International Journal of Innovative Information Systems & Technology Research 19(4):84-95, Oct-Dec, 2026

© SEAHI PUBLICATIONS, 2026 www.seahipublications.org ISSN: 2467-8562

DOI:10.5281/zenodo.23337754

Compressed Hybrid CNN-LSTM Intrusion Detection for Small IoT Device Networks

[1] Eldad Otokiniye Alalibo *[2] Prof. N. D. Nwiabu *[3] Dr. C. G. Igiri

[1,2&3]Department of Computer Science, Rivers State University, Port-Harcourt, Nigeria
eldad.alalibo@ust.edu.ng[1], nwiabu.nuka@ust.edu.ng[2] and giri.chima@ust.edu.ng[3]

ABSTRACT
Internet of Things (IoT) networks need intrusion detection models that can identify multiple attack categories while remaining small and fast enough for edge-oriented deployment. Hybrid deep learning models can improve detection capability, but their storage and inference costs may limit their usefulness on small IoT devices. This article presents a compressed hybrid Convolutional Neural Network-Long Short-Term Memory (CNN-LSTM) method for multi-class IoT intrusion detection. The method uses a full-precision teacher CNN-LSTM model, a compact student model trained through knowledge distillation, and TensorFlow Lite dynamic-range, float16, and INT8 variants. It is evaluated on CIC-IoT-2023 using the grouped attack_class label, covering benign traffic and seven attack categories. The evaluation used 39 numerical flow features, a class-balanced training sample capped at 100,000 records per class, and validation and test samples that preserved the prepared split distribution. Detection quality was measured using accuracy, macro precision, macro recall, macro F1-score, false positive rate, per-class metrics, and confusion matrices, while deployment efficiency was measured using model size and inference latency. XGBoost achieved the strongest overall macro F1-score of 0.7217, with accuracy of 0.8124, model size of 10.466 MB, and latency of 0.035 ms per sample. Among the compressed CNN-LSTM models, Student TFLite Dynamic achieved the strongest macro F1-score of 0.6655, with accuracy of 0.8003, model size of 0.133 MB, and latency of 0.358 ms per sample. The results show that compression substantially reduced model size and preserved low latency, but minority-class detection and false positive behaviour remained limiting factors.
Keywords: IoT intrusion detection; CNN-LSTM; knowledge distillation; TensorFlow Lite; model compression; quantisation; CIC-IoT-2023; multi-class classification.

I. INTRODUCTION

Internet of Things networks have expanded the number of connected devices used in smart homes, wearable systems, healthcare monitoring, industrial sensing, and other edge environments. These devices generate heterogeneous network traffic and are frequently deployed with limited processing power, memory, storage, and energy availability. Intrusion detection is therefore an important defensive function because IoT systems must identify malicious or policy-violating traffic before it causes service disruption, unauthorised access, botnet activity, or data compromise.

Machine learning has become a common approach for intrusion detection because it can learn traffic patterns from labelled network-flow data. Traditional machine learning models such as Decision Tree, Random Forest, and XGBoost remain relevant for tabular traffic features. Deep learning models can learn more complex non-linear representations, and hybrid architectures such as CNN-LSTM models combine convolutional feature extraction with recurrent sequence modelling. In intrusion detection, this combination is useful because traffic features may contain local feature-level patterns as well as ordered behavioural relationships [1], [2].

However, strong offline detection performance is not sufficient for small IoT deployment. A model that performs well in a controlled test may still be unsuitable if it requires excessive storage, memory, or inference time. Deployment-oriented comparisons must therefore include model size and latency alongside detection metrics. This is especially important for multi-class intrusion detection, where accuracy can be misleading under class imbalance. A model may classify dominant DDoS or DoS traffic correctly while performing poorly on rare but operationally important classes such as Brute Force or Web-based attacks.

Model compression provides a practical route toward small-device deployment. Knowledge distillation trains a compact student model to approximate a larger teacher model, while quantisation reduces numerical precision to decrease storage and computation cost [3], [4]. TensorFlow Lite supports compressed deployment formats such as dynamic-range, float16, and INT8 conversion. These methods can reduce model footprint, but they may also reduce detection performance. Their effect should therefore be evaluated empirically rather than assumed.

This article presents a compressed hybrid CNN-LSTM method for multi-class IoT traffic classification. The method is evaluated on CIC-IoT-2023, a contemporary IoT intrusion detection dataset containing large-scale attacks against IoT devices [5]. The article compares a full-precision teacher CNN-LSTM model, a distilled student CNN-LSTM model, TensorFlow Lite compressed variants, and traditional machine learning baselines under the same data split, sampling procedure, and metrics. The comparison shows how distillation and TensorFlow Lite conversion affect the size, latency, and class-balanced detection of CNN-LSTM IDS models.

II. RELATED WORK

IoT intrusion detection research has developed across deep learning, hybrid learning, explainable detection, federated learning, and model compression. Deep learning approaches are widely used because IoT attacks can produce complex traffic signatures that may not be captured by manually designed rules. Banaamah and Ahmad [6] evaluated deep learning approaches for IoT intrusion detection and showed the relevance of convolutional and recurrent models. Elsayed et al. [1] applied deep learning to secure IoT and software-defined networking systems, while Selvam et al. [2] used a hybrid convolutional-recurrent model for multi-class IoT intrusion detection in a federated learning setting.

Other studies have examined alternative deep learning designs. Lazzarini et al. [7] proposed a stacking ensemble of deep learning models for IoT intrusion detection. Keshk et al. [8] combined deep learning with explainability for IoT intrusion detection, and Sharma et al. [9] similarly emphasised explainable deep learning for IoT network security. Hassini et al. [10] proposed an end-to-end CNN1D approach for Industrial IoT intrusion detection. These studies support the value of deep learning, but many focus mainly on detection performance rather than deployment efficiency.

