HPC for SDN Intrusion Detection: Lessons from Literature and Open Challenges

Mariana Almeida Lima[1] **, Dernier Bruno Teixeira**[1]
Antonio Wendell de O. Rodrigues[1]

[1]Instituto Federal do Ceará (IFCE)

mariana.almeida07@aluno.ifce.edu.br,{dernier,wendell}@ifce.edu.br

Abstract. *Deep learning-based Intrusion Detection Systems (IDS) for Software-Defined Networks (SDN) have achieved high accuracy but often neglect computational efficiency. This paper analyzes nine recent works (2020–2025) on IDS for SDN, examining the gap between accuracy-focused research and practical HPC deployment. We identify three patterns: (1) deep learning dominance without scalability evaluation, (2) GPU acceleration treated as a secondary concern, and (3) lack of real-world performance benchmarks. We propose a taxonomy of HPC strategies and highlight the need for efficiency-accuracy trade-off studies.*

Resumo. *Sistemas de Detecção de Intrusão (IDS) baseados em aprendizado profundo para Redes Definidas por Software (SDN) atingem alta precisão mas frequentemente negligenciam eficiência computacional. Este trabalho analisa nove trabalhos recentes (2020–2025) sobre IDS para SDN, examinando a lacuna entre pesquisa focada em acurácia e implantação prática de HPC. Identificamos três padrões: (1) domínio de aprendizado profundo sem avaliação de escalabilidade, (2) aceleração GPU tratada como preocupação secundária, e (3) falta de benchmarks de desempenho em tempo real. Propomos uma taxonomia de estratégias HPC e destacamos a necessidade de estudos sobre trade-offs eficiência-acurácia.*

1. Introduction

Software-Defined Networking enables programmable network control but introduces new attack vectors requiring real-time intrusion detection. While recent works employ deep learning for SDN security, a critical gap exists: *papers report high accuracy (often >99%) but provide minimal performance metrics such as latency, throughput, or scalability.* This discrepancy matters for practitioners. A CNN-BiLSTM achieving 99.5% accuracy is impractical if it requires specialized GPUs unavailable in resource-constrained environments. Yet, most literature does not address this trade-off.

We analyzed nine recent works (2020–2025) on IDS for SDN, focusing on how HPC techniques are integrated. Our finding is stark: *computational efficiency is treated as an afterthought, not a primary design goal.*

2. Methodology and Analysis

We selected works via keyword searches (”SDN intrusion detection GPU”, ”parallel IDS SDN”) spanning 2020–2025. Selection criteria: (1) address IDS in SDN, (2) mention

computational techniques (GPU, parallelism, distributed), (3) published in peer-reviewed venues. Rather than list papers, we organize findings by **HPC strategy**:

2.1. Pattern 1: Deep Learning Without Scalability

Ben Said et al. (2023), Zhang et al. (2025), and Hnamte et al. (2024) employ CNN-BiLSTM, GNNs, and autoencoders respectively. All report >95% detection accuracy. *None evaluate latency scaling with packet volume, CPU/GPU memory requirements, or cost-per-detection.*

This is a methodological failure. A 99% accurate IDS that processes only 1,000 packets/sec is useless at 10 Gbps link rates.

2.2. Pattern 2: GPU as Enabler, Not Subject

Javeed et al. (2023) and Muthanna et al. (2022) use CUDA acceleration but report only overall accuracy. Missing: (a) training/inference time comparison (GPU vs. CPU), (b) speedup factors, (c) hardware cost.

Haugerud et al. (2021) and Shu et al. (2020) adopt distributed processing, reducing packet loss. Yet they lack detailed scalability analysis across controller replicas or network sizes.

2.3. Pattern 3: Real-World Deployment Gap

Eight of nine works use simulated datasets (NSL-KDD, UNSW-NB15, CICIDS2017). Only Muthanna et al. (2022) claim validation on operational SDN. *Simulation-to-deployment gaps are well-known: real traffic distributions, encryption patterns, and anomaly prevalence differ significantly.*

3. Critical Insights

(1) Accuracy-centric incentives: Publication venues reward accuracy metrics. HPC metrics (throughput, latency percentiles, energy) are less prestigious, creating publication bias where efficiency is deprioritized.

(2) Domain fragmentation: IDS and HPC communities operate separately. IDS papers rarely cite parallel computing literature; HPC papers rarely address security constraints. This lack of cross-pollination leaves gaps unaddressed.

(3) Hardware heterogeneity: SDN deployments span edge devices (ARM-based microcontrollers) to data centers (GPU clusters). One-size-fits-all solutions fail. Adaptive approaches that adjust model complexity to hardware are missing.

4. A Taxonomy of HPC for SDN IDS

No work combines multiple strategies adaptively.

