Uma Abordagem Baseada em Big Data para Detecção de Intrusões com Aprendizado de Máquina sobre Dados de Múltiplos Domínios

Vinicius M. S. de Oliveira[1], Henrique M. S. de Oliveira[1], Gabriel M. Santos[1], Jhonatan Geremias[1], Eduardo K. Viegas[1]

[1]Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
80.215-901 – Curitiba – PR

{vinicius.oliveira, henrique.oliveira, gabriel.santos,
jhonatan.geremias, eduardo.viegas}@ppgia.pucpr.br

Abstract. *Neste trabalho, propomos um NIDS distribuído baseado em ensemble para melhorar a precisão e escalabilidade em redes de grande escala. Utilizando Apache Spark e Kafka, desacoplamos a ingestão de eventos da inferência, garantindo processamento em alta velocidade. O uso de múltiplos classificadores aumenta a generalização e reduz a perda de precisão em diferentes conjuntos de dados. Avaliações com os conjuntos UNSW-NB15, CS-CIC-IDS e BoT-IoT mostram que o modelo supera abordagens tradicionais, com ganhos de até 0,46 no F-Measure e processamento de 1,07 milhão de eventos por segundo.*

1. Introdução

O desenvolvimento de um Network Intrusion Detection System (NIDS) baseado em Machine Learning (ML) com capacidade de generalização, voltado para redes de alta velocidade e ataques hipervolumétricos atuais, é frequentemente negligenciado na literatura [Ye et al. 2024]. As abordagens atuais geralmente buscam detectar uma gama mais ampla de ataques e tráfego normal ao aumentar a complexidade do classificador subjacente, frequentemente utilizando Deep Neural Networks (DNNs) [Shrestha et al. 2023]. Como resultado, embora esses sistemas possam ter potencial para melhorar a generalização, muitas vezes são impraticáveis em redes de alta velocidade devido aos elevados custos computacionais da fase de inferência. Em redes de alta velocidade, a inferência deve ser realizada em larga escala com o mínimo de custo computacional; em contraste, as abordagens atuais frequentemente exigem grande uso de memória e impõem exigências computacionais inviáveis [Hussen et al. 2023].

Alcançar um NIDS baseado em ML com capacidade de generalização para redes de alta velocidade requer sua implementação como um sistema distribuído, capaz de operar em larga escala [Abid et al. 2023]. Esse processo envolve diversos desafios que devem ser enfrentados para garantir a implantação eficaz de um NIDS baseado em ML em redes de alta velocidade. Primeiramente, o armazenamento e a disponibilização de modelos de ML para inferência exigem mecanismos eficientes de versionamento e acesso de baixa latência, a fim de permitir atualizações frequentes e re-treinamento dos modelos [Horchulhack et al. 2024]. Sem uma estratégia otimizada de armazenamento e recuperação, o sistema pode sofrer atrasos significativos na detecção de ameaças. Em

segundo lugar, projetar um mecanismo distribuído de ingestão de eventos para processamento quase em tempo real exige pipelines de dados com alta vazão, capazes de lidar com volumes massivos de tráfego de rede, minimizando a sobrecarga de processamento [Akili et al. 2024]. Por fim, permitir a escalabilidade da arquitetura geralmente requer que o sistema seja projetado como uma implementação baseada em microsserviços, permitindo a distribuição dinâmica das cargas de trabalho entre múltiplos nós [Rodrigues et al. 2025].

Infelizmente, a literatura existente sobre NIDSs baseados em ML frequentemente ignora o desafio da generalização, assumindo que modelos treinados em conjuntos de dados específicos terão um desempenho confiável em diferentes ambientes de rede [Cantone et al. 2024]. Quando a generalização é considerada, os estudos normalmente se concentram em aumentar a robustez dos modelos por meio de arquiteturas mais complexas, como as DNN, negligenciando, porém, a viabilidade de implantar esses modelos em larga escala [Espindola et al. 2021]. Por outro lado, as abordagens que tratam da escalabilidade geralmente focam em um único aspecto do sistema, como a otimização da eficiência da inferência, deixando de lado os desafios mais amplos de integração. Em especial, muitas implementações escaláveis deixam de considerar componentes essenciais como armazenamento de modelos, inferência em tempo real e ingestão distribuída de eventos, tratando esses elementos como tarefas isoladas em vez de partes interconectadas de um sistema completo.

