Avaliação de Modelos Leves para Detecção de Anomalias em Redes IoT Domésticas

Euler da Silva Lima[1], **Ana Carolina Xavier Castro**[1], **Cauê Rodrigues de Aguiar**[1]
Geraldo Pereira Rocha Filho[1]

[1]Departamento de Ciências Exatas e Tecnológicas
Universidade Estadual do Sudoeste da Bahia (UESB)
Vitória da Conquista – BA – Brasil

{euler410,anacarolina.acxc}@gmail.com, cauaguiar@outlook.com.br,
geraldo.rocha@uesb.edu.br

Abstract. *The proliferation of IoT devices in home networks has expanded the residential attack surface, yet few studies jointly evaluate predictive performance and computational cost for edge deployment. This work compares three machine learning models, Isolation Forest, Autoencoder, and Random Forest, for anomaly detection in IoT network traffic, assessing both classification metrics and resource consumption. Experiments on two public datasets show that unsupervised models can achieve competitive detection performance without requiring labeled attack data, supporting the viability of lightweight agents for anomaly detection in real home IoT networks.*

Resumo. *A proliferação de dispositivos IoT em redes domésticas expandiu a superfície de ataque residencial, contudo, poucos estudos avaliam conjuntamente o desempenho preditivo e o custo computacional para implantação na borda. Este trabalho compara três modelos de aprendizado de máquina, Isolation Forest, Autoencoder e Random Forest, para detecção de anomalias no tráfego de redes IoT, avaliando as métricas de classificação e o consumo de recursos. Experimentos em dois conjuntos de dados públicos demonstram que modelos não supervisionados podem alcançar desempenho de detecção competitivo sem a necessidade de dados de ataque rotulados, comprovando a viabilidade de agentes leves para detecção de anomalias em redes IoT domésticas reais.*

1. Introdução

No período de 2015 a 2025, a proporção de domicílios com acesso à internet no Brasil cresceu de 51% para 86% [Comitê Gestor da Internet no Brasil 2025]. Esse crescimento das redes domésticas acompanha a evolução do ecossistema digital, que inclui, entre outros fatores, a disseminação de dispositivos IoT (*Internet of Things*).

Entretanto, a maioria desses dispositivos opera com recursos computacionais limitados, *firmware* raramente atualizado e, frequentemente, credenciais padrão inalteradas, características que ampliam a superfície de ataque das redes domésticas [Adhikari et al. 2024]. Nesse cenário, os IDS (*Intrusion Detection System*) baseados em anomalias, surgem como alternativa [Rafique et al. 2024]. Para redes domésticas, abordagens não supervisionadas são interessantes pois aprendem o comportamento normal do tráfego da rede e sinalizam desvios como possíveis ameaças, não dependentes de dados previamente rotulados como maliciosos para o treinamento [Alaghbari et al. 2023].

Diversos trabalhos têm investigado a detecção de anomalias em redes IoT utilizando algoritmos de aprendizado de máquina não supervisionado, como Isolation Forest e Autoencoders, obtendo resultados promissores em ambientes controlados [Rafique et al. 2024]. Entretanto, a maioria desses estudos avalia predominantemente o desempenho preditivo dos modelos, sem examinar com o mesmo rigor o custo computacional associado, aspecto fundamental para a implantação em dispositivos embarcados com recursos restritos [Javed et al. 2024]. Como consequência, ainda não está claro qual abordagem oferece o melhor equilíbrio entre capacidade de detecção e viabilidade de implantação em redes domésticas IoT.

O presente trabalho realiza uma análise comparativa de três algoritmos de aprendizado de máquina para detecção de anomalias em tráfego de rede IoT: Isolation Forest, Autoencoder e Random Forest, com o objetivo de identificar o modelo mais adequado para agentes leves de detecção em redes domésticas. As principais contribuições deste trabalho são: (i) avaliação comparativa dos três modelos em dois *datasets* públicos amplamente utilizados na literatura de segurança em redes IoT, o N-BaIoT e o CICIoT2023; (ii) análise conjunta de métricas de classificação (F1-Score e AUC-ROC) e custo computacional (tamanho em disco, latência e tempo de inferência por amostra), aspectos pouco explorados de forma integrada na literatura; e (iii) evidência empírica de que modelos não supervisionados, em especial o Autoencoder, oferecem viabilidade de implantação em dispositivos embarcados sem exigir tráfego malicioso rotulado no treinamento.

O restante deste artigo está organizado da seguinte forma: a Seção 2 apresenta a fundamentação teórica dos algoritmos utilizados; a Seção 3 discute os trabalhos relacionados; a Seção 4 detalha a metodologia; a Seção 5 apresenta e analisa os resultados obtidos; e, por fim, a Seção 6 conclui o trabalho e indica direções futuras de pesquisa.

2. Fundamentação Teórica

A detecção de intrusões em redes é fundamentada nos paradigmas de assinatura, porém é eficaz apenas contra ameaças conhecidas, onde em um cenário de ataques *zero-day* e de anomalia se encontra limitado, devido a desvios de comportamento padrão previamente aprendido [Khraisat et al. 2019]. Esta última abordagem mostra-se essencial para ambientes de IoT domésticos, onde a falta de atualizações de segurança e a possibilidade de existir uma grande quantidade de dispositivos inviabilizam catálogos de assinaturas estáticas, tornando-as alvos fáceis para invasores e comprometendo a integridade geral da rede [Alotaibi 2024]. Diante disso, este trabalho visa avaliar técnicas não supervisionadas como Autoencoder [Torabi et al. 2023] e Isolation Forest [Liu et al. 2008] comparando com o desempenho de referência do modelo supervisionado Random Forest [Breiman 2001].

2.1. Isolation Forest

O Isolation Forest é um algoritmo de aprendizado de máquina não supervisionado, mais focado em *outliers* em grande conjunto de dados [Liu et al. 2008]. Nele, ao invés de modelar o comportamento normal e identificar a diferença e desvios, ocorre a detecção de anomalias de forma direta, onde é explorado o fato de que observações anômalas são mais simples de isolar do que a normalidade.

