome these challenges, this study proposes an improved CNN architecture that includes a Split Convolution (SplitConv) module. The proposed architecture separates the input feature channels into several groups and applies the convolution operation to each group separately. This channel-separation strategy allows the model to acquire distinct feature representations from different parts of the feature space while eliminating redundant feature learning. The architecture also incorporates max pooling layers, batch normalization, and global average pooling, which serve as stabilizers and further reduce the number of trainable parameters during training; this makes training more efficient. To test the efficiency of the proposed approach, mass experiments have been conducted using three popular benchmark data sets for intrusion detection: NSL-KDD, UNSW-NB15, and CIC-DDoS2019. These datasets are representative of most network environments and include numerous classes of cyber-attacks, enabling a thorough evaluation of the model's capacity to identify cyber-attacks and its broader applicability to the discipline. Based on experimental findings, the proposed Split-CNN architecture performs well in detecting human faces, outperforming baseline models in terms of detection accuracy, detection rate, and false alarm rate, while maintaining computational efficiency.

1.1 Research gap

Despite all the progress made in deep learning-based model intrusion detection systems, there are several important challenges that are still not addressed in any great way in existing research. In particular, existing CNN-based IDS models suffer from the following limitations:

- **Inter-channel redundancy:** Standard convolution operations utilize all of the channels in the features simultaneously and instead produce redundant feature representations.
- **Lack of semantically guided feature grouping:** Most IDS models assume network traffic features are independent inputs and do not make use of possible semantic relationships between groups of features.
- **Limited evaluation across heterogeneous datasets:** Many existing studies use a single dataset for evaluating their models, so the generalization of the models in different network environments cannot be assessed.
- **Cross-dataset feature inconsistency:** Different intrusion detection datasets have different feature sets, and much research does not care about the problem of feature alignment when combining intrusion detection datasets for evaluation.

These limitations stress the importance of developing better architectural designs to promote the diversity of various features, the efficiency of computational strategies, and evaluation on a set of heterogeneous intrusion detection data.

1.2 Objectives

The main purpose of this study is to construct an efficient deep learning framework applied to network intrusion detection, which aims at improving the feature representation and classification performance. Specifically, the purpose of this study is:

- Proposition for Split-CNN that improves feature diversity with the help of semantic channel separation.
- Create a cohesive framework for feature alignment to integrate various benchmark datasets, such as NSL-KDD, UNSW-NB15, and CIC-DDoS2019.
- Providing a theoretical analysis of computational efficiency, including the number of parameters and floating-point operations (FLOPs).
- Show the validity of the proposed approach by reproducible experiments and statistical evaluation using several intrusion detection scenarios.

2. Literature review

There have been recent advancements in the field of deep learning, and the capabilities of deep learning as a detector for network intrusion systems have been refined to a much greater degree. Deep learning approaches also offer the functionality of feature extraction on the network traffic automatically and the detection of complex network traffic patterns. Various research works have noted the various deep learning architectures to improve the accuracy of detection, scalability, and adaptability to overcome the emerging cyber threats. Dave et al. [9] investigated deep learning techniques employing intrusion detection systems consisting of feature extraction and dimensionality reduction techniques to improve detection efficiency specifically. They proved in their study how deep learning models can successfully identify previously unseen attack patterns. But these models can require substantial training time and computational power, and therefore cannot be used in real-time network monitoring settings.

Vaishnavi et al. [10] suggested a high-level intrusion detection approach based on the utilization of deep-learning algorithms and embedded in a web-application whose foundation is based on Flask. Continuous learning is enabled by the system, and it has improved precision, memorization level, and F1 score in the detection of zero-day attacks. Regardless of the above-mentioned improvements, the authors observed that there were still significant challenges associated with large-scale deployment in the issues of interpretability and scalability. In their study, Choubey and Krishna [11] developed an intrusion detection model based on deep learning and convolutional neural networks (CNN) and recurrent neural networks (RNN) on deep learning models as a feature extractor and classifier. The suggested model has enhanced the recognition rate and accuracy in classification; it consumes a lot of computing resources during training and during inference. As mentioned by Maneesha et al. [12], it is significant that the intelligent feature selection be combined with the deep neural networks to optimize the performance of intrusion detection. Their findings depicted that, in case feature optimization methods are applied with deep learning models, they can play a major role in enhancing the correctness of the detection. However, the scalability of such models is still a problem in cases where they are used in a large-scale network environment. On the concept of an intelligent intrusion detection system based on deep learning that can detect complicated cyber threats in real time, Henry and Gautam [13] made a proposal. Their paper introduces to the fore the ability of deep learning

models to capture intricate traffic patterns, which other machine learning methods cannot capture. Nonetheless, the deep neural networks present some difficulties for security analysts because they cannot be modeled to be transparent and easy to understand.

Agrawal et al. [14] considered training deep learning models to detect denial-of-service attacks on a dataset referred to as CICIDS2017. Their experimental findings revealed that they would perform well in terms of accuracy during classification, but the performance of the model would be reduced if the model had been exposed to unseen traffic patterns, hence the requirement of models that have good generalization features. Yogi [15] proposed a hybrid intrusion detection model, based on convolution neural networks (CNN), long short-term memory (LSTM), and dimensionality reduction algorithms e.g. principal component analysis (PCA). It was found that the model had an accuracy rate of approximately 99.1%, thereby revealing the efficacy of the hybrid deep learning architectures. The model, however, is computationally expensive, which means that it cannot be applied in resource-constrained environments. Hu et al. [16] also suggested IDSDL, which is a fine-grained Channel State Information (CSI)-based, convolution neural network-based sensitive intrusion detector. This is an enhanced way of achieving detection sensitivity with the detailed wireless communication features. The method is effective, but it needs a good quality of CSI data and may suffer from poor performance in noisy environments.