Contribuição. Diante disso, este artigo propõe uma nova arquitetura escalável de Big Data para NIDS baseado em ML com capacidade de cruzar conjuntos de dados, implementada em duas etapas principais. Primeiramente, projetamos a tarefa de classificação como um ensemble (conjunto) de classificadores rasos, utilizando um mecanismo de votação majoritária. Nossa abordagem seleciona os classificadores mais eficazes com base no desempenho entre conjuntos de dados cruzado, garantindo uma melhor generalização ao mesmo tempo em que mantém baixos custos computacionais. Em segundo lugar, implementamos o sistema como uma arquitetura distribuída baseada em microsserviços, construída sobre uma plataforma de Big Data. A arquitetura proposta integra armazenamento e disponibilização de modelos, ingestão distribuída de eventos e inferência escalável para lidar com tráfego de rede em alta velocidade de maneira eficiente. Como resultado, nosso sistema aprimora a generalização da classificação e garante escalabilidade, tornando-o adequado para implantação no mundo real em ambientes com alto volume de tráfego

2. Trabalhos Relacionados

Nos últimos anos, diversos trabalhos propuseram NIDSs baseados em ML com alta acurácia [Filho et al. 2025,Simioni et al. 2025]. Em geral, os esquemas propostos buscam maiores taxas de detecção, mas acabam negligenciando aspectos como a generalização e os custos de processamento dos modelos [Abreu et al. 2017]. Por exemplo, Z. Ye *et al.* [Ye et al. 2024] propuseram um ensemble de classificadores de ML construído por meio de uma estratégia de seleção de características O modelo proposto aumenta a acurácia em um único conjunto de dados, mas ignora as capacidades de generalização e os custos de processamento resultantes. De forma semelhante, C. Hazman *et al.* [Hazman et al. 2022] propuseram uma estrutura baseada em AdaBoost, também construída com uma estratégia de seleção de características. Embora o esquema possa melhorar a acurácia e reduzir os custos computacionais da inferência, não aborda o impacto na generalização

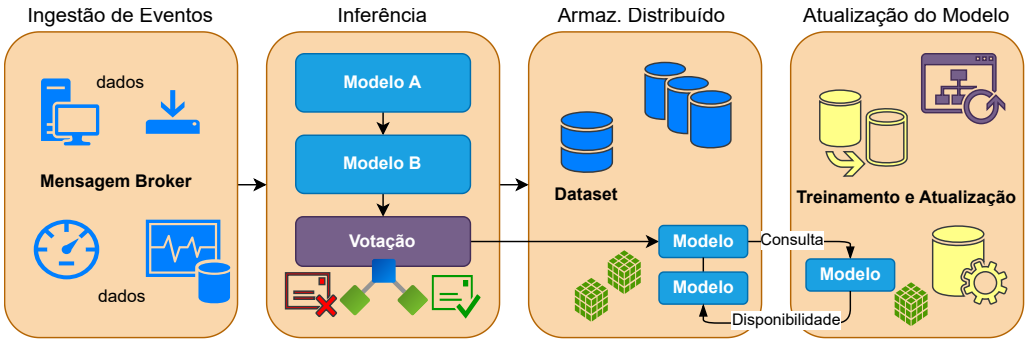

Figure 1. Visão geral da arquitetura proposta para viabilizar NIDS baseados em ML em redes de alta velocidade.

