Abordagem em duas fases para detecção e classificação de ataques em redes IoT com computação em névoa

Matheus Antony Souza Pereira[1], **Vitor Barbosa Souza**[1], **Daniel Louzada Fernandes**[1]

[1]Departamento de Informática (DPI) – Centro de Ciências Exatas (CCE)
Universidade Federal de Viçosa (UFV) – Viçosa – MG – Brazil

{matheus.antony,vitor.souza,daniel.louzada}@ufv.br

Abstract. *The current Internet comprises millions of IoT devices, such as cameras, sensors, and smart light bulbs, characterized by limited processing power and a reduced emphasis on security, which makes them potential entry points for large-scale cyberattacks. Moreover, these inherent constraints prevent such devices from running complex machine learning models, which are increasingly used in IDS/IPS systems. This work adopts a fog computing architecture in which more capable intermediary nodes monitor network traffic to detect attacks targeting an IoT network. To this end, in a first phase, the traffic is grouped using unsupervised techniques to form clusters that separate benign traffic from distinct attack types. In a second phase, a supervised model is used to predict the labels of the clusters formed in the previous phase. The results indicate that the proposed approach can capture coherent and statistically significant attack patterns at the network edge.*

Resumo. *A internet atual abrange milhões de dispositivos IoT, como câmeras, sensores e lâmpadas inteligentes, caracterizados pelo baixo poder de processamento e limitado foco em segurança, o que os torna potenciais vetores para ataques cibernéticos de larga escala. Além disso, essas limitações intrínsecas impedem que os mesmos executem complexos modelos de aprendizado de máquina, cada vez mais utilizados em IDS/IPS. Este trabalho adota uma arquitetura de computação em névoa, na qual nós intermediários mais robustos monitoram o tráfego de rede para detectar ataques à rede IoT. Para isso, em uma primeira fase, o tráfego é agrupado por meio de técnicas não supervisionadas, de forma a criar clusters, segregando tráfego normal e diferentes tipos de ataques. Em uma segunda fase, um modelo supervisionado é utilizado para predizer os rótulos de cada um dos clusters formados na fase anterior. Os resultados indicam que a abordagem proposta é capaz de capturar padrões de ataque coerentes e estatisticamente significativos na borda da rede.*

1. Introdução

A segurança em redes tornou-se ainda mais desafiadora com a expansão da Internet das Coisas (IoT, do inglês *Internet of Things*), na qual milhões de dispositivos simples (sensores, câmeras, atuadores e equipamentos domésticos) passam a se comunicar continuamente pela Internet, muitas vezes sem terem sido projetados com foco em segurança e com recursos computacionais limitados [Chen et al. 2018]. Nesse cenário, a computação de névoa (*fog computing*) surge como alternativa para deslocar parte do processamento

para nós intermediários mais robustos (roteadores, *gateways* e servidores próximos à rede), capazes de monitorar o tráfego e apoiar a detecção de comportamento malicioso [Mohamed and Ismael 2023].

Com a computação de névoa, técnicas de aprendizado de máquina podem ser executadas fora dos dispositivos finais utilizando registros de fluxos de rede, incluindo dados como endereços IP, portas de origem/destino, protocolos, serviços e estatísticas de bytes e pacotes, para identificar e prever ataques de forma escalável e com baixa latência [Prazeres et al. 2023]. Como exemplo, técnicas de aprendizado não supervisionado, em especial as baseadas em *clustering*, podem ajudar a descobrir "padrões típicos" de tráfego que correspondam a ataques ou anomalias, mesmo sem informação prévia explícita [Perumal et al. 2025]. Desta forma, a arquitetura proposta neste trabalho é mostrada na Figura 1.

Este trabalho investiga se, a partir de um conjunto de dados de tráfego de rede com múltiplos tipos de ataque, é possível utilizar apenas os atributos de conexão, sem recorrer aos rótulos, para obter agrupamentos alinhados tanto à separação entre tráfego normal e ataque quanto à distinção entre diferentes tipos de ataque. Essa investigação se justifica pela necessidade de lidar com cenários sem rótulos, nos quais a rotulagem manual é cara e inviável, e pela importância de verificar se os rótulos existentes correspondem a "ilhas naturais" no espaço de atributos ou refletem convenções do ambiente de coleta, como endereçamento IP, portas e serviços. Assim, técnicas de *clustering* podem reduzir o esforço de rotulagem e gerar *insights* sobre a qualidade do *dataset* e a robustez das categorias rotuladas.

Este trabalho deve ser entendido, contudo, como uma investigação experimental de natureza do potencial dessa abordagem, servindo de base para trabalhos futuros envolvendo *edge intelligence* aplicada à segurança transparente em cenários IoT e ambientes ubíquos e pervasivos, permitindo detecção local com baixa latência e operação contínua mesmo sob conectividade limitada. Desta forma, sem a pretensão de ser uma solução consolidada em ambiente operacional.