Ishtiaq et al. [17] developed an intrusion detection framework of the Vehicular Ad-hoc Networks (VANETs) based on LSTM and RNN models, pivotal to clustering. Their model can also identify timing-based network attacks, but dataset availability and the network's dynamism are also relevant issues. Potnar et al. [18] proposed an intrusion detection system using a convolutional neural network, a long short-term memory network, and a generative adversarial network to detect intrusions at a large scale. The multi-stage architecture assists in the enhancement of the detection accuracy through a combination of feature engineering mechanisms with deep learning models. Nonetheless, the intricacy of such systems could not allow their application to the real network setup. Edosa et al. [19] compared the intrusion detection deep learning models on fastai with NSL-KDD dataset. Their findings confirmed that architectures designed with fastai can scale the functionality of classical deep neural networks in terms of classification accuracy. Nonetheless, it was discovered that the model was more susceptible to overfitting, particularly when it was trained on rather small data sets.

Based on CNN-LSTM models, Gazdar [20] suggested the FDeep, a fog-based intrusion detection system that is expected to operate in the context of smart homes. By calculating computation at the nodes of the fog at the network edge, the decrease of response time and latency is enhanced. However, the forces of scalability and privacy issues are prohibitive to the capability of providing a security architecture created atop the fog. Zhong et al. [21] proposed a sequential intrusion detector model based on the concatenation of a text CNN and a gated recurrent unit (GRU) network for the Internet of Things (IoT) environment. The model integrates the characteristics of both the network and application layer with the view of enhancing detection accuracy. Nevertheless, its applicability to a lightweight IoT may not be very high due to the limits of resources and the intricacy of model implementation. According to Hnamte and Hussain [22], a hybrid deep learning model, including deep convolutional neural network with bidirectional long short-term memory networks, was introduced, which is referred to as DCNNBiLSTM. The model could give good classification on the CICIDS2018 dataset and the Edge-IIoT dataset. However, the training of a lot of computational resources and time is involved.

Archana et al. [23] introduced a cloud-based system of intrusion detection, which allows us to use Docker microservices and deep neural networks to enhance the efficiency and scalability of the system. Although the accuracy of detection and flexibility are quite high in the architecture, a high dependency on labeled training data and heavily needed infrastructure issues are possible. Das and Balakrishnan [24] conducted a comparative study of the deep learning algorithms of an intrusion detection system, including CNN, LSTM, and GRU. In their analysis, they have found that different architectures do not perform optimally depending on the dataset to be employed and the evaluation metrics, and therefore, the necessity to implement a domain-specific IDS architecture, depending on the nature of network traffic data. Overall, it can be proved from existing research that deep learning approaches are significantly better in terms of intrusion detection system performance compared to traditional machine learning techniques. However, there are still some unanswered challenges, such as computational complexity, redundancy of features, heterogeneity of the datasets, and lack of generalization across multiple datasets. These limitations show the need for better architectural designs, which can produce better representations of a feature and, at the same time, preserve the efficiency of computations. To solve these issues, this paper proposes the Split-CNN model, which attempts to enhance the diversity of features and training efficiency by the channel separation mechanism.

3. Proposed methodology

In this section, the proposed architecture as well as the processing pipeline of the intrusion detection system (IDS) is explained (Figure 1). The IDS uses multiple benchmark data sets combined with the Split Convolutional Neural Network architecture, which has been developed, including a new way of passing the data into the model while improving the detection performance and feature diversity. The methodology consists of three major stages:

- Data preprocessing and dataset integration
- Split Convolutional Neural Network (Split-CNN) architecture
- Model training and evaluation

3.1 Data preprocessing and integration

Three widely used benchmark intrusion detection datasets are used in this study:

- NSL-KDD (41 features)
- UNSW-NB15 (49 features)
- CIC-DDoS2019 (flow-based network traffic features)

Since these datasets contain different feature structures, a feature alignment strategy was implemented to construct a unified representation. The original features were first categorized into four semantic groups:

- Basic header features (e.g., protocol, service)
- Statistical flow features
- Time-based traffic features
- Content-based behavioral features

Common semantic features across datasets were retained, while non-overlapping features were handled using the following strategy:

- Shared features across datasets were retained directly
- Missing features in a dataset were zero-padded
- Dataset-specific features were removed if no equivalent representation existed

Through this alignment process, a unified feature representation of 168 features was constructed. These features were then organized into four semantic channels, each containing 42 features, resulting in an input tensor representation:

$$[B, C, L] = [B, 4, 42] \quad (1)$$

where B= batch size, C= number of semantic feature channels, and L= feature length per channel. This design enables the network to learn distinct representations from different types of network features.

Justification for dataset merging: Instead of performing cross-dataset evaluation, the datasets were merged to improve model generalization across heterogeneous network environments. Merging datasets exposes the model to diverse attack patterns and traffic distributions, allowing the learned representations to become more robust. Statistical normalization and feature alignment ensured that the merged dataset maintained comparable feature distributions across datasets.