Seu funcionamento se baseia na construção de um conjunto de árvores de decisão

binárias, chamadas de *isolation trees*, onde cada árvore é construída selecionando aleatoriamente uma *feature* e um valor de corte entre o mínimo e o máximo, particionando de forma recursiva os dados até que cada observação seja isolada em uma folha. A intuição central é que uma anomalia, por ser distante das demais observadas, é isolada com poucos cortes, ou seja, gera um caminho curto da raiz até a folha, por estar concentrado em regiões densas dos dados, onde requer muito mais partições para ser isolado, resultando em um caminho longo.

Então, a pontuação de anomalia (*anomaly score*) de uma amostra $\mathbf{x}$ é calculada com base no comprimento médio do caminho $h(\mathbf{x})$ ao longo de todas as árvores da floresta, normalizado pelo comprimento esperado para um conjunto de dados de tamanho n, como é possível ver na Equação 1, onde $\mathbb{E}[h(\mathbf{x})]$ é o comprimento médio do caminho nas *iTrees* e $c(n) = 2H(n-1) - \frac{2(n-1)}{n}$ é o comprimento médio esperado em uma árvore de busca binária com n nós, usado como fator de normalização, sendo $H(i)$ o número harmônico estimado por $\ln(i) + 0{,}5772$ (constante de Euler-Mascheroni) . A pontuação varia entre 0 e 1: valores próximos de 1 indicam anomalia, valores próximos de 0,5 indicam comportamento normal.

$$s(\mathbf{x}, n) = 2^{-\frac{\mathbb{E}[h(\mathbf{x})]}{c(n)}} \tag{1}$$

Neste trabalho, o Isolation Forest é treinado exclusivamente com amostras de tráfego benigno, seguindo uma estratégia não supervisionada. O parâmetro *contamination* (que estima a proporção de anomalias esperada nos dados de treino) foi fixado em 0,1, e o *ensemble* é composto por 100 árvores (`n_estimators = 100`).

2.2. Autoencoder

O Autoencoder é uma rede neural não supervisionada projetada para aprender uma representação comprimida dos dados de entrada e reconstruí-la na saída [Torabi et al. 2023]. A arquitetura é composta por um *encoder* f_e, que mapeia a entrada $\mathbf{x} \in \mathbb{R}^d$ para um espaço latente $\mathbf{z} \in \mathbb{R}^k$ (com $k \ll d$), e um *decoder* f_d, que reconstrói a entrada a partir desse vetor comprimido, tal que $\hat{\mathbf{x}} = f_d(f_e(\mathbf{x}))$. Dessa forma, o treinamento minimiza o Erro Quadrático Médio (MSE) entre a entrada original e a reconstruída, conforme a Equação 2:

$$\mathcal{L}(\mathbf{x}) = \frac{1}{d}\sum_{i=1}^{d}(x_i - \hat{x}_i)^2 \tag{2}$$

A aplicação de Autoencoders para detecção de anomalias em redes IoT baseia-se no princípio de que um modelo treinado exclusivamente com tráfego benigno aprende a reconstruir fielmente apenas padrões normais [Meidan et al. 2018]. Diante de uma amostra anômala, o espaço latente aprendido não captura adequadamente suas características, resultando em erro de reconstrução elevado [Alaghbari et al. 2023]. Assim, dado um limiar τ, uma amostra é classificada segundo a regra apresentada na Equação 3:

$$\hat{y} = \begin{cases} 1 & \text{se } \mathcal{L}(\mathbf{x}) > \tau \\ 0 & \text{caso contrário} \end{cases} \tag{3}$$

Neste trabalho, o Autoencoder é composto por camadas densas com funções de ativação ReLU (*Rectified Linear Unit*), seguindo a arquitetura proposta por [Alaghbari et al. 2023]. O limiar τ é definido como o *percentil* 95 do MSE calculado sobre um subconjunto de validação composto exclusivamente por tráfego benigno, estratégia que equilibra sensibilidade e especificidade sem exigir amostras rotuladas de ataque [Torabi et al. 2023]. Ademais, abordagens semelhantes foram adotadas por [Ayad et al. 2024] com resultados expressivos no *dataset* BoT-IoT, atingindo 99,99% de acurácia, reforçando a viabilidade do modelo em cenários IoT com restrições computacionais [Adhikari et al. 2024].

2.3. Random Forest

O Random Forest é um algoritmo de aprendizado supervisionado baseado no conceito de *ensemble learning*, no qual múltiplas árvores de decisão são combinadas para produzir uma predição final mais robusta. Proposto por [Breiman 2001], o método constrói diversas árvores de decisão a partir de subconjuntos aleatórios dos dados de treinamento, reduzindo o risco de *overfitting* e aumentando a capacidade de generalização do modelo.

Durante o treinamento, cada árvore é construída utilizando uma amostra aleatória dos dados obtida por meio da técnica de *bootstrap sampling*. Para um conjunto de treinamento D com N instâncias, cada árvore T_k é treinada a partir de um subconjunto D selecionado aleatoriamente com reposição, de forma que aproximadamente um terço das instâncias originais não é incluído em cada subconjunto (as chamadas amostras *out-of-bag* [Breiman 2001]). Após o treinamento das árvores, a predição final do modelo para uma amostra $\mathbf{x}$ é obtida por meio da agregação das predições individuais das árvores, geralmente utilizando votação majoritária no caso de classificação, conforme a Equação 4, onde K representa o número total de árvores na floresta, produzindo por esta combinação um modelo robusto com alta capacidade preditiva [Breiman 2001].

$$\hat{y} = \text{mode}\{T_1(x), T_2(x), \ldots, T_K(x)\} \tag{4}$$

Outro aspecto importante do Random Forest é a seleção aleatória de um subconjunto de atributos em cada divisão da árvore. Onde em vez de avaliar todas as variáveis disponíveis, o algoritmo seleciona aleatoriamente m atributos dentre p atributos totais, sendo comum utilizar $m = \sqrt{p}$ em problemas de classificação. Dessa maneira, a estratégia reduz a correlação entre as árvores e aumenta a diversidade do *ensemble*, melhorando o desempenho preditivo [Hastie et al. 2009].