do modelo. A generalização de modelos em NIDS raramente é considerada na literatura, onde geralmente se assume que a acurácia obtida no conjunto de *teste* será refletida em ambientes reais. M. Cantone *et al.* [Cantone et al. 2024] avaliaram a acurácia de classificadores de ML amplamente utilizados para NIDS em um cenário de conjunto de dados cruzados. A avaliação demonstrou que os esquemas atuais sofrem degradação significativa de acurácia quando avaliados em conjuntos de dados diferentes daqueles usados na fase de treinamento. Para abordar esse problema, M. Wang *et al.* [Wang et al. 2024] combinaram conjuntos de dados utilizando uma estratégia de amostragem entre conjuntos (*cross-dataset sampling*) para treinar modelos baseados em DNN. Essa abordagem melhora a acurácia quando um cenário de conjunto de dado cruzados é considerado, no entanto, os custos de processamento associado ao uso de DNN são ignorados. Lidar com o tráfego de rede de alta velocidade para NIDS baseados em ML também não é algo facilmente alcançado na literatura A. Abid *et al.* [Abid et al. 2023] utilizaram Computação em Nuvem e Big Data para realizar fusão de dados na detecção de intrusões em redes de alta velocidade. Essa abordagem melhora a acurácia, mas ignora a generalização do modelo F. Jemili *et al.* [Jemili et al. 2023] propuseram um ensemble de classificadores de ML em um ambiente de Big Data. A abordagem melhora a acurácia e aborda redes de alta velocidade, mas negligencia como a generalização dos modelos pode ser tratada.

3. Uma Arquitetura de Big Data para NIDS Baseado em ML com Capacidade de Conjuntos de Dados Cruzados

A arquitetura proposta é implementada como uma estrutura de processamento de Big Data para enfrentar os desafios mencionados relacionados aos NIDS baseados em ML quase em tempo real em redes de alta velocidade. Na prática, ela busca resolver três desafios principais associados às redes de alta velocidade: ***Generalização de Modelos*** Projetamos um ensemble de classificadores simples, otimizados para desempenho de conjuntos de dados cruzado. ***Inferência em Escala e Quase em Tempo Real*** Nossa arquitetura permite o processamento de tráfego de rede em alta velocidade com latência mínima, mantendo alta taxa de transferência (*throughput*). ***Treinamento e Atualização Distribuída de Modelos*** Incorporamos um mecanismo escalável de armazenamento e fornecimento de modelos que permite o reprocessamento contínuo com tráfego de rede atualizado.

Figura 1 ilustra a implementação da nossa arquitetura proposta Ela inclui o *Pipeline de Ingestão de Eventos*, *Pipeline de Inferência*, *Armazenamento Distribuído*, e *Pipeline de Atualização de Modelo*. O *Pipeline de Ingestão de Eventos* utiliza um *message*

broker para coletar e processar eventos de rede de múltiplas fontes de dados de forma eficiente e em quase tempo real. O *Pipeline de Inferência* é implementado como um serviço distribuído em uma estrutura de Big Data, que processa os eventos recebidos aplicando um ensemble de classificadores rasos por meio de um procedimento de votação majoritária. Esse ensemble é construído com classificadores que demonstraram bom desempenho em avaliações em conjuntos de dados cruzados, garantindo melhor generalização entre diferentes ambientes de rede. O componente de *Armazenamento Distribuído* é responsável por armazenar tanto os conjuntos de dados de treinamento quanto diferentes versões dos modelos de ML. Por fim, o *Pipeline de Atualização de Modelo* recupera os dados de treinamento do armazenamento distribuído para retreinar periodicamente o modelo de ML.

3.1. Classificação e Treinamento de Modelo

Realizar uma classificação eficaz com NIDS baseados em ML que garanta generalização enquanto opera em redes de alta velocidade e quase em tempo real, apresenta desafios significativos. O tráfego de rede é intrinsecamente dinâmico, com novos serviços e padrões de ataque surgindo continuamente, o que exige modelos capazes de se generalizar para diversos ambientes. No entanto, alcançar essa generalização geralmente requer modelos de ML complexos, que podem ser computacionalmente custosos e inadequados para processamento em tempo real em redes de alta velocidade. Além disso, integrar mecanismos escaláveis de ingestão de eventos e inferência distribuída para lidar com grandes volumes de tráfego de forma eficiente, mantendo a detecção com baixa latência.

