HETCLNN: A Lightweight Intrusion Detection Network with Class-Aware Self-Knowledge Distillation

Chenghao Liu[1] and Wenjun Yang[1]

[1] School of Computer Science and Engineering, Tianjin University of Technology, China

yangwj@tjut.edu.cn

Abstract. With the proliferation of edge computing, building efficient NIDS faces challenges related to limited computational resources and class imbalance. To address the issues of high overhead and poor detection of rare attacks in existing models, this paper proposes a lightweight intrusion detection model based on Class-Aware Self-Knowledge Distillation (CASKD). Architecturally, we design a lightweight network utilizing Heterogeneous Convolution (HetConv)-based residual and inverted residual structures. For training, a temperature-based CASKD method is introduced to tackle extreme class imbalance. Experimental results on CIC-IDS2017 and Bot-IoT datasets demonstrate classification accuracies exceeding 99%. The proposed method significantly reduces computational overhead while improving detection precision for rare attacks, achieving an optimal balance between model compactness and performance.

Keywords: NIDS, CASKD, HetConv, Lightweight Model, Class Imbalance

.

1 Introduction

The Internet of Things (IoT) is rapidly expanding across smart homes, healthcare, and industrial control, yet heterogeneous access across cloud–edge–device layers also amplifies security risks, from data leakage to malicious intrusions. Network Intrusion Detection Systems (NIDS) provide a key proactive defense and are commonly grouped into signature-based and anomaly-based approaches: signature-based methods match traffic to known patterns with high precision but offer limited protection against zero-day attacks and require continual rule maintenance, whereas anomaly-based methods learn normal behavior to flag deviations but can suffer from high false positive rates in real deployments. [1]

As anomaly-based NIDS increasingly adopt deep learning (DL) for its representation learning and automatic feature extraction, practical deployment at the IoT edge exposes a central tension—resource-limited devices must run models that are often parameter-heavy and FLOP-intensive, making real-time inference difficult. This has motivated compression techniques such as pruning [6], quantization [5], and knowledge distilla-

tion (KD) [8]. KD is particularly attractive because it transfers soft targets, together with their inter-class structure, from a strong teacher to a compact student without increasing inference cost. However, many KD implementations adopt a globally fixed temperature, an assumption that conflicts with the severe class imbalance and varying class difficulty typical of NIDS datasets; as a result, majority traffic can dominate the distilled supervision and minority attacks may be underrepresented, weakening their feature learning. Against this background, the main contributions are as follows:

1) We propose a Class-Aware Self-Knowledge Distillation (CASKD) strategy. By deriving distinct distillation temperatures based on the sample quantity of different classes, this approach enables the model to learn richer internal information and knowledge representations, thereby enhancing its robustness and stability.

2) We utilize Heterogeneous Convolution (HetConv) and Squeeze-and-Excitation (SE) mechanisms to construct lightweight residual and inverted residual blocks, respectively, and propose the HETCLNN model. This model improves computational efficiency while ensuring effective attack identification.

The remainder of this paper is organized as follows. Section 2 reviews related studies on intrusion detection, lightweight models, and knowledge distillation. Section 3 introduces the proposed HETCLNN framework, including the lightweight network design and the Class-Aware Self-Knowledge Distillation (CASKD) strategy. Section 4 presents the datasets, experimental setup, evaluation metrics, and performance comparisons. Section 5 concludes the paper and presents future work.

2 Related Work

Recent intrusion detection studies span both classical machine learning and deep learning. Farnaaz and Jabbar [4] showed that a Random Forest ensemble can reduce the overfitting of single trees and achieve strong results on NSL-KDD, while Azimjonov and Kim [2] combined Ridge Regression–based feature selection with an SGDC classifier to lower computation without sacrificing precision—an attractive direction when deployment budgets are tight.

Deep models excel at capturing spatiotemporal patterns, yet imbalanced traffic and edge-side budgets often limit their practical use. Res-TranBiLSTM [17] integrates ResNet, Transformer, and BiLSTM and applies SMOTE-ENN, yet reports higher false positives likely tied to normal-class bias; to improve deployability, Self-KD [20] distills internal deep features to guide shallow layers, and BERT-of-Theseus compression [18] replaces modules probabilistically to obtain smaller Transformer variants. DuGAT-LSTM [3] combines Dual GAT and LSTM with chaotic optimization to enhance convergence, illustrating that performance gains often come with added design complexity. Overall, these efforts highlight a recurring tension between representation power, computational overhead, and robustness under imbalanced traffic.