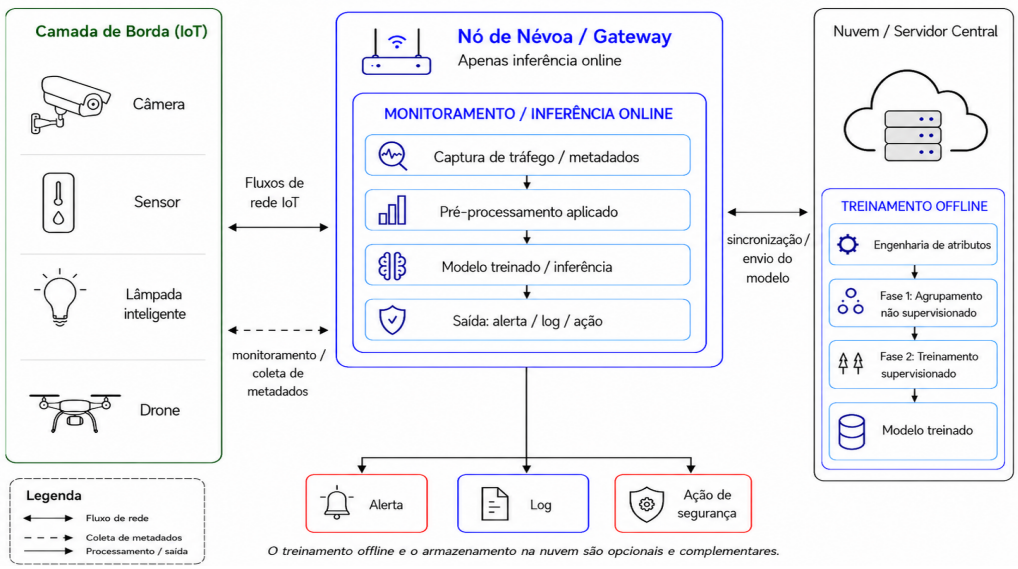


Figura 1. Arquitetura para detecção e classificação de ataques em redes IoT/WiFi.

2. Trabalhos Relacionados

O uso de *clustering* em sistemas de detecção de intrusão é relativamente consolidado na literatura. Em um *survey* sobre o tema, [Bohara et al. 2020] destacam a forte dependência de bases clássicas, como o *KDDCup99*[1], e a recorrência do *K*-Means como técnica amplamente empregada. De forma complementar, [Rahman et al. 2025] mostram que, em redes IoT, a literatura recente tem avançado para abordagens de aprendizado de máquina e aprendizado profundo voltadas ao tratamento de tráfego heterogêneo, classes desbalanceadas e restrições de custo computacional.

No contexto de abordagens que combinam aprendizado não supervisionado e supervisionado, [Qaddoura et al. 2021] propuseram um *pipeline* para IDS em IoT que combina *k*-means++, *oversampling* e classificação com SLFN, utilizando o agrupamento como etapa intermediária para melhorar o classificador final. De forma semelhante, [Perumal et al. 2025] propuseram o CAEP (*Cluster Autoencoder Pair*), que combina *clustering* e *autoencoders* para detecção de anomalias a partir do erro de reconstrução. Em ambos os casos, o foco recai sobre a melhoria do desempenho preditivo ou da detecção de anomalias, e não sobre o alinhamento entre os agrupamentos obtidos e as classes reais.

No cenário específico de IoT e computação em névoa, parte da literatura tem enfatizado arquiteturas distribuídas e o uso de atributos de fluxo de rede para deslocar o processamento para nós mais robustos, próximos à borda, como [Mohamed and Ismael 2023]. Nesse contexto, também ganham espaço abordagens que combinam processamento distribuído e modelos híbridos para lidar com a complexidade do tráfego em ambientes IoT. Por exemplo, [Salehiyan et al. 2025] investigam uma arquitetura Transformer–GAN–AE para detecção de intrusão em sistemas Edge/IIoT. Em comum, essas propostas reforçam a importância de soluções distribuídas para ambientes IoT, mas tendem a priorizar a maximização do desempenho preditivo por meio de modelos mais sofisticados e, em geral, mais custosos computacionalmente.

Em contraste com esses trabalhos, este artigo não busca propor uma nova arquitetura profunda, mas investigar em que medida agrupamentos obtidos sem acesso aos rótulos se alinham à separação entre tráfego normal e ataque e às diferentes famílias de ataque em um cenário multiclasse, bem como o potencial de utilizar esses agrupamentos em modelos mais simples e leves, ainda que de modo experimental. Para isso, adota-se o *dataset* TON_IoT, introduzido por [Moustafa 2021], derivado de uma arquitetura com camadas borda–névoa–nuvem e concebido para representar padrões legítimos e maliciosos mais aderentes a cenários IoT do que bases clássicas amplamente utilizadas. Assim, a ênfase recai sobre a análise do alinhamento entre *clusters* e classes reais, sob avaliação *group-wise*, com uso de atributos de fluxo e modelos de menor complexidade relativa.

3. Materiais e Métodos

O conjunto de dados utilizado neste trabalho foi o *TON_IoT Network Dataset*[2], criado por [Moustafa 2021]. Segundo o autor, os dados foram coletados em um *testbed* que simula interações entre as camadas borda–névoa–nuvem, contemplando cenários simultâneos de tráfego benigno e de ataque.