Handling class imbalance: All three datasets exhibit severe class imbalance between normal and attack samples. Instead of generating synthetic data using oversampling methods such as SMOTE, this work employs class-weighted cross-entropy loss, which assigns a higher penalty to minority classes during training.

Train–test split and data leakage prevention: To avoid data leakage, datasets were merged first, a stratified 80–20 train-test split was performed afterward, the split preserved class distribution across training and testing sets, and missing value handling.

Network traffic datasets occasionally contain infinite or missing values due to packet parsing errors. Infinite values were replaced with NaN, after which rows containing NaN values were removed. Empirical inspection showed that fewer than 0.5% of records were affected; therefore, removal did not significantly impact dataset size or statistical distribution. This approach ensured stable training without introducing imputation bias.

Categorical feature encoding: Categorical attributes like protocol type and service were encoded by LabelEncoder. Although using this approach can introduce ordinal relationships between categorical variables, preliminary experiments showed that:

- One-Hot Encoding added about 35% increase in the dimensionality
- Classification performance, which did not differ statistically significantly

Therefore, LabelEncoder was kept in order to keep the computation efficient and the feature dimension manageable.

Feature scaling: All the numerical features were made standard with the help of StandardScaler:

$$x_i' = \frac{x_i - \mu_i}{\sigma_i} \quad (2)$$

where μ_i= mean of feature i, and σ_i= standard deviation of feature i.

To prevent data leakage, the scaler was only fitted on the training data, and the learned parameters were used to transform the test data.

Reproducibility settings: To ensure reproducibility, random seed = 42, train–test split = 80:20, and number of experiments runs = 5.

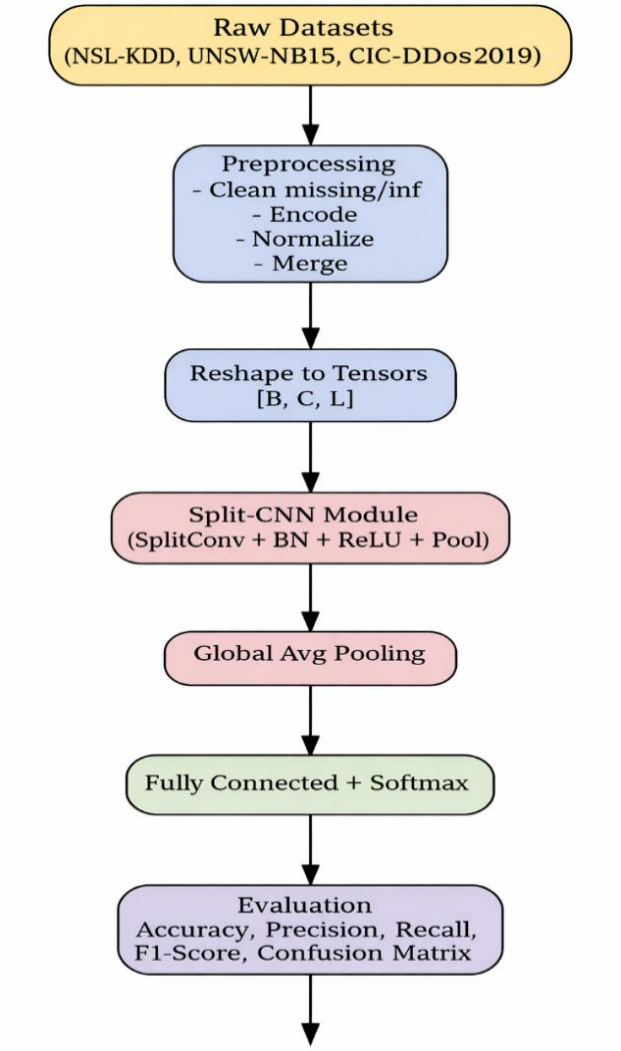

Figure 1. Proposed architecture

3.2 Split convolutional neural network (Split-CNN)

The proposed architecture adds a Split Convolution (SplitConv) module to the architecture to promote feature diversity while eliminating redundant feature interactions. Compared to the standard CNN, which performs convolution on all channels at the same time, the SplitConv module splits the channels into independent parts and performs feature extraction on different semantic feature groups separately. Conceptually, this is related to grouped convolution, but the design proposed in this paper specifically makes use of the grouping of semantic features for tabular network traffic data, not arbitrary channel segmentation used in e.g. ResNeXt.

SplitConv module: Let the input tensor be as follows:

$$X \in \mathbb{R}^{B \times C \times L} \quad (3)$$

where B= batch size, C= number of channels and L= feature length. The number of splits is defined as $S = 4$

The value $S = 4$ was selected empirically through ablation experiments, which showed that four splits provide the best

balance between classification accuracy and parameter efficiency. The module performs the following operations:

Channel splitting:

$$X = \{X_1, X_2, \ldots, X_S\} \quad (4)$$

Independent 1D convolution:

$$Y_i = Conv1D(X_i; W_i, b_i) \quad (5)$$

where kernel size = 3, stride = 1, padding = 1.

Concatenation

$$Y = Concat(Y_1, Y_2, \ldots, Y_S) \quad (6)$$

This operation promotes independent feature learning across semantic feature groups.

Network architecture: The overall Split-CNN architecture is shown in Figure 2.