Traditional machine learning remains important because many network intrusion detection datasets are represented as tabular flow features. Balega et al. [11] evaluated representative machine learning models for IoT anomaly detection, including XGBoost and support vector machines. Kantharaju et al. [12] also investigated machine learning for IoT security attack detection. These works show that compressed neural models should not be evaluated in isolation; tree-based models may provide competitive detection and faster inference for tabular traffic data.

Model compression is central to edge deployment. Hinton et al. [3] introduced knowledge distillation as a method for transferring predictive behaviour from a larger model to a smaller one. Jacob et al. [4] demonstrated integer quantisation for efficient neural network inference, while Gholami et al. [13] reviewed quantisation methods and showed that their effectiveness varies across architectures and deployment conditions. Umar et al. [14] applied knowledge distillation and quantisation to energy-efficient intrusion detection for edge computing. However, their architecture and evaluation setting differ from a hybrid CNN-LSTM comparison across dynamic-range, float16, and INT8 TensorFlow Lite variants.

CIC-IoT-2023 provides a recent benchmark for IoT intrusion detection. Neto et al. [5] introduced the dataset as a large-scale IoT attack benchmark with real IoT devices and multiple attack categories. Its scale and class imbalance make it useful for realistic evaluation, but also make accuracy alone

insufficient. A deployment-oriented IDS evaluation should include macro F1-score, false positive rate, per-class performance, model size, and latency.

Recent IoT intrusion detection studies also show the breadth of model choices available for this problem. Aldaej et al. [15] used an ensemble IDS for IoT-edge settings, Shebl et al. [16] evaluated deep convolutional models for binary and multi-class intrusion detection, and Alrayes et al. [17] used a denoising autoencoder for IoT intrusion detection. These studies reinforce the need to compare compressed CNN-LSTM models against practical alternatives rather than treating one neural architecture as sufficient. Broader compression literature also supports this position: Li et al. [18] show that compression methods differ in how they affect memory, computation, and predictive behaviour, making empirical comparison necessary.

The gap addressed in this article is the limited comparative evidence on compressed hybrid CNN-LSTM intrusion detection for small IoT deployment. Existing work demonstrates the relevance of deep learning, hybrid models, and compression, but fewer comparisons examine knowledge distillation and multiple TensorFlow Lite compression variants on the same multi-class IoT task with traditional machine learning baselines.

III. METHOD

The experimental design combines a full-precision teacher model, a smaller distilled student model, TensorFlow Lite compressed variants, validation-based score calibration, and traditional machine learning baselines. The architecture in Figure 1 focuses on the CNN-LSTM distillation and compact student deployment path.

Figure 1 shows the CNN-LSTM distillation and deployment architecture.

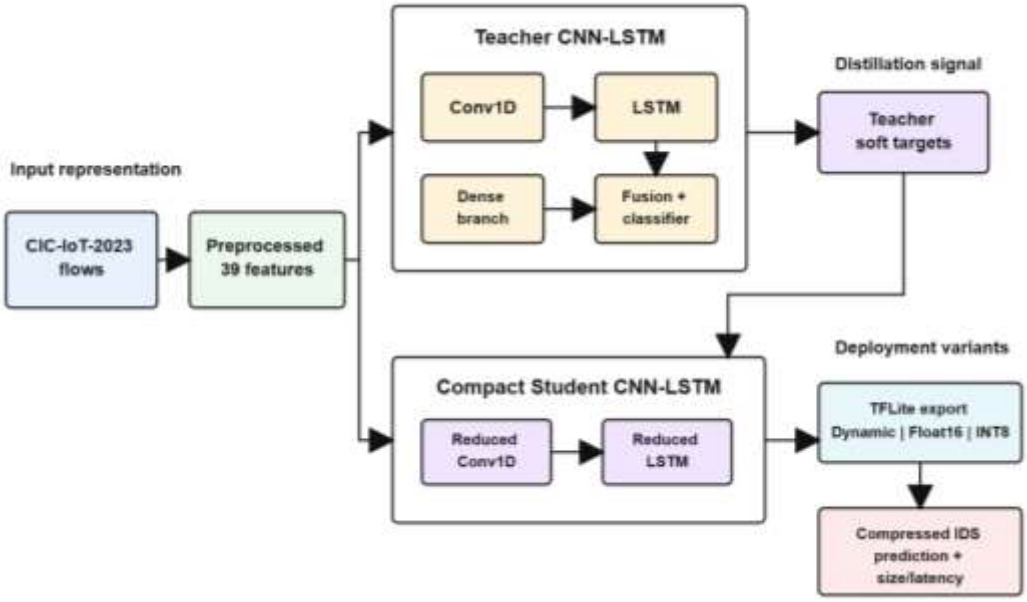

Figure 1: Compressed hybrid CNN-LSTM intrusion detection architecture

The method begins with CIC-IoT-2023 flow records represented as 39 preprocessed numerical features. These features are used to train a teacher CNN-LSTM model with convolutional-recurrent and dense branches. A compact student CNN-LSTM receives the same preprocessed feature representation and is trained using the teacher's soft targets. The compact student is then exported to TensorFlow Lite dynamic-range, float16, and INT8 variants for compressed IDS prediction and deployment measurement.

Dataset and Preprocessing

The evaluation used the CIC-IoT-2023 dataset released by the Canadian Institute for Cybersecurity [5]. The grouped **attack_class** label was used for multi-class classification. The classes were Benign, Brute Force, DDoS, DoS, Mirai, Reconnaissance, Spoofing, and Web-based.

The prepared dataset contained separate train, validation, and test splits. The full train split contained 30,806,432 records, while the validation and test splits contained 3,850,805 and 3,850,804 records respectively. The evaluation used 39 numerical flow features after excluding the label columns **Label**, **attack_class**, and **label**.

Training used balanced class sampling with a maximum of 100,000 records per class. Because Brute Force and Web-based had fewer available training records than the cap, the final training sample contained 630,149 records. Validation and test sampling preserved the prepared split distribution, with 199,998 validation records and 200,000 test records. This design reduced majority-class dominance during training while keeping evaluation close to the deployment-facing imbalance.

