Universidade Federal de São Carlos– UFSCar
Centro de Ciências Exatas e de Tecnologia– CCET
Departamento de Computação– DC
Programa de Pós-Graduação em Ciência da Computação– PPGCC

Wagner Rafael Giarini

Mapas Auto-Organizáveis Crescentes para Classificação de logs de Firewall em Fluxos Contínuos de Dados

São Carlos
2026

Wagner Rafael Giarini

Mapas Auto-Organizáveis Crescentes para Classificação de logs de Firewall em Fluxos Contínuos de Dados

Dissertação apresentada ao Programa de Pós-Graduação em Ciência da Computação do Centro de Ciências Exatas e de Tecnologia da Universidade Federal de São Carlos, como parte dos requisitos para a obtenção do título de Mestre em Ciência da Computação.

Área de concentração: Metodologias e Técnicas de Computação

Orientador: Ricardo Cerri

São Carlos
2026

Folha de Aprovação

Defesa de dissertação de mestrado do(a) candidato(a) Wagner Rafael Giarini, realizada em 13/03/2026

Comissão Julgadora

Prof(a) Dr(a) Ricardo Cerri (USP)

Prof(a) Dr(a) Jean Paul Barddal (PUC-PR)

Prof(a) Dr(a) Helio Crestana Guardia (UFSCar)

O relatório de defesa assinado pelos membros da comissão Julgadora encontra-se arquivado junto ao Programa de Pós-Graduação em Ciência da Computação

Agradecimentos

Agradeço a todos que estiveram ao meu lado e contribuíram para a conclusão deste mestrado.

Ao meu orientador, Prof. Dr. Ricardo Cerri, pelas orientações e por ter me guiado ao longo de todo o desenvolvimento desta pesquisa.

À minha esposa, por ser minha conselheira. Muito obrigado por compreender as horas destinadas ao estudo e por ter tido paciência para ouvir meus desabafos, e me ajudando a manter o foco nos momentos em que precisei.

Aos meus amigos e colegas de laboratório, pelas discussões técnicas, troca de ideias e por tornarem o ambiente de pesquisa mais produtivo e colaborativo.

Aos meus amigos e colegas de trabalho, pelo companheirismo. Obrigado por entenderem a minha divisão de foco entre as demandas diárias e as entregas necessárias para esta dissertação.

Resumo

O aumento exponencial no volume de dados e a natureza dinâmica das redes de computadores têm imposto desafios significativos à segurança da informação. Nesse contexto, a classificação de dados em Fluxo Contínuo de Dados (FCDs) torna-se essencial, especialmente em Sistemas de Segurança de Rede (SSR). Essas aplicações demandam técnicas que lidem com restrições de memória, de tempo de processamento e de dados não estacionários. Este trabalho propõe uma adaptação do Mapa Auto-Organizável Crescente (GSOM) para a classificação de registros de *firewall* em FCDs. A arquitetura do GSOM foi considerada adequada, pois ajusta os pesos dos neurônios para acompanhar a evolução gradual dos padrões e expande sua estrutura para representar novos comportamentos. Para avaliação, foram utilizados dados reais de um *firewall* em operação, classificados nas ações "*Allow*", "*Deny*", "*Reset-Both*" ou "*Drop*". Os experimentos seguiram uma abordagem em fases *offline* (treinamento inicial) e *online*, em que três estratégias de atualização foram propostas: (1) localização do neurônio vencedor mais próximo, mantendo o mapa fixo; (2) expansão dinâmica do mapa, adicionando novos neurônios para representar padrões emergentes; e (3) refinamento dos pesos dos neurônios existentes, ajustando o mapa sem alterar sua estrutura. O desempenho do GSOM foi avaliado em quatro cenários temporais, incluindo um cenário combinado, e comparado com métodos estabelecidos como SVM *Incremental* e *Adaptive Random Forest* (ARF). A comparação estabeleceu um limite superior de desempenho para o cenário ideal de latência nula dos rótulos. Embora o ARF tenha demonstrado superioridade geral, seu alto custo computacional e a dependência de rotulação imediata limitam sua aplicação prática. Conclui-se que o GSOM constitui uma solução viável para o cenário de classificação em FCDs com latência infinita de rótulos, em que a supervisão completa é operacionalmente inviável.

Palavras-chave: fluxos contínuos de dados, mapas auto-organizáveis, logs de *firewall*.

Abstract

The exponential increase in data volume and the dynamic nature of computer networks have imposed significant challenges on information security. In this context, classification in Continuous Data Streams (CDS) becomes essential, especially for Network Security Systems (NSS). These applications require techniques that handle memory constraints, processing time, and non-stationary data. This work proposes an adaptation of the Growing Self-Organizing Map (GSOM) for classifying firewall logs in CDS. The GSOM architecture was considered suitable, as it adjusts neuron weights to follow the gradual evolution of patterns and expands its structure to represent new behaviors as they emerge. For evaluation, real data from an operational firewall was used, classified into the actions "Allow", "Deny", "Reset-Both", or "Drop". The experiments followed an approach with *offline* (initial training) and *online* phases, where three update strategies were proposed: (1) nearest winner neuron location, keeping the map fixed; (2) dynamic map expansion, adding new neurons to represent emerging patterns; and (3) refinement of existing neuron weights, adjusting the map without altering its structure. The performance of the GSOM was evaluated in four temporal scenarios, including a combined scenario, and compared with established methods such as Incremental SVM and Adaptive Random Forest (ARF). The comparison established an upper bound of performance for the ideal scenario of zero-label latency. Although ARF demonstrated overall superiority, its high computational cost and dependence on immediate labeling limit its practical application. It is concluded that GSOM is a viable solution for real-world classification in CDS with infinite label latency, where complete supervision is operationally unfeasible.

Keywords: continuous data streams, self-organizing maps, firewall logs.

Lista de ilustrações

Figura 1 – Posicionamento e função de um *firewall* 30
Figura 2 – Mudanças nos fluxos de dados 38
Figura 3 – Etapas para agrupamento em FCDs 40
Figura 4 – Arquitetura de funcionamento do SOM 44
Figura 5 – Estrutura dos Mapas Auto-Organizáveis 45
Figura 6 – Diagrama esquemático de uma rede bidimensional de neurônios 46
Figura 7 – Ajuste de pesos do neurônio vencedor e dos seus vizinhos 47
Figura 8 – Função de vizinhança gaussiana 48
Figura 9 – Topografia da grade de neurônios 49
Figura 10 – Visualização da auto-organização através do mapeamento dos dados de entrada para o espaço de neurônios 52
Figura 11 – Configuração inicial da rede GSOM com quatro neurônios. 57
Figura 12 – Geração de novos neurônios na fronteira da rede. 58
Figura 13 – Inicialização de peso de novos neurônios 59
Figura 14 – Fluxo dos vetores de peso 60
Figura 15 – Palo Alto PA-3260 Firewall 67
Figura 16 – Interface de monitoramento de tráfego de rede 69
Figura 17 – Projeção PCA dos dados coletados em agosto de 2024 72
Figura 18 – Projeção PCA dos dados coletados em fevereiro de 2025 73
Figura 19 – Projeção PCA dos dados coletados em outubro de 2025 73
Figura 20 – Projeções PCA por lotes selecionados do conjunto de dados de agosto de 2024 76
Figura 21 – Projeções PCA de lotes selecionados do conjunto de dados de fevereiro de 2025. 78
Figura 22 – Projeções PCA por lotes selecionados do conjunto de dados de outubro de 2025. 80
Figura 23 – Mapas das três estratégias do GSOM no Experimento 1. 90

Figura 24 – Mapas das três estratégias do GSOM no Experimento 2. 94
Figura 25 – Mapas das três estratégias do GSOM no Experimento 3. 99
Figura 26 – Mapas das três estratégias do GSOM no Experimento 4. 104