[1] `https://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html`
[2] `https://research.unsw.edu.au/projects/toniot-datasets`

Neste trabalho, foi utilizado um recorte fornecido pelo criador do dataset contendo 211 043 instâncias, em que cada instância corresponde a uma sessão completa de comunicação entre cliente e servidor. Os metadados extraídos pelo Zeek podem ser organizados em quatro grupos: 12 atributos de conexão, 4 atributos estatísticos, 25 atributos centrados em serviço e 3 atributos de violação/anomalia [Moustafa 2021]. Esses atributos incluem variáveis categóricas, como endereços IP, portas, protocolo de transporte, serviço e estado da conexão, e variáveis numéricas, como duração da sessão, número de pacotes, volume de bytes e total de bytes de cabeçalhos IP em cada direção.

O processo de rotulagem utilizou correspondência com o *ground truth* do *testbed* [Moustafa 2021]. Assim, o conjunto possui dois atributos de rótulo: *label* (binário), que indica tráfego normal (0) ou ataque (1), e *type* (string), que diferencia tráfego normal e nove famílias de ataque (*backdoor*, *ddos*, *dos*, *injection*, *mitm*, *password*, *ransomware*, *scanning* e *xss*). A distribuição dessas classes é mostrada na Tabela 1, evidenciando um *dataset* multiclasse e desbalanceado, com 161 043 instâncias de ataque (76,31%) e 50 000 de tráfego normal.

Tabela 1. Distribuição de classes do atributo `type` no recorte utilizado

Rótulo	Nº amostras	%	Descrição breve
normal	50 000	23,69	Fluxo legítimo / comportamento normal
dos	20 000	9,48	Negação de serviço (DoS)
ddos	20 000	9,48	Negação de serviço distribuída (DDoS)
backdoor	20 000	9,48	Acesso não autorizado persistente
injection	20 000	9,48	Injeção de código (ex.: SQL *Injection*)
password	20 000	9,48	Ataques a credenciais / tentativas de senha
ransomware	20 000	9,48	Sequestro/criptografia de dados
scanning	20 000	9,48	Varredura / descoberta de portas e serviços
xss	20 000	9,48	*Cross-Site Scripting* (XSS)
mitm	1 043	0,49	Interceptação (*Man-in-the-Middle*)

3.1. Metodologia de separação do conjunto de dados

Em tráfego de rede, conexões muito semelhantes podem se repetir com pequenas variações. Quando amostras quase idênticas aparecem simultaneamente em treino e teste, o modelo pode aparentar desempenho elevado por vazamento de dados (*data leakage*), sem necessariamente demonstrar capacidade real de generalização. Para reduzir esse risco, adotou-se uma divisão *group-wise*, na qual grupos não se repetem entre treino, validação e teste. O identificador de grupo foi construído pela agregação dos atributos endereço IP de origem, endereço IP de destino, serviço e protocolo de transporte, na forma `src_ip|dst_ip|service|proto`. As portas não foram incluídas porque, especialmente no caso das portas de origem, diferentes valores podem ocorrer dentro de um mesmo evento de interesse, o que poderia fragmentar artificialmente amostras semelhantes.

A utilização dos endereços IP foi tratada com cautela. Embora muitos fluxos compartilhem os mesmos pares de origem e destino, o que poderia induzir memorização de identificadores específicos do ambiente de coleta, remover completamente essa informação descartaria sinais relevantes para ataques, como concentração anormal de conexões e comunicação repetida entre pares. Por isso, os IPs foram usados na definição dos grupos e na construção de atributos agregados de conectividade, mas não foram mantidos como identificadores categóricos livres na matriz final de agrupamento.

A separação dos *splits* foi realizada com validação cruzada estratificada por grupos

(SGKF, do inglês *Stratified Group K-Fold*), utilizando estratificação pelo atributo `type` para preservar, em cada partição, a proporção original das 10 classes do conjunto de dados e, ao mesmo tempo, garantir que todas as amostras de um mesmo grupo permanecessem no mesmo *fold*. Inicialmente, definiu-se um *fold* externo, utilizado como conjunto de teste por grupos, de modo que cada grupo pertencesse exclusivamente ao conjunto de teste ou ao conjunto de treino e validação. Em seguida, dentro do conjunto de treino e validação, aplicou-se novamente a SGKF para obtenção dos *folds* internos de validação por grupos. Embora o particionamento tenha sido realizado com auxílio da SGKF e o *dataset* seja suficientemente grande, utilizou-se apenas um *split* interno fixo para validação. Dessa forma, os dados foram divididos em conjuntos de treino, validação e teste sem interseção de grupos entre as partições, sendo o *fold* externo reservado à avaliação final e os *folds* internos empregados no ajuste de hiperparâmetros e na seleção de modelos. No recorte adotado, os conjuntos resultantes possuem 183 158 instâncias de treino (86,79%), 16 901 de validação (8,01%) e 10 984 de teste (5,20%).

3.2. Pré-processamento e preparação dos dados

Para se obter melhores resultados na fase de treinamento, estratégias foram utilizadas para a preparação dos dados. Em primeiro lugar, para simular o cenário sem rótulos, as colunas `label` e `type` foram removidas antes do *clustering*. Além disso, como detalhado a seguir, o *pipeline* foi estruturado para lidar com dados dos tipos numéricos e categóricos, alta dimensionalidade após a codificação e as diferenças de escala.