Table 1: Sampled Dataset Split Sizes

Split	Number of records
Training	630,149
Validation	199,998
Test	200,000

Table 2: Class Distribution in the Sampled Dataset

Class	Training	Validation	Test
Benign	100,000	5,703	5,703
Brute Force	10,394	70	69
DDoS	100,000	133,595	133,613
DoS	100,000	40,715	40,745
Mirai	100,000	13,684	13,639
Reconnaissance	100,000	3,586	3,584
Spoofing	100,000	2,515	2,514
Web-based	19,755	130	133

Figure 2 shows the class distribution across the sampled splits.

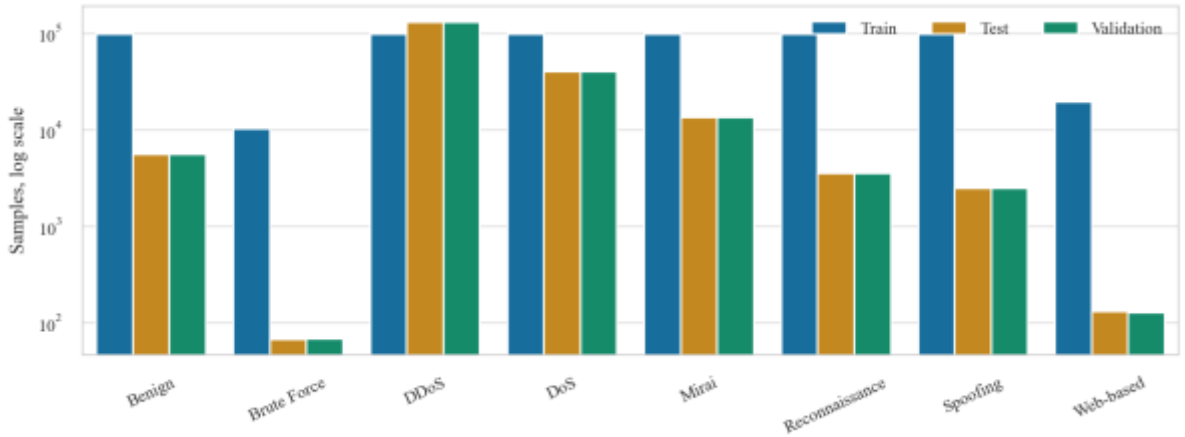

Figure 2: Class distribution of the sampled CIC-IoT-2023 splits

Model Architecture and Compression

The teacher CNN-LSTM model used a convolutional-recurrent branch and a parallel tabular branch. The convolutional-recurrent branch applied Conv1D layers, batch normalisation, max pooling, and an LSTM layer to the sequence-shaped traffic representation. The tabular branch flattened the input and passed it through a dense layer to retain direct feature-level information. The two branches were concatenated before dense classification into the eight traffic classes.

The student model followed the same general structure but used fewer filters, fewer LSTM units, and a smaller dense representation. Knowledge distillation trained the student using both hard class labels and teacher-guided soft targets. The distillation temperature was 3.0 and the distillation alpha was 0.3. The student was therefore optimised to learn from the teacher while retaining a compact architecture.

Three TensorFlow Lite compression variants were exported for the neural models:

1. Dynamic-range TensorFlow Lite conversion, which applies post-training optimisation while keeping conversion risk low.
2. Float16 TensorFlow Lite conversion, which stores compatible weights in float16 precision.
3. INT8 TensorFlow Lite conversion, which uses representative calibration samples and integer input/output tensors.

INT8 conversion used up to 5,000 training samples for representative calibration.

Baseline Models

Three traditional machine learning baselines were trained on the same scaled tabular features. Random Forest used 120 trees, maximum depth 24, minimum leaf size 2, square-root feature sampling, and balanced subsampling. Decision Tree used balanced class weights and minimum leaf size 2. XGBoost used 300 estimators, maximum depth 7, learning rate 0.08, subsample 0.9, column sample 0.9, and the **multi:softprob** objective.

These baselines were included because IoT traffic-flow data is tabular. A compressed neural network is only useful in deployment if it compares favourably against simpler and faster models under the same conditions.

Validation-Based Score Calibration

For models that produced class probability scores, validation predictions were used to select class-specific score multipliers before final test evaluation. The selected multipliers were then applied unchanged to test predictions. This separated calibration from final testing and prevented the test set from influencing the selection process.

The validation objective balanced macro F1-score and false positive rate:

$$S = \text{MacroF1}_{val} - \lambda \text{FPR}_{val} \tag{1}$$

where S is the validation selection score and λ is the false positive rate penalty. The value of λ was set to 0.10.

Evaluation Metrics

Detection performance was evaluated using accuracy, macro precision, macro recall, macro F1-score, false positive rate, per-class precision, per-class recall, per-class F1-score, and confusion matrices. Macro F1-score was emphasised because the validation and test sets were highly imbalanced. Deployment efficiency was evaluated using model size in megabytes and inference latency in milliseconds per sample.

Model size and latency describe deployment cost, while false positive rate and macro F1-score indicate whether compression preserves usable detection behaviour under class imbalance.

IV. RESULTS AND DISCUSSION

Overall Performance

Table 3 presents the performance of all evaluated models on the same 200,000-record test sample. XGBoost achieved the strongest overall result, with accuracy of 0.8124 and macro F1-score of 0.7217. Random Forest achieved the second-highest macro F1-score of 0.7142 but had a very large model size of 704.666 MB. Among the compressed CNN-LSTM models, Student TFLite Dynamic achieved the strongest macro F1-score of 0.6655, with a model size of 0.133 MB.

Figure 3 compares macro F1-score across the evaluated models.