Lista de tabelas

Tabela 1 – Resumo dos principais trabalhos relacionados 65
Tabela 2 – Descrição das classes de ação do firewall 69
Tabela 3 – Atributos utilizados na base de dados. 71
Tabela 4 – Distribuição em valores absolutos e porcentagem das classes em três coletas 72
Tabela 5 – Distribuição das classes nos blocos em agosto de 2024. 75
Tabela 6 – Distribuição das classes nos blocos de 50.000 instâncias em fevereiro de 2025. 77
Tabela 7 – Distribuição das classes nos blocos de 50.000 instâncias durante outubro de 2025. 79
Tabela 8 – Distribuição detalhada das três coletas por fase e ação 84
Tabela 9 – Distribuição detalhada dos experimentos por fase e ação 84
Tabela 10 – Configuração do treinamento *offline* do GSOM nos experimentos realizados 85
Tabela 11 – Matriz de confusão do GSOM na primeira coleta (estratégia *get_winner*) em valores absolutos e porcentuais. 90
Tabela 12 – Desempenho do GSOM por classe na primeira coleta (estratégia *get_winner*) 91
Tabela 13 – Médias de desempenho do GSOM na primeira coleta (estratégia *get_winner*) 91
Tabela 14 – Matriz de confusão do GSOM na primeira coleta (estratégia *growing_phase*) em Valores Absolutos e em Porcentagem. 92
Tabela 15 – Desempenho do GSOM por classe na primeira coleta (estratégia *growing_phase*) 92
Tabela 16 – Médias de desempenho do GSOM na primeira coleta (estratégia *growing_phase*) 92
Tabela 17 – Matriz de confusão do GSOM na primeira coleta (estratégia *smoothing_phase*) em Valores Absolutos e em Porcentagem. 93
Tabela 18 – Desempenho do GSOM por classe na primeira coleta (estratégia *smoothing_phase*) 93

Tabela 19 – Médias de desempenho do GSOM na primeira coleta (estratégia *smoothing_phase*) 94
Tabela 20 – Matriz de confusão do GSOM na segunda coleta (estratégia *get_winner*) em Valores Absolutos e em Porcentagem. 95
Tabela 21 – Desempenho do GSOM por classe na segunda coleta (estratégia *get_winner*) 95
Tabela 22 – Médias de desempenho do GSOM na segunda coleta (estratégia *get_winner*) 96
Tabela 23 – Matriz de confusão do GSOM na segunda coleta (estratégia *growing_phase*) em Valores Absolutos e em Porcentagem. 96
Tabela 24 – Desempenho do GSOM por classe na segunda coleta (estratégia *growing_phase*) 97
Tabela 25 – Médias de desempenho do GSOM na segunda coleta (estratégia *growing_phase*) 97
Tabela 26 – Matriz de confusão do GSOM na segunda coleta (estratégia *smoothing_phase*) em valores absolutos e porcentagem. 98
Tabela 27 – Desempenho do GSOM por classe na segunda coleta (estratégia *smoothing_phase*) 98
Tabela 28 – Médias de desempenho do GSOM na segunda coleta (estratégia *smoothing_phase*) 98
Tabela 29 – Matriz de confusão do GSOM na terceira coleta (estratégia *get_winner*) em Valores Absolutos e em Porcentagem. 100
Tabela 30 – Desempenho do GSOM por classe na terceira coleta (estratégia *get_winner*) 100
Tabela 31 – Médias de desempenho do GSOM na terceira coleta (estratégia *get_winner*) 100
Tabela 32 – Matriz de confusão do GSOM na terceira coleta (estratégia *growing_phase*) em Valores Absolutos e em Porcentagem. 101
Tabela 33 – Desempenho do GSOM por classe na terceira coleta (estratégia *growing_phase*) 101
Tabela 34 – Médias de desempenho do GSOM na terceira coleta (estratégia *growing_phase*) 101
Tabela 35 – Matriz de confusão do GSOM na terceira coleta (estratégia *smoothing_phase*) em Valores Absolutos e em Porcentagem. 102
Tabela 36 – Desempenho do GSOM por classe na terceira coleta (estratégia *smoothing_phase*) 102
Tabela 37 – Médias de desempenho do GSOM na terceira coleta (estratégia *smoothing_phase*) 103
Tabela 38 – Divisão dos dados no Experimento 4: treinamento e teste com coletas temporais distintas 103
Tabela 39 – Matriz de confusão do GSOM no Experimento 4 (estratégia *get_winner*) em Valores Absolutos e em Porcentagem. 104
Tabela 40 – Desempenho do GSOM por classe no Experimento 4 (estratégia *get_winner*) 105
Tabela 41 – Médias de desempenho do GSOM no Experimento 4 (estratégia *get_winner*) 105
Tabela 42 – Matriz de confusão do GSOM no Experimento 4 (estratégia *growing_phase*) em Valores Absolutos e em Porcentagem. 106
Tabela 43 – Desempenho do GSOM por classe no Experimento 4 (estratégia *growing_phase*) 106

Tabela 44 – Médias de desempenho do GSOM no Experimento 4 (estratégia *growing_phase*) 107
Tabela 45 – Matriz de confusão do GSOM no Experimento 4 (estratégia *smoothing_phase*) em Valores Absolutos e em Porcentagem. 107
Tabela 46 – Desempenho do GSOM por classe no Experimento 4 (estratégia *smoothing_phase*) 108
Tabela 47 – Médias de desempenho do GSOM no Experimento 4 (estratégia *smoothing_phase*) 108
Tabela 48 – Matriz de confusão do *Support Vector Machine* (SVM) *Incremental* na primeira coleta em Valores Absolutos e Porcentagem 110
Tabela 49 – Desempenho do SVM *Incremental* por classe na primeira coleta 110
Tabela 50 – Médias de desempenho do SVM *Incremental* na primeira coleta 111
Tabela 51 – Matriz de confusão do SVM *Incremental* na segunda coleta em Valores Absolutos e Porcentagem 111
Tabela 52 – Desempenho do SVM *Incremental* por classe na segunda coleta 112
Tabela 53 – Médias de desempenho do SVM *Incremental* na segunda coleta 112
Tabela 54 – Matriz de confusão do SVM *Incremental* na terceira coleta em Valores Absolutos e Porcentagem 113
Tabela 55 – Desempenho do SVM *Incremental* por classe na terceira coleta 113
Tabela 56 – Médias de desempenho do SVM *Incremental* na terceira coleta 113
Tabela 57 – Matriz de confusão do SVM *Incremental* no Experimento 4 em Valores Absolutos e Porcentagem 114
Tabela 58 – Desempenho do SVM *Incremental* por classe no Experimento 4 114
Tabela 59 – Médias de desempenho do SVM *Incremental* no Experimento 4 115
Tabela 60 – Matriz de confusão do ARF na primeira coleta em Valores Absolutos e Porcentagem 116
Tabela 61 – Desempenho do ARF por classe na primeira coleta 116
Tabela 62 – Médias de desempenho do ARF na primeira coleta 117
Tabela 63 – Matriz de confusão do ARF na segunda coleta em Valores Absolutos e Porcentagem 117
Tabela 64 – Desempenho do ARF por classe na segunda coleta 118
Tabela 65 – Médias de desempenho do ARF na segunda coleta 118
Tabela 66 – Matriz de confusão do ARF na terceira coleta em Valores Absolutos e Porcentagem 119
Tabela 67 – Desempenho do ARF por classe na terceira coleta 119
Tabela 68 – Médias de desempenho do ARF na terceira coleta 119
Tabela 69 – Matriz de confusão do ARF no Experimento 4 em Valores Absolutos e Porcentagem 120
Tabela 70 – Desempenho do ARF por classe no Experimento 4 121
Tabela 71 – Médias de desempenho do ARF no Experimento 4 121

Lista de siglas

AM Aprendizado de Máquina

AD Árvore de Decisão

AP Aprendizagem Profunda

ARF *Adaptive Random Forest*

AIDS Sistema de Detecção de Intrusão baseado em Anomalias

BMU Best Matching Unit

DDoS Ataque de Negação de Serviço Distribuído

DoS Negação de Serviço

DPI Inspeção Profunda de Pacotes

FCDs Fluxo Contínuo de Dados

FIFO *First-In, First-Out*

GSOM Mapa Auto-Organizável Crescente

GT *Growth Threshold*

HIDS Sistema de Detecção de Intrusões baseado em Host

IDS Sistemas de Detecção de Intrusão

IA Inteligência Artificial

IoT Internet das Coisas

KNN *K-Nearest Neighbor*

MD Mineração Dados

MLP *Multilayer Perceptron*

NIDS Sistema de Detecção de Intrusões baseado em Rede

NB *Naive Bayes*

NSOM Mapa Auto-Organizável de Rede

NGFW *Firewalls* da Próxima Geração

PCA *Principal Component Analysis*

RF *Random Forest*

RNAs Redes Neurais Artificiais

SOM Mapa Auto-Organizável

SVM *Support Vector Machine*

SSR Sistemas de Segurança de Rede

SGD *Stochastic Gradient Descent*

SIDS Sistema de Detecção de Intrusão baseado em Assinatura

SF *Spread Factor*

UTI Unidade de Terapia Intensiva

Sumário

1 INTRODUÇÃO 23
1.1 Motivação 24
1.2 Objetivos 25

2 SEGURANÇA EM REDES 27
2.1 Conceitos fundamentais 27
2.2 Vulnerabilidades de Sistemas em Rede 28
2.3 Sistema de Detecção de Intrusão 29
2.4 *Firewalls* 30
2.4.1 Evolução dos *Firewalls* 30
2.4.2 Funcionamento do *Firewall* utilizado no estudo 32
2.5 Considerações finais 33