3 Method

This chapter presents the proposed intrusion detection framework based on improved feature selection and lightweight deep learning. The framework primarily consists of two core components: a lightweight model architecture utilizing Heterogeneous Convolution (HetConv)[15], and a training optimization strategy based on Class-Aware Self-Knowledge Distillation (CASKD).

3.1 HetConv

In conventional CNNs, convolutional layers usually employ a homogeneous design: the same $K \times K$ kernel is applied to every input channel when producing each output channel. Empirically, however,feature maps often exhibit notable redundancy along the channel dimension (**Fig. 1**),meaning that large kernels are not equally necessary everywhere. While some channels benefit from broader spatial/temporal context,
others mainly require lightweight channel mixing, and treating them identically leads to avoidable parameters and FLOPs.

HetConv addresses this inefficiency by splitting each filter into two complementary parts. A small subset of channels ($1/P$)is processed with $K \times K$ kernels (the K-part) to capture local context, whereas the remaining channels use 1×1 pointwise kernels (the 1-part) for efficient cross-channel fusion.To prevent any single channel from being consistently excluded from contextual extraction,HetConv further rotates the K-part assignment across filters via a cyclic shift, so that different output channels "see" different channel groups under $K \times K$ operations. With this heterogeneous structure in place, we next quantify the compactness gain by comparing the parameter counts of standard convolution and HetConv. Let C_{in} and C_{out} denote the input and output channels, and $K \times K$ the kernel size. In standard convolution, the parameter count W_{SC} is:

$$W_{SC} = C_{in} \times C_{out} \times K \times K \tag{1}$$

The total number of parameters in Heterogeneous Convolution (HetConv), denoted as W_{Het} ,consists of the linear superposition of two components: the retained standard convolution kernels (e.g., $K \times K$) and the substituted pointwise convolution kernels (e.g., 1×1). Based on the partitioning of input channels defined by the hyperparameter P ,the specific derivation of the calculation formula is as follows:

$$W_{Het} = C_{in} \cdot C_{out} \cdot \left(\frac{K^2}{P} + \left(1 - \frac{1}{P} \right) \right) \tag{2}$$

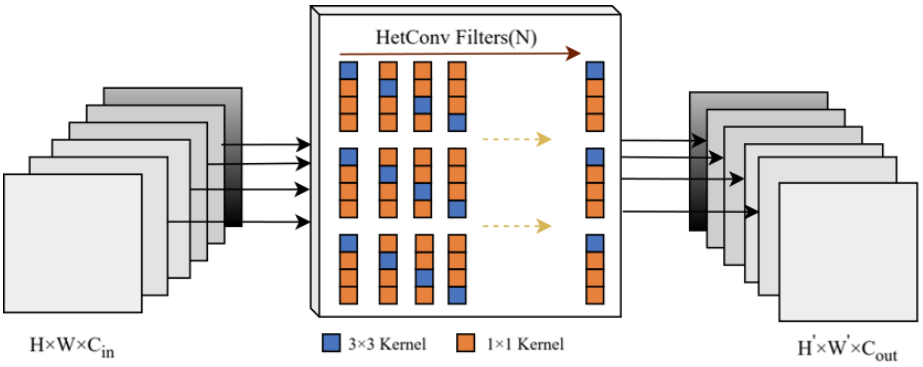

Fig. 1. The structure of HetConv

3.2 HETCLNN

To reconcile limited edge resources with high detection accuracy in IoT settings, we propose HETCLNN, a lightweight architecture that balances effectiveness and efficiency. As shown in **Fig. 2**, the network is built around lightweight residual blocks and inverted residual blocks. A DSConv stem—adopted following [19]—decouples spatial filtering from channel fusion, mapping raw traffic features into a higher-dimensional representation while avoiding the parameter and FLOP burden of standard convolutions. For deeper semantic modeling under tight budgets, we employ lightweight residual blocks that couple HetConv with SE attention: HetConv improves parameter efficiency via channel-wise heterogeneous processing, the SE module adaptively recalibrates channel responses, and skip connections preserve feature manifolds and stabilize optimization by alleviating gradient degradation.