Table 3: Overall Model Performance on the Test Set

Model	Accuracy	Macro precision	Macro recall	Macro F1	FPR	Size (MB)	Latency (ms/sample)
Teacher Baseline	0.7745	0.6755	0.6644	0.6622	0.1026	4.032	0.416
Teacher TFLite Dynamic	0.7754	0.6715	0.6569	0.6566	0.1010	0.377	1.088
Teacher TFLite Float16	0.7745	0.6755	0.6645	0.6622	0.1026	0.699	0.479
Teacher INT8	0.6586	0.5663	0.5507	0.5421	0.2462	0.391	1.573
Student FP32	0.7997	0.6653	0.6706	0.6638	0.1238	0.416	0.115
Student TFLite Dynamic	0.8003	0.6674	0.6719	0.6655	0.1252	0.133	0.358
Student TFLite Float16	0.7997	0.6654	0.6707	0.6638	0.1236	0.220	0.160
Student INT8	0.6981	0.5872	0.5608	0.5614	0.2167	0.147	0.572
Random Forest	0.8098	0.7570	0.6931	0.7142	0.1117	704.666	0.015
Decision Tree	0.7639	0.6171	0.6834	0.6339	0.2439	22.204	0.001
XGBoost	0.8124	0.7512	0.7023	0.7217	0.1038	10.466	0.035

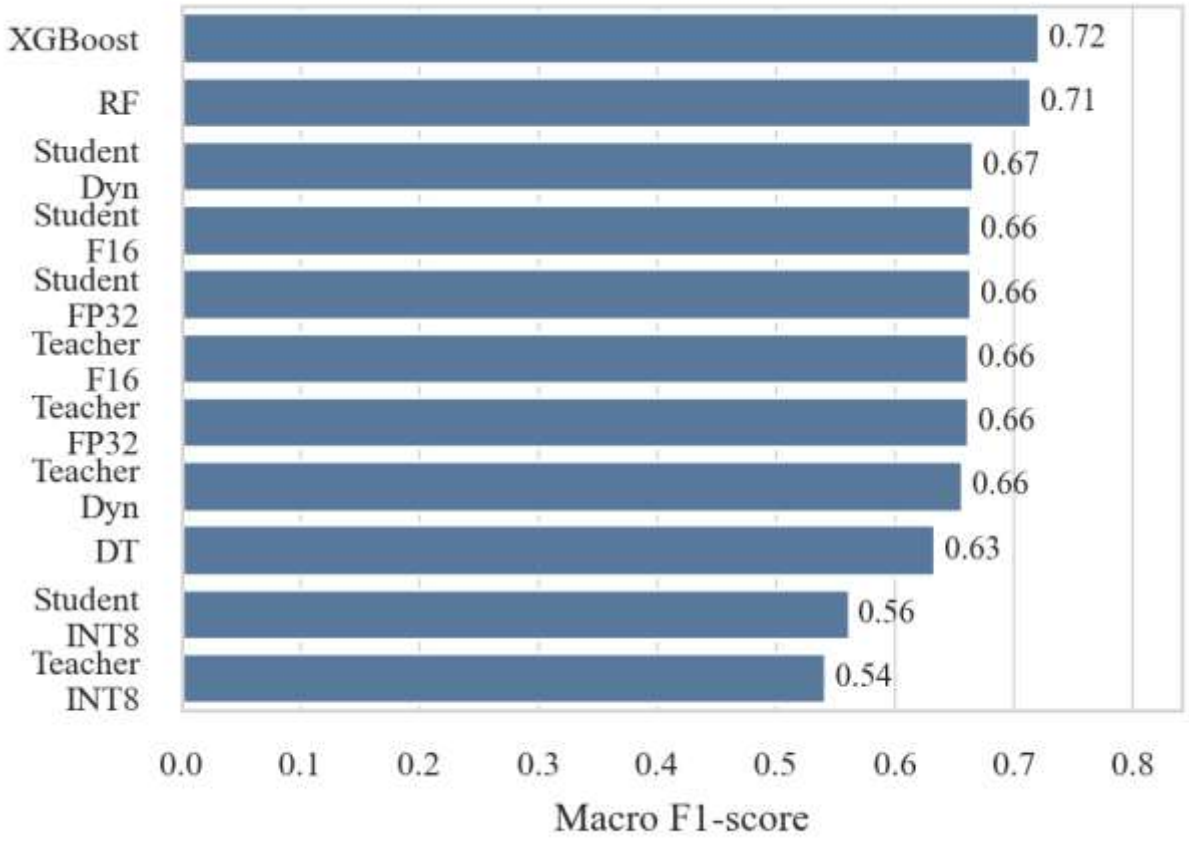

Figure 3: Macro F1-score comparison across evaluated models

The results show that accuracy alone gives an incomplete view of performance. The test set is dominated by DDoS and DoS records, so a model can achieve moderate or high accuracy while still underperforming on rare attack categories. Macro F1-score is therefore more informative for this evaluation because it gives equal weight to all classes.

Distillation Results

Knowledge distillation produced a smaller student model that preserved a similar detection profile to the teacher. The teacher baseline achieved accuracy of 0.7745 and macro F1-score of 0.6622. The Student FP32 model achieved accuracy of 0.7997 and macro F1-score of 0.6638. It also reduced model size from 4.032 MB to 0.416 MB and reduced latency from 0.416 ms to 0.115 ms per sample.

This indicates that the compact student was not simply a weaker version of the teacher. The student architecture and teacher-guided training improved deployment efficiency while preserving class-balanced detection close to the full-precision teacher. However, the small macro F1-score gain shows that distillation improved efficiency more clearly than class-balanced detection.

Compression Results

TensorFlow Lite conversion produced substantial storage reductions across the compressed CNN-LSTM variants. Table 4 presents the compression and latency results relative to the teacher baseline.