3 FLUXOS CONTÍNUOS DE DADOS 35
3.1 Conceitos fundamentais 35
3.2 Agrupamento em Fluxo Contínuo de Dados 39
3.3 Classificação em Fluxo Contínuo de Dados 41
3.4 Considerações finais 42

4 MAPAS AUTO-ORGANIZÁVEIS 43
4.1 Conceitos fundamentais 43
4.2 Estruturas dos Mapas Auto-Organizáveis 44
4.2.1 Processo Competitivo 46
4.2.2 Processo Cooperativo 48
4.2.3 Processo Adaptativo 49
4.3 Configuração dos hiperparâmetros da rede 50
4.4 Análise de agrupamento 51

4.5 Classificação em Mapas Auto-Organizáveis . . . 52
4.6 Considerações Finais . . . 53

5 MAPAS AUTO-ORGANIZÁVEIS CRESCENTES . . . 55
5.1 Conceitos fundamentais . . . 55
5.2 Fase de Inicialização . . . 56
5.3 Fase de Crescimento . . . 57
5.3.1 Geração de Novos Neurônios . . . 58
5.3.2 Inicialização de Peso de Novos Neurônios . . . 58
5.4 Fase de Suavização . . . 60
5.5 Considerações finais . . . 60

6 TRABALHOS RELACIONADOS . . . 61
6.1 Classificação de logs de firewall . . . 61
6.2 Classificação em Sistemas de Detecção de Intrusão . . . 63
6.3 Tabela comparativa dos trabalhos relacionados . . . 64
6.4 Considerações finais . . . 65

7 MATERIAIS E MÉTODOS . . . 67
7.1 Análise Temporal dos Dados . . . 70
7.1.1 Agosto de 2024 . . . 75
7.1.2 Fevereiro de 2025 . . . 76
7.1.3 Outubro de 2025 . . . 78
7.2 Pré-processamento e Particionamento dos Dados . . . 80
7.3 Treinamento e Operação Contínua do GSOM . . . 85
7.3.1 Fase *offline* . . . 85
7.3.2 Fase *online* . . . 86
7.4 Tarefa de Classificação . . . 87

8 EXPERIMENTOS E RESULTADOS . . . 89
8.1 Experimento 1: agosto de 2024 . . . 89
8.1.1 Estratégia *get_winner* . . . 90
8.1.2 Estratégia *growing_phase* . . . 91
8.1.3 Estratégia *smoothing_phase* . . . 93
8.2 Experimento 2: fevereiro de 2025 . . . 94
8.2.1 Estratégia *get_winner* . . . 95
8.2.2 Estratégia *growing_phase* . . . 96
8.2.3 Estratégia *smoothing_phase* . . . 97
8.3 Experimento 3: Outubro de 2025 . . . 98
8.3.1 Estratégia *get_winner* . . . 99
8.3.2 Estratégia *growing_phase* . . . 100

8.3.3 Estratégia *smoothing_phase* . . . 102
8.4 Experimento 4: Dados Combinados . . . 103
8.4.1 Estratégia *get_winner* . . . 104
8.4.2 Estratégia *growing_phase* . . . 105
8.4.3 Estratégia *smoothing_phase* . . . 107
8.5 Comparação entre os Experimentos . . . 108
8.6 Comparação com Métodos Supervisionados . . . 109
8.6.1 SVM *Incremental* . . . 109
8.6.2 Adaptive Random Forest . . . 115
8.6.3 Análise Comparativa com Métodos Supervisionados . . . 121
8.6.4 Considerações Finais do Capítulo . . . 122

9 CONCLUSÃO . . . 125
9.1 Principais Contribuições . . . 125
9.2 Limitações . . . 126
9.3 Trabalhos Futuros . . . 126

A DISPONIBILIDADE DO CÓDIGO E DOS DADOS . . . 129

REFERÊNCIAS . . . 131

Capítulo 1

Introdução

O avanço exponencial das tecnologias de Internet, computação em nuvem, Internet das Coisas (IoT) e Inteligência Artificial (IA) tem provocado um crescimento sem precedentes no volume de dados trafegados em redes de computadores. Esse fenômeno é acompanhado por uma crescente complexidade das infraestruturas de rede e pelo aumento de dispositivos conectados, o que amplia a superfície de ataque e torna as redes mais vulneráveis a novas ameaças (QU et al., 2021; HARIPRIYA et al., 2024).

Os Sistemas de Segurança de Rede (SSR), como *firewalls* e Sistemas de Detecção de Intrusão (IDS), desempenham um papel central na proteção contra ameaças, monitorando e controlando o fluxo de dados em tempo real (ALJABRI et al., 2022). Contudo, é fundamental diferenciar seus papéis, que são complementares, mas distintos: o *firewall* exerce uma função primária de controle de acesso, aplicando regras predefinidas para permitir ou negar tráfego; o Sistemas de Detecção de Intrusão (IDS), por sua vez, desempenha uma função de detecção de ameaças, monitorando o tráfego interno em busca de padrões suspeitos. Este trabalho foca nos logs gerados pelas decisões do *firewall*, que refletem o resultado direto das políticas de segurança aplicadas.

Nesse cenário, a análise de tráfego de rede tornou-se uma etapa fundamental para garantir a eficiência, a confiabilidade e a segurança das redes modernas. Contudo, a natureza dinâmica e não estacionária dos dados gerados por esses sistemas, aliada ao surgimento constante de novos tipos de ataques, como *phishing*, *ransomware* e Ataque de Negação de Serviço Distribuído (DDoS), desafia a eficácia dos mecanismos tradicionais de defesa e análise (BABCOCK et al., 2002; ALJABRI et al., 2022).

Além disso, a crescente adoção de criptografia no tráfego de rede, a integração de múltiplos serviços em ambientes híbridos e a dissolução das fronteiras tradicionais das redes corporativas impõem desafios adicionais para o monitoramento, a inspeção e a resposta

a incidentes de segurança. O gerenciamento manual das regras de *firewall*, baseado em ações como "*Allow*", "*Deny*", "*Reset-Both*" ou "*Drop*", torna-se cada vez mais complexo e propenso a erros, podendo resultar em vulnerabilidades críticas e brechas exploráveis por agentes maliciosos (UCAR; OZHAN, 2017).

A maioria das abordagens tradicionais de análise de tráfego e de classificação de logs de *firewall* ainda se baseia em processamento em lote (*batch*) e assume que os dados são estacionários e totalmente rotulados. No entanto, as redes atuais operam em Fluxo Contínuo de Dados (FCDs), nos quais os dados chegam em alta velocidade, apresentam mudanças de conceito ao longo do tempo e, muitas vezes, a obtenção de rótulos é limitada, onerosa ou impraticável (HARIPRIYA et al., 2024). Essas características tornam os métodos convencionais insuficientes para lidar com a escala, a dinâmica e a complexidade dos ambientes modernos de rede.

Portanto, há uma demanda crescente por soluções automatizadas, adaptativas e escaláveis capazes de processar, analisar e classificar grandes volumes de dados em tempo real, mesmo diante de padrões de tráfego em constante evolução e de restrições de memória e de processamento. É nesse contexto que se insere a presente pesquisa, que busca avançar o estado da arte na classificação de logs de *firewall* em FCDs por meio de técnicas de Aprendizado de Máquina (AM).