No contexto deste trabalho, na detecção de intrusões em redes IoT, o Random Forest é amplamente utilizado devido à sua capacidade de lidar com *datasets* de alta dimensionalidade e padrões complexos de tráfego. Diversas pesquisas científicas demonstram que esse algoritmo alcança elevados níveis de precisão e F1-score em tarefas de classificação de tráfego malicioso em redes IoT, sendo frequentemente adotado como modelo de referência *baseline* para comparação com métodos não supervisionados e modelos baseados em aprendizado profundo [Hussein et al. 2022, Ntayagabiri et al. 2025].

3. Trabalhos Relacionados

Diversos estudos recentes exploram o uso de técnicas de aprendizado de máquina para a detecção de anomalias em tráfego de rede, com ênfase em ambientes de IoT, nos quais o

elevado número de dispositivos conectados e a variedade de padrões de tráfego ampliam os desafios relacionados à segurança. Nesse contexto, diferentes abordagens têm sido propostas na literatura para identificar comportamentos maliciosos ou anômalos em redes, usando tanto métodos supervisionados quanto não supervisionados.

[Meidan et al. 2018] propõem um método de detecção de ataques em dispositivos IoT utilizando *deep autoencoders*. Nesse estudo, os autores utilizaram o *dataset* N-BaIoT, composto por tráfego de rede coletado de dispositivos IoT reais infectados com *malwares* como Mirai e BASHLITE, sendo o modelo treinado apenas com tráfego benigno e utilizou o erro de reconstrução para identificar comportamentos anômalos. Assim, os resultados demonstraram que *autoencoders* podem ser eficazes na detecção de ataques em dispositivos IoT, alcançando altos níveis de precisão na identificação de tráfego malicioso. De maneira análoga, o presente trabalho também utiliza *autoencoders* para detecção de anomalias em tráfego de rede, no entanto, realiza uma análise comparativa entre diferentes algoritmos de aprendizado de máquina, avaliando não apenas o desempenho preditivo, mas também o custo computacional dos modelos.

Dando continuidade à exploração de *autoencoders* para detecção de anomalias, [Ayad et al. 2024] apresentaram um modelo leve que combina um Autoencoder assimétrico empilhado com uma rede neural profunda, atingindo 96,27% de detecção em 0,27s no BoT-IoT, com acurácia de 99,99% e F1-score de 97,69%. Apesar dos resultados expressivos, o modelo híbrido impõe maior complexidade, contrastando com a abordagem deste estudo, que avalia modelos isolados (Autoencoder, Isolation Forest e Random Forest) para identificar claramente a contribuição de cada um, sem efeito de sinergia, facilitando comparação e replicação, alinhado ao objetivo de análise comparativa.

[Liu et al. 2012] propuseram o algoritmo Isolation Forest, um método de detecção de anomalias baseado na ideia de que observações raras tendem a ser isoladas mais rapidamente em árvores geradas aleatoriamente. Dessa maneira, o algoritmo utiliza o comprimento médio do caminho necessário para isolar uma observação como métrica de anomalia, sendo amplamente aplicado em tarefas de segurança de redes e detecção de fraudes. Com base nessa abordagem, [Obidiagha et al. 2025] propuseram o *framework* IforestExplain, que combina o Isolation Forest com técnicas de IA explicável (SHAP e LIME) para detecção de ataques DoS e DDoS em ambientes IoT, alcançando F1-score de até 0,94 e AUC superior a 0,99 no *dataset* CICIoT2023. A atual pesquisa, por sua vez, avalia o Isolation Forest em sua configuração padrão, sem oversampling sintético, e inclui métricas de tamanho de modelo e latência, aspectos fundamentais para aferir sua viabilidade em dispositivos de borda, indo além da análise preditiva e possibilitando comparações diretas com outras técnicas de aprendizado de máquina, o que permite discutir não só eficácia, mas também viabilidade prática de implantação.

[Hussein et al. 2022] indicaram um sistema de detecção de intrusões baseado em Random Forest com um método duplo de seleção de atributos aplicado ao *dataset* IoTID20, reduzindo a dimensionalidade dos dados e melhorando a precisão do modelo ao mesmo tempo em que diminui o custo computacional. De forma complementar, [Ntayagabiri et al. 2025] realizaram uma análise comparativa entre algoritmos supervisionados no CICIoT2023, na qual o Random Forest obteve o melhor desempenho entre os algoritmos avaliados (99,29% de acurácia) na detecção de tráfego malicioso. Nesta investigação científica, o Random Forest é utilizado como limite superior de desempenho

supervisionado, servindo como referência para comparação com modelos não supervisionados em cenários onde não há dados rotulados disponíveis.

Portanto, embora os trabalhos citados apresentem contribuições relevantes, poucos comparam desempenho preditivo e custo computacional dos modelos, aspecto crucial para viabilizar a implantação em *hardware* de borda com recursos restritos. A presente pesquisa preenche essa lacuna ao avaliar de forma sistemática os algoritmos Autoencoder, Isolation Forest e Random Forest sob ambas as perspectivas, utilizando os N-BaIoT e CICIoT2023 e métricas, contribuindo para o avanço de soluções de segurança mais leves, escaláveis e alinhadas aos princípios de privacidade desde a concepção.