Para capturar padrões operacionais de rede relevantes a ataques, definiu-se uma engenharia de atributos que cria indicadores de portas de origem e destino bem conhecidas para portas < 1024, categorias top-N para as portas mais frequentes no conjunto de treino (aplicadas separadamente a origem e destino, com as demais agrupadas como `other`), contagens agregadas de conectividade por par, por origem e por destino, com versões transformadas logaritmicamente para estabilizar a escala, e remove endereços IP isolados após agregação, reduzindo o risco de "memorização" dos identificadores na etapa de agrupamento. Para variáveis categóricas, aplicou-se a técnica de codificação *One-Hot Encoding* (OHE) com os parâmetros `handle_unknown="ignore"` e `min_frequency=10`, com o objetivo de reduzir a explosão de categorias raras. Para variáveis numéricas, aplicou-se normalização (e, quando aplicável, transformação logarítmica em contagens).

Após a codificação OHE, a matriz resultado pode apresentar uma alta dimensionalidade e um elevado grau de esparsidade. Sendo assim, utilizou-se o método baseado na decomposição de valores singulares, denominado Truncated SVD, para projeção em um espaço de baixa dimensão (50 componentes) antes da aplicação de um algoritmo de agrupamento (*clustering*). Com isso, foi possível reduzir o custo computacional, melhorar a estabilidade do agrupamento, e facilitar tanto a comparação quanto a visualização dos *embeddings* (representações vetoriais densas em um espaço de baixa dimensionalidade).

3.3. Algoritmos de aprendizado de máquina

Para a etapa não supervisionada, foram considerados os algoritmos Mini-Batch *K*-Means, DBSCAN e Hierárquico (Agglomerative). O Mini-Batch *K*-Means [Sculley 2010] foi empregado a partir da definição prévia de K, com avaliação de diferentes valores para esse parâmetro; o DBSCAN foi utilizado como alternativa baseada em densidade, por

meio do ajuste de `eps` e `min_samples`; e o método Hierárquico foi testado com diferentes critérios de *linkage*. Entretanto, a análise aprofundada concentrou-se no Mini-Batch *K*-Means, por apresentar melhor equilíbrio entre qualidade, escalabilidade e custo computacional no recorte analisado. Técnicas mais avançadas não foram utilizadas nesta etapa justamente por implicarem maior complexidade computacional.

Após a formação dos *clusters*, na segunda fase, utilizou-se o algoritmo *Random Forest* para predizer os rótulos removidos antes da criação dos agrupamentos — `label` (binário) e `type` (10 classes) — além do rótulo das classes produzido pela fase não supervisionada. Os valores utilizados para os parâmetros do algoritmo foram baseados em [Moustafa 2021], de modo que o número de árvores e a semente para o gerador de números aleatórios foram definidos, respectivamente, como 150 e 10000, além de se considerar o desbalanceamento entre classes e a utilização de paralelismo. O pré-processamento supervisionado usou codificação categórica de modo consistente, com separação numérica/categórica. O desempenho final reportado neste trabalho é calculado no conjunto de teste por grupos, definido pelo SGKF, de modo a aproximar um cenário de generalização para conexões/serviços não vistos. Como referência adicional, pode-se aplicar validação cruzada apenas no treino; porém, quando essa validação não é *group-aware*, ela pode superestimar o desempenho por redundância de seções de comunicação semelhantes em *folds* distintos.

3.4. Métricas de avaliação e estatísticas

As métricas internas são utilizadas para avalizar a qualidade geométrica dos *clusters* usando apenas os atributos (coesão e separação), sem recorrer a rótulos. Foram utilizados neste trabalho: Coeficiente de Silhueta, que mede a separação e a coesão; Davies–Bouldin (DB), que quantifica quão separados e compactos são os clusters; Calinski–Harabasz (CH), que mede a razão entre a dispersão entre clusters e a dispersão dentro dos clusters, ajustada pelos graus de liberdade; e a Inércia, que é calculada pela soma de distâncias intra-cluster, sendo usada no método visual do cotovelo [Fahad et al. 2014].

Já as métricas externas comparam a partição obtida com uma referência rotulada, quantificando o quanto os agrupamentos se alinham às classes reais. Assim, após a criação dos clusters e a geração dos respectivos rótulos, os rótulos `label` e `type` foram reinseridos para avaliação através das seguintes métricas externas: AMI (*Adjusted Mutual Information*); *V-measure* (média harmônica de homogeneidade e completude); pureza por cluster e tabelas de contingência (*cluster*×`type`, *cluster*×`label`) [Fahad et al. 2014].

Além disso, para avaliar se o alinhamento observado entre *cluster* e classe pode ser explicado por um acaso, aplicou-se um teste de permutação, que consistiu em: manter fixa a partição de clusters; permutar $N = 200$ vezes os rótulos reais (`type`/`label`); recomputar AMI/*V-measure* por permutação; estimar um p-valor como a fração de permutações com *score* maior ou igual ao observado. Os resultados obtidos são mostrados na Tabela 2, incluindo os respectivos valores de q_{95} e *p*-valor.

4. Resultados e Discussão

Nesta seção, são apresentados os resultados obtidos neste trabalho. As primeiras subseções são dedicadas à avaliação da formação dos *clusters*, enquanto a última aborda a qualidade da inferência supervisionada.

