Otimização e Análise Crítica de um Classificador de Ataques de Rede por Similaridade no Dataset CIC-IDS-2018

Nícolas Warmeling[1], **Laura Klippel**[1], **Carlo Mantovani**[1], **Tiago Ferreto**[1]

[1]Escola Politécnica – Pontifícia Universidade Católica do Rio Grande do Sul (PUCRS)
90619-900 – Porto Alegre – RS – Brasil

{nicolas.warmeling,laura.klippel,c.mantovani,tiago.ferreto}@edu.pucrs.br

Abstract. *This work optimizes a network packet classification model based on similarity vectors for the CIC-IDS-2018 dataset. To address memory challenges exceeding 100 GB RAM, the FAISS library replaced the k-NN algorithm, enabling analysis on a high-performance VM. Replicating original test scenarios with additional precision metrics revealed low effectiveness in detecting underrepresented attacks, reflecting the accuracy paradox in imbalanced datasets. While optimizations proved successful, data balancing is necessary for robust intrusion detection systems.*

Resumo. *Este trabalho otimiza um modelo de classificação de pacotes de rede baseado em vetores de similaridade para o dataset CIC-IDS-2018. Para enfrentar desafios de memória superiores a 100 GB de RAM, a biblioteca FAISS substituiu o algoritmo k-NN, possibilitando análise em uma VM de alta performance. A replicação dos cenários de teste originais com métricas adicionais de precisão revelou baixa efetividade na detecção de ataques subrepresentados, refletindo o paradoxo da acurácia em datasets desbalanceados. Embora as otimizações tenham sido bem-sucedidas, o balanceamento de dados é necessário para sistemas de detecção de intrusão robustos.*

1. Introdução

Com a crescente dependência global em tecnologia, esses ativos se tornam cada vez mais valiosos. Consequentemente, a necessidade de protegê-los torna-se uma prioridade [IBM X-Force 2022]. Diante deste cenário, os Network Intrusion Detection System (NIDS) trabalham recebendo tráfego de rede e comparando os padrões (assinaturas) com seu banco de dados de ameaças conhecidas. Se houver correspondência, a intrusão é identificada [Sharafaldin et al. 2018].

O projeto desenvolvido utiliza NIDS aprimorados com Aprendizado de Máquina (ML). Esta abordagem visa superar as limitações das NIDS tradicionais, cuja rigidez de dados impede de identificar ameaças novas e evoluídas [Singhal 2001]. No entanto, a migração de classificadores existentes para datasets modernos e massivos, como o CIC-IDS-2018, apresenta desafios computacionais e de análise. Este trabalho aborda a otimização de um classificador baseado em similaridade para lidar com esse volume de dados e analisa criticamente seu desempenho em um cenário de classes desbalanceadas.

2. Objetivos

Este trabalho tem como objetivo investigar o desempenho de um classificador de ataques de rede baseado em similaridade, modernizado através da substituição do algoritmo

k-NN pela biblioteca FAISS para operar com eficiência no volume massivo do dataset CIC-IDS-2018. A análise busca validar essa otimização comparando-a com resultados obtidos em datasets anteriores (CIC-IDS-2017) e aplicando métricas estatísticas robustas, como F1-Score e médias por classe. O foco central é evidenciar o impacto crítico do desbalanceamento de dados, quantificando o paradoxo da acurácia e a limitações do modelo na detecção de ameaças sub-representadas.

3. Análise Comparativa dos Datasets

A transição para o dataset CIC-IDS-2018 impõe desafios de volume e complexidade superiores aos do CIC-IDS-2017 [Sharafaldin et al. 2018]. A motivação para a arquitetura proposta decorre de experimentos preliminares no dataset de 2017, onde a classificação por similaridade demonstrou alto potencial, atingindo **93,25%** de precisão em cenários controlados e balanceados. Contudo, a técnica mostrou-se inviável em larga escala devido ao custo da busca em força bruta. A necessidade de reduzir a base de treino para apenas 25% das amostras resultou em perda de representatividade, derrubando a precisão global para **63,58%**. Para superar essa limitação e processar o CIC-IDS-2018 em sua totalidade, este trabalho implementa a otimização via FAISS em infraestrutura de alto desempenho, concretizando a necessidade de paralelização apontada nos experimentos anteriores para recuperar a eficácia do modelo.