4. Metodologia

4.1. Visão geral da *Pipeline*

A metodologia adotada, apresentada na Figura 1, foi organizada em quatro etapas principais. Na primeira etapa, são selecionados e caracterizados os dois *datasets* públicos utilizados no estudo. Na segunda e terceira, os dados passam por pré-processamento e particionamento em conjuntos de treino e teste. Na quarta, os modelos são treinados de acordo com seus respectivos paradigmas de aprendizado. Por fim, na quinta etapa, os modelos são avaliados tanto por métricas de classificação quanto por métricas de custo computacional, de modo a analisar sua viabilidade para *deployment* em dispositivos embarcados. As subseções a seguir aprofundam cada uma dessas etapas.

Os experimentos foram executados em um computador com processador AMD Ryzen 5 5600 (6 núcleos, 3,5 GHz), 16 GB de RAM e Windows 11, utilizando apenas CPU. Por ser um hardware significativamente mais potente que o alvo de *deployment*, as latências medidas devem ser interpretadas como limite inferior da latência esperada em produção, servindo como referência relativa entre os modelos sob condições idênticas.

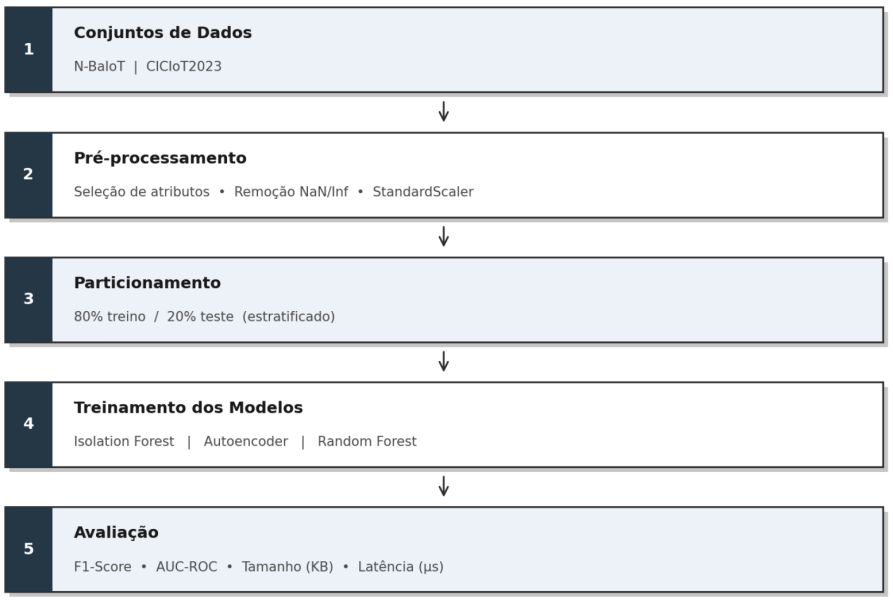

Figura 1. Pipeline para detecção de anomalias em redes IoT domésticas.

4.2. *Datasets*

O presente trabalho utiliza dois *datasets* públicos com altos números de referências na literatura de segurança em redes IoT.

O N-BaIoT [Meidan et al. 2018], que foi construído a partir do tráfego de rede capturado em nove dispositivos IoT domésticos reais, sendo eles câmeras de segurança (Provision PT-737E e PT-838, SimpleHome XCS7-1002 e XCS7-1003), uma *webcam* (Samsung SNH-1011-N), uma babá eletrônica (Philips B120N), um monitor de porta (Danmini Doorbell), um monitor de porta adicional (Ennio Doorbell) e um termostato inteligente (Ecobee). Cada dispositivo foi infectado em um ambiente controlado com dois *malwares*: o Mirai, responsável por ataques de negação de serviço distribuído (DDoS, do inglês *Distributed Denial of Service*) de grande escala [Antonakakis et al. 2017], e o BASHLITE (também conhecido como *Gafgyt*), voltado à formação de redes de dispositivos infectados (*botnets*) [Marzano et al. 2018]. As *features* de cada amostra correspondem a estatísticas de fluxo de rede extraídas em janelas de tempo (média, variância, magnitude, raio e correlação de pacotes), totalizando 115 atributos por amostra.

Já o CICIoT2023 [Neto et al. 2023] foi desenvolvido pelo Instituto Canadense de Cibersegurança (*Canadian Institute for Cybersecurity*) e contempla 33 tipos de ataque distintos, organizados em sete categorias: negação de serviço distribuído (DDoS), negação de serviço (DoS), reconhecimento, ataques baseados em *web*, força bruta, *spoofing* e Mirai. O *dataset* foi gerado em uma topologia controlada com 105 dispositivos IoT. Sua inclusão neste trabalho serve como conjunto de validação cruzada, modelos que apresentam bom desempenho em ambos os *datasets* demonstram maior capacidade de generalização frente a ataques heterogêneos.

Vale ressaltar que em ambos os datasets, as amostras foram definidas em duas classes: tráfego benigno (0) e tráfego de ataque (1). Para viabilidade computacional, foram amostradas até 50.000 instâncias de cada classe por *dataset*.

4.3. Pré-processamento

Para maximizar sua usabilidade, ambos os *datasets* foram submetidos às seguintes etapas de pré-processamento:

Seleção de atributos numéricos: Nesta etapa, algumas colunas não numéricas foram removidas antes do treinamento, posteriormente usadas como metadados de rastreabilidade.

Tratamento de valores ausentes ou infinitos: Durante a avaliação, foram identificados valores infinitos e negativos, sendo necessário suas substituições por NaN (*Not a Number*). Além disso colunas com mais de 30% de valores ausentes foram descartadas, seguindo a recomendação de [Hair et al. 2019], onde sugerem que atributos com perdas superiores a 20-30% podem comprometer a integridade estatística da amostra (as demais colunas tiveram seus valores ausentes preenchidos com a mediana da coluna correspondente, conforme o necessário).

Remoção de *features* com variância zero: Atributos constantes (aqueles que não carregam informação discriminante) foram removidos.