Nosso modelo proposto aborda esse desafio por meio do uso de um ensemble de classificadores rasos, garantindo tanto eficiência quanto capacidade de generalização em redes de alta velocidade. O ensemble é construído a partir da seleção dos modelos com melhor desempenho no modo conjuntos de dados cruzados, permitindo que ele se adapte a variados comportamentos de rede mantendo baixa complexidade computacional. Essa abordagem aumenta a acurácia da detecção em diferentes ambientes e viabiliza a inferência em tempo quase real, sendo adequada para implantação em cenários de rede de alta velocidade e larga escala.

3.2. Pipeline de Inferência

Implementar a inferência em redes de alta velocidade apresenta um desafio significativo, pois requer não apenas a classificação do tráfego de rede recebido, mas também a ingestão de eventos em alta velocidade. Nesses ambientes, o volume e a velocidade dos dados tornam essencial o processamento e a classificação dos eventos em quase tempo real, sem introduzir atrasos. Essa exigência de ingestão rápida de eventos e inferência ágil impõe dificuldades, pois o sistema precisa ser capaz de processar grandes quantidades de dados continuamente, aplicando modelos de ML para classificação precisa e pontual. Equilibrar essas duas tarefas de forma eficiente em redes de alta velocidade exige arquiteturas escaláveis, que possam lidar com a alta taxa de transferência sem comprometer o desempenho da inferência.

Consideramos uma arquitetura que ingere eventos de rede a partir de múltiplas fontes de dados (*Data Sources*, Fig. 1). Os eventos são coletados por um *Message Broker*, que é implementado de forma distribuída para lidar com altos volumes de dados quase em tempo real. Os eventos coletados são então encaminhados ao *Endpoint de Inferência*,

que aplica o ensemble de classificadores (ver Seção 3.1) de forma distribuída para realizar essa tarefa em escala. Adicionalmente, o *endpoint* de inferência consulta periodicamente o sistema de armazenamento distribuído para obter a versão mais recente do modelo de ML garantindo que sempre se utilize o modelo mais atualizado para a classificação.

Separamos os processos de inferência e ingestão de eventos para garantir que cada tarefa possa ser escalada e otimizada de forma independente, evitando gargalos e aumentando a eficiência do sistema. Ao separar essas duas funções, é possível alocar recursos de maneira mais flexível, permitindo que o pipeline de ingestão se concentre em lidar com os eventos recebidos em alta velocidade sem ser impactado pelas exigências computacionais da inferência. Esse design também facilita o balanceamento de carga e permite escalar cada processo de acordo com as necessidades específicas do ambiente de rede, assegurando uma operação contínua mesmo sob condições de tráfego intenso. Além disso, utilizamos um *Armazenamento Distribuído* para armazenar o modelo de ML, o que melhora ainda mais a eficiência do nosso pipeline de inferência, permitindo uma recuperação rápida e escalável da versão mais recente do modelo para inferência.

3.3. Pipeline de Atualização de Modelo

Implementamos o pipeline de treinamento de modelo de forma distribuída, com o objetivo de lidar eficientemente com grandes volumes de dados de treinamento. O conjunto de dados de treinamento, armazenado em um sistema de armazenamento distribuído, é recuperado e processado em paralelo entre múltiplos nós, acelerando significativamente o processo de treinamento. O treinamento do modelo é realizado de maneira distribuída, utilizando o poder computacional do sistema para tratar conjuntos de dados em larga escala. Adicionalmente, podemos realizar atualizações periódicas dos modelos, garantindo que o classificador se adapte a comportamentos de rede em evolução e a novos padrões de ataque. Quando um novo modelo é treinado, ele é publicado e armazenado no sistema de armazenamento distribuído, ficando disponível para consulta pelo *endpoint* de inferência e assegurando que a versão mais recente do modelo esteja sempre em uso nas tarefas de classificação. Essa abordagem permite uma melhoria contínua e escalabilidade do sistema, sem comprometer o desempenho.

4. Protótipo

Implementamos uma proposta de protótipo em um ambiente distribuído. É considerado a implementação de uma arquitetura distribuída de processamento Big Data executando o esquema proposto (ver Seção 3). Para conseguir isso, os componentes de hardware e software do protótipo foram projetados para criar uma arquitetura de Big Data comumente disponível, capaz de ingestão de eventos de rede em escala para fins de inferência. Cada nó da arquitetura é executado com o Ubuntu $v.24.04$ equipado com um processador Intel i7 de 4 núcleos e 16GB de memória RAM. Os elementos da infraestrutura são implantados como *containers* isolados através do Docker $v24.0$.