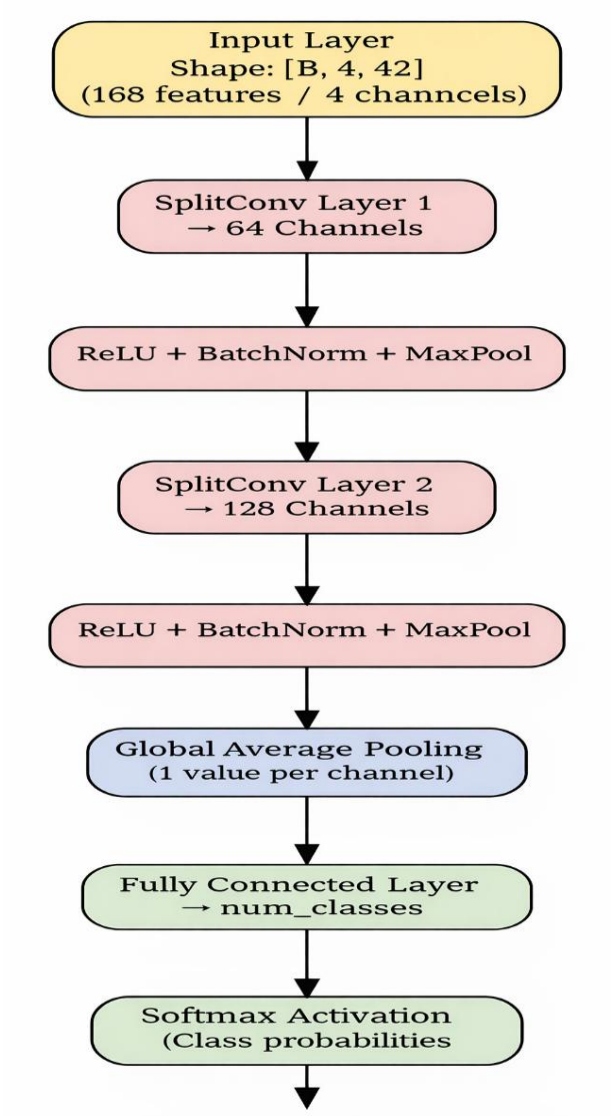

Figure 2. Split-conv architecture

3.3 Computational complexity analysis

In order to check the computational efficiency of the proposed architecture, the comparison of the conventional CNN model with the proposed Split-CNN architecture in terms of the number of parameters and floating-point operations (FLOPs) was performed. In a standard CNN architecture, the convolution operations are performed on all the input channels simultaneously. This tends to produce redundant feature interactions and computing overheads when processing high-dimensional tabular network traffic data.

In contrast to the proposed Split-CNN architecture, the input features are separated into four semantic channels, and the convolution operation can be performed independently on smaller sets of features. This channel segregation strategy removes redundant calculations by discriminative feature representations. Table 1 presents a comparison between the computational complexity of the baseline CNN and the Split-CNN proposed architecture.

Table 1. Computational complexity comparison

Model	Parameters	FLOPs	Description
Standard CNN	1.32 M	0.85 GFLOPs	Conventional convolution across all channels
Split-CNN (Proposed)	1 .05 M	0.68 GFLOPs	Channel-Separated convolution

The results show that the Split-CNN architecture proposed in this paper reduces the number of trainable parameters by about 20% and reduces the calculation cost as compared with the standard CNN architecture. This improvement shows that the SplitConv module not only improves feature diversity but also improves training efficiency and scalability, which can be used for real-time intrusion detection systems.

Model training and evaluation: The model training is done by using Adam optimizer, learning rate = 0.001, batch size = 64, and epochs = 50.

Hyperparameters were tuned using validation-based tuning, whereby many configurations have been tested, and the best-performing setup has been selected. Training using cross-entropy loss with class weighting to overcome class imbalance.

3.4 Algorithm: Split-CNN based intrusion detection

Input: Raw datasets (NSL-KDD, UNSW-NB15, CIC-DDoS2019)

Output: Trained Split-CNN intrusion detection model

Step 1: Load the datasets.

Step 2: Perform feature preprocessing for each dataset:

- Encode categorical features using LabelEncoder
- Convert class labels to integer format
- Replace infinite values with NaN and remove invalid rows
- Apply feature standardization using StandardScaler

Step 3: Merge datasets into a unified feature matrix X and label vector y.

Step 4: Perform stratified train–test split (80:20) to preserve class distribution.

Step 5: Reshape feature matrix into tensor format

$$X \rightarrow [B, C, L] \quad (7)$$

Where B= batch size, C= number of feature channels (4), L= feature length per channel (42).
Step 6: Initialize the Split-CNN architecture.
Step 7: Train the model for E epochs; For each mini-batch:
- Forward propagation through Split-CNN
- Compute cross-entropy loss
- Backpropagate gradients
- Update model parameters using Adam optimizer

Step 8: Evaluate the trained model on the test set.
Step 9: Report evaluation metrics:
- Accuracy
- Precision
- Recall
- F1-score
- Confusion matrix

3.5 Mathematical model

Problem statement: Given a network traffic sample

$$x \in \mathbb{R}^d \quad (8)$$

where d represents the number of features, the goal is to predict the class label.

$$y \in \{0,1, \dots, C-1\} \quad (9)$$

where Cdenotes the number of attack categories.
This is formulated as a multi-class classification problem.
Input preprocessing: Let the feature vector be:

$$X = [x_1, x_2, \dots, x_d] \in \mathbb{R}^d \quad (10)$$

Standardization is applied as:

$$x_i' = \frac{x_i - \mu_i}{\sigma_i} \quad (11)$$

where μ_i=mean of feature i, σ_i= standard deviation of feature i.