Tabela 2. Teste de significância por permutação para o alinhamento entre *clusters* e rótulos

Split	**Referência**	**AMI**	q_{95}	***V-measure***	q_{95}	p**-valor**
Validação	`type` ↔ clusters	0,6920	0,0005	0,6928	0,0031	0,0050
Validação	`label` ↔ clusters	0,4706	0,0003	0,4708	0,0008	0,0050
Teste	`type` ↔ clusters	0,6023	0,0008	0,6041	0,0055	0,0050
Teste	`label` ↔ clusters	0,3692	0,0005	0,3696	0,0012	0,0050

4.1. Resultados do clustering

4.1.1. Avaliação interna

Inicialmente foi considerado o uso do Mini-Batch K-Means para a formação dos clusters realizando uma varredura de k, com valores variando de 2 a 60, para a seleção de um valor adequado. A semente do gerador aleatório e o tamanho do lote usado em cada iteração foram definidos, respectivamente, como 42 e 4096 amostras.

A Figura 2a apresenta a variação do Coeficiente de Silhueta em função de k (valores maiores são considerados melhores), enquanto a Figura 2b, a Figura 2c e a Figura 2d mostram, respectivamente, Davies–Bouldin (quanto menor, melhor), Calinski–Harabasz (quanto maior, melhor) e a Inércia (quanto menor, melhor). Essas curvas permitem observar regiões com melhor separação/coesão no *embedding*, bem como regimes em que o particionamento passa a sobre-segmentar o espaço.

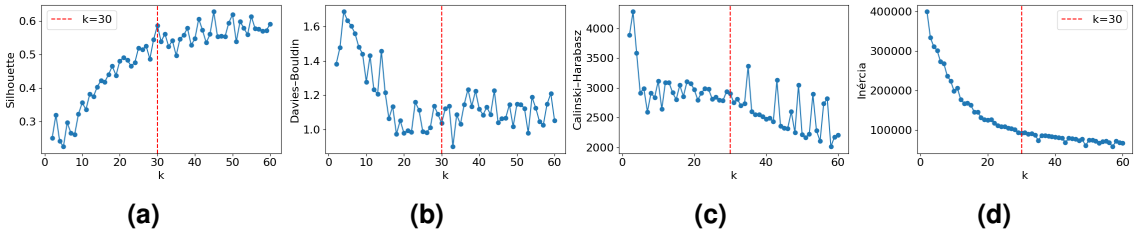


Figura 2. Variação de cada métrica em função do valor de k: (a) Coeficiente de Silhueta; (b) Davies–Bouldin; (c) Calinski–Harabasz; (d) Inércia.

Adicionalmente, a Tabela 3 apresenta os 10 melhores valores avaliados para k segundo as métricas internas computadas na amostra de 20 000 pontos no *embedding*. Através da análise conjunta da Figura 2 e da Tabela 3, neste trabalho, adotou-se $k = 30$ como região de interesse e configuração principal, por conciliar qualidade interna, além de custo e aderência externa após reinserção dos rótulos, como será evidenciado pelos resultados da avaliação externa.

Tabela 3. 10 melhores valores de k, ordenados pelo Coeficiente de Silhueta

k	**Coeficiente de Silhueta**	**Davies–Bouldin**	**Calinski–Harabasz**	**Inércia**
45	0.627735	1.039980	2323.449961	78013.906255
50	0.619337	1.147636	2219.083717	74105.552254
55	0.613779	1.125021	2103.377061	71704.637152
41	0.606375	1.084910	2489.771595	80653.535368
52	0.598407	1.123156	2221.089645	71630.386985
49	0.594675	1.016283	3047.913774	59691.072015
60	0.591422	1.050585	2210.567047	67058.917948
30	0.586466	1.037019	2908.084437	91592.465004
53	0.579155	0.979662	2891.959254	66197.726795
38	0.578868	1.134789	2549.257595	83627.592104

Destaca-se que, embora os algoritmos DBSCAN e Hierárquico também tenham sido analisados, a análise aprofundada deste trabalho concentra-se no Mini-Batch *K*-Means por sua melhor adequação ao protocolo experimental adotado. A Tabela 4 apresenta uma comparação resumida entre configurações representativas dos três algoritmos avaliados. Observa-se que DBSCAN e o método hierárquico apresentam resultados competitivos em algumas métricas, especialmente nas medidas internas e no alinhamento externo com o atributo `type`. No entanto, o Mini-Batch *K*-Means foi adotado como método principal por apresentar desempenho externo competitivo, maior escalabilidade para conjuntos com centenas de milhares de conexões e melhor compatibilidade com a avaliação *group-wise*. Diferentemente de DBSCAN e Agglomerative Clustering, o Mini-Batch *K*-Means aprende centróides no conjunto de treino e permite atribuir, de forma direta, rótulos de *cluster* a novas instâncias de validação e teste, o que facilita sua integração à fase supervisionada, fase dois do *pipeline*.