Furthermore, we incorporate Lightweight Inverted Residual Blocks derived from MobileNetV2 [13]. Distinct from the 'compress-then-expand' logic of traditional ResNet bottlenecks [7], this module adopts an 'expand-then-compress' inverted bottleneck structure. This design utilizes linear bottlenecks to eliminate channel redundancy and ensure compact feature transmission, rendering it highly adaptable to resource-constrained nodes. Finally, Global Average Pooling (GAP) replaces traditional fully connected layers to minimize model size, followed by a Dropout layer to improve regularization and reduce overfitting in supervised traffic classification.

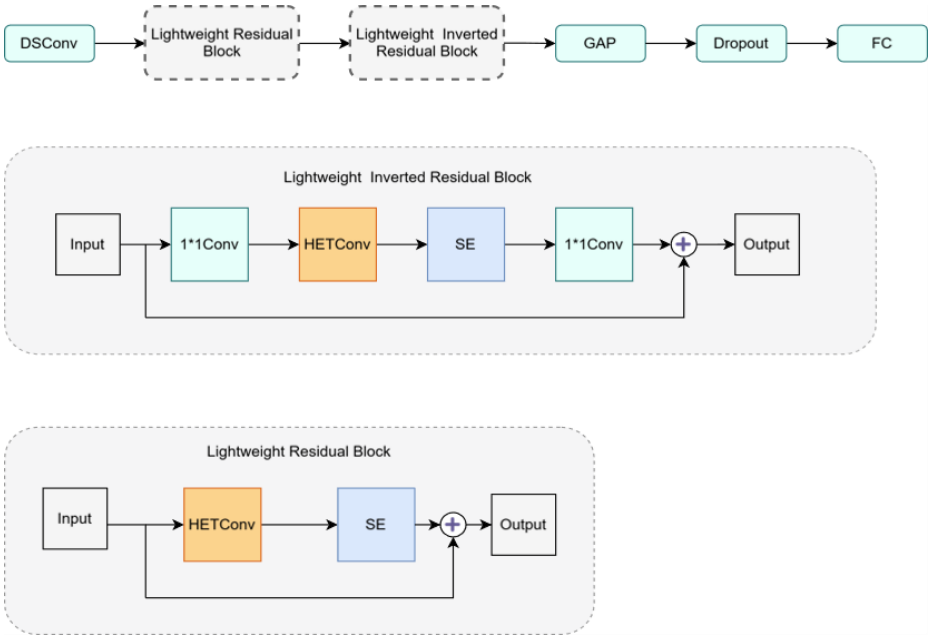

Fig. 2. The structure of HETCLNN Model

3.3 CASKD

Knowledge distillation (KD) is a canonical teacher–student compression paradigm in which a lightweight student is guided to imitate a stronger teacher, often exceeding what can be achieved by hard-label supervision alone. Its effectiveness largely comes from temperature-scaled soft targets, whose non-peaked distributions expose inter-class similarity structure and provide richer supervision than one-hot labels.

To exploit this benefit without introducing an external teacher, we adopt self-knowledge distillation (SKD), where the model's probability distribution from the previous epoch serves as a soft teacher signal for the current epoch. In practice, this temporal self-evolution encourages the network to integrate ground-truth labels with the structural information retained in its historical predictions, improving generalization and predictive stability.The optimization objective thus balances correct label fitting with consistency to past knowledge.Given the severe class imbalance typical of network traffic, a single global temperature can misrepresent per-class frequency and difficulty, often leaving minority attacks under-distilled. We therefore adopt a class-aware, dynamic temperature that modulates softening according to the class prior in the training set.For class c , the distillation temperature T_c is defined as:

$$T_c = \min\left(T_{cap}, T_{min} + (T_{max} - T_{min}) \times \frac{\log(n_{max} / n_c)}{\log(n_{max} / n_{min})}\right) \tag{3}$$