Table 4: Compression and Latency Results Relative to the Teacher Baseline

Model	Size (MB)	Compression ratio	Size reduction (%)	Latency (ms/sample)
Teacher Baseline	4.032	1.00	0.00	0.416
Teacher TFLite Dynamic	0.377	10.71	90.66	1.088
Teacher TFLite Float16	0.699	5.77	82.67	0.479
Teacher INT8	0.391	10.30	90.29	1.573
Student FP32	0.416	9.69	89.68	0.115
Student TFLite Dynamic	0.133	30.41	96.71	0.358
Student TFLite Float16	0.220	18.29	94.53	0.160
Student INT8	0.147	27.44	96.36	0.572

Student TFLite Dynamic achieved the smallest compressed CNN-LSTM model size at 0.133 MB and the highest compressed CNN-LSTM macro F1-score at 0.6655. Student TFLite Float16 achieved a similar macro F1-score of 0.6638, lower latency of 0.160 ms per sample, and a slightly lower false positive rate than Student TFLite Dynamic. The practical choice between these two variants therefore depends on whether minimum storage or lower latency is more important.

Student INT8 achieved a small model size of 0.147 MB but reduced macro F1-score to 0.5614 and increased false positive rate to 0.2167. The teacher INT8 variant showed a similar pattern, with macro F1-score falling to 0.5421. These results indicate that full integer quantisation introduced a stronger detection penalty than dynamic-range or float16 conversion.

Figure 4 shows the model size and latency trade-off, while Figure 5 shows compression ratio relative to the teacher baseline.

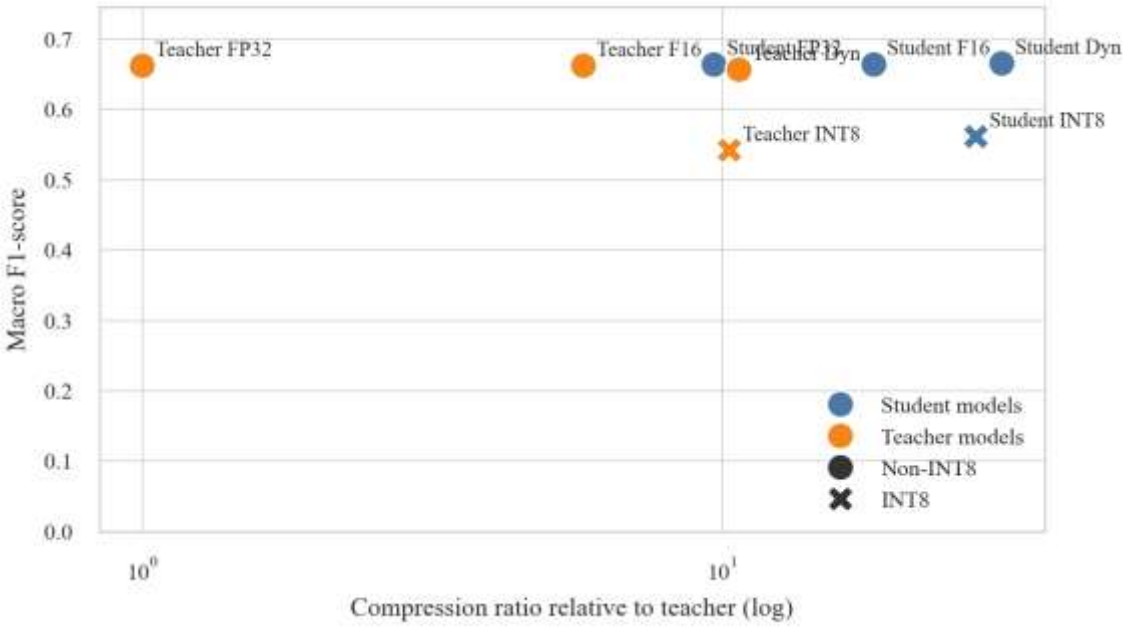

Figure 4: Model size and latency trade-off

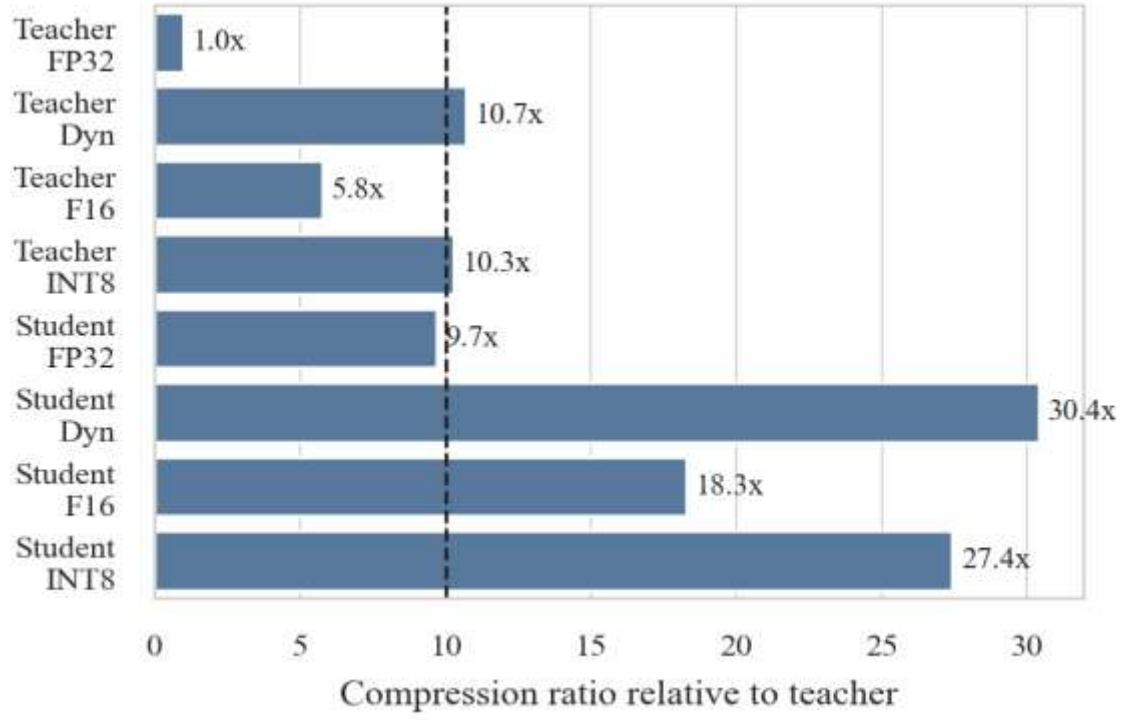

Figure 5: Compression ratio relative to the teacher baseline

Baseline Comparison

The traditional machine learning results show that compressed deep learning is not automatically superior for tabular IoT intrusion detection. XGBoost achieved the best macro F1-score and a practical model size of 10.466 MB. It also had low latency at 0.035 ms per sample. Random Forest achieved a close macro F1-score but required 704.666 MB, making it much less compact than the neural and XGBoost alternatives. Decision Tree had the fastest inference latency but weaker macro F1-score and a high false positive rate.