1.1 Motivação

Apesar dos avanços recentes em AM aplicados à segurança de redes, diversos desafios persistem na análise de logs de *firewall* em FCDs. Os métodos tradicionais, como *K-Nearest Neighbor* (KNN) (COVER; HART, 1967), *Random Forest* (RF) (BREIMAN, 2001) e Redes Neurais Artificiais (RNAs) em *batch*, como, por exemplo, Mapa Auto-Organizável (SOM) (KOHONEN, 1988), *Multilayer Perceptron* (MLP) (POPESCU et al., 2009), embora amplamente utilizados para classificação e análise de tráfego (LIAO; VEMURI, 2002; PATGIRI et al., 2018; BREIMAN, 2001), apresentam limitações importantes nesse contexto.

Primeiramente, abordagens convencionais frequentemente dependem de hiperparâmetros predefinidos e de conjuntos de dados rotulados, o que limita sua capacidade de adaptação a mudanças nos padrões de tráfego e dificulta a aplicação em cenários dinâmicos e não estacionários (DUA et al., 2019). Em ambientes reais, a obtenção de rótulos é custosa e demorada, tornando inviável o uso de métodos supervisionados em larga escala. Além disso, algoritmos tradicionais geralmente não são projetados para atualização contínua a partir de dados que chegam em tempo real, pois assumem que todos os dados estão disponíveis previamente, em um esquema de aprendizado em lote, o que não reflete a natureza dos FCDs (GAMA, 2010).

Nesse cenário, SOM destaca-se como alternativa promissora devido à sua capacidade

de aprendizado contínuo, redução de dimensionalidade e visualização de dados complexos (KOHONEN, 1988). Contudo, a topologia fixa em SOMs tradicionais impõe restrições à sua adaptabilidade, tornando-os menos adequados para ambientes em que padrões de tráfego e distribuições de dados mudam ao longo do tempo. A necessidade de pré-definir o tamanho e a estrutura da rede pode resultar em modelos subdimensionados ou superdimensionados, o que impacta negativamente a acurácia e a eficiência do processo de classificação (ALAHAKOON; HALGAMUGE; SRINIVASAN, 2000).

Para superar essas limitações, o Mapa Auto-Organizável Crescente (GSOM) surge como uma solução inovadora. O GSOM é capaz de expandir dinamicamente sua estrutura durante o treinamento, permitindo a incorporação de novos neurônios à medida que padrões emergentes são detectados nos dados (ALAHAKOON; HALGAMUGE; SRINIVASAN, 2000). Essa característica torna a GSOM adequada para cenários de FCDs, em que a distribuição dos dados pode variar significativamente ao longo do tempo, exigindo modelos flexíveis e adaptativos.

Além disso, GSOM pode ser treinado de forma incremental, ajustando seus pesos e sua estrutura sem a necessidade de retreinamento completo, o que reduz o custo computacional e a latência na resposta a novas ameaças ou mudanças de conceito. Essa abordagem é particularmente relevante em ambientes de segurança de redes, onde a detecção rápida e precisa de padrões anômalos é fundamental para a prevenção de ataques e a garantia da integridade dos sistemas.

Para avaliar a adaptabilidade do GSOM em cenários dinâmicos, este trabalho propõe uma abordagem experimental dividida em fases *offline* (treinamento inicial) e *online* (adaptação contínua), com base em dados reais coletados em períodos distintos. Os detalhes metodológicos e os resultados são apresentados nos Capítulos 7 e 8.

Diante desse contexto, este trabalho propõe investigar e adaptar GSOM para a classificação de logs de *firewall* em FCDs, avaliando sua robustez, capacidade de adaptação e desempenho em comparação com métodos supervisionados consolidados. Embora esses métodos também possam ser utilizados em cenários de fluxo, eles dependem da disponibilidade de rótulos durante a atualização do modelo, o que nem sempre condiz com o cenário considerado neste trabalho. Assim, busca-se contribuir para a análise de segurança em redes por meio de uma abordagem adequada a ambientes dinâmicos.

1.2 Objetivos

O objetivo principal deste trabalho é investigar a aplicação do GSOM na classificação de logs de *firewall* em FCDs, utilizando logs reais de um ambiente acadêmico. Os objetivos específicos incluem:

- Realizar uma análise temporal com base em conjuntos de dados coletados em agosto de 2024, fevereiro de 2025 e outubro de 2025, com o objetivo de investigar possíveis

mudanças no conceito do padrão de tráfego ao longo do tempo.

- Avaliar o desempenho de classificação de logs de *firewall* utilizando GSOM em um cenário multiclasse, considerando ações como

Capítulo 2

Segurança em Redes

A segurança em redes de computadores representa um desafio diante do volume e da complexidade do tráfego de dados em ambientes digitais. A necessidade de proteger informações sensíveis e garantir a privacidade dos usuários exige mecanismos eficazes para a análise de vulnerabilidades, a detecção de intrusões e o controle de acesso. Este capítulo apresenta os principais conceitos e desafios da segurança em redes, estabelecendo as bases teóricas para as estratégias abordadas neste trabalho.

2.1 Conceitos fundamentais

A segurança em redes de computadores é essencial diante do crescente volume de dados sensíveis que trafegam diariamente pela internet. Transações financeiras, e-mails e o armazenamento de dados pessoais e corporativos são exemplos de atividades que dependem da segurança das redes para garantir a confidencialidade, integridade e disponibilidade das informações (CHEHRI; FOFANA; YANG, 2021). O principal objetivo da segurança em redes é proporcionar um ambiente em que os usuários possam operar com liberdade, sem o temor de ver seus direitos e interesses comprometidos (MICHAEL et al., 2019). Para isso, é necessário compreender que a segurança de redes de computadores é um processo contínuo que exige monitoramento e atualização constantes de políticas e práticas de segurança para proteger dados confidenciais contra ameaças em constante evolução (SCARFONE; MELL, 2007).

2.2 Vulnerabilidades de Sistemas em Rede

As vulnerabilidades de sistemas em rede são fraquezas presentes no software, no hardware, nas políticas de segurança e até mesmo no comportamento dos usuários, que podem ser exploradas por agentes maliciosos para obter acesso não autorizado, comprometer a integridade dos dados ou interromper serviços (KIZZA, 2014). Essas vulnerabilidades não se limitam a questões técnicas, mas também abrangem falhas humanas e organizacionais, o que torna o cenário de segurança ainda mais complexo. Entre as principais fontes de vulnerabilidades destacam-se:

- **Falhas no projeto**: Erros de arquitetura ou implementação de sistemas e protocolos podem gerar vulnerabilidades de difícil correção;
- **Gestão de segurança ineficiente**: Ausência de políticas claras, atualização irregular de sistemas e falta de monitoramento;
- **Implantação incorreta de serviços**: Configurações inadequadas de *firewalls*, servidores e aplicações;
- **Vulnerabilidades tecnológicas**: Softwares desatualizados, falhas não corrigidas e uso de tecnologias ultrapassadas;
- **Evolução das técnicas de ataque**: Métodos de invasão tornam-se cada vez mais sofisticados, exigindo atualização constante das defesas;
- **Dificuldade na correção de vulnerabilidades**: A demora na aplicação de atualizações e correções de segurança deixa sistemas vulneráveis por longos períodos;
- **Limitações das soluções de segurança**: Mesmo ferramentas avançadas podem apresentar falhas ou cobertura insuficiente para todas as ameaças;
- **Engenharia social**: Técnicas de manipulação psicológica são utilizadas para obter informações confidenciais ou para obter acesso indevido.

Além disso, o ambiente de rede está sujeito a ataques como Ataque de Negação de Serviço Distribuído (DDoS), exploração de vulnerabilidades em aplicações web, infecção por *malware* e roubo de credenciais. Esses ataques podem causar desde indisponibilidade temporária até perda irreparável de dados sensíveis, prejuízos financeiros e danos à reputação das organizações (ASLAN et al., 2023).

O combate eficaz a essas ameaças requer um processo contínuo de segurança, que inclui: monitoramento permanente, atualização frequente de sistemas, treinamento de usuários, revisão periódica de políticas e adoção de camadas adicionais de proteção, como Sistemas de Detecção de Intrusão (IDS). O conhecimento das táticas

utilizadas por intrusos e a capacidade de adaptação às novas ameaças são fundamentais para manter a integridade e confidencialidade dos dados em ambientes de rede (KIZZA, 2014).