Here, n_c and n_{max} the sample counts of class c and the majority class, respectively, and T_{cap} is the clipping threshold. The logarithmic mapping dampens extreme imbalance by assigning higher temperatures to minority classes—encouraging richer knowledge transfer—while keeping lower temperatures for majority classes to retain prediction confidence.With T_c in place, we rewrite the distillation loss,denoted as L_{KD} , into a sample-adaptive form:

$$L_{KD} = \frac{1}{N}\sum_{i=1}^{N} T_{y_i}^2 \cdot \mathcal{D}_{KL}\left(\sigma\left(\frac{z_s^{(i)}}{T_{y_i}}\right) \| \sigma\left(\frac{z_t^{(i)}}{T_{y_i}}\right)\right) \tag{4}$$

In this formulation, Z_s and Z_t are the student and teacher logits, and $\sigma(\cdot)$ denotes the Softmax function.The temperature T_{y_i} is selected according to the ground-truth class of sample i .To offset the gradient attenuation induced by higher temperatures, we include the scaling term $T_{y_i}^2$, which helps keep gradient contributions from frequent and rare classes more comparable during backpropagation.

We optimize HETCLNN with a joint objective that combines Focal Loss and the dynamic temperature–based distillation loss. For a mini-batch of B samples, the total loss L_{total} is defined as:

$$L_{\text{total}} = (1-\alpha_t)L_{\text{FL}} + \alpha_t L_{\text{KD}} \tag{5}$$

Here, L_{FL} is the focal-loss term, and α_t is a linearly decaying weight over epochs. A larger α_t early on emphasizes distilling “dark knowledge” from soft labels, while its gradual decay shifts training toward fitting ground-truth targets as optimization stabilizes.The overall design is illustrated in **Fig. 3**.

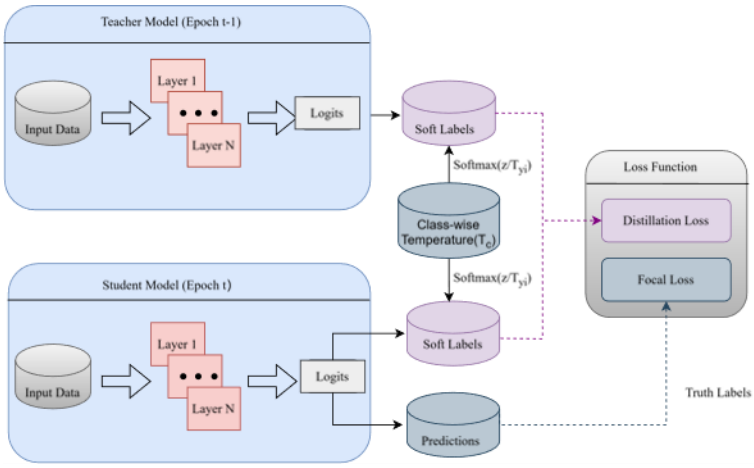

Fig. 3. The structure of CASKD

4 Experimental Setup and Results

The experiments were conducted on a server with Intel Xeon CPUs and an NVIDIA RTX 3090 GPU.

4.1 Dataset and Data Preprocessing

In this study, we utilized two benchmark datasets: CIC-IDS2017 [14] and Bot-IoT [11]. The datasets were partitioned into three distinct subsets. Specifically, 20% of the data was strictly reserved as the test set for final evaluation. The remaining 80% constituted the training set, from which 20% was further allocated as a validation set for hyperparameter tuning and overfitting monitoring.

Experiments are conducted on two complementary benchmarks. For enterprise-style traffic, we use the Wednesday-workingHours.csv subset of CIC-IDS2017, covering Application-layer DoS and Heartbleed attacks, with each flow represented by 78 CICFlowMeter-extracted statistical features; the class distribution is reported in **Table 1**. To evaluate IoT-oriented threats, we adopt Bot-IoT, which simulates botnet traffic in smart-home/industrial scenarios (e.g., MQTT and CoAP) using 46 flow- and window-based features; its extreme class imbalance, reflective of background-dominated traffic during botnet outbreaks, provides a stringent test of lightweight-model robustness under resource constraints (**Table 2**).