Normalização: Esta etapa foi realizada utilizando o StandardScaler (ou padronização *z-score*), uma técnica de pré-processamento de dados que subtrai a média e

divide pelo desvio padrão de cada *feature*, assim produzindo distribuições com média zero e variância unitária, com o objetivo de facilitar a convergência dos algoritmos de aprendizado [scikit-learn developers 2024]. Para evitar o vazamento de dados (*data leakage*), o ajuste (*fit*) do *scaler* foi realizado exclusivamente sobre o conjunto de treinamento, conforme definido por [Hastie et al. 2009], garantindo que a avaliação no conjunto de teste reflita o desempenho do modelo em dados não vistos anteriormente.

Por fim, a divisão dos dados, onde 80% foram destinados ao treino e 20% ao teste, uma abordagem amplamente adotada na literatura para garantir um volume de dados suficiente para o ajuste dos parâmetros e uma avaliação estatisticamente significativa [Gholamy et al. 2018]. Para evitar riscos associados ao desbalanceamento de classes, comum em *datasets* de tráfego de rede, aplicou-se a amostragem estratificada, assegurando que a proporção original de amostras benignas e de ataque seja preservada em ambas as partições, evitando vieses na estimativa de desempenho do modelo [Hastie et al. 2009].

4.4. Estratégia de Avaliação

Para a análise do desempenho e viabilidade técnica, os modelos foram avaliados sob as seguintes métricas:

F1-Score: Uma das principais métricas de performance preditiva por ser a média harmônica entre precisão e revocação, sendo superior a acurácia em cenários de detecção de anomalias, onde o desbalanceamento de classes tornaria a acurácia uma métrica excessivamente otimista [Sokolova and Lapalme 2009].

AUC-ROC: Empregado para medir a capacidade discriminativa dos modelos independente do limiar (*threshold*) de decisão, permitindo comparar de forma equivalente modelos supervisionados e não supervisionados, onde saídas variam em escalas distintas [Fawcett 2006].

Métricas de Recursos Computacionais: Com o objetivo de realizar o *deployment* em *hardware* embarcado, foram medidos:

- Tamanho do modelo serializado (KB).
- Latência total de inferência (ms).
- Tempo médio de inferência por amostra (μs)
  As medições de latência seguiram um protocolo de *benchmarking* rigoroso, descartando a primeira execução (*warmup*) para evitar o efeito de *cold start* e calculou-se a média de cinco execuções consecutivas para assegurar estabilidade estatística dos tempos reportados.

A avaliação foi conduzida independentemente em cada *dataset*, permitindo comparar tanto desempenho absoluto quanto a capacidade de generalização entre vários cenários de ataque.

Embora os experimentos tenham sido conduzidos em hardware de propósito geral, os valores de tamanho de modelo e latência obtidos permitem estimar a compatibilidade com dispositivos embarcados. Tomando como referência o Raspberry Pi 3 Model B+, amplamente adotado na literatura como gateway de borda para redes IoT domésticas [Asulba et al. 2024, Tuncer et al. 2021], e equipado com processador ARM Cortex-A53 quad-core a 1,4 GHz e 1 GB de RAM, os modelos avaliados situam-se bem abaixo dos limites de armazenamento dessa plataforma. Quanto à latência, os valores de inferência por

amostra medidos neste trabalho devem ser interpretados como limite inferior da latência esperada em produção, dado o menor poder de processamento de plataformas ARM embarcadas em relação a CPUs x86 de propósito geral. Ainda assim, mesmo considerando essa diferença de desempenho, os modelos mais leves avaliados permanecem em faixas compatíveis com análise em tempo real de tráfego em redes domésticas IoT.

5. Resultados

5.1. Desempenho da Classificação

A tabela 1 mostra os valores de F1-Score e de AUC-ROC obtidos pelos três modelos treinados pelos *datasets* anteriormente avaliados.

Tabela 1. Métricas de classificação por modelo e dataset.

Modelo	F1 N-BaIoT	AUC N-BaIoT	F1 CICIoT2023	AUC CICIoT2023
Isolation Forest	0,970	0,938	0,827	0,949
Autoencoder	0,880	0,965	0,967	0,993
Random Forest*	**1,000**	**1,000**	**0,991**	**0,998**

*Modelo supervisionado utilizado como referência superior.
VN = Verdadeiro Negativo; FP = Falso Positivo; FN = Falso Negativo; VP = Verdadeiro Positivo.

Dentre eles os modelos treinados, o Random Forest apresentou os melhores resultados. No N-BaIoT teve F1 de 1,000 e AUC-ROC de 1,000, um desempenho perfeito confirmado pela matriz de confusão, onde não teve nenhum erro de classificação. Já no CICIoT2023, o modelo manteve desempenho quase perfeito, apresentando um F1 de 0,991 e AUC-ROC de 0,998, registrando 6 falsos positivos e 174 falsos negativos. Porém, estes resultados muito precisos eram esperados aqui por se tratar de um método de aprendizado de máquina supervisionado, ou seja, há acesso a exemplos rotulados de ambas as classes durante o processo, assim tendo uma vantagem direta sobre os modelos não supervisionados. Contudo, essa exigência da existência de um *dataset* de ataques rotulados, é uma condição que não pode ser garantida em uma rede doméstica.

Entre os modelos não supervisionados, houve um comportamento diferente do Autoencoder entre os *datasets*. No N-BaIoT, ele obteve um F1 = 0,880 e AUC = 0,965, um resultado abaixo do Isolation Forest (IF) que obteve um F1 de 0,970, porém em AUC o IF teve um desempenho abaixo em AUC atingindo 0,938. Esse padrão indica que o Autoencoder tem maior capacidade de separação linear entre classes, embora seu *threshold* fixo no *percentil* 95 sacrifique revocação em favor de precisão neste dataset. Já no CICIoT2023 os papeis se invertem, houve uma melhoria significativa no Autoencoder, que alcançou um F1 de 0,967 e AUC de 0,993, superando o IF que obteve um F1 de 0,827 e um AUC de 0,949 nas métricas. A diversidade de ataques neste *dataset* (33 tipos) parece favorecer o aprendizado de representações compactas do tráfego benigno pelo Autoencoder, tornando anomalias mais distinguíveis pelo erro de reconstrução.