2.3 Sistema de Detecção de Intrusão

Os Sistemas de Detecção de Intrusão (IDS) são ferramentas fundamentais para a proteção de redes de computadores, monitorando atividades maliciosas e tentativas de acesso não autorizado (KHRAISAT et al., 2019). Os IDS podem ser classificados em dois grandes grupos:

- **Sistema de Detecção de Intrusão baseado em Assinatura (SIDS):** identificam ataques conhecidos ao comparar o tráfego de rede ou eventos do sistema com um conjunto de padrões previamente definidos (assinaturas) (KHRAISAT et al., 2019);
- **Sistema de Detecção de Intrusão baseado em Anomalias (AIDS):** detectam desvios em relação ao comportamento considerado normal, utilizando técnicas estatísticas ou métodos de aprendizado de máquina para identificar atividades potencialmente suspeitas, mesmo que desconhecidas previamente (KHRAISAT et al., 2019).

Além da classificação por método de detecção, os IDS também podem ser categorizados de acordo com o local de análise dos dados, sendo as principais abordagens:

- **Sistema de Detecção de Intrusões baseado em Host (HIDS):** Monitora dados provenientes do próprio *host*, como logs do sistema operacional, logs de aplicações e registros de auditoria. O HIDS é capaz de detectar ataques internos, incluindo ações de usuários privilegiados, e pode analisar tráfego criptografado, pois opera após a decodificação dos dados. No entanto, exige instalação em cada *host* e pode consumir recursos locais (KHRAISAT et al., 2019);
- **Sistema de Detecção de Intrusões baseado em Rede (NIDS):** Analisa o tráfego de rede em tempo real, capturando e examinando pacotes transmitidos pela rede. O NIDS pode monitorar múltiplos *hosts* simultaneamente e identificar ataques externos antes que se espalhem. Entretanto, enfrenta dificuldades para analisar grandes volumes de tráfego em redes de alta velocidade e não consegue inspecionar tráfego criptografado antes da decodificação (KHRAISAT et al., 2019).

2.4 *Firewalls*

Em tecnologia da informação, o termo *firewall* é utilizado para designar um conjunto de sistemas e equipamentos que implementam mecanismos de proteção de perímetro entre redes. Esses sistemas são projetados para atuar como uma barreira entre uma rede confiável (*trusted*) e uma rede não confiável (*untrusted*), como a Internet. O *firewall* é normalmente instalado em pontos estratégicos, nas fronteiras entre as redes, de modo a possibilitar o gerenciamento de todo e qualquer tráfego, aplicando as restrições definidas em sua configuração (ALSAQOUR; MOTMI; ABDELHAQ, 2021).

A Figura 1 ilustra esse conceito, mostrando o *firewall* posicionado entre a Internet e a rede interna, atuando como barreira de proteção e controlando o tráfego de dados entre esses ambientes. Dessa forma, o *firewall* auxilia na redução do risco de ataques a dispositivos localizados em uma rede protegida, controlando quais dispositivos podem estabelecer sessões de comunicação com outros dispositivos em redes externas. Seu funcionamento é fundamental para proteger a confidencialidade, a integridade e a disponibilidade dos dados em uma rede.

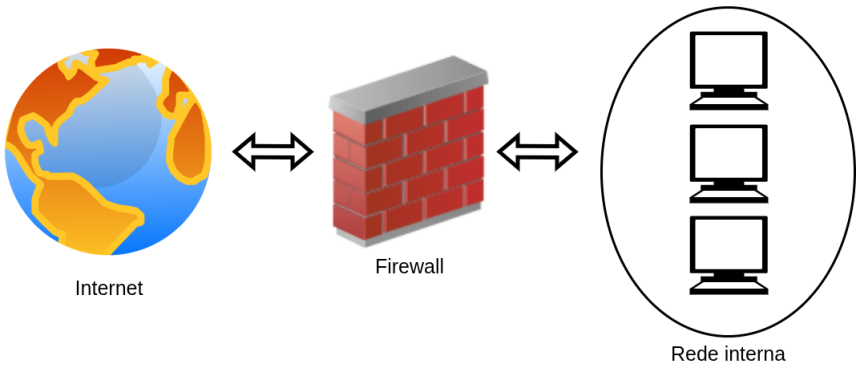


Figura 1 – Posicionamento e função de um *firewall*

Fonte: Elaborado pelo autor.

2.4.1 Evolução dos *Firewalls*

Ao longo do tempo, diferentes arquiteturas e funcionalidades de *firewalls* foram desenvolvidas para responder à crescente complexidade do tráfego e à sofisticação das ameaças digitais. Atualmente, existem diversas categorias de *firewalls*, como filtragem de pacotes, inspeção de estado, firewalls de aplicação e *Firewalls* da Próxima Geração (NGFW), cada uma com características, vantagens e limitações específicas. A seguir, são apresentadas as principais gerações de *firewalls* e suas características (GUPTA, 2024).

Primeira Geração: *Firewalls* de Filtragem de Pacotes: Os *firewalls* de filtragem de pacotes monitoram o tráfego que circula entre diferentes segmentos de rede, tomando decisões de bloqueio ou permissão com base em políticas estabelecidas que consideram endereços IP de origem e de destino, protocolos e portas utilizados. Atuam nas camadas de rede e de transporte do modelo OSI, ou seja, avaliam apenas as informações presentes nos cabeçalhos dos pacotes, sem inspecionar o conteúdo transmitido.

As regras de filtragem aplicadas por esses dispositivos são organizadas em listas que definem, de forma sequencial, qual ação deve ser tomada para cada pacote avaliado. A ordem dessas regras pode afetar o resultado da filtragem, exigindo atenção ao planejamento das políticas para evitar conflitos ou comportamentos inesperados. Caso um pacote não atenda a nenhum dos critérios definidos, é comum adotar uma postura preventiva de bloquear o tráfego por padrão, descartando o pacote por segurança (GOEL; KUMAR; RAJA, 2014).

Segunda Geração: Firewalls de Inspeção com Estado: Os *firewalls* de inspeção com estado representam uma evolução significativa na proteção de redes, pois, além de analisar os cabeçalhos dos pacotes, acompanham o estado das conexões estabelecidas. Isso significa que o *firewall* mantém uma tabela dinâmica que registra informações sobre as sessões ativas, como os endereços IP envolvidos, as portas utilizadas e o status da comunicação. Dessa forma, esses sistemas conseguem identificar se um pacote faz parte de uma conexão válida ou se é uma tentativa maliciosa de acesso, aumentando a segurança contra ataques que tentam explorar falhas na sequência ou autenticidade dos pacotes.

Esse acompanhamento detalhado das sessões permite que o *firewall* tome decisões mais inteligentes, liberando automaticamente o tráfego de resposta e bloqueando pacotes suspeitos fora do contexto. Essa abordagem reduz falsos positivos e melhora a eficiência da filtragem em comparação com modelos que avaliam apenas pacotes isolados. Porém, a necessidade de manter e consultar constantemente a tabela de estados exige que o sistema seja eficiente no gerenciamento desses dados, para não comprometer o desempenho da rede, especialmente em ambientes com alto volume de tráfego (ROECKL; DIRECTOR, 2004).

Terceira Geração: *Firewalls* de Aplicação e Proxy: Os *firewalls* de aplicação atuam no nível mais alto do modelo OSI, inspecionando o conteúdo efetivo das comunicações entre usuários e aplicações. Essa inspeção detalhada permite que esses *firewalls* detectem ameaças específicas a protocolos e aplicativos, como ataques de injeção SQL, adulteração de parâmetros e *scripts* maliciosos, protegendo diretamente serviços como HTTP, FTP e e-mail. Eles aplicam políticas configuradas para permitir ou bloquear tráfego com base na

análise dos dados da aplicação, oferecendo controle sobre as interações entre clientes e servidores.

Frequentemente implementados como *firewalls proxy*, esses dispositivos atuam como intermediários entre o usuário e o servidor de destino, estabelecendo conexões separadas e filtrando o tráfego de forma mais criteriosa. Embora essa abordagem proporcione maior segurança e controle, ela pode implicar maior uso de recursos e introduzir alguma latência, especialmente em redes com tráfego intenso (BASILE; LIOY, 2013).