3.1. Volume de Dados

O dataset CIC-IDS2018 apresenta um volume significativamente maior de dados em comparação ao CIC-IDS2017. Como consequência, houve um aumento substancial na demanda de processamento, exigindo aproximadamente 80 GB de memória RAM, com picos que chegaram a 100 GB durante a execução dos experimentos. Além disso, observa-se uma diferença marcante na proporção entre amostras benignas e maliciosas, o que introduz um desbalanceamento acentuado o que impacta diretamente as etapas de análise e modelagem.

3.2. Diferenças de Rotulagem entre Datasets

As divergências entre os dois datasets (CIC-IDS2017 e CIC-IDS2018) não se limitam a um simples aumento no volume de dados; a forma como as informações são organizadas também difere. Algumas diferenças relevantes incluem:

- **Remoção de ataques específicos:** Os ataques `Heartbleed` e `PortScan`, presentes no dataset de 2017, foram eliminados no conjunto de 2018.
- **Reorganização de categorias de ataque:** A categoria `DDoS`, que no CIC-IDS2017 se fazia presente como uma única *label* (rótulo) genérica, foi subdividida no CIC-IDS2018 em classificações mais específicas, como `DDoS attack-HOIC` e `DDoS attack-LOIC-UDP`.
- **Alterações de nomenclatura:** Ataques como `DoS Hulk`, `GoldenEye`, `SlowHTTPTest`, `Slowloris`, `FTP-BruteForce`, `SSH-Bruteforce`, `Bot`, `Infiltration`, `Web Brute Force`, `Web XSS` e `SQL Injection` foram mantidos no novo dataset, porém sofreram pequenas modificações em seus nomes ou padronizações.

A distribuição de rótulos de cada dataset está representada na Tabela 1 e Tabela 2.

Tabela 1. Distribuição de rótulos no CICIDS 2017

Nome	Ocorrência
BENIGN	2.272.895
DoS Hulk	231.072
PortScan	158.930
DDoS	128.027
DoS GoldenEye	10.293
FTP-Patator	7.938
SSH-Patator	5.897
DoS slowloris	5.796
DoS Slowhttptest	5.499
Bot	1.966
Web Attack Brute Force	1.507
Web Attack XSS	652
Infiltration	36
Web Attack Sql Injection	21
Heartbleed	11

Tabela 2. Distribuição de rótulos no CICIDS 2018

Nome	Ocorrência
Benign	6.095.415
DDOS attack-HOIC	668.461
DoS attacks-Hulk	434.873
Bot	282.310
Infiltration	161.897
SSH-Bruteforce	117.322
DoS attacks-GoldenEye	41.455
FTP-BruteForce	39.352
DoS attacks-SlowHTTPTest	19.462
DoS attacks-slowloris	10.285
DDOS attack-LOIC-UDP	1.730
Brute Force -Web	611
Brute Force -XSS	230
SQL Injection	87

4. Metodologia

A metodologia adotada consiste em um pipeline de processamento de dados para extração de características, seguido pela construção de um classificador baseado em similaridade vetorial otimizado.

4.1. Seleção e Pré-processamento de Dados

O conjunto de dados CIC-IDS-2018 foi submetido a um processo de limpeza e transformação. Inicialmente, foram selecionadas 23 *features* estatísticas e de cabeçalho consideradas mais relevantes para a caracterização de fluxos, descartando atributos redundantes ou de baixo valor preditivo. As variáveis selecionadas foram:

- **Identificação e Cabeçalho:** *Protocol, Fwd Header Len, Bwd Header Len*.
- **Tempo e Fluxo:** *Flow IAT Mean, Flow IAT Min, Flow IAT Max, Fwd IAT Mean, Bwd IAT Mean, Idle Mean, Active Mean, Active Min, Active Max*.
- **Estatísticas de Pacotes:** *Fwd Pkt Len Mean, Bwd Pkt Len Mean, Fwd Pkt Len Min, Fwd Pkt Len Max, Pkt Len Min, Pkt Len Max*.
- **Flags TCP:** *SYN Flag Cnt, PSH Flag Cnt, ACK Flag Cnt, URG Flag Cnt, FIN Flag Cnt*.