A figura 2 mostra a comparação dos desempenhos de F1 e AUC-ROC do N-BaIoT.

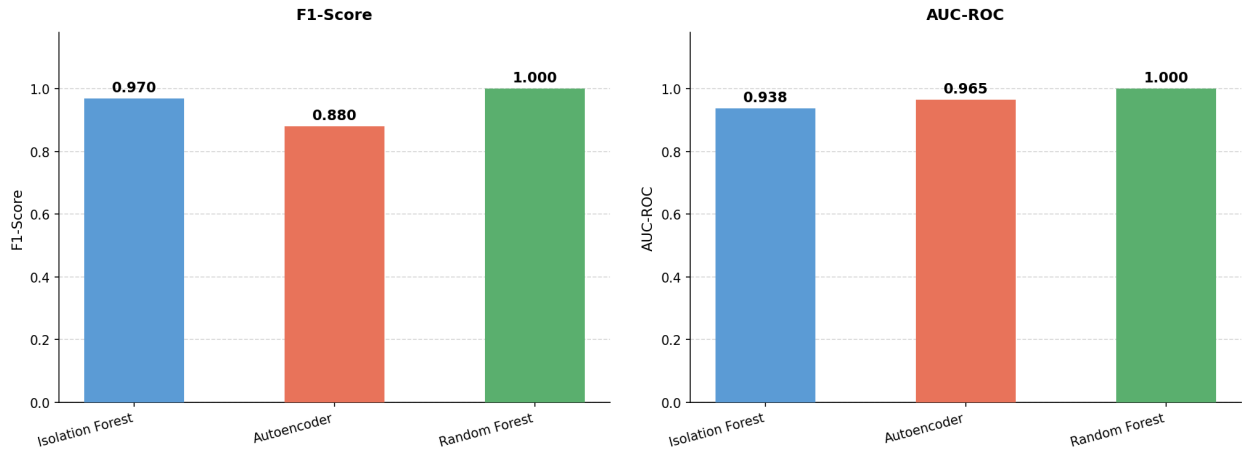

Figura 2. Comparação de Modelos N-BaloT.

Ja a figura 3 mostra a comparação dos desempenhos de F1 e AUC-ROC do CICIoT2023.

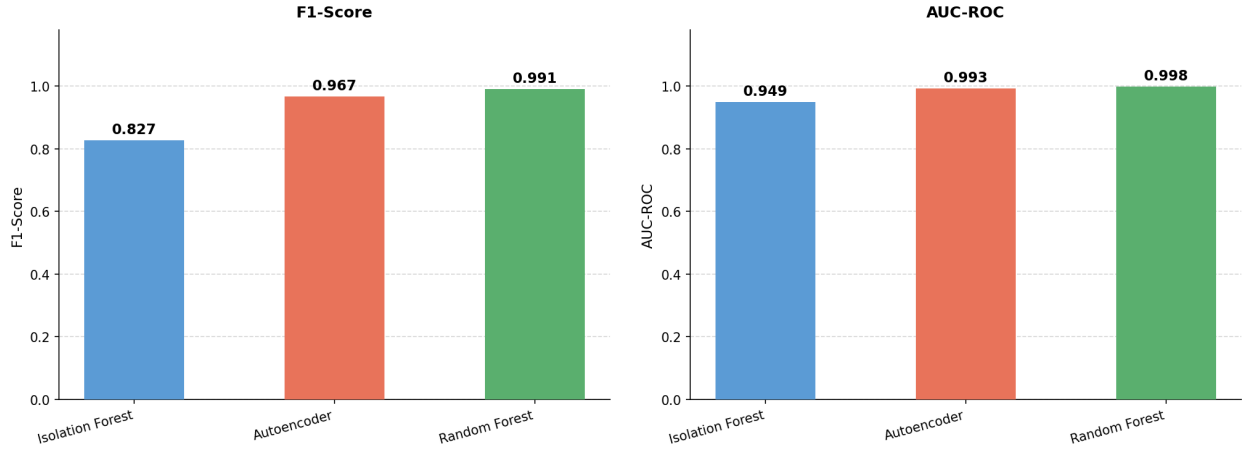

Figura 3. Comparação de Modelos CICIoT2023.

Por fim, a tabela 2 detalha as matrizes de confusão resumidas, para complementar a análise de F1 e AUC.

Tabela 2. Matrizes de confusão resumidas.

Modelo / Dataset	VN	FP	FN	VP	Precisão
Isolation Forest / N-BaIoT	4858	542	54	9546	94,6%
AutoEncoder / N-BaIoT	5148	252	1865	7735	96,8%
RandomForest / N-BaIoT*	5400	**0**	**0**	9600	100%
IsolationForest / CICIoT2023	8965	1035	2214	7786	88,3%
AutoEncoder / CICIoT2023	9494	506	175	9825	95,1%
RandomForest / CICIoT2023*	**9994**	6	174	**9826**	**99,9%**

*VN = Verdadeiro Negativo; FP = Falso Positivo; FN = Falso Negativo; VP = Verdadeiro Positivo.

*Random Forest supervisionado (referência). Latências medidas em CPU.

Ao analisar a tabela, é possível notar que o Isolation Forest demonstrou bons resultados no N-BaIoT, mas seu desempenho caiu no CICIoT2023, devido ao isolamento por cortes aleatórios. O Autoencoder oferece um melhor balanceamento entre a detecção e eficiência, atingindo apenas 175 falsos negativos, mesmo sendo um modelo não supervisionado. Por fim, o Random Forest confirma um desempenho superior por ser um modelo supervisionado, atingindo excelentes pontuações de F1 nos dois *datasets*, com poucos ou nenhum falso negativo e positivo, porém, em redes domésticas reais o tráfego malicioso rotulado raramente estará disponível de antemão.