During data preprocessing, irrelevant and potentially leakage-related fields were removed, including auxiliary label columns, flow identifiers, timestamp-related fields, and non-numeric attributes that could not be reliably converted into numerical representations. Categorical fields were converted into numerical values using label encoding. Infinite values were first replaced with NaN. Columns containing no valid values were removed, and the remaining missing values were imputed using the median value of each corresponding feature.

To avoid data leakage, feature normalization was performed using min-max normalization fitted only on the training set. Specifically, the minimum and maximum values of each feature were computed from the training data, and the same normalization parameters were then applied to the validation and test sets. The normalization operation is defined as follows:

$$x' = \frac{x - x_{\min^{\text{train}}}}{x_{\max^{\text{train}}} - x_{\min^{\text{train}}}} \tag{6}$$

where x denotes the original feature value, x' denotes the normalized value, $x_{\min^{\text{train}}}$ and $x_{\max^{\text{train}}}$ represent the minimum and maximum values of the corresponding feature calculated from the training set.

After preprocessing, each network traffic sample was represented as a one-dimensional feature vector of length L. The feature order was determined by the column order

after preprocessing. When feature selection was applied, the final feature order followed the selected feature indices produced by the feature selection method; otherwise, the original preprocessed column order was preserved. In this study, feature selection reduced the feature dimension of CIC-IDS2017 to 45 and that of Bot-IoT to 32.

To improve the reliability of the results, we repeated each experiment five times. Each run used a different random seed. All runs used the same preprocessing pipeline and data-splitting protocol. The random seeds affected model initialization, mini-batch shuffling, and training stochasticity. The final results are reported as mean ± standard deviation. This repeated-run setting provides a more robust estimate of model performance and reduces the risk of over-interpreting single-run observations, especially for minority classes under severe class imbalance.

Table 1. Class Distribution of the CIC-IDS2017

Category	Training Set	Validation Set	Test Set
Benign	281609	70403	88003
GoldenEye	6587	1647	2059
Hulk	147886	36971	46215
Slowhttptest	3519	880	1100
Slowloris	3710	927	1159
Heartbleed	7	2	2
Total	443318	110830	138538

Table 2. Class Distribution of the Bot-IoT

Category	Training Set	Validation Set	Test Set
Normal	306	76	95
DoS	1,056,166	264,042	330,052
DDoS	1,233,039	308,260	385,325
Reconnaissance	58,292	14,573	18,217
Theft	50	13	16
Total	2,347,853	586,964	733,705

4.2 Evaluation Metrics

To further show the effectiveness of our method, we compared it with well-known lightweight intrusion detection models.

4.3 Classification Results of HETCLNN

We train HETCLNN on the preprocessed data with the CASKD strategy. Since overall accuracy may obscure minority-class behavior under severe class imbalance, we further report confusion matrices and per-class precision, recall, and F1-score on both CIC-

IDS2017 and Bot-IoT. As shown in **Table 3** and **Fig. 4**, the proposed model achieves strong performance on the major classes in CIC-IDS2017. The Heartbleed class is correctly classified in the current test split; however, its support is extremely limited, so this result should be interpreted as preliminary evidence rather than a statistically stable conclusion. A similar pattern is observed on Bot-IoT (**Fig. 4** and **Fig. 5Fig. 5.** Confusion matrix on Bot-IoT), where the model performs well on the major categories and shows promising minority-class performance. In particular, the Theft class attains high recall with limited support, while its relatively lower precision indicates that false positives remain an issue for ultra-minority classes.