The strongest compressed CNN-LSTM model was much smaller than XGBoost. Student TFLite Dynamic was 0.133 MB, compared with 10.466 MB for XGBoost. However, XGBoost achieved stronger class-balanced detection. This creates a practical trade-off: XGBoost achieved the strongest detection result among compact, low-latency models, while Student TFLite Dynamic was the best compressed neural option for storage-constrained IoT settings.

Class-Level Errors

Student TFLite Dynamic was selected for detailed per-class analysis because it achieved the highest macro F1-score among the compressed CNN-LSTM variants. Table 5 shows its per-class results.

Table 5: Per-Class Performance of Student TFLite Dynamic

Class	Precision	Recall	F1-score	Support
Benign	0.7628	0.8748	0.8150	5,703
Brute Force	0.2870	0.4493	0.3503	69
DDoS	0.8688	0.8497	0.8591	133,613
DoS	0.5411	0.5792	0.5595	40,745
Mirai	0.9947	0.9977	0.9962	13,639
Reconnaissance	0.7761	0.6897	0.7304	3,584
Spoofing	0.8625	0.7164	0.7827	2,514
Web-based	0.2458	0.2180	0.2311	133
Macro average	0.6674	0.6719	0.6655	200,000
Weighted average	0.8053	0.8003	0.8023	200,000

The model detected Mirai most effectively, with an F1-score of 0.9962. DDoS, Benign, Spoofing, and Reconnaissance also produced stronger results than the rarest categories. Brute Force and Web-based attacks were the weakest classes, with F1-scores of 0.3503 and 0.2311 respectively. This result reflects the effect of class imbalance and limited minority-class support in the evaluation set.

Figure 6 presents the per-class recall comparison across models, and Figure 7 presents the per-class F1-score of the selected compressed CNN-LSTM model.

Table 6 presents the confusion matrix for Student TFLite Dynamic.