Para garantir a identificação única de cada fluxo e permitir o rastreamento durante a classificação, foi criado um identificador composto denominado `Flow_Label`. Este identificador é gerado pela concatenação dos valores brutos das colunas selecionadas, servindo como chave primária para os vetores.

O pré-processamento seguiu as seguintes etapas:

1. **Limpeza:** Remoção de duplicatas baseadas no `Flow_Label` e tratamento de valores nulos.

2. **Codificação:** Os rótulos das classes (*Labels*) foram convertidos utilizando *One-Hot Encoding*, permitindo a classificação multilabel.
3. **Padronização:** Foi aplicado o algoritmo `StandardScaler` (Z-Score), ajustando as features para média 0 e desvio padrão 1. Esta etapa é crítica para algoritmos baseados em distância, impedindo que variáveis com grandes magnitudes dominem o cálculo vetorial.

4.2. Arquitetura do Classificador e Otimização com FAISS

A classificação baseia-se na premissa de que fluxos de ataques semelhantes possuem vetores de características próximos no espaço vetorial. A métrica utilizada para quantificar essa proximidade foi a Similaridade de Cosseno.

Na implementação inicial utilizando a biblioteca `scikit-learn` (*Nearest-Neighbors*), a complexidade computacional da busca em força bruta inviabilizou a execução no dataset completo devido ao consumo excessivo de memória RAM e tempo de CPU.

Para solucionar este gargalo, o módulo de busca de vizinhos foi substituído pela biblioteca **FAISS** (Facebook AI Similarity Search). A implementação utilizou o índice `IndexFlatIP` (*Inner Product*), que realiza o cálculo exato da similaridade. Para equivaler à similaridade de cosseno, os vetores de consulta e de base foram submetidos a uma normalização L2 prévia ($||\mathbf{v}|| = 1$).

O pipeline de classificação final opera da seguinte forma:

1. O conjunto de treino é indexado no FAISS.
2. Para cada fluxo do conjunto de teste, o sistema recupera os k vizinhos mais próximos ($k = 50$).
3. A classificação é atribuída com base na frequência majoritária dos rótulos presentes nesses vizinhos recuperados.

4.3. Ambiente Experimental

Os experimentos foram executados em um ambiente virtualizado de alto desempenho, configurado para suportar a carga de dados massiva do CIC-IDS-2018.

A infraestrutura de hardware alocada para a máquina virtual (VM) consistiu em:

- **Processamento:** 32/64 cores de CPU Intel(R) Xeon(R) Silver 4310.
- **Memória:** 160 GB de RAM.
- **Armazenamento:** 500 GB de disco rígido (HDD)

A implementação do software foi realizada em linguagem **Python 3**, utilizando o ecossistema de bibliotecas `pandas` e `numpy` para estruturação de dados, `scikit-learn` para o pipeline de pré-processamento, e `faiss-cpu` para a motorização da busca vetorial.

5. Discussão e Resultados

Para avaliar o desempenho do novo dataset e das estratégias de otimização realizadas, foram definidos quatro cenários de teste:

- **Primeiro Cenário:** Foca na detecção de ataques com baixo volume de amostras, avaliando a capacidade do modelo em identificar classes raras.
- **Segundo Cenário:** Treina o modelo exclusivamente com os ataques mais frequentes, submetendo-o posteriormente a um ambiente diversificado para verificar sua capacidade de generalização.
- **Terceiro Cenário:** Avalia o desempenho utilizando uma divisão percentual de todos os dados entre treino e teste.
- **Quarto Cenário:** Utiliza 70% do dataset para treino e 30% para teste, permitindo uma análise mais padronizada e comparável com estudos similares.