Tabela 4. Comparação resumida entre os algoritmos de agrupamento avaliados

Algoritmo	Configuração	Silhueta	DB	AMI/V$_{type}$
Mini-Batch K-Means	$k = 30$	0,433	1,498	0,610 / 0,611
DBSCAN	$\epsilon = 4{,}33$, minPts=30	0,715	0,562	0,635 / 0,637
Hierárquico	Ward, $k = 30$	0,396	1,409	0,639 / 0,640

4.1.2. Avaliação externa

Tendo os 30 *clusters* gerados (rotulados de 0 a 29 e, doravante, referenciados como `cluster_kmeans30`), os rótulos reais foram reinseridos para análise e cálculo das métricas AMI e *V-measure*. Os resultados indicaram alinhamento consistente entre os agrupamentos induzidos e as classes reais, tanto em validação (`type`: AMI = 0,6920 e *V-measure* = 0,6928; `label`: AMI = 0,4706 e *V-measure* = 0,4708) quanto em teste (`type`: AMI = 0,6023 e *V-measure* = 0,6041; `label`: AMI = 0,3692 e *V-measure* = 0,3696), com significância estatística evidenciada na Tabela 2. A Figura 3, normalizada por linha, mostra que diversas famílias de ataque formam "ilhas" bem definidas: o *cluster* 1 é praticamente puro para `DoS` (≈0,99) e o *cluster* 3 para `backdoor` (≈0,99). Para `DDoS`, observam-se pelo menos dois *clusters* majoritariamente dominados (17 com ≈0,92 e 26 com ≈0,90), além de um *cluster* de fronteira (16), em que `DDoS` se mistura com `normal` (≈0,56 vs. ≈0,37). Já o tráfego `normal` aparece distribuído em múltiplos modos quase puros, especialmente nos *clusters* 2, 5, 6, 7, 8, 14, 18, 19 e 23, com proporções entre ≈0,96 e 1,00, sugerindo heterogeneidade do tráfego benigno e contribuindo para o alinhamento externo observado.

Em contraste, algumas classes não se manifestam como uma única ilha bem separada, aparecendo de forma sistematicamente misturada com outras. Isso ocorre para `injection`, que domina *clusters* mistos (4 com ≈0,42 e 28 com ≈0,56), frequentemente compartilhando massa com classes associadas a padrões de aplicação (por exemplo, `password` e `xss`). De modo semelhante, percebe-se a existência de *clusters* altamente mistos (por exemplo, 22 e 24) em que nenhuma classe domina fortemente, indicando regiões de sobreposição no espaço de atributos e possíveis subcomportamentos dentro das famílias rotuladas. Por fim, `mitm` não forma um *cluster* dominante (o maior valor observado na Figura3 fica abaixo de ≈0,20), o que é consistente com um baixo

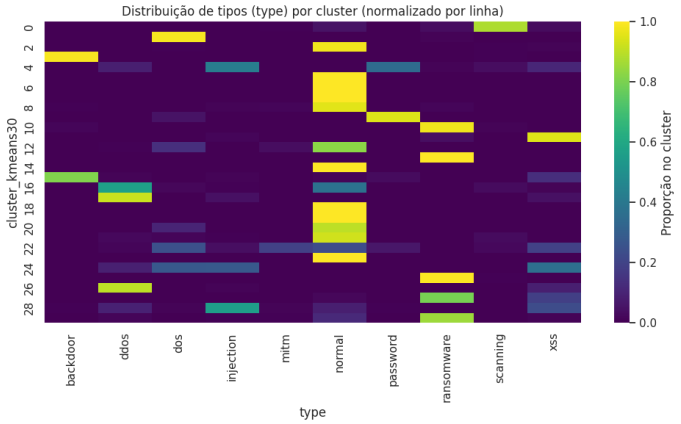


Figura 3. Distribuição dos tipos de ataque por *cluster* (normalizada por linha).

suporte (desbalanceamento) e/ou padrões de fluxo pouco distintivos para essa família na amostra analisada.

Além das métricas externas, a inspeção de pureza por *cluster* (isto é, a proporção da classe majoritária em cada agrupamento) permite caracterizar quais grupos são "quase puros" e quais misturam famílias de ataque diferentes. Em geral, pureza baixa pode refletir decomposição de um mesmo `type` em múltiplos *clusters* distintos, sobreposição entre tipos de ataques com padrões semelhantes de portas, serviços e volume de tráfego, ou ainda *clusters* de fronteira que capturam regiões intermediárias do espaço de atributos. A análise de pureza por *cluster* em relação ao tipo de ataque e a relação entre cobertura e limiar de pureza são mostradas na Figura 4; os limiares 0,90 e 0,95 são usados apenas como referências operacionais para destacar *clusters* quase puros.

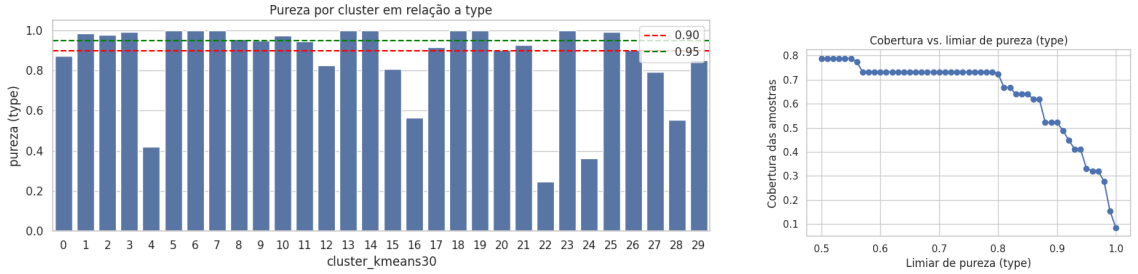


Figura 4. Análise da pureza dos *clusters* em relação ao tipo de ataque: (a) pureza por *cluster*; (b) cobertura versus limiar de pureza.