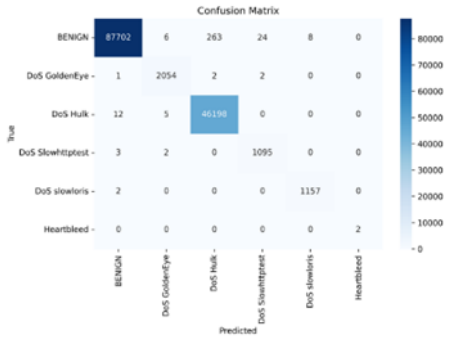

Fig. 4. Confusion matrix on CIC-IDS2017

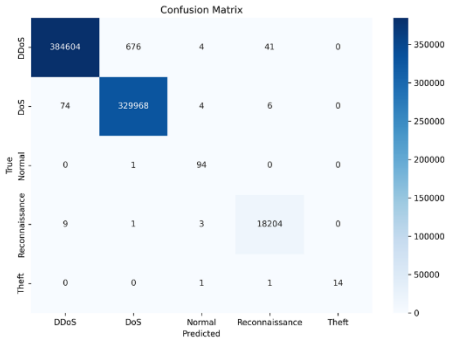

Fig. 5. Confusion matrix on Bot-IoT

Table 3. Per-class metrics on the CIC-IDS2017 test set.

Class	Precision	Recall	F1-score	Support
BENIGN	0.9998	0.9966	0.9982	88003
DoS GoldenEye	0.9928	0.9976	0.9951	2059
DoS Hulk	0.9943	0.9996	0.9970	46215
DoS Slowhttptest	0.9768	0.9955	0.9861	1100
DoS Slowloris	0.9931	0.9983	0.9957	1159

Heartbleed	1.0000	1.0000	1.0000	2
Macro Avg	0.9928	0.9979	0.9954	--
Weighted Avg	0.9981	0.9981	0.9981	--

Table 4. Per-class metrics on the Bot-IoT test set

Class	Precision	Recall	F1-score	Support
DDoS	0.9998	0.9981	0.9990	385325
DoS	0.9979	0.9997	0.9988	330052
Normal	0.8868	0.9895	0.9353	95
Reconnaissance	0.9974	0.9993	0.9983	18217
Theft	1.0000	0.8750	0.9333	16
Macro Avg	0.9764	0.9723	0.9730	--
Weighted Avg	0.9989	0.9989	0.9989	--

4.4 Ablation Studies

To further examine the effectiveness of the proposed method, we conducted an extended ablation study on both CIC-IDS2017 and Bot-IoT. In addition to the variant without CASKD, we also evaluated two architectural variants by removing HetConv and the Inverted Residual structure, respectively. This experimental design allows us to assess the individual contribution of the main components in the proposed framework.

As shown in **Table 5**, the full model achieves the best overall performance on both datasets. On CIC-IDS2017, removing CASKD decreases the accuracy by 0.11%, while removing HetConv or the Inverted Residual structure also results in consistent drops in Precision, Recall, and F1-score. On Bot-IoT, the impact of CASKD is more pronounced, with the accuracy dropping from 99.89% to 99.63% when CASKD is removed. The two architectural ablation variants also perform worse than the full model, indicating that both HetConv and the Inverted Residual structure contribute positively to the final results.

In summary, the ablation study confirms that CASKD is an important source of performance gain, especially under severe class imbalance, while the lightweight architectural components provide additional complementary improvements. The results support the effectiveness of the overall HETCLNN design.

Table 5. Ablation study of different components on two datasets.

Dataset	Case	Accuracy	Precision	Recall	F1-score
CIC-IDS2017	w/o CASKD	99.65 ± 0.03%	99.64 ± 0.03%	99.67 ± 0.03%	99.41 ± 0.05%
	w/o HetConv	99.68 ± 0.03%	99.53 ± 0.03%	99.54 ± 0.03%	99.45 ± 0.03%
	w/o Inverted Residual	99.61 ± 0.02%	99.58 ± 0.02%	99.57 ± 0.03%	99.45 ± 0.05%

	Our Model	99.76 0.02%	±	99.75 0.02%	±	99.79 0.02%	±	99.54 0.11%	±
Bot-IoT	w/o CASKD	99.63 0.04%	±	99.64 0.04%	±	99.67 0.04%	±	99.61 0.04%	±
	w/o HetConv	99.80 0.05%	±	99.78 0.04%	±	99.73 0.03%	±	99.72 0.05%	±
	w/o Inverted Residual	99.72 0.03%	±	99.70 0.03%	±	99.72 0.03%	±	99.70 0.04%	±
	Our Model	99.89 0.05%	±	99.89 0.04%	±	99.86 0.03%	±	99.89 0.05%	±

4.5 Comparison with Other Intrusion Detection Methods

Comparison with Baseline Deep Learning Models: To further verify the effectiveness of HETCLNN, we conducted comparative experiments with both traditional machine learning and deep learning baselines, including Random Forest(RF), CNN,DNN and RNN, under a unified experimental protocol. All models used the same preprocessing pipeline, data splits, and metrics. This ensured a fair comparison. **Table 6** reports the classification performance and model complexity of different models on the two datasets.