O *Armazenamento Distribuído* (Fig. 1) é implementado por meio de um *cluster* Hadoop Distributed File System (HDFS) composto por três DataNodes e um NameNode. Os DataNodes armazenam o *Dataset de Treinamento* quando necessário para fins de treinamento e atualização de modelo, além do modelo de ML que vai ser utilizado para inferência. Implementamos o *Pipeline de Ingestão de Eventos* com um *message broker*

do Apache Kafka $v.3.7.0$. Os eventos utilizados para inferência são publicados como um tópico Kafka e posteriormente lidos pelo pipeline de inferência, em quase tempo real. O *Pipeline de Inferência* é executado sobre o Apache Spark $v.3.5.4$. Esse pipeline roda como um job do Apache Spark, com até 3 Workers simultâneos, realizando ingestão contínua de eventos para inferência via o broker Apache Kafka. Os eventos ingeridos são utilizados para inferência por meio da aplicação do modelo de ML previamente treinado, implementado com a biblioteca Apache Spark Machine Learning Library (MLlib). Para atingir tal objetivo, na fase de implantação do job, o modelo disponível no modulo *Armazenamento Distribuído* é lido e utilizado para a tarefa de inferência. O *Pipeline de Atualização de Modelo* também é implementado sobre o Apache Spark como um job com até 3 Workers. Nas atualizações de modelo, ele lê um conjunto de dados previamente armazenado no HDFS e constrói um novo modelo de ML por meio da API do Apache Spark MLlib. O modelo resultante é então armazenado novamente no HDFS.

5. Avaliação

Nossos experimentos conduzidos visam responder às seguintes Research Questions (RQs):

- ***RQ1:*** *Quais são as capacidades de generalização dos NIDSs tradicionais baseados em ML?*
- ***RQ2:*** *Nosso esquema de classificação proposto melhora a generalização?*
- ***RQ3:*** *Quais são as capacidades de escalabilidade do nosso esquema?*

5.1. Construção do Modelo

Avaliamos o nosso esquema proposto com base no protótipo descrito anteriormente (ver Seção 4). Para atingir tal objetivo, avaliamos três classificadores, nomeadamente Decision Tree (DT), Gradient Boosting (GBT), e Random Forest (RF). O classificador DT utiliza o critério de impureza de Gini para divisão dos nós. O classificador GBT é configurado com uma taxa de aprendizado de 0.1, utiliza a função de perda deviance e é composto por um ensemble de 10 árvores de decisão como modelos base. O classificador RF é composto por um ensemble de 10 árvores de decisão como modelos base, cujas predições são agregadas via votação majoritária. Os classificadores selecionados foram implementados utilizando a biblioteca Apache Spark MLlib, permitindo o treinamento e inferência de forma distribuída.

Avaliamos o desempenho dos algoritmos selecionados utilizando três conjuntos de dados de referência: UNSW-NB15 [Moustafa and Slay 2015], CS-CIC-IDS [Sharafaldin et al. 2018], e BoT-IoT [Moustafa et al. 2021]. Esses conjuntos de dados fornecem diversas características de tráfego de rede, permitindo uma avaliação abrangente das capacidades de generalização do nosso modelo em diferentes ambientes. Cada conjunto de dados é dividido aleatoriamente sem reposição em três subconjuntos *treinamento*, *validação*, e *teste*, cada um composto de 40%, 30%, e 30% de amostras, respectivamente. O conjunto de *treinamento* é utilizado para o aprendizado do modelo. O conjunto de *validação* é usado para ajuste fino (fine-tuning) do modelo. Por fim, o conjunto de *teste* é utilizado para medir a acurácia final do modelo. O desempenho relatado neste trabalho é medido com base no conjunto de *teste*.