4.1. GPU Acceleration (4 works)

GPU-based approaches dominate the sample. Ben Said et al. (2023) train CNN-BiLSTM models on NVIDIA GPUs, achieving 99.5% accuracy on NSL-KDD. Javeed et al. (2023) use CUDA for healthcare IoT IDS. However, none report:

Table 1. HPC Strategies: Trade-offs and Gaps

Strategy	Works	Strength	Gap
GPU accel.	4	Fast inference	Energy, cost
Distributed	2	Scalability	Sync overhead
Edge/NFV	2	Low latency	Resource limits
Dimensionality	1	CPU-friendly	Accuracy loss

- Training/inference time ratio (GPU vs. CPU baseline)
- Memory requirements (VRAM, system RAM)
- Cost amortization (GPU acquisition, electricity)
- Batch processing limits

This is a methodological gap. A 10x speedup on GPU is impressive, but meaningless without knowing base CPU performance.

4.2. Distributed Processing (2 works)

Shu et al. (2020) and Haugerud et al. (2021) use distributed IDS across multiple SDN controllers. Benefits: packet loss reduction, load balancing. Costs: synchronization overhead, controller coordination.

Critically, neither quantifies synchronization latency. In a 10 Gbps network, even 1ms coordination delay translates to 1.25 MB of unprocessed packets. No work addresses this.

4.3. Edge/NFV Deployment (2 works)

Sood et al. (2023) optimize IDS for NFV edge nodes. Approach: dimensionality reduction to cut CPU requirements. Trade-off: explicit but not quantified. How much accuracy is lost per 10% CPU reduction? Unclear.

This reflects a deeper issue: *no work systematically sweeps an accuracy-efficiency frontier.* Publishing a single point (99% accuracy, 50ms latency) is less useful than a curve showing options.

5. Recommendations

For all researchers in domain: (1) Publish latency/throughput as first-class metrics, not appendices. (2) Establish accuracy-efficiency Pareto curves, not single points. (3) Release real SDN datasets (NSL-KDD from 1998 is obsolete). (4) Compare hardware architectures: CPU, GPU, edge devices. Benchmark CNN-BiLSTM (Ben Said et al. 2023) on three platforms: GPU, x86 CPU, Jetson edge device. Measure: training time, inference latency, peak memory, power, cost per inference. Expected output: show that best-for-accuracy $\neq$ best-for-edge.

6. Efficiency Metrics Gap

All nine analyzed papers report detection accuracy; however, efficiency metrics are largely absent. Ben Said et al. (2023) achieves 99.5% accuracy but provides no training time, inference latency, memory requirements, or power consumption data. Analysis across the nine works shows: 100% report accuracy metrics, $<$30% report any efficiency metric (latency or memory), and 0% report power or cost analysis. This gap represents the core methodological difference between accuracy-centric and HPC-aware research.

7. Why the Gap Persists

Incentive misalignment: Venues reward accuracy (>99%) over efficiency. Publication bias deprioritizes HPC metrics.

Siloed research: IDS and HPC communities operate separately. Nobody bridges the gap.

Experimental convenience: Real SDN datasets are expensive; simulated datasets (NSL-KDD, CICIDS2017) are free. Naturally, convenience wins.

8. Conclusion

Current research on IDS for SDN achieves high accuracy but ignores practical HPC concerns. A path forward requires bridging accuracy and efficiency, moving from publications optimized for high metrics to solutions optimized for deployment. Future work should focus on adaptive IDS designs that trade detection accuracy for computational efficiency based on network state, and on benchmarking existing deep learning models across heterogeneous hardware platforms to establish practical deployment guidelines.

References

Ben Said, R., Sabir, Z., and Askerzade, I. (2023). Cnn-bilstm: A hybrid deep learning approach for network intrusion detection system in sdn. *IEEE Access*.

Haugerud, H., Tran, H. N., Aitsaadi, N., and Yazidi, A. (2021). A dynamic and scalable parallel network intrusion detection system using intelligent rule ordering and network function virtualization. *IEEE Access*.

Hnamte, V., Najar, A. A., Nguyen, H. N., Hussain, J., and Sugali, M. N. (2024). Ddos attack detection and mitigation using deep neural network in sdn environment. *IEEE Access*.

Javeed, D., Gao, T., Saeed, M. S., Kumar, P., Kumar, R., and Jolfael, A. (2023). A softwarized intrusion detection system for iot-enabled smart healthcare system. *IEEE Access*.

Muthanna, M. S. A., Alkahel, R., Muthanna, A., Rafiq, A., and Abdullah, W. A. M. (2022). Towards sdn-enabled intelligent intrusion detection system for internet of things. *IEEE Access*.

Shu, J., Zhou, L., Zhang, W., Du, X., and Guizani, M. (2020). Collaborative intrusion detection for vanets: A deep learning-based distributed sdn approach. *IEEE Access*.

Sood, K., Nosouhi, M. R., Nguyen, D. D. N., Jiang, F., Chowdhury, M., and Doss, R. (2023). Intrusion detection scheme with dimensionality reduction in next generation networks. *IEEE Access*.

Yang, L., Song, Y., Gao, S., Hu, A., and Xiao, B. (2022). Griffin: Real-time network intrusion detection system via ensemble of autoencoder in sdn. *IEEE Access*.

Zhang, Y., Jue, C., Liu, W., and Ma, Y. (2025). Gran: A sdn intrusion detection model based on graph attention network and residual learning. *IEEE Access*.