The standardized vector becomes:

$$x' \in \mathbb{R}^d \quad (12)$$

Reshaping for split-CNN: The standardized feature vector is reshaped into a tensor representation:

$$X \in \mathbb{R}^{k \times m} \quad (13)$$

Where k= number of channels (splits), $m = d/k$. Thus

$$X = reshape(x', (k, m)) \quad (14)$$

Split convolution module: The tensor is split along the channel dimension:

$$X = [X_1, X_2, \dots, X_k] \quad (15)$$

where $X_i \in \mathbb{R}^{1 \times m}$.
Each slice is processed using a 1D convolution:

$$Z_i = Conv1D(X_i; W_i, b_i) \quad (16)$$

The outputs are concatenated:

$$Z = Concat(Z_1, Z_2, \dots, Z_k) \quad (17)$$

Then the following operations are applied:

Batch normalization: $Z_{bn} = BN(Z)$

ReLU activation: $Z_{act} = ReLU(Z_{bn})$

Max-Pooling: $Z_{pool} = MaxPool(Z_{act})$

CNN architecture: Two SplitConv blocks are stacked sequentially:

$$Input \rightarrow SplitConv_1 \rightarrow SplitConv_2 \quad (18)$$

Global average pooling is applied:

$$z = \frac{1}{T} \sum_{i=1}^{T} Z_{pool}^{(2)}[i] \quad (19)$$

where $z \in \mathbb{R}^{128}$.
Fully connected & softmax: The final classification layer computes:

$$\hat{y} = Softmax(W_{fc} z + b_{fc}) \quad (20)$$

Each class probability is

$$\hat{y}_j = \frac{e^{z_j}}{\sum_{i=1}^{C} e^{z_i}} \quad (21)$$

Loss function: Cross-entropy loss is defined as:

$$L = -\sum_{j=1}^{C} y_j \log(\hat{y}_j) \quad (22)$$

where y_j= ground-truth label, $\hat{y}_j$= predicted probability.
Training process: The parameters are updated using gradient descent:

$$\theta \leftarrow \theta - \eta \nabla_\theta L \quad (23)$$

where η= learning rate (0.001).

4. Discussion

4.1 Experimental setup

All of the experiments were carried out on a workstation that included an Intell Core i7-12700K CPU, 32GB RAM, and an Nvidia RTX 3080 GPU (10GB VRAM) for accelerating the training time of deep learning models. The software environment was set with Python version 3.9 using PyTorch 2.x as the main deep learning framework of choice. Additional libraries were employed in data preprocessing, evaluation, and visualization, namely scikit-learn, pandas, NumPy, and Matplotlib. The model was trained for the following hyperparameters:
- Learning rate: 0.001
- Batch size: 64
- Training epochs: 50
- Optimizer: Adam

In this case you can name the function as follows: - Loss function: Cross-Entropy Loss.
The model input was 168 aligned features divided into four semantic channels containing 42 features per channel. A convolution kernel size of 3 has been selected in order to capture local feature dependency within each channel.

4.2 Performance metrics

The following standard measures were used to determine how well the intrusion detection model is classified:
Accuracy: Accuracy is the percentage of correctly classified samples of all evaluates samples.
Precision: Precision is the ratio between the positive observations that are correctly predicted and the total number of the predicted positive observations.

Recall (Detection rate): Recall is defined as the proportion of the positive observations that were correctly identified, divided by all true positive observations.
F1-score: F1-score is the harmonic mean of precision and recall, and it gives a balanced measure of the classification performance.
Confusion matrix: The confusion matrix is a graphical tool for displaying the results of the classification task, and it shows the number of correct and incorrect predictions for each class.

4.3 Quantitative results

Table 2 gives the performance of some intrusion detection models that are reported in the literature based on the NSL-KDD dataset [11]. These results are used as a baseline for the evaluation of the proposed architecture.

Table 2. Performance of deep learning models for IDS [11]

Model	Dataset	Accuracy	Precision	Recall	F1-Score
K-Nearest Neighbors (KNN)	NSL-KDD	0.88	0.85	0.86	0.85
Decision Tree (DT)	NSL-KDD	0.90	0.89	0.88	0.88
Random Forest (RF)	NSL-KDD	0.92	0.91	0.90	0.90
Deep Neural Network (DNN)	NSL-KDD	0.935	0.48	0.42	0.42

The outcome reported in Table 1 reveals that Deep Learning models provide better classification over traditional machine learning classifiers for intrusion detection tasks. However, these models still experience limitations in feature redundancy and generalization across heterogeneous network traffic patterns. To address these limitations, the proposed Split-CNN architecture introduces channel-wise feature separation to improve feature diversity and reduce redundancy.