4.2. Resultados do aprendizado supervisionado

4.2.1. Predição do rótulo `type` e `label`

A Tabela 5 resume o desempenho do *Random Forest* no conjunto de teste para as tarefas de predição de `type`, `label` e `cluster_kmeans30`. Como o conjunto é desbalanceado, especialmente pela baixa representatividade da classe `mitm`, a análise considera não apenas a acurácia, mas também métricas macro, métricas ponderadas e o coeficiente de correlação de Matthews (MCC). Na predição de `type`, os valores de F1 macro e revocação macro são inferiores aos das métricas ponderadas, pois as métricas macro atribuem o mesmo peso a todas as classes e, portanto, evidenciam de modo mais claro o

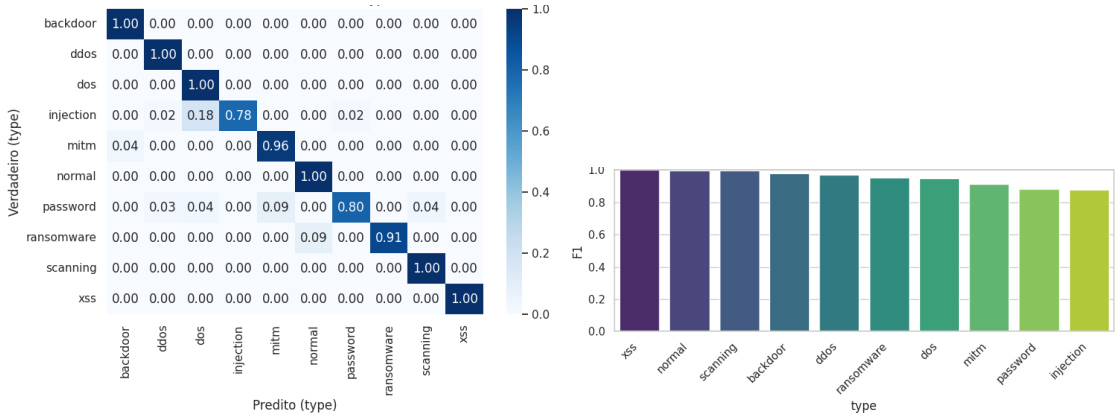
impacto de classes raras ou mais difíceis.

Tabela 5. Resultados do aprendizado supervisionado no conjunto de teste

Tarefa	Acc.	M-P	M-R	M-F1	W-P	W-R	W-F1	MCC
`type`	0,9951	0,9651	0,9460	0,9525	0,9953	0,9951	0,9950	0,9928
`label`	0,9995	0,9996	0,9991	0,9993	0,9995	0,9995	0,9995	0,9987
`cluster_kmeans30`	0,9144	0,8326	0,7679	0,7648	0,9556	0,9144	0,9265	0,8977

Legenda: Acc. = acurácia; M-P = precisão macro; M-R = revocação macro; M-F1 = F1 macro; W-P = precisão ponderada; W-R = revocação ponderada; W-F1 = F1 ponderado; MCC = coeficiente de correlação de Matthews.

Na Figura 5, a concentração na diagonal da matriz de confusão indica boa separação global entre as classes no conjunto de teste *group-wise*. As reduções de desempenho aparecem sobretudo nas classes com menor suporte, refletindo o desbalanceamento do *dataset*, e nas categorias cujos padrões de tráfego são mais próximos no espaço de atributos. Esse comportamento é consistente com a diferença entre métricas ponderadas (dominadas por classes frequentes) e métricas macro (sensíveis a classes raras) reportadas na Tabela 5.



(a) **(b)**

Figura 5. Resultados do classificador para o tipo de ataque com base no conjunto de **teste**: (a) Matriz de confusão normalizada; (b) F1-score por classe.

Resultados complementares, incluindo matrizes de confusão, métricas por classe e análises adicionais para a predição de `type`, `label` e `cluster_kmeans30`, estão disponíveis no repositório complementar do trabalho[3].

4.2.2. Predição de `cluster_kmeans30`

Ao utilizar o Random Forest para predizer o `cluster_kmeans30` (30 classes), observou-se que uma validação cruzada não *group-aware* (10-fold no treino) superestima fortemente o desempenho, alcançou-se um macro-F1 de $0{,}9993 \pm 0{,}00039$. Em comparação, quando se adota o particionamento *group-wise* (SGKF), o macro-F1 médio no treino cai para $0{,}7967 \pm 0{,}1044$, como observável na Tabela 6 e, no teste por grupos, o macro-F1 é de $0{,}7648$, evidenciado na Tabela 5. Em termos relativos, a validação cruzada não-*group-aware* é aproximadamente 25,4% maior que a validação cruzada *group-aware* no treino e 30,7% maior que o resultado no teste por grupos, evidenciando o vazamento

[3] `https://github.com/Math-BUG/iot-fog-ids-two-phase`

por repetição de instâncias altamente similares em *folds* distintos quando os grupos não são respeitados.