Table 1. Desempenho de classificação dos classificadores selecionados em um cenário entre conjuntos de dados (cross-dataset), medido pelo F-Measure.

Amb. de Treinamento.	Classificador	Ambiente de Teste.		
		UNSW-NB15	CS-CIC-IDS	BoT-IoT
UNSW-NB15	DT	**0.97**	0.02	0.68
	GBT	**0.96**	0.04	0.65
	RF	**0.96**	0.10	0.16
CS-CIC-IDS	DT	0.32	**0.98**	0.08
	GBT	0.03	**0.97**	0.65
	RF	0.02	**0.97**	0.08
BoT-IoT	DT	0.07	0.23	**1.00**
	GBT	0.03	0.30	**1.00**
	RF	0.25	0.54	**1.00**
Nosso		**0.77**	**0.56**	**1.00**

5.2. Desafio da Generalização

Nosso primeiro experimento tem como objetivo responder à RQ1, investigando as capacidades de generalização dos NIDSs tradicionais baseados em ML. Para atingir esse objetivo, avaliamos o desempenho de classificação dos classificadores selecionados em múltiplos conjuntos de dados (ver Seção 5.1). Especificamente, avaliamos se a acurácia das técnicas selecionadas pode ser mantida quando aplicadas a um conjunto de dados diferente, em um cenário de avaliação entre conjuntos (cross-dataset). Essa abordagem nos permite determinar quão bem um modelo treinado em um ambiente de rede específico desempenha-se quando exposto a um conjunto de dados diferente.

A Tabela 1 apresenta o desempenho de classificação medido pelo F-Measure dos classificadores selecionados em um cenário entre conjuntos de dados. É possível observar que todos os classificadores selecionados alcançam altas taxas de detecção quando avaliados no mesmo ambiente em que foram treinados. Por exemplo, os classificadores atingiram uma média de F-Measure de 0.95, 0.96, e 1.00 nos conjuntos UNSW-NB15, CS-CIC-IDS e BoT-IoT, respectivamente. No entanto, quando esses modelos são avaliados em um ambiente diferente, seu desempenho se deteriora significativamente, evidenciando o desafio da generalização em NIDSs baseados em ML. Esse fenômeno é particularmente evidente no caso do classificador RF. Por exemplo, ao ser treinado no conjunto UNSW-NB15, o classificador RF apresenta forte desempenho de classificação no próprio dataset, mas quando avaliado nos conjuntos CS-CIC-IDS e BoT-IoT, seu F-Measure cai para apenas 0.10 e 0.16, respectivamente. Esse resultado indica que as características aprendidas de um conjunto de dados podem não generalizar bem para outro, já que diferentes datasets capturam características distintas do tráfego de rede, padrões de ataque e distribuições subjacentes.

Nosso segundo experimento tem como objetivo responder à RQ2 e investigar como o esquema de classificação proposto pode melhorar a capacidade de generalização dos NIDS baseados em ML. Para isso, implementamos nosso esquema de classificação baseado em ensemble (ver Seção 3.1) em cima do nosso protótipo proposto. Considerando que utilizamos três conjuntos de dados, construímos nosso ensemble $\mathcal{E}$ composto por três classificadores. Nesse caso, o ensemble é composto pelos classificadores

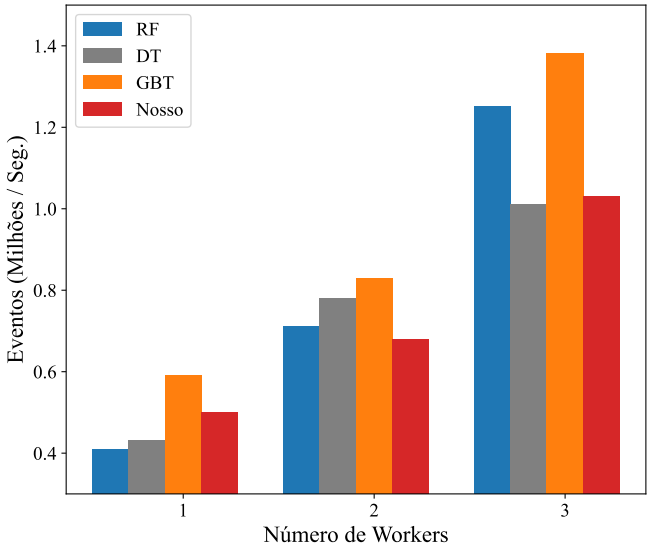

Figure 2. Escalabilidade da nossa proposta.