5.2. Análise dos Erros de Reconstrução

O *threshold* do Autoencoder é definido como o *percentil* 95 do MSE de reconstrução no conjunto de validação *benign*. Os valores obtidos foram 0,3531 para o N-BaIoT e 0,0220 para o CICIoT2023. Essa diferença de magnitude reflete a escala das *features* após normalização em cada *dataset*.

A separabilidade entre as distribuições de erro de amostras benignas e de ataque é mais pronunciada no CICIoT2023, onde os erros de ataque concentram-se em faixas muito superiores ao *threshold*, explicando o alto F1 obtido. No N-BaIoT, há maior sobreposição na cauda superior da distribuição *benign* com ataques de baixa intensidade, resultando nos 1865 falsos negativos observados na matriz de confusão.

Esse comportamento é esperado em *autoencoders* treinados exclusivamente com tráfego normal, o modelo aprende a reconstruir com fidelidade o padrão dominante do conjunto de treino. Os desvios sutis (como ataques que exploram variações de baixa amplitude) podem acabar não gerando erro de reconstrução suficiente para ultrapassar o *threshold*.

5.3. Custo computacional e Escolha do modelo

A tabela 3 mostra as métricas de recursos dos três modelos, onde temos o tamanho em disco, a latência medida sobre o conjunto de testes completo, e µs/amostra (latência normalizada por amostra).

Tabela 3. Recursos computacionais por modelo e *dataset*.

Modelo	*Dataset*	Tamanho (KB)	Latência (ms)	μs/amostra	Tipo
Isolation Forest	N-BaIoT	753,9	49,45	3,297	Não-sup.
Isolation Forest	CICIoT2023	824,7	69,79	3,490	Não-sup.
Autoencoder	N-BaIoT	78,8	6,39	0,426	Não-sup.
Autoencoder	CICIoT2023	**40,5**	**3,54**	0,177	Não-sup.
Random Forest*	N-BaIoT	308,6	29,23	1,949	Superv.
Random Forest*	CICIoT2023	4234,4	37,40	1,870	Superv.

*Random Forest supervisionado (referência). Latências medidas em CPU.

Como podemos analisar na tabela, o Autoencoder se destaca por ser mais eficiente em todas as dimensões, ou seja, ocupa o menor espaço em disco (40,5kb no CICIoT2023 e 78,8 kb no N-BaIoT) e realiza inferência em 0,177µs/amostra no CICIoT2023

e 0,426µs/amostra no N-BaIoT, sendo assim aproximadamente 8x mais rápido por amostra, e 4-7x menor em tamanho, comparado ao Isolation Forest.

Já o Random Forest apresenta um crescimento em espaço de disco significativo comparado ao Autoencoder, principalmente no CICIoT2023. Embora sua latência por amostra (1,870µs-1,949µs) seja inferior ao IF, o tamanho de memória maior inviabiliza o uso em cenários que visam o menor custo computacional possível.

Em síntese, os resultados indicam que o Autoencoder é o modelo mais adequado ao cenário de detecção de anomalias em redes domésticas com agentes leves, operando rótulos de ataque, e resiliente em *datasets* heterogêneos como o CICIoT2023, e consome poucos recursos computacionais, com latência satisfatória para uma possível análise em tempo real.

6. Conclusão e Trabalhos Futuros

O presente trabalho avaliou a viabilidade de agentes leves de aprendizado de máquina para detecção de anomalias em redes domésticas com dispositivos IoT, cenário caracterizado pela ausência de tráfego malicioso rotulado para treinamento. Foram avaliados três modelos: Isolation Forest, Autoencoder e Random Forest, em dois *datasets* públicos, o N-BaIoT e o CICIoT2023. Adicionalmente, experimentos complementares investigaram a viabilidade de integração com extratores de features em tempo real, evidenciando desafios práticos relevantes para o *deployment* em redes reais. Os experimentos demonstraram que o Autoencoder não supervisionado oferece o melhor equilíbrio entre desempenho de detecção e eficiência computacional no cenário proposto. No CICIoT2023 o modelo alcançou F1 = 0,967 e AUC = 0,993, com apenas 175 falsos negativos em 10.000 amostras de teste. Esse resultado é praticamente equivalente ao do Random Forest supervisionado (174 falsos negativos, F1 = 0,991), que requer rótulos de ataque durante o treinamento. Do ponto de vista computacional, o Autoencoder ocupa 40,5 KB no CICIoT2023 e realiza inferência em 0,177 µs por amostra. Esses valores são compatíveis com execução em dispositivos embarcados de médio porte.

O principal trabalho futuro é o retreinamento do Autoencoder com tráfego IoT real capturado via Zeek [The Zeek Project 2026] diretamente no ambiente de *deployment*. O procedimento envolve, primeiramente, um período de quarentena com monitoramento da rede doméstica em paralelo com um IDS baseado em assinaturas para validação, após isso, uma geração de um *dataset* local com as 26 *features* mapeadas, e por fim, o retreino e validação do Autoencoder neste *dataset*. Esse pipeline constituiria uma prova de conceito completa de *deployment* em rede real. Uma segunda direção relevante é o ajuste dinâmico do *threshold*. Estratégias baseadas em janelas deslizantes de tráfego recente, por exemplo, permitiriam que o Autoencoder se adaptasse a mudanças no padrão de tráfego benigno sem necessidade de retreinamento completo.

Em síntese, este trabalho demonstra que é possível detectar anomalias de rede em dispositivos IoT domésticos com modelos não supervisionados sem exigir tráfego malicioso rotulado no treinamento. O desafio remanescente, o (*domain shift* entre *datasets* públicos e tráfego IoT real), constitui a fronteira natural entre este trabalho e sua aplicação em produção, delineando-se como uma pesquisa futura.