Tabela 6. Resultados supervisionados no **treino**: Macro-F1 em validação cruzada (10-*fold*, média $\pm$ desvio-padrão)

Tarefa	Macro-F1 (validação cruzada no treino, 10-fold)
Predizer `type` (10 classes)	$0{,}7349 \pm 0{,}1967$
Predizer `label` (binário)	$0{,}9945 \pm 0{,}0093$
Predizer `cluster_kmeans30` (30)	$0{,}7967 \pm 0{,}1044$

Observa-se que a predição de `label` (normal vs. ataque) atinge desempenho próximo ao ótimo no teste, sugerindo que, neste recorte do dataset, a separação binária é altamente aprendível a partir dos atributos disponíveis. Assim, as tarefas mais informativas e desafiadoras são: (i) distinguir `type` (múltiplas famílias de ataque) e (ii) reproduzir os rótulos induzidos pelo *clustering*, que refletem padrões comportamentais potencialmente mais sutis.

5. Conclusão e Trabalhos Futuros

Este trabalho propôs e implementou um *pipeline* completo para treinamento e inferência de diferentes tipos de ataques em redes de computadores a partir de metadados relacionados a sessões de comunicação de dispositivos na borda da rede. O *pipeline* proposto incluiu o agrupamento de conexões sem uso dos rótulos, a comparação de algoritmos de agrupamento e a seleção de configurações com base em métricas internas, a quantificação do alinhamento entre *clusters* e classes reais por métricas externas e teste de permutação, a adoção de um protocolo de avaliação *group-wise* para reduzir vazamento e aproximar um cenário realista de generalização e, finalmente, o treinamento de modelos supervisionados para predizer rótulos originais e *clusters*. Do ponto de vista de aplicação, os resultados reforçam a viabilidade de executar etapas de engenharia de atributos e inferência em nós na névoa (por exemplo, *gateways* e roteadores), de modo que dispositivos IoT com recursos limitados não precisem manter modelos localmente. Nesse cenário, o *clustering* pode servir como ferramenta de descoberta e organização de padrões, enquanto modelos supervisionados podem operacionalizar detecção e classificação com baixa latência na borda.

Em trabalhos futuros pretende-se explorar estratégias de agrupamento mais avançadas, como HDBSCAN, incorporar variáveis temporais, validação temporal janelas de observação, consolidar resultados visuais do SHAP (*SHapley Additive exPlanations*) para `label`, `type` e `cluster`, avaliar a estabilidade do *clustering* (múltiplas sementes, amostragens e variação de k) e emprego de datasets distintos para aprimorar a generalização, fundamental em ambientes pervasivos. Além disso, pretende-se formalizar a arquitetura de segurança para o cenário IoT/névoa e emular um cenário operacional em nós de névoa representativos (por exemplo, *gateways* de baixo custo), visando avaliar a latência fim-a-fim de detecção de anomalias, além do custo computacional, do *overhead* de comunicação e da escalabilidade sob tráfego real.

Agradecimentos

Os autores agradecem FAPEMIG, CNPq e CAPES pelo financiamento deste trabalho.

Referências

Bohara, B., Bhuyan, J., Wu, F., and Ding, J. (2020). A survey on the use of data clustering for intrusion detection system in cybersecurity. *International Journal of Network Security & Applications*, 12(1):1–18.

Chen, K., Zhang, S., Li, Z., et al. (2018). Internet-of-things security and vulnerabilities: Taxonomy, challenges, and practice. *Journal of Hardware and Systems Security*, 2:97–110.

Fahad, A., Alshatri, N., Tari, Z., Alamri, A., Khalil, I., Zomaya, A., Foufou, S., and Bouras, A. (2014). A survey of clustering algorithms for big data: Taxonomy and empirical analysis. *IEEE Transactions on Emerging Topics in Computing*, 2(3):267–279.

Mohamed, D. and Ismael, O. (2023). Enhancement of an iot hybrid intrusion detection system based on fog-to-cloud computing. *Journal of Cloud Computing*, 12:41.

Moustafa, N. (2021). A new distributed architecture for evaluating AI-based security systems at the edge: Network TON_IoT datasets. *Sustainable Cities and Society*, 72:102994.

Perumal, S., Sujatha, P. K., Krishnaa, S., and Krishnan, M. (2025). Clusters in chaos: A deep unsupervised learning paradigm for network anomaly detection. *Journal of Network and Computer Applications*, 235:104083.

Prazeres, N., Costa, R. L. d. C., Santos, L., and Rabadão, C. (2023). Engineering the application of machine learning in an ids based on iot traffic flow. *Intelligent Systems with Applications*, 17:200189.

Qaddoura, R., Al-Zoubi, A. M., Almomani, I., and Faris, H. (2021). A multi-stage classification approach for iot intrusion detection based on clustering with oversampling. *Applied Sciences*, 11(7).

Rahman, M. M., Al Shakil, S., and Mustakim, M. R. (2025). A survey on intrusion detection system in iot networks. *Cyber Security and Applications*, 3:100082.

Salehiyan, A., Moghaddam, P. S., and Kaveh, M. (2025). An optimized transformer–gan–ae for intrusion detection in edge and iiot systems: Experimental insights from wustl-iiot-2021, edgeiiotset, and ton_iot datasets. *Future Internet*, 17(7):279.

Sculley, D. (2010). Web-scale k-means clustering. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 1177–1178, New York, NY, USA. Association for Computing Machinery.