DT, RF e GBT, construídos a partir dos conjuntos de dados UNSW-NB15, CSE-CIC-IDS e BoT-IoT, respectivamente. Os classificadores foram escolhidos conforme nossa estratégia de construção do ensemble, baseada na acurácia alcançada em cada conjunto de dados. A combinação de classificações do ensemble resultante é conduzida por meio de uma estratégia de votação majoritária.

A Tabela 1 apresenta o desempenho de classificação do nosso modelo proposto nos diferentes conjuntos de dados de teste. É evidente que nossa abordagem melhora substancialmente o F-Measure no cenário de avaliação entre conjuntos de dados (cross-dataset), demonstrando capacidade de generalização superior em comparação aos NIDSs tradicionais baseados em ML. Embora o F-Measure do nosso modelo permaneça ligeiramente inferior ao de classificadores treinados e testados no mesmo conjunto de dados, ele consistentemente supera os modelos avaliados em ambientes diferentes daqueles em que foram treinados. Por exemplo, uma comparação direta com o classificador RF destaca ainda mais as vantagens da nossa abordagem. Como discutido anteriormente, o classificador RF apresenta uma queda drástica no F-Measure quando aplicado a datasets diferentes daquele utilizado no treinamento, atingindo apenas *0.10* ao ser treinado em CS-CIC-IDS e avaliado no BoT-IoT. Em contraste, nosso modelo proposto mantém um F-Measure significativamente superior sob as mesmas condições, atingindo $0,56$, reforçando sua capacidade de generalizar efetivamente em diferentes ambientes de rede.

5.3. Escalabilidade da Detecção em NIDS Baseados em ML

Por fim, respondemos à RQ3 e investigamos as capacidades de escalabilidade do nosso modelo proposto ao operar com o esquema de classificação baseado em ensemble. Para isso, analisamos como o nosso ensemble proposto pode escalar tanto a inferência quanto o treinamento do modelo, conforme implementado em nosso protótipo (ver Seção 4). Neste caso, avaliamos o desempenho de escalabilidade do nosso esquema $vs.$ à abordagem tradicional ao variar o número de trabalhadores (workers) do Apache Spark implantados.

A Figura 2 ilustra a escalabilidade da inferência do nosso modelo proposto con-

forme aumenta o número de workers implantados. Os resultados demonstram que nossa abordagem escala de maneira eficaz em um ambiente distribuído, permitindo o processamento eficiente de eventos de rede, mantendo alta performance de classificação. Além disso, nosso esquema de classificação baseado em ensemble proposto alcança uma taxa de inferência comparável às técnicas tradicionais de ML. Mesmo incorporando múltiplos classificadores, o ensemble não introduz uma sobrecarga computacional excessiva, possibilitando a classificação em escala quase em tempo real. Isso é particularmente evidente ao comparar nosso modelo com o classificador RF. Por exemplo, quando implantado com três workers, o classificador RF atinge uma vazão de inferência de aproximadamente ≈ 1.27 milhões de eventos por segundo, enquanto nosso modelo proposto alcança cerca de ≈ 1.07 milhões de eventos por segundo, demonstrando que nossa abordagem entrega uma taxa semelhante com capacidade de generalização significativamente superior. No geral, esses resultados destacam a eficácia da nossa implementação distribuída.

6. Conclusão

Neste trabalho propusemos um NIDS distribuído baseado em ensemble, que utiliza uma estrutura escalável de Big Data para o treinamento, inferência e atualização eficiente de modelos. Nossa arquitetura desacopla a ingestão de eventos e inferência, garantindo o processamento em alta velocidade sem comprometer o desempenho de classificação. O sistema proposto recupera dinamicamente as versões mais recentes dos modelos a partir de um armazenamento distribuído, permitindo atualizações contínuas e a adaptação a novas ameaças. Ademais, ao implementar a inferência e as atualizações de modelo sobre o Apache Spark, nossa abordagem assegura que ambas as tarefas escalem eficientemente conforme aumenta o número de recursos computacionais implantados.