***Firewalls* de Próxima Geração:** Os *Firewalls* da Próxima Geração (NGFW)s representam uma evolução em relação às gerações anteriores, pois integram diversas funcionalidades avançadas em um único dispositivo. Além da filtragem de pacotes e da inspeção por estado, esses equipamentos oferecem recursos como Inspeção Profunda de Pacotes (DPI), IDS, controle e identificação de aplicações, filtragem de *URLs*, proteção contra *malwares* e inspeção de tráfego criptografado (SSL/TLS). Essa combinação permite analisar o tráfego de forma mais detalhada e eficaz, identificando tentativas sofisticadas de ataque que poderiam passar despercebidas a soluções convencionais.

Outra característica marcante dos NGFWs é a possibilidade de gerenciar políticas de segurança de forma centralizada, com automação das respostas a incidentes e integração com sistemas de identificação de usuários. Isso permite aplicar regras específicas com base em usuários, grupos ou dispositivos, além de manter atualizações automáticas diante de novas ameaças. Esses *firewalls* exigem configuração cuidadosa, atualização contínua e, em alguns casos, infraestrutura mais robusta para lidar com o alto volume de inspeção e processamento de dados de rede (GUPTA, 2024).

2.4.2 Funcionamento do *Firewall* utilizado no estudo

O *firewall* utilizado neste estudo é classificado como um NGFW, incorporando funcionalidades de DPI, filtragem de URL, análise de ameaças avançadas, identificação de usuários, controle de aplicações, inspeção de tráfego criptografado, monitoramento de sessões, entre outros recursos modernos (GUPTA, 2024). Entretanto, a abordagem deste trabalho está restrita à análise dos logs de tráfego, referentes às regras de filtragem de acesso estabelecidas pelo administrador. A partir desses registros, é possível monitorar e avaliar eventos como conexões permitidas, bloqueadas ou descartadas, independentemente das demais capacidades do NGFW.

Essas regras de filtragem, definidas conforme a política de segurança da organização, especificam como o tráfego deve ser tratado, por meio de ações como "*Allow*",

"*Deny*", "*Reset-Both*" ou "*Drop*" (ALJABRI et al., 2022). As regras são aplicadas com base em critérios como:

- Endereços IP: Permite ou bloqueia o tráfego com base no endereço IP de origem ou destino;
- Portas e Protocolos: Controla o acesso a serviços específicos, como HTTP (porta 80) ou SSH (porta 22);
- Conteúdo dos Pacotes: Analisa o conteúdo dos pacotes para identificar atividades maliciosas, como vírus e tentativas de exploração de vulnerabilidades.

A efetividade das ações de filtragem aplicadas pelo *firewall* está diretamente relacionada à qualidade e à precisão das regras configuradas pelo administrador do equipamento. Regras de filtragem excessivamente permissivas podem permitir atividades não autorizadas, o que pode evidenciar falhas de segurança. Por outro lado, regras excessivamente restritivas podem comprometer o funcionamento normal dos serviços da rede, ocasionando bloqueios indevidos e dificultando o acesso a recursos. Portanto, para alcançar um equilíbrio entre proteção e funcionalidade, é fundamental revisar e ajustar regularmente as regras implementadas, observando tendências de ameaças e os padrões identificados nos próprios logs do sistema analisado.

As ações de um *firewall* são geralmente categorizadas em quatro classes principais (ALJABRI et al., 2022):

- *Allow*: Permite que o tráfego passe pelo *firewall*, desde que esteja em conformidade com as regras de segurança. Exemplo: Permitir o acesso a um servidor web na porta 80;
- *Deny*: Bloqueia o tráfego e notifica o remetente de que a conexão foi rejeitada. Exemplo: Bloquear tentativas de acesso a uma porta não utilizada;
- *Reset-Both*: Envia um pacote de redefinição TCP para ambos os lados da conexão, interrompendo-a imediatamente. Exemplo: Encerrar uma conexão suspeita de ser um ataque de negação de serviço;
- *Drop*: Ignora o tráfego em silêncio, sem notificar o remetente. Exemplo: Descartar pacotes de um ataque de varredura de portas.

2.5 Considerações finais

Este capítulo apresentou os conceitos e desafios da segurança em redes, com ênfase na identificação de vulnerabilidades, no funcionamento dos IDS e no papel dos *firewalls* como barreiras de proteção. Embora os IDS sejam fundamentais para a identificação de ameaças, este trabalho concentra-se na análise de logs de *firewall*

como a principal fonte de dados para a investigação de padrões de tráfego e a identificação de comportamentos anômalos.

A escolha por analisar logs de *firewall*, em vez de dados provenientes de IDS, deve-se ao fato do *firewall* ser o principal sistema de proteção utilizado pela instituição de ensino onde a pesquisa foi realizada. Além disso, os logs de *firewall*, gerados a partir das regras de filtragem aplicadas, proporcionam uma visão abrangente e confiável do tráfego de rede, essencial para o desenvolvimento e validação das técnicas propostas neste trabalho. Essa abordagem, amplamente adotada na literatura, também permite comparações com estudos similares.

Capítulo 3

Fluxos Contínuos de Dados

O processamento de grandes volumes de dados em tempo real tornou-se um desafio para a segurança de redes. A necessidade de analisar e classificar eventos em alta velocidade, com dados que chegam de forma contínua e não estacionária, exige técnicas capazes de lidar com fluxos dinâmicos e padrões em constante evolução. Este capítulo apresenta os principais conceitos e desafios associados aos Fluxo Contínuo de Dados (FCDs), destacando as abordagens mais relevantes para a análise e a classificação em ambientes de dados em fluxo, e fundamentando as estratégias empregadas neste trabalho.

3.1 Conceitos fundamentais

O Aprendizado de Máquina (AM) é uma área da IA que se preocupa com o desenvolvimento de algoritmos capazes de melhorar o desempenho de sistemas computacionais em tarefas específicas, com base em experiência adquirida com dados anteriores. Esses algoritmos aprendem a identificar padrões, realizar previsões e tomar decisões automáticas, ajustando seus modelos à medida que novos dados são disponibilizados (MITCHELL, 1997).

Durante muito tempo, o AM concentrou-se em algoritmos de aprendizado *offline*, em que os conjuntos de dados de treinamento estão sempre disponíveis para a geração de um modelo que aprende a partir deles. Além disso, esses modelos são estáticos, não sendo atualizados com novos dados (GAMA, 2010).

Segundo Guha e Mishra (2016), os avanços recentes em *software* e *hardware* possibilitaram a aquisição de dados em grande escala, caracterizando ambientes dinâmicos,

enquanto as bases de dados tradicionais supõem cenários estáticos. Nesse contexto, as inovações tecnológicas têm favorecido o surgimento de inúmeras aplicações do mundo real que se alinham a essas características. Alguns exemplos típicos incluem:

- **Cidades inteligentes:** sistemas monitoram dados urbanos em tempo real e se adaptam automaticamente a mudanças no ambiente, como alterações no tráfego ou no clima, garantindo respostas mais precisas (AL-MALIKI et al., 2022);
- **Indústria 4.0:** sensores e IA analisam séries temporais para prever falhas em máquinas, otimizar a produção e detectar anomalias, reduzindo custos e paradas inesperadas (KASHPRUK; PISKOR-IGNATOWICZ; BARANOWSKI, 2023);
- **Meio ambiente:** sensores de baixo custo monitoram continuamente a qualidade do ar e usam técnicas de aprendizado de máquina para recalibrar automaticamente os modelos quando ocorrem mudanças nas condições ambientais, garantindo medições mais precisas e confiáveis (D'ELIA et al., 2024);
- **Jogos *online*:** sistemas utilizam inteligência artificial para analisar o comportamento de jogadores em tempo real, detectar trapaças e prever abandono de partidas, personalizando a experiência e aumentando a segurança nas plataformas (DONTIREDDY et al., 2024);
- **Sistemas de segurança:** utilizam câmeras que enviam imagens em tempo real para sistemas que analisam e identificam automaticamente movimentos, pessoas ou situações suspeitas, facilitando a detecção de possíveis ameaças e reduzindo o armazenamento ao salvar só o que é realmente importante (DUTT; KALRA, 2016);
- **Redes de Computadores:** sistemas analisam continuamente o tráfego da rede para identificar padrões incomuns e detectar automaticamente acessos suspeitos, ajudando a prevenir ataques e falhas de segurança (MOHAMMED; AKAY, 2023);
- **Mercado Financeiro:** sistemas analisam continuamente dados de transações financeiras para detectar fraudes em cartões de crédito e operações suspeitas, protegendo investidores e instituições (JHA; SIVASANKARI; VENUGOPAL, 2020);
- **Medicina:** sistemas de inteligência artificial monitoram continuamente dados de pacientes em Unidade de Terapia Intensiva (UTI) para detectar riscos, como infecções ou falhas de órgãos, gerando alertas em tempo real e ajudando a equipe médica a agir mais rapidamente (KHAN, 2025).