Comparison with Established Lightweight Architectures: To further show the effectiveness of our method, we compared it with well-known lightweight intrusion detection models.In **Table 7**,our model exhibits significantly lower computational costs, a reduced number of parameters, and a smaller model size compared to these established architectures. This empirical evidence substantiates that our proposed model is sufficiently lightweight and well-suited for deployment in resource-constrained environments.

Comparison with Other Deep Learning Methods: To place our approach in a broader context, we benchmark it against several representative deep learning–based intrusion detection models. **Table 8** summarizes the reported results of these methods on the two datasets, providing a direct reference for cross-model comparison.

Table 6. Comparison of Different Deep Learning Models on CIC-IDS2017 and Bot-IoT

Dataset	Model	Accuracy	Precision	Recall	F1-score	Model Size
CIC-IDS2017	RF	98.20%	98.25%	98.27%	98.22%	0.04MB
	DNN	98.88%	99.50%	99.52%	99.55%	0.20MB
	CNN	98.55%	99.47%	98.55%	98.40%	1.5 MB
	RNN	99.30%	99.30%	99.30%	99.30%	0.05 MB

	Our Model	99.76%	99.75%	99.79%	99.54%	0.12 MB
Bot-IoT	RF	98.34%	98.37%	98.37%	98.32%	0.04MB
	DNN	98.10%	98.40%	98.41%	98.52%	0.20MB
	CNN	97.54%	97.55%	97.54%	97.54%	1.5 MB
	RNN	99.86%	99.86%	99.86%	99.86%	0.05 MB
	Our Model	99.89%	99.89%	99.86%	99.87%	0.12 MB

Table 7. Model-level efficiency comparison under the CIC-IDS2017 and Bot-IoT input settings

Model	CIC-IDS2017			Bot-IoT		
	Params	FLOPs	Model Size	Params	FLOPs	Model Size
MobileNet_V2	3.5×10^6	3.6×10^6	13.4 MB	3.5×10^6	3.6×10^6	13.4 MB
MobileNet_V3_Small[9]	2.5×10^6	2.6×10^6	9.8 MB	2.5×10^6	2.6×10^6	9.8 MB
EfficientNet_B0[16]	5.3×10^6	5.4×10^6	20.1 MB	5.3×10^6	5.4×10^6	20.1 MB
ShuffleNet_V2_x2_0[12]	7.4×10^6	7.4×10^6	28.1 MB	7.4×10^6	7.4×10^6	28.1 MB
Our Model	3.1×10^4	2.8×10^4	0.12 MB	3.1×10^4	2.8×10^4	0.12 MB

Table 8. Performance Comparison on CIC-IDS2017 and Bot-IoT

Dataset	Model	Accuracy	Precision	Recall	F1-score
CIC-IDS2017	KD-TCNN [19]	99.44%	99.48%	99.47%	99.46%
	BT-TPF [18]	99.60%	99.60%	99.60%	99.60%
	Our Model	99.76%	99.75%	99.79%	99.54%
Bot-IoT	CL-SKD-L [20]	97.54%	97.55%	97.54%	97.54%
	PNet-IDS [10]	99.86%	99.86%	99.86%	99.86%
	Our Model	99.89%	99.89%	99.86%	99.87%

5 Conclusion

This work introduces HETCLNN to bridge the gap between strong intrusion detection performance and the tight compute budgets typical of IoT edge deployments. By pairing lightweight heterogeneous convolutions with inverted residual blocks, the model substantially reduces computational complexity without sacrificing representational capacity. To further cope with the skewed traffic distributions that routinely appear in practice, we design a Class-Aware Self-Knowledge Distillation (CASKD) strategy that adapts soft supervision to each class and refines feature learning under imbalance. Experiments on CIC-IDS2017 and Bot-IoT show that HETCLNN delivers consistently

high accuracy while keeping the parameter footprint minimal compared with state-of-the-art baselines, supporting its potential suitability for resource-constrained deployment. Looking ahead, we plan to extend the framework toward detecting previously unseen, zero-day vulnerabilities.