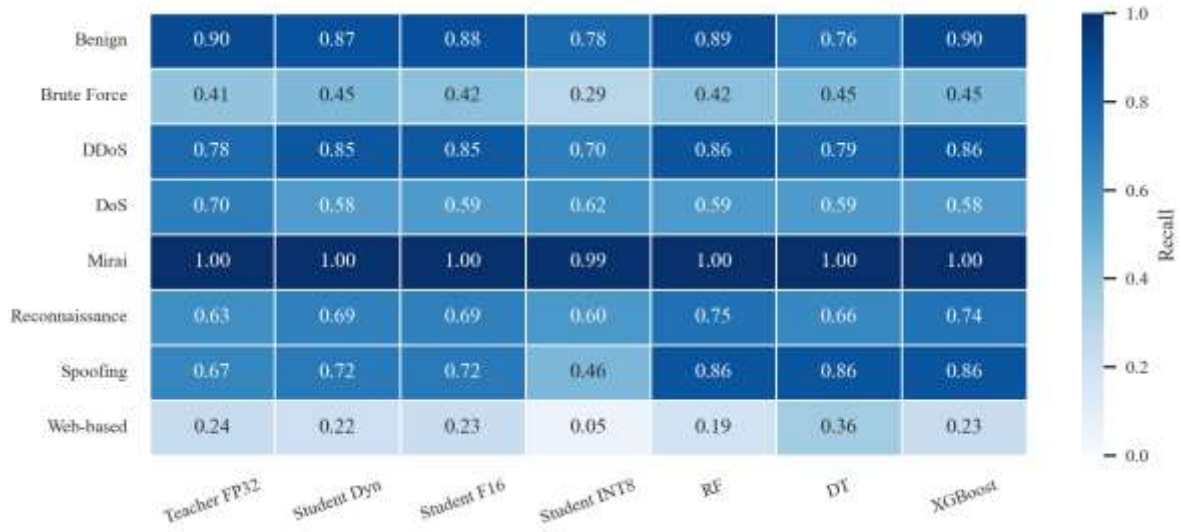

Figure 6: Per-class recall comparison across models

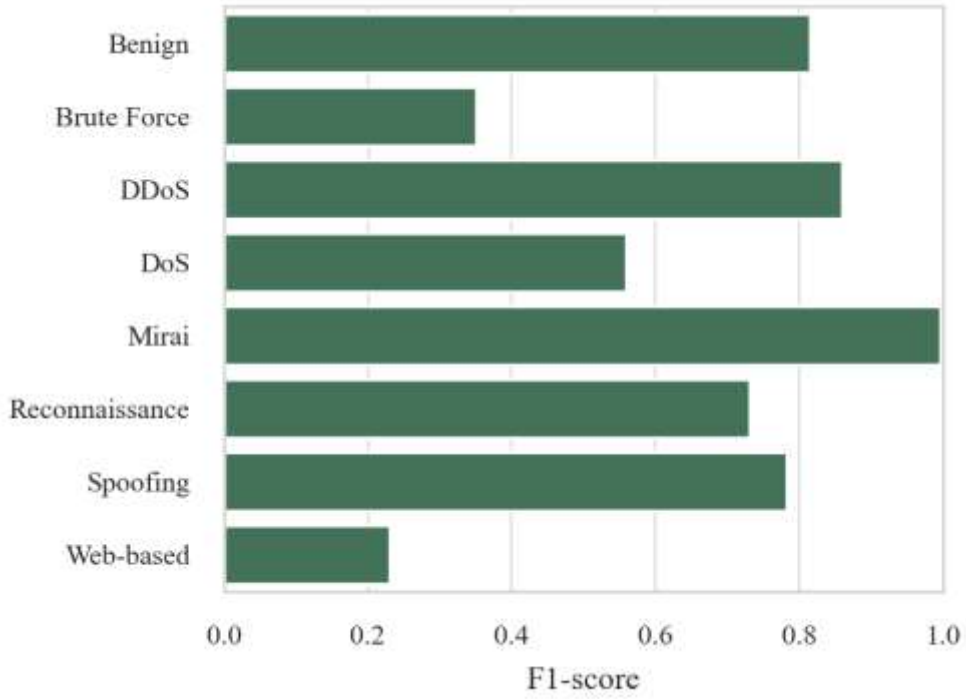

Figure 7: Per-class F1-score of Student TFLite Dynamic

Table 6: Confusion Matrix for Student TFLite Dynamic

Actual class	Benign	Brute Force	DDoS	DoS	Mirai	Reconnaissance	Spoofing	Web-based
Benign	4,989	19	0	0	0	545	126	24
Brute Force	22	31	0	0	0	15	0	1
DDoS	2	0	113,526	20,005	59	6	15	0
DoS	0	0	17,121	23,600	13	3	8	0
Mirai	0	0	19	10	13,608	0	2	0
Reconnaissance	906	45	0	0	0	2,472	122	39
Spoofing	576	10	0	0	0	102	1,801	25
Web-based	45	3	0	0	0	42	14	29

The confusion matrix shows two main error patterns. First, DDoS and DoS were frequently confused, with 20,005 DDoS records classified as DoS and 17,121 DoS records classified as DDoS. This is plausible because both attack categories involve service disruption traffic and may share similar flow-level characteristics. Second, Web-based and Brute Force attacks were often confused with Benign, Reconnaissance, and Spoofing. These errors explain why macro F1-score remained moderate despite the model's accuracy of 0.8003.

Deployment Discussion

The results indicate that compression reduced footprint and latency more clearly than it improved detection quality. Student TFLite Dynamic, Student TFLite Float16, and Student INT8 were all below 1 MB, while Random Forest required 704.666 MB. However, macro F1-score and false positive rate remained limiting factors across all evaluated models.

Figure 8 compares false positive rates across the evaluated models.

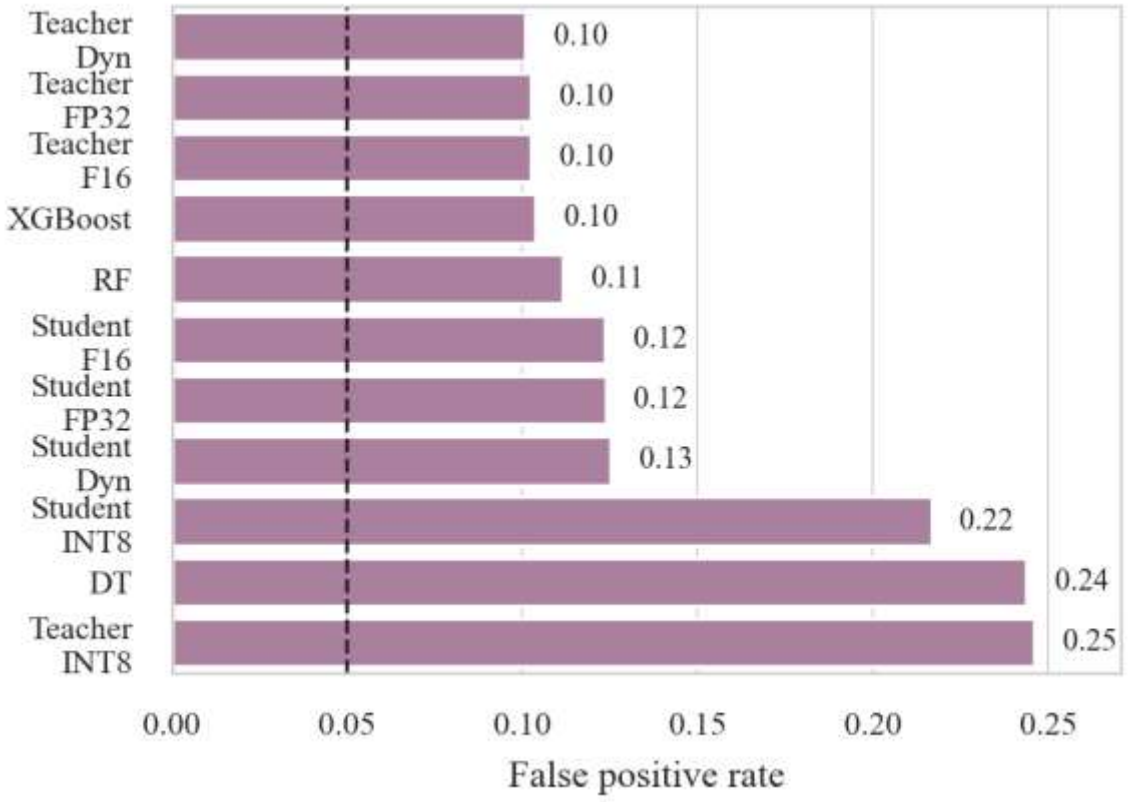

Figure 8: False positive rate comparison across evaluated models

The result is therefore mixed. Compression made the neural models small and fast enough for edge-oriented IoT constraints, but efficiency alone did not make them ready for field use. The main limiting factor was detection quality, particularly the moderate macro F1-score and elevated false positive rate. For IDS use, false positives are important because benign traffic incorrectly flagged as malicious can produce alert fatigue and reduce trust in the detection system.