Referências

Adhikari, D., Jiang, W., Zhan, J., Rawat, D. B., and Bhattarai, A. (2024). Recent advances in anomaly detection in Internet of Things: Status, challenges, and perspectives. *Computer Science Review*, 54:100665.

Alaghbari, K. A., Lim, H.-S., Saad, M. H. M., and Yong, Y. S. (2023). Deep autoencoder-based integrated model for anomaly detection and efficient feature extraction in IoT networks. *IoT*, 4(3):345–365.

Alotaibi, B. (2024). An efficient flow-based anomaly detection system for enhanced security in iot networks. *Sensors*, 24(22):7408.

Antonakakis, M., April, T., Bailey, M., Bernhard, M., Bursztein, E., Cochran, J., Durumeric, Z., Halderman, J. A., Invernizzi, L., Kallitsis, M., Kumar, D., Lever, C., Ma, Z., Mason, J., Menscher, D., Seaman, C., Sullivan, N., Thomas, K., and Zhou, Y. (2017). Understanding the Mirai botnet. In *26th USENIX Security Symposium (USENIX Security 17)*, pages 1093–1110, Vancouver, BC. USENIX Association.

Asulba, B., Souto, P. F., and Almeida, L. (2024). Bringing IoT intrusion detection to the edge. In *Proceedings of the 8th International Conference on Future Networks and Distributed Systems (ICFNDS '24)*, Marrakech, Morocco. ACM.

Ayad, A. G., El-Gayar, M. M., Hikal, N. A., and Sakr, N. A. (2024). Efficient real-time anomaly detection in iot networks using one-class autoencoder and deep neural network. *Electronics*, 14(1).

Breiman, L. (2001). Random forests. *Machine learning*, 45(1):5–32.

Comitê Gestor da Internet no Brasil (2025). Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros: Tic domicílios 2025. Pesquisa realizada pelo Centro Regional de Estudos para o Desenvolvimento da Sociedade da Informação (Cetic.br).

Fawcett, T. (2006). An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874.

Gholamy, A., Kreinovich, V., and Kosheleva, O. (2018). Why 70/30 or 80/20 relation between training and testing sets: A pedagogical explanation. Technical Report 1209, Departmental Technical Reports (CS), University of Texas at El Paso.

Hair, J. F., Black, W. C., Babin, B. J., and Anderson, R. E. (2019). *Multivariate Data Analysis*. Cengage, United Kingdom, 8th edition.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). *The Elements of Statistical Learning*. Springer, 2 edition.

Hussein, A. Y., Falcarin, P., and Sadiq, A. T. (2022). Iot intrusion detection using modified random forest based on double feature selection methods. In *Emerging Technology Trends in Internet of Things and Computing*, pages 61–78, Cham. Springer International Publishing.

Javed, A., Ehtsham, A., Jawad, M., Awais, M. N., Qureshi, A. H., and Larijani, H. (2024). Implementation of lightweight machine learning-based intrusion detection system on IoT devices of smart homes. *Future Internet*, 16(6):200.

Khraisat, A., Gondal, I., Vamplew, P., and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):20.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2008). Isolation forest. In *2008 Eighth IEEE International Conference on Data Mining*, pages 413–422. IEEE.

Liu, F. T., Ting, K. M., and Zhou, Z.-H. (2012). Isolation-based anomaly detection. *ACM Trans. Knowl. Discov. Data*, 6(1).

Marzano, A., Alexander, D., Fazzion, E., Fonseca, O., Ítalo Cunha, Hoepers, C., Steding-Jessen, K., Chaves, M. H. P. C., Guedes, D., and Jr., W. M. (2018). Monitoramento e caracterização de botnets bashlite em dispositivos iot. In *Anais do XXXVI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos (SBRC 2018)*, Vitória, ES, Brasil. Sociedade Brasileira de Computação - SBC.

Meidan, Y., Nolane, M., Doitshman, Y., Yashkaniv, S., Bakalo, R., Breitenbacher, D., Shabtai, A., and Elovici, Y. (2018). N-baiot—network-based detection of iot botnet attacks using deep autoencoders. *IEEE Pervasive Computing*, 17(3):12–22.

Neto, E. C. P., Dadkhah, S., Ferreira, R., Zohourian, A., Lu, R., and Ghorbani, A. A. (2023). Ciciot2023: A real-time dataset and benchmark for large-scale attacks in iot environment. *Sensors*, 23(13).

Ntayagabiri, J. P., Bentaleb, Y., Ndikumagenge, J., and El Makhtoum, H. (2025). A comparative analysis of supervised machine learning algorithms for iot attack detection and classification. *Journal of Computing Theories and Applications*, 2(3):312–326.

Obidiagha, C. C., Rahouti, M., Drid, H., Hamouid, K., Medjadba, Y., and Jagatheesaperumal, S. K. (2025). Iforestexplain: a dual-stage isolation forest framework with explainable ai for dos/ddos anomaly detection in iot networks. *Cluster Computing*, 28(10):668.

Rafique, S. H., Abdallah, A., Musa, N. S., and Murugan, T. (2024). Machine learning and deep learning techniques for Internet of Things network anomaly detection—current research trends. *Sensors*, 24(6):1968.

scikit-learn developers (2024). StandardScaler – scikit-learn documentation. Acesso em: fev. 2026.

Sokolova, M. and Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437.

The Zeek Project (2026). Zeek network security monitor. Accessed: 2026-03-14.

Torabi, H., Mirtaheri, S. L., and Greco, S. (2023). Practical autoencoder based anomaly detection by using vector reconstruction error. *Cybersecurity*, 6(1):1.

Tuncer, T., Dogan, S., and Acharya, U. R. (2021). Analysis of machine learning algorithms for anomaly detection on edge devices. *Sensors*, 21(14):4946.