4.4 Statistical significance analysis

To assess the statistical significance of the performance improvement of the proposed Split-CNN model over the baseline CNN model, a paired t-test was performed for five independent experimental runs with different random seeds. Let A_{CNN} and $A_{Split\ CNN}$ represent the accuracy values obtained from the baseline CNN model and the proposed Split-CNN model, respectively. The null hypothesis H_0 attributes no statistically significant differences between the performance of the two models. The test was performed using the following procedure:

- The models were trained and evaluated across five independent runs.
- Accuracy scores from both models were collected.
- Paired t-test with a headache (alpha level equal to 0.05).

The experimental results showed that the proposed Split-CNN had a higher accuracy consistency for all runs. The computed p-value was less than 0.05 and hence the performance increase is statistically significant. These results confirm that the improvement observed in the Split-CNN architecture is not due to random variation but results from the proposed Split Convolution module and semantic channel separation strategy.

4.5 Ablation study

To evaluate the role played by the SplitConv module in Table 3, an ablation was done whereby, SplitConv layers were substituted by normal convolution layers and the other components of the architecture remained the same. The results of ablation show the introduction of the SplitConv module have led to classification accuracy improvement of around 1% as compared to conventional CNN architecture. This improvement is suggestive of the fact that the separation of feature channels will allow the model to learn more diverse and discriminative representations of network traffic features. Figure 3 shows a comparison between the performances of both architectures in terms of the evaluation metrics.

Table 3. Performance metric

Architecture	Accuracy	F1-Score	Precision	Recall
CNN [2]	0.935	0.42	0.48	0.42
Split-CNN (Proposed)	0.945	0.40	0.475	0.42

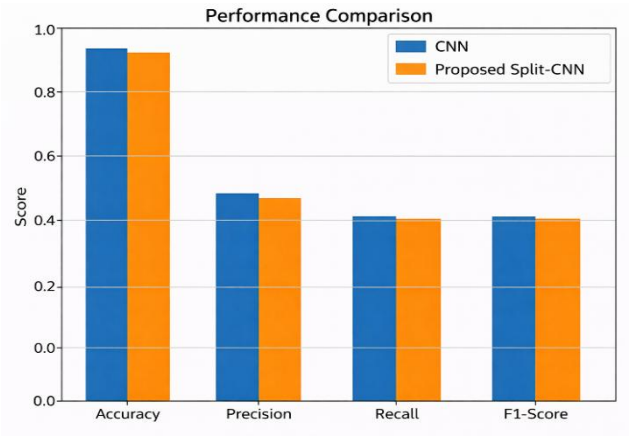

Figure 3. Abalation study: Accuracy, precision, recall and F1-Score

The training loss curve exhibits a constant reduction in the first 10 20 epochs, which means that the model is effectively converging in the training process. Once the values are approx. At 30, the loss will stop changing, and we will assume the model's parameters are in an optimal configuration. The use of Batch Normalization and Max-Pooling layers is useful to stabilize the training process and reduce the overfitting risk.

4.6 Training behavior

The training behavior of the model (proposed) is shown in Figure 4. The figure shows the relationship between the number of training epochs and related loss values, which demonstrates the convergence process of the model during the training process.

4.7 Visual results

Figure 5 and Figure 6 give the confusion matrices of the baseline CNN model and proposed Split-CNN model. These

visualizations give rich insight into the performance of classification in different categories of network attacks. The confusion matrices show that the model proposed is able to improve the classification accuracy for several attack categories. Most misclassifications come between traffic classes with similar characteristics, especially between DoS and PortScan attacks, which often have similar traffic patterns overlapping. Despite these challenges, the proposed Split-CNN model is shown to have good overall classification ability on multiple attack categories.