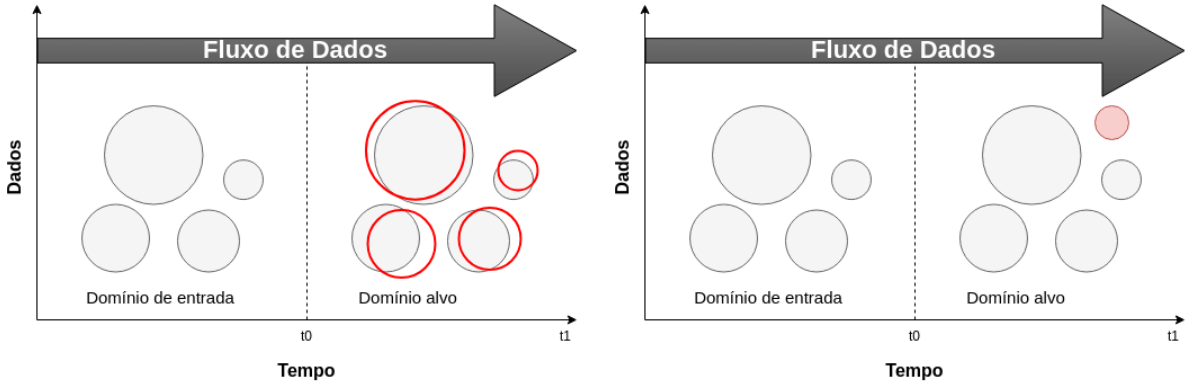
Conforme destacado por diversos autores na literatura, as características dos FCDs incluem (PAIVA, 2014; BABCOCK et al., 2002; AGGARWAL, 2007; GAMA, 2010):

- Os dados apresentam uma distribuição não-estacionária, indicando que as características dos dados se alteram ao longo do tempo;
- A chegada de dados ocorre de forma contínua e, geralmente, em alta velocidade;
- O sistema não possui controle sobre a ordem de chegada dos exemplos;
- O fluxo de dados pode ser considerado infinito, o que implica que os exemplos não podem ser armazenados em memória, sendo necessário processar cada exemplo apenas uma vez;
- O sistema deve estar apto a realizar predições a qualquer momento.

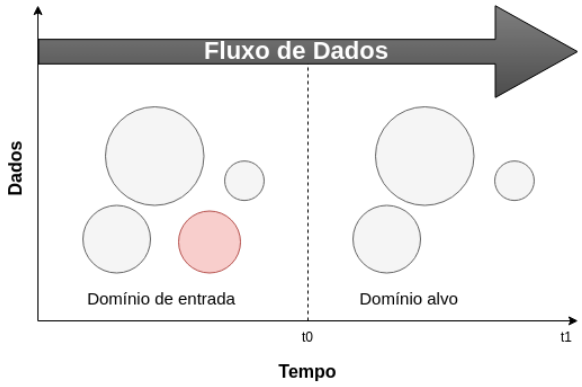
Essas características não permitem que algoritmos tradicionais de Mineração Dados (MD) sejam facilmente aplicados em FCDs (MAHDIRAJI, 2009). Para superar essas limitações e explorar efetivamente os FCDs, é fundamental compreender outros termos neste contexto. Esses termos incluem:

- **Mudança de conceito**: ocorre quando a relação entre os atributos de entrada e a classe de saída se altera ao longo do tempo. (ELWELL; POLIKAR, 2011). No contexto deste trabalho, que envolve a classificação de dados em quatro classes predefinidas, a mudança de conceito ocorre quando os padrões dos dados em uma classe se alteram, conforme ilustrado na Figura 2a. O algoritmo proposto é capaz de detectar essas mudanças e ajustar o modelo de decisão para manter a acurácia das predições. Para isso, ele utiliza técnicas de aprendizado incremental, que permitem atualizar o modelo à medida que novos dados chegam, sem a necessidade de retreinamento completo. Essa abordagem garante que o algoritmo se adapte dinamicamente às variações na distribuição dos dados, mantendo a eficiência e a precisão ao longo do tempo.
- **Evolução de conceito**: ocorre quando o número de classes sofre alterações ao longo do fluxo e novas classes podem surgir, conforme ilustrado na Figura 2b, o que representa uma evolução (MASUD et al., 2010). No contexto deste trabalho, a evolução de conceito não é aplicável, pois o número de classes é fixo em quatro e não há surgimento ou desaparecimento de novas classes ao longo do fluxo de dados. As quatro classes são predefinidas e permanecem constantes ao longo de todo o processo. O algoritmo proposto não precisa lidar com a criação ou remoção de classes, o que simplifica sua implementação e foca sua capacidade de adaptação às mudanças de conceito.
- **Esquecimento de conceitos**: consiste em esquecer conceitos desatualizados que não têm utilidade na atividade atual do fluxo e ocupam espaço, conforme

ilustrado na Figura 2c (ABDALLAH et al., 2016). No contexto deste trabalho, o esquecimento de conceito não é relevante, pois as quatro classes são sempre válidas e não há necessidade de descartar informações antigas. O foco do algoritmo é manter o modelo atualizado com base nas mudanças na distribuição dos dados dentro das classes existentes.



(a) Mudança de Conceito (b) Evolução de Conceito



(c) Esquecimento de Conceito

Figura 2 – Mudanças nos fluxos de dados

Fonte: Adaptado de Abdallah et al. (2016).

FCDs podem ser definidos como um conjunto D contendo exemplos multidimensionais $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n, \ldots$ que é potencialmente infinito ($n \to \infty$). Os dados chegam em instantes de tempo $t_1, t_2, \ldots, t_n, \ldots$, e cada exemplo é descrito por um vetor de atributos de dimensão d $\mathbf{x}_i = (x_{i1}, x_{i2}, \ldots, x_{id})$ (AGGARWAL et al., 2003).

Diante disso, novas estruturas de dados e técnicas de mineração em FCDs são necessárias, pois as abordagens tradicionais não são adequadas para lidar com a natureza dinâmica e o volume dos dados de fluxo contínuo (MAHDIRAJI, 2009). Aggarwal (2007) salienta que os dois principais desafios da mineração em FCDs são: o processamento rápido dos dados de forma incremental e lidar com mudanças na distribuição dos dados. Sobre o primeiro desafio, alguns algoritmos tradicionais, como o KNN e o *Naive Bayes* (NB) (LANGLEY et al., 1992), são, por natureza, incre-

mentais. Outros, como Árvore de Decisão (AD) (MAIMON; ROKACH, 2014) e SVM (CORTES; VAPNIK, 1995), não apresentam essa capacidade e necessitam de adaptações para esse propósito. Contudo, embora algoritmos incrementais sejam necessários, não são suficientes, dada a distribuição não estacionária dos dados nos sistemas com FCDs (GAMA, 2010). Para contornar esse problema, os algoritmos de AM devem ser capazes de detectar e se adaptar a tais mudanças, não apenas incorporando novas informações ao modelo de decisão, mas também eliminando informações desatualizadas (NGUYEN; WOON; NG, 2015).

3.2 Agrupamento em Fluxo Contínuo de Dados

O processo de agrupamento de dados envolve a identificação de grupos que descrevam um conjunto de dados (FAYYAD et al., 1996), procurando agrupar dados que sejam semelhantes entre si e maximizar a diferença entre os diferentes grupos.