Agradecimentos

Este trabalho foi parcialmente financiado pelo Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), número do processo 302937/2023-4, 407879/2023-4, e 442262/2024-8.

References

Abid, A., Jemili, F., and Korbaa, O. (2023). Real-time data fusion for intrusion detection in industrial control systems based on cloud computing and big data techniques. *Cluster Computing*, 27(2):2217–2238.

Abreu, V., Santin, A. O., Viegas, E. K., and Stihler, M. (2017). A multi-domain role activation model. In *2017 IEEE International Conference on Communications (ICC)*, page 1–6. IEEE.

Akili, S., Purtzel, S., and Weidlich, M. (2024). Decopa: Query decomposition for parallel complex event processing. *Proceedings of the ACM on Management of Data*, 2(3):1–26.

Cantone, M., Marrocco, C., and Bria, A. (2024). Machine learning in network intrusion detection: A cross-dataset generalization study. *IEEE Access*, 12:144489–144508.

Espindola, A., Viegas, E. K., Traleski, A., Pellenz, M. E., and Santin, A. O. (2021). A deep autoencoder and rnn model for indoor localization with variable propagation loss.

In *2021 17th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*. IEEE.

Filho, A. G., Viegas, E. K., Santin, A. O., and Geremias, J. (2025). A dynamic network intrusion detection model for infrastructure as code deployed environments. *Journal of Network and Systems Management*, 33(4).

Hazman, C., Guezzaz, A., Benkirane, S., and Azrour, M. (2022). lids-sioel: intrusion detection framework for iot-based smart environments security using ensemble learning. *Cluster Computing*, 26(6):4069–4083.

Horchulhack, P., Viegas, E. K., Santin, A. O., and Simioni, J. A. (2024). Network-based intrusion detection through image-based cnn and transfer learning. In *2024 International Wireless Communications and Mobile Computing (IWCMC)*, page 386–391. IEEE.

Hussen, N., Elghamrawy, S. M., Salem, M., and El-Desouky, A. I. (2023). A fully streaming big data framework for cyber security based on optimized deep learning algorithm. *IEEE Access*, 11:65675–65688.

Jemili, F., Meddeb, R., and Korbaa, O. (2023). Intrusion detection based on ensemble learning for big data classification. *Cluster Computing*, 27(3):3771–3798.

Moustafa, N., Keshk, M., Choo, K.-K. R., Lynar, T., Camtepe, S., and Whitty, M. (2021). Dad: A distributed anomaly detection system using ensemble one-class statistical learning in edge networks. *Future Generation Computer Systems*, 118:240–251.

Moustafa, N. and Slay, J. (2015). Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, page 1–6. IEEE.

Rodrigues, M. G., Viegas, E. K., Santin, A. O., and Enembreck, F. (2025). A mlops architecture for near real-time distributed stream learning operation deployment. *Journal of Network and Computer Applications*, 238:104169.

Sharafaldin, I., Habibi Lashkari, A., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy*. SCITEPRESS - Science and Technology Publications.

Shrestha, S., Pathak, S., and Viegas, E. K. (2023). Towards a robust adversarial patch attack against unmanned aerial vehicles object detection. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, page 3256–3263. IEEE.

Simioni, J. A., Viegas, E. K., Santin, A. O., and de Matos, E. (2025). An energy-efficient intrusion detection offloading based on dnn for edge computing. *IEEE Internet of Things Journal*, 12(12):20326–20342.

Wang, M., Yang, N., Guo, Y., and Weng, N. (2024). Learn-ids: Bridging gaps between datasets and learning-based network intrusion detection. *Electronics*, 13(6):1072.

Ye, Z., Luo, J., Zhou, W., Wang, M., and He, Q. (2024). An ensemble framework with improved hybrid breeding optimization-based feature selection for intrusion detection. *Future Generation Computer Systems*, 151:124–136.