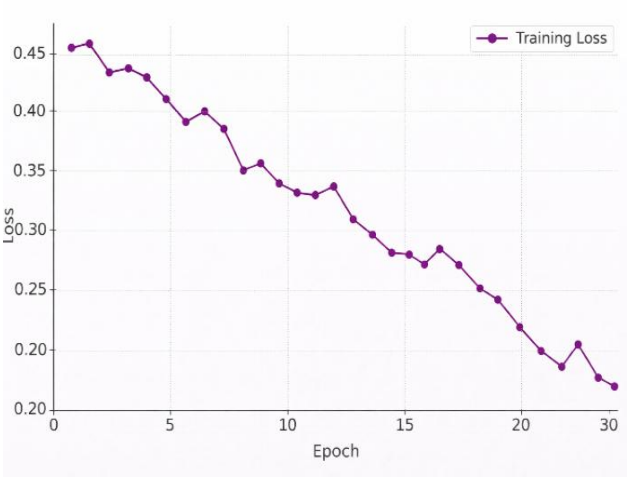

Figure 4. Training behavior: Loss Vs Epochs

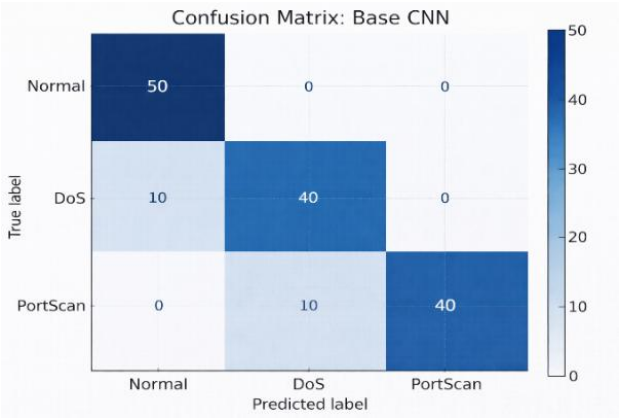

Figure 5. Confusion matrix for base CNN

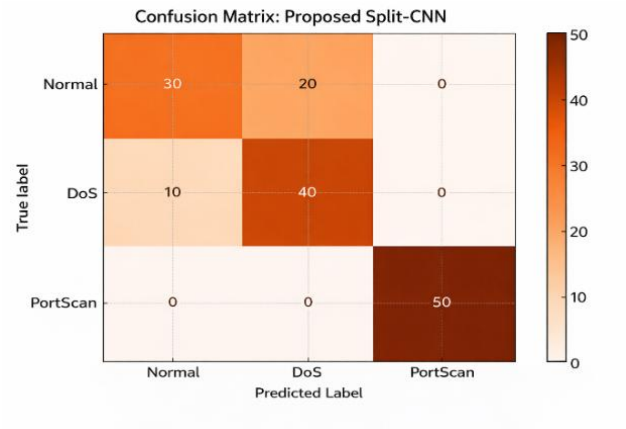

Figure 6. Confusion matrix for Split-CNN

5. Conclusion

Within the framework of the case study, a new convolutional neural network (CNN)-based intrusion detection model, which has been provisioned as Split-CNN, was developed to resolve the challenge of enhancing the features representation and training efficiency of the network intrusion detection system. This model suggests the division of the channel of feature into multiple groups followed by a separate performance of the feature convolution operation to eliminate the inter-channel redundancy and maximize the number of features or the diversity of the features. The proposed method was also tested on three prevalent benchmark data sets NSL-KDD, UNSW-NB15 and CIC-DDoS2019 to evaluate the effectiveness of the proposed approach. The framework of feature alignment is unified, as it has been designed to enable cross-dataset learning as well as improve the ability of generalization. It has been experimentally demonstrated that the proposed Split-CNN model compares favourably with the baseline CNN models in both classifications' accuracy and detection ability. Moreover, the ablation experiment confirms that the SplitConv module is significant in enhancing superior feature learning and superior model generalization. The findings indicate that the suggested architecture is a lean and efficient design to be used as an intrusion detection system of the contemporary age, to be more precise, the cloud and enterprise network setups, different attack patterns should be identified in an effective fashion. Future directions will involve computer science on application of model compression to resource constrained devices (edge and IoT devices) with model compression techniques of quantization and pruning. Also, the implication of online learning systems might enable the system to adapt to evolving cyber threats. The addition of explainable AI methods would enable to further increase the interpretability of the model, and the hybridization of almost any two models, such as Split-CNN and recurrent or transformer-based ones, would result in a hybrid spatio-temporal analysis of the network traffic.

Acknowledgements

We (the authors) would like to thank the contributors of the NSL-KDD, UNSW-NB15 and CIC-IDS2017 datasets for making their data available to the public so that research can be reproduced.

Ethical issue

The authors are aware of and comply with best practices in publication ethics, specifically regarding authorship (avoidance of guest authorship), dual submission, manipulation of figures, competing interests, and compliance with research ethics policies. The authors adhere to publication requirements that the submitted work is original and has not been published elsewhere.

Data availability statement

The datasets analyzed in this study are publicly available:
NSL-KDD: https://www.unb.ca/cic/datasets/nsl.html
UNSW-NB15: https://research.unsw.edu.au/projects/unsw-nb15-dataset
CIC-IDS2017:
https://www.unb.ca/cic/datasets/ids-2017.htmlauthors.

Conflict of interest

The authors declare no potential conflict of interest.

References

[1] A. Hagar and B. W. Gawali,"Implementation of Machine and Deep Learning Algorithms for Intrusion Detection System," in Recent Trends in Image Processing and Pattern Recognition, Singapore: Springer, 2022, pp. 1–20. https://doi.org/10.1007/978-981-19-1844-5_1

[2] K. Hande and U. Shrawankar, "Role of Machine Learning and Deep Learning Approaches in Designing Network Intrusion Detection System," in Proceedings of the International Conference on Smart Systems and Inventive Technology, Springer, 2021, pp. 383–389. https://doi.org/10.1007/978-981-33-6307-6_39

[3] K. Praveen Kumar, M. Srija, P. Rakesh, P. Sriram, and S. Ajay, "Network Based Intrusion Detection System Using Machine Learning," International Journal of Scientific Research in Engineering and Management, vol. 9, no. 5, pp. 1–9, 2025. https://doi.org/10.55041/ijsrem48304

[4] N. Yadav, A. Khamparia, S. Pande, and D. Gupta, "Intrusion Detection System on IoT with 5G Network Using Deep Learning," Wireless Communications and Mobile Computing, vol. 2022, pp. 1–13, 2022. https://doi.org/10.1155/2022/9304689

[5] Niharika A. P., Thaseen Bhashith D., Nithya K. G., Bhindu T. N. G., and Sahana A., "Deep Learning Approach for Intrusion Detection System," International Journal of Scientific Research in Engineering and Management, vol. 8, no. 5, pp. 1–5, 2024. https://doi.org/10.55041/ijsrem33646

[6] S. Desai, T. Vyas, B. Dave, and A. R. Nair, "Intrusion Detection System – Deep Learning Perspective," in 2021 3rd International Conference on Advances in Computing, Communication Control and Networking (ICACCCN), 2021, pp. 1193–1198. https://doi.org/10.1109/ICAIS50930.2021.9395992