5.1. Métricas de Avaliação

Inicialmente, a análise baseou-se nas métricas clássicas Precision e Recall [Powers 2011]. No entanto, em contextos de desbalanceamento de classes, essas métricas isoladas podem não refletir o desempenho real nas classes minoritárias. Para superar essa limitação, este trabalho incorporou indicadores mais robustos, calculados com o auxílio da biblioteca Scikit-learn [Pedregosa et al. 2011]:

- **Suporte:** Representa o número absoluto de amostras de cada classe no conjunto de teste. É fundamental para diferenciar o desempenho entre ataques massivos e raros.
- **Micro Average:** Calcula a performance global, somando os verdadeiros positivos, falsos negativos e falsos positivos de todas as classes. Tende a ser dominada pelas classes com maior suporte e reflete a acurácia geral do sistema.
- **Macro Average:** Calcula a média aritmética simples das métricas de cada classe, sem ponderar pelo suporte. Diferente da Micro, a Macro trata todas as classes com igual importância, e penaliza severamente a pontuação final caso o modelo falhe na detecção de ataques raros, servindo como o principal indicador de viabilidade do sistema em cenários reais.

5.2. Testes

Os quatro testes demonstram que, apesar da acurácia global alta, o classificador falha consistentemente em ataques raros. No teste 70/30 (Gráfico 1) e no Teste 3 (Gráfico 4), o modelo tem ótimo desempenho nas classes comuns (**Micro Avg de 0.99 e 0.94**, respectivamente), mas a queda abrupta na média por classe (**Macro Avg de 0.57 e 0.56**) revela que o F1-Score é próximo de zero nas classes com pouco suporte. O Teste 1 (Gráfico 2) confirma isso ao focar nos ataques menos frequentes, resultando em uma **Macro Avg de apenas 0.45**, no qual somente as classes com mais exemplos têm resultados satisfatórios. Já no Teste 2 (Gráfico 3), treinado somente com ataques comuns, a **Macro Avg reduz para 0.14**, indicando que o desempenho nas demais classes praticamente desaparece. Em conjunto, os testes demonstram que o desbalanceamento é o principal limitador do modelo, independentemente do volume total de dados.