O agrupamento em FCDs requer algoritmos capazes de agrupar os dados de forma contínua, respeitando as restrições de tempo e de memória. Gama (2010) destaca que a dificuldade reside na alta velocidade e na distribuição não estacionária dos dados. Tendo em mente essas restrições, algoritmos para o agrupamento em FCDs devem apresentar as seguintes propriedades (SILVA et al., 2013):

- Prover os resultados em tempo viável, realizando processamento de maneira rápida e incremental;
- Adaptar-se rapidamente à dinâmica dos dados, *i.e.*, os algoritmos devem detectar quando novos grupos aparecem ou desaparecem;
- Ser escalável ao número de dados que surgem continuamente;
- Prover um modelo que não seja apenas compacto, mas que também o seu tamanho não cresça com o número de dados processados;
- Rapidamente detectar *outliers* e reagir de forma adequada;

Em geral, o desafio do agrupamento em FCDs é manter um agrupamento eficiente dos dados processados, considerando a limitação de memória disponível. A quantidade de memória necessária pode ser afetada pelo volume de dados, pela taxa de chegada de novos dados e pela complexidade dos algoritmos utilizados. Dessa maneira, vários algoritmos de agrupamento em FCDs foram propostos na literatura para abordar esses desafios (SILVA et al., 2013).

Para contornar o problema de memória e de espaço limitado, preservando ao mesmo tempo os dados originais, Silva et al. (2013) destaca o uso de estruturas de sumarização. As estruturas mais comuns na literatura incluem o Vetor de Características, o Vetor de Protótipos, a Árvore de Conjunto de Objetos Representativos e a Malha.

Entre os algoritmos de agrupamento propostos para FCDs, métodos baseados em SOMs têm recebido atenção por sua capacidade de organizar os dados preservando relações topológicas. Nesse contexto, Galete (2012) apresentou um método de agrupamento em FCDs, utilizando o algoritmo de agrupamento Mapa Auto-Organizável (SOM) com base nas etapas de pré-processamento, processamento e pós-processamento.

Na etapa de pré-processamento, o fluxo contínuo de dados é tratado por meio de uma fila do tipo *First-In, First-Out* (FIFO), que organiza a chegada dos exemplos. Técnicas como amostragem e agregação podem ser aplicadas para reduzir a sobrecarga. Os dados são processados por meio de uma janela de leitura, que define o intervalo de dados a serem encaminhados ao agrupamento.

Na etapa de processamento, o agrupamento é realizado por meio do algoritmo SOM, escolhido por sua capacidade de identificar automaticamente diferentes tipos de agrupamentos, como grupos bem separados, densos ou com formatos variados, sem a necessidade de intervenção de um especialista na definição prévia dos grupos.

No pós-processamento, os grupos formados são avaliados por meio de técnicas de validação não supervisionada, como o coeficiente de silhueta, que mede a qualidade dos agrupamentos quanto à coesão e à separação entre os grupos.

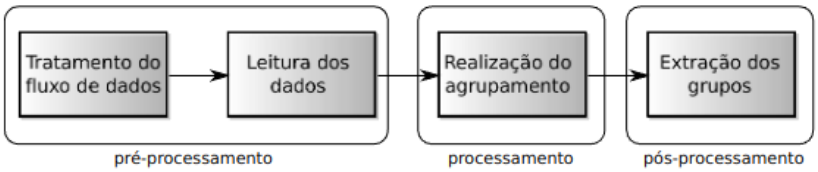


Figura 3 – Etapas para agrupamento em FCDs

Fonte: (GALETE, 2012).

Além do método apresentado por Galete (2012), a literatura de agrupamento em FCDs mostra que uma parcela expressiva dos algoritmos adota uma arquitetura em duas fases: uma fase *online*, responsável por processar continuamente o fluxo e manter estruturas resumidas dos dados, e uma fase *offline*, encarregada de gerar ou refinar os agrupamentos a partir dessas estruturas. Esse tipo de organização busca conciliar restrições de memória e tempo de resposta com a necessidade de representar a evolução temporal dos dados (AGGARWAL et al., 2003; GHESMOUNE; LEBBAH; AZZAG, 2016).

Nesse contexto, diferentes estratégias têm sido propostas. No CluStream, os dados do fluxo são resumidos continuamente na forma de microgrupos durante a fase *online*. Depois, esses microgrupos podem ser utilizados em uma etapa *offline* para formar os agrupamentos finais (AGGARWAL et al., 2003). De modo semelhante, o DenStream também atualiza estruturas ao longo do fluxo, mas com maior foco em

lidar com ruído e em distinguir grupos mais estáveis de exemplos isolados (CAO et al., 2006).

Embora esses métodos sejam adequados para resumir grupos no fluxo, abordagens baseadas em SOM se destacam por organizar padrões semelhantes em regiões próximas do mapa, facilitando a visualização dos dados. Isso também permite observar mudanças ao longo do tempo, como surgimento, desaparecimento ou deslocamento de grupos (DENNY; WILLIAMS; CHRISTEN, 2010; LICEN; ASTEL; TSAKOVSKI, 2023).

No contexto deste trabalho, essa característica é especialmente relevante, pois o interesse não está apenas em formar agrupamentos, mas também em associar regiões do mapa às ações do *firewall* e acompanhar como essa organização se altera ao longo do fluxo. Por isso, mapas auto-organizáveis são atrativos, já que ajudam a organizar padrões semelhantes em regiões próximas e tornam essa estrutura mais fácil de visualizar e interpretar. Além disso, variantes com crescimento dinâmico, como o GSOM, são adequadas quando a complexidade dos dados não é conhecida previamente, pois permitem expandir o mapa conforme a necessidade de representação (ALAHAKOON; HALGAMUGE; SRINIVASAN, 2000; GALETE, 2012).

3.3 Classificação em Fluxo Contínuo de Dados

A classificação em FCDs tem como objetivo prever, com alta acurácia, a classe de novos exemplos que chegam continuamente ao longo do fluxo (PAIVA, 2014). No aprendizado supervisionado, fornece-se ao algoritmo um conjunto de exemplos rotulados, o que permite a construção de um modelo capaz de identificar, de forma automática e correta, a classe dos novos exemplos à medida que o fluxo evolui e novos dados são inseridos.

Muitos algoritmos de classificação foram desenvolvidos para operar em cenários tradicionais em lote (*batch*), incluindo AD, SVM e NB. Nesses casos, o ambiente é estático, o modelo de decisão permanece inalterado, e o algoritmo pressupõe que todos os dados estão disponíveis na memória (MARKOU; SINGH, 2003a; MARKOU; SINGH, 2003b; MARSLAND; SHAPIRO; NEHMZOW, 2002; SCHOLKOPF et al., 2000; HOFFMANN, 2007). Além disso, para a construção do modelo, os algoritmos assumem que todos os dados estão na memória e, portanto, fazem várias varreduras sobre eles (PAIVA, 2014).

Em um contexto comum de classificação de fluxo, o rótulo da classe real ($y^{(t)}$) associado a cada ponto de dados $x^{(t)}$ torna-se disponível após um certo período de atraso $\delta \in [0, \infty)$, que também é denominado latência de rótulo ou latência de verificação (SOUZA et al., 2015).

Em um cenário de latência de rótulo zero, o classificador é constantemente atualizado assim que um exemplo de teste é processado. Isso implica que, quando um dado é recebido, seu rótulo real é disponibilizado para atualização do modelo (SOUZA et al., 2015). Por outro lado, na latência infinita de rótulo, os rótulos dos dados nunca chegam após a inicialização do modelo de classificação (DAS et al., 2020). Como na condição de atraso infinito não há dados rotulados durante a fase de classificação, presume-se que uma pequena quantidade de dados rotulados esteja disponível antes do início da classificação. Esses dados inicialmente rotulados fornecem uma informação geral do número de classes e dos respectivos espaços de características para apoiar a classificação (DYER; CAPO; POLIKAR, 2013).

Conforme Aggarwal (2007), o desafio mais complexo na classificação em cenários de FCDs está relacionado à mudança de conceito, ou seja, à alteração na distribuição dos dados ao longo do tempo. Além disso, é necessário abordar outros desafios, como a evolução de conceitos ao longo do tempo, a identificação e eliminação de ruídos e *outliers*, e a preservação da acurácia diante da evolução dos dados. O algoritmo precisa ser projetado para atualizar o modelo de decisão de forma incremental, permitindo um aprendizado contínuo e eficiente.

3.4 Considerações finais

O objetivo desse capítulo foi apresentar uma visão geral sobre FCDs, destacando sua importância e os desafios associados. Inicialmente, foram discutidas as características essenciais dos FCDs, incluindo a distribuição não estacionária, a chegada contínua de