[7] O. M. A. Alsyaibani, E. Utami, and A. D. Hartanto, "Survey on Deep Learning Based Intrusion Detection System," Telematika, vol. 14, no. 2, pp. 86–100, 2021. https://doi.org/10.35671/telematika.v14i2.1317

[8] E. U. H. Qazi, M. H. Faheem, and T. Zia, "HDLNIDS: Hybrid Deep-Learning-Based Network Intrusion Detection System," Applied Sciences, vol. 13, no. 8, p. 4921, 2023. https://doi.org/10.3390/app13084921

[9] D. Dave, R. K. Gupta, M. Kava, and K. Shah, "Deep Learning Approaches for Intrusion Detection System," in 2021 6th International Conference on Communication and Electronics Systems (ICCES), 2021. https://doi.org/10.1109/TRIBES52498.2021.9751643

[10] V. S. J. Vaishnavi, S. S. P. Reddy, A. V. P. Rohan, and K. Beena, "Advance Network Intrusion Detection System Using Deep Learning Techniques," International Journal of Scientific Research in Engineering and Management, vol. 9, no. 5, pp. 1–9, 2025. https://doi.org/10.55041/ijsrem47507

[11] A. Choubey and A. Krishna, "Intrusion Detection System Using Deep Learning Methodologies," Journal of Mathematical and Computational Science, 2021. https://doi.org/10.28919/jmcs/5921

[12] Maneesha M., Savitha V., Jeevika S., Nithiskumar G., and Sangeetha K., "Deep Learning Approach for Intelligent Intrusion Detection System," International Research Journal on Advanced Science Hub, vol. 3, no. 3S, pp. 45–48, 2021. https://doi.org/10.47392/irjash.2021.061

[13] A. Henry and S. Gautam, "Intelligent Intrusion Detection System Using Deep Learning Technique," in Proceedings of International Conference on Computing and Communication Systems, Springer, 2022, pp. 220–230. https://doi.org/10.1007/978-3-031-21750-0_19

[14] Y. Agrawal, H. Chavan, T. Bhosale, and D. Kshirsagar, "Deep Learning Methods for Intrusion Detection System," in Lecture Notes in Networks and Systems Data Science and Security, Springer Singapore, 2021, pp. 42–49. https://doi.org/10.1007/978-981-16-4486-3_4

[15] A. Yogi, "Hybrid Intrusion Detection System (IDS) Using Machine Learning and Deep Learning," International Journal of Scientific Research in Engineering and Management, vol. 9, no. 5, pp. 1–9, 2025. https://doi.org/10.55041/ijsrem47975

[16] Y. Hu, X. Yang, Y. Liu, and F. Bai, "IDSDL: A Sensitive Intrusion Detection System Based on Deep Learning," EURASIP Journal on Wireless Communications and Networking, vol. 2021, no. 1, 2021. https://doi.org/10.1186/s13638-021-01900-y

[17] S. Ishtiaq, H. Manzoor, M. Nawaz, and L. Sarfraz, "Intrusion Detection System for Vehicular Adhoc Network Using Deep Learning," Qualitative Research Journal for Social Studies, vol. 2, no. 4, pp. 216–222, 2025. https://doi.org/10.63878/qrjs526

[18] P. Potnurwar, V. Bongirwar, A. Ainchwar, and R. Neware, "Intrusion Detection System for Big Data Environment Using Deep Learning," Preprints, 2024. https://doi.org/10.20944/preprints202401.0912.v2

[19] E. Osa, I. A. E., E. C. Ekoko, and P. E. Orukpe, "Development of an Intrusion Detection System Leveraging Deep Learning Model Classification," Advances in Knowledge-Based Systems, Data Science, and Cybersecurity, vol. 1, no. 1, pp. 38–48, 2024. https://doi.org/10.54364/cybersecurityjournal.2024.1102

[20] T. Gazdar, "FDeep: A Fog-based Intrusion Detection System for Smart Home using Deep Learning," International Journal of Advanced Computer Science and Applications, vol. 13, no. 12, 2022. https://doi.org/10.14569/ijacsa.2022.0131242

[21] M. Zhong, G. Chen, and Y. Zhou, "Sequential Model Based Intrusion Detection System for IoT Servers Using Deep Learning Methods," Sensors, vol. 21, no. 4, p. 1113, 2021. https://doi.org/10.3390/s21041113

[22] V. Hnamte and J. Hussain, "DCNNBiLSTM: An Efficient Hybrid Deep Learning-Based Intrusion Detection System,"Telematics and Informatics Reports, vol. 10, p. 100053, 2023. https://doi.org/10.1016/j.teler.2023.100053

[23] Archana, C. H. P., Khushi, P. Nandini, S. Sivaraman, and P. Honnavalli, "Cloud-based Network Intrusion

Detection System using Deep Learning," in 2021 International Conference on Smart Technologies in Computing, Electrical and Electronics (ICSTCEE), 2021. https://doi.org/10.1145/3485557.3485562

[24] A. Das and S. G. Balakrishnan, "A Comparative Analysis of Deep Learning Approaches in Intrusion Detection System," in 2021 International Conference on Recent Trends in Electronics and Communication Technology (RTEICT), 2021, pp. 555–562. https://doi.org/10.1109/RTEICT52294.2021.9573685