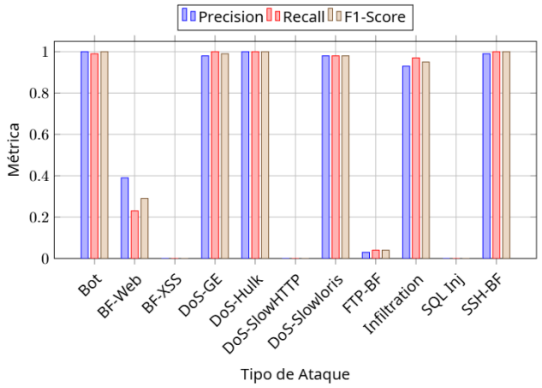

Figura 1. Resultados do Teste 70/30

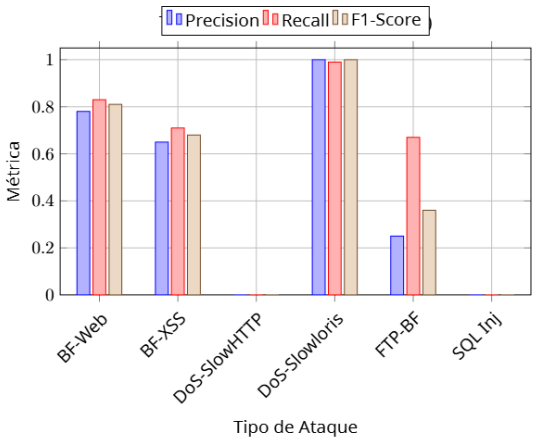

Figura 2. Resultados do Teste 1

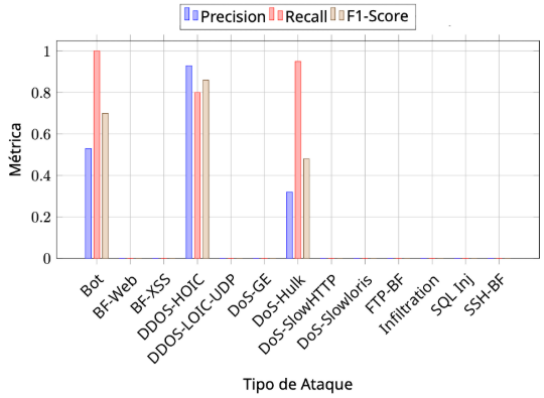

Figura 3. Resultados do Teste 2

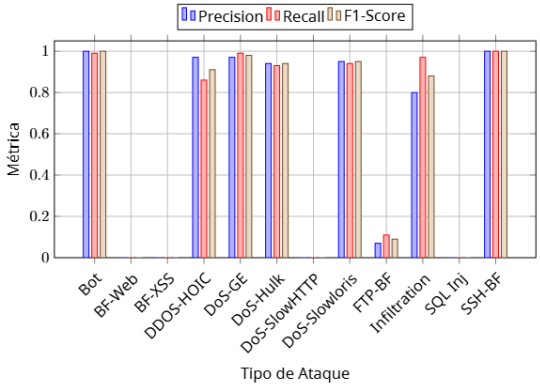

Figura 4. Resultados do Teste 3

6. Trabalhos Relacionados

A literatura lida com o desbalanceamento de classes variando desde reamostragem clássica até *Deep Learning*. O modelo IDS-SMOTE-RF [Alshamy et al. 2021] estabelece o *benchmark* padrão, utilizando a técnica SMOTE para gerar dados sintéticos simples antes da classificação. Em contraste, abordagens modernas evoluíram para o uso de Redes Generativas Adversárias (GANs) [Barkah et al. 2023] e arquiteturas híbridas com *Autoencoders* e *Transformers* [Kamal and Mashaly 2025], que buscam aprender a distribuição real dos ataques para gerar amostras complexas.

A contribuição deste trabalho diferencia-se ao demonstrar empiricamente que, mesmo resolvendo o gargalo de performance computacional com o uso do FAISS, a dificuldade do modelo com ataques raros persiste no CIC-IDS-2018. Isso confirma que o aumento de volume de dados não substitui a necessidade crítica da aplicação dessas técnicas de balanceamento.

7. Conclusão

O classificador de rede foi implementado para o dataset CIC-IDS-2018 com sucesso, superando as limitações computacionais observadas em trabalhos anteriores com o dataset

de 2017. Enquanto estudos passados viram a precisão cair para 63% devido à incapacidade de processar todo o volume de dados, a utilização da biblioteca FAISS aliada a uma infraestrutura de 160GB de RAM permitiu, neste trabalho, a indexação completa do vetor base sem perda de características por restrição de hardware.

Neste trabalho, teve-se a oportunidade de demonstrar, na prática e em larga escala, as limitações de um classificador treinado com dados desbalanceados. As métricas de análise permitiram avaliar seu desempenho de forma mais precisa, revelando que, apesar da alta acurácia geral, o modelo apresentava falhas críticas na detecção de ataques minoritários, um ponto que a acurácia, isoladamente, não evidencia [Powers 2011]. Com base nessas conclusões, trabalhos futuros podem investigar outras técnicas de balanceamento de dados para aprimorar a detecção de ataques raros, como a geração de dados sintéticos por meio de Redes Generativas Adversárias (GANs) [Arjovsky et al. 2017]. A melhoria da robustez do classificador contra classes minoritárias permanece como o principal desafio para a evolução deste trabalho.

Referências

Alshamy, R., Ghurab, M., Othman, S., and Alshami, F. (2021). *Intrusion Detection Model for Imbalanced Dataset Using SMOTE and Random Forest Algorithm*, pages 361–378.

Arjovsky, M., Chintala, S., and Bottou, L. (2017). Wasserstein GAN. In *Proceedings of the 34th International Conference on Machine Learning*, pages 214–223. PMLR.

Barkah, A., Selamat, S. R., Abidin, Z., and Wahyudi, R. (2023). Data generative model to detect the anomalies for ids imbalance cicids2017 dataset. *TEM Journal*, 12:80–89.

IBM X-Force (2022). X-force threat intelligence index 2022. Technical report, IBM Corp.

Kamal, H. and Mashaly, M. (2025). Hybrid deep learning-based autoencoder-dnn model for intelligent intrusion detection system in iot networks. pages 1–6.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830. Acesso em: 27 jun. 2025.

Powers, D. M. W. (2011). Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. *Journal of Machine Learning Technologies*, 2(1):37–63.

Sharafaldin, I., Lashkari, A. H., 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 (ICISSP)*, pages 108–116, Funchal, Madeira, Portugal. SciTePress. Dataset available at `https://www.unb.ca/cic/datasets/ids-2018.html`.

Singhal, A. (2001). Modern information retrieval: a brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43.