References

1. Abdulganiyu, O.H., Ait Tchakoucht, T., Saheed, Y.K.: A systematic literature review for network intrusion detection system (IDS). International Journal of Information Security 22(5), 1125–1162 (2023). https://doi.org/10.1007/s10207-023-00682-2
2. Azimjonov, J., Kim, T.: Stochastic gradient descent classifier-based lightweight intrusion detection systems using the efficient feature subsets of datasets. Expert Systems with Applications 237, 121493 (2024). Article number: 121493
3. Devendiran, R., Turukmane, A.V.: DUGAT-LSTM: Deep learning based network intrusion detection system using chaotic optimization strategy. Expert Systems with Applications 245, 123027 (2024). https://doi.org/10.1016/j.eswa.2023.123027
4. Farnaaz, N., Jabbar, M.A.: Random forest modeling for network intrusion detection system. Procedia Computer Science 89, 213–217 (2016)
5. Gong, Y., Liu, L., Yang, M., Bourdev, L.: Compressing deep convolutional networks using vector quantization. arXiv:1412.6115 (2014)
6. Han, S., Mao, H., Dally, W.J.: Deep Compression: Compressing deep neural networks with pruning, trained quantization and Huffman coding. arXiv:1510.00149 (2015)
7. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 770–778 (2016)
8. Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. arXiv:1503.02531 (2015)
9. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., et al.: Searching for MobileNetV3. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 1314–1324 (2019)
10. Iliyasu, A.S., Siddiqui, A.J., Song, H., Abdu, F.J.: PNet-IDS: A lightweight and generalizable convolutional neural network for intrusion detection in Internet of Things. IEEE Access (2025)
11. Koroniotis, N., Moustafa, N., Sitnikova, E., Turnbull, B.: Towards the development of realistic botnet dataset in the Internet of Things for network forensic analytics: Bot-IoT dataset. Future Generation Computer Systems 100, 779–796 (2019)
12. Ma, N., Zhang, X., Zheng, H.T., Sun, J.: ShuffleNet V2: Practical guidelines for efficient CNN architecture design. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 116–131 (2018)
13. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: MobileNetV2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4510–4520 (2018)
14. Sharafaldin, I., Lashkari, A.H., Ghorbani, A.A.: Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: Proceedings of the International Conference on Information Systems Security and Privacy (ICISSP), vol. 1, pp. 108–116 (2018)

15. Singh, P., Verma, V.K., Rai, P., Namboodiri, V.P.: HetConv: Heterogeneous kernel-based convolutions for deep CNNs. arXiv:1903.04120 (2019). https://doi.org/10.48550/arXiv.1903.04120
16. Tan, M., Le, Q.: EfficientNet: Rethinking model scaling for convolutional neural networks. In: Proceedings of the International Conference on Machine Learning (ICML), pp. 6105–6114. PMLR (2019)
17. Wang, S., Xu, W., Liu, Y.: Res-TranBiLSTM: An intelligent approach for intrusion detection in the Internet of Things. Computer Networks 235, 109982 (2023). Article number: 109982
18. Wang, Z., Li, J., Yang, S., et al.: A lightweight IoT intrusion detection model based on improved BERT-of-Theseus. Expert Systems with Applications 238, 122045 (2024). https://doi.org/10.1016/j.eswa.2023.122045
19. Wang, Z., Li, Z., He, D., Chan, S.: A lightweight approach for network intrusion detection in industrial cyber-physical systems based on knowledge distillation and deep metric learning. Expert Systems with Applications 206, 117671 (2022). Article number: 117671
20. Wang, Z., Zhou, R., Yang, S., et al.: A novel lightweight IoT intrusion detection model based on self-knowledge distillation. IEEE Internet of Things Journal 12(11), 16912–16930 (2025). https://doi.org/10.1109/JIOT.2025.3533092