V. Conclusion

This article presented a compressed hybrid CNN-LSTM method for multi-class IoT intrusion detection using CIC-IoT-2023. The method combined a full-precision teacher CNN-LSTM model, a distilled student CNN-LSTM model, TensorFlow Lite dynamic-range, float16 and INT8 variants, validation-based score calibration, and traditional machine learning baselines. Knowledge distillation reduced the CNN-LSTM architecture, while TensorFlow Lite conversion reduced the storage footprint of teacher and student models for edge-oriented evaluation.

Student TFLite Dynamic reduced model size to 0.133 MB, representing a 30.41x compression ratio relative to the teacher baseline, while achieving 0.358 ms latency per sample. Student TFLite Float16 also provided a strong compressed result, with 0.220 MB model size and 0.160 ms latency per sample. Dynamic-range and float16 conversion preserved detection performance more effectively than INT8 conversion. The strongest overall model was XGBoost, with macro F1-score of 0.7217, accuracy of 0.8124, model size of 10.466 MB, and latency of 0.035 ms per sample. These results show that compressed hybrid CNN-LSTM models can be made small and fast enough for edge-oriented IoT constraints, but the evaluated models still need stronger class-balanced detection for field use. Future work should focus on improving minority-class detection through stronger class-balancing methods, cost-sensitive learning, focal loss, feature selection, quantisation-aware training, lightweight alternative architectures, and testing on actual edge hardware.

REFERENCES

[1] Elsayed, R. A., Hamada, R. A., Abdalla, M. I., & Elsaid, S. A. (2023). Securing IoT and SDN systems using deep-learning-based automatic intrusion detection. *Ain Shams Engineering Journal, 14*(10), Article 102211. https://doi.org/10.1016/j.asej.2023.102211

[2] Selvam, P., Karthikeyan, P., Manochitra, S., Sujith, A. V. L. N., Ganesan, T., Ayyasamy, R., Shuaib, M., Alam, S., & Rajendran, A. (2025). Federated learning-based hybrid convolutional recurrent neural network for multi-class intrusion detection in IoT networks. *Discover Internet of Things, 5*, Article 39. https://doi.org/10.1007/s43926-025-00130-8

[3] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv*. https://arxiv.org/abs/1503.02531

[4] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., & Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 2704-2713). https://doi.org/10.1109/CVPR.2018.00286

[5] Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., & Ghorbani, A. A. (2023). CICIoT2023: A real-time dataset and benchmark for large-scale attacks in IoT environment. *Sensors, 23*(13), Article 5941. https://doi.org/10.3390/s23135941

[6] Banaamah, A. M., & Ahmad, I. (2022). Intrusion detection in IoT using deep learning. *Sensors, 22*(21), Article 8417. https://doi.org/10.3390/s22218417

[7] Lazzarini, R., Tianfield, H., & Charissis, V. (2023). A stacking ensemble of deep learning models for IoT intrusion detection. *Knowledge-Based Systems, 279*, Article 110941. https://doi.org/10.1016/j.knosys.2023.110941

[8] Keshk, M., Koroniotis, N., Pham, N., Moustafa, N., Turnbull, B., & Zomaya, A. Y. (2023). An explainable deep learning-enabled intrusion detection framework in IoT networks. *Information Sciences, 639*, Article 119000. https://doi.org/10.1016/j.ins.2023.119000

[9] Sharma, B., Sharma, L., Lal, C., & Roy, S. (2024). Explainable artificial intelligence for intrusion detection in IoT networks: A deep learning-based approach. *Expert Systems with Applications, 238*, Article 121751. https://doi.org/10.1016/j.eswa.2023.121751

[10] Hassini, K., Khalis, S., Habibi, O., Chemmakha, M., & Lazaar, M. (2024). An end-to-end learning approach for enhancing intrusion detection in Industrial-Internet of Things. *Knowledge-Based Systems, 294*, Article 111785. https://doi.org/10.1016/j.knosys.2024.111785

[11] Balega, M., Farag, W., Wu, X.-W., Ezekiel, S., & Good, Z. (2024). Enhancing IoT security: Optimizing anomaly detection through machine learning. *Electronics, 13*(11), Article 2148. https://doi.org/10.3390/electronics13112148

[12] Kantharaju, V., Suresh, H., Niranjanamurthy, M., Ansarullah, S. I., Amin, F., & Alabrah, A. (2024). Machine learning based intrusion detection framework for detecting security attacks in internet of things. *Scientific Reports, 14*, Article 30275. https://doi.org/10.1038/s41598-024-81535-3

[13] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., & Keutzer, K. (2022). A survey of quantization methods for efficient neural network inference. In *Low-power computer vision* (pp. 291-326). Chapman and Hall/CRC. https://doi.org/10.1201/9781003162810-13

[14] Umar, H. G. A., Yasmeen, I., Aoun, M., Mazhar, T., Khan, M. A., Jaghdam, I. H., & Hamam, H. (2025). Energy-efficient deep learning-based intrusion detection system for edge computing: A novel DNN-KDQ model. *Journal of Cloud Computing, 14*, Article 32. https://doi.org/10.1186/s13677-025-00762-9

[15] Aldaej, A., Ullah, I., Ahanger, T. A., & Atiquzzaman, M. (2024). Ensemble technique of intrusion detection for IoT-edge platform. *Scientific Reports, 14*, Article 11703. https://doi.org/10.1038/s41598-024-62435-y

[16] Shebl, A., Elsedimy, E. I., Ismail, A., Salama, A. A., & Herajy, M. (2024). DCNN: A novel binary and multi-class network intrusion detection model via deep convolutional neural network. *EURASIP Journal on Information Security, 2024*, Article 35. https://doi.org/10.1186/s13635-024-00184-1

[17] Alrayes, F. S., Zakariah, M., Amin, S. U., Khan, Z. I., & Helal, M. (2024). Intrusion detection in IoT systems using denoising autoencoder. *IEEE Access, 12*, 122401-122425. https://doi.org/10.1109/ACCESS.2024.3451726

[18] Li, Z., Li, H., & Meng, L. (2023). Model compression for deep neural networks: A survey. *Computers, 12*(3), Article 60. https://doi.org/10.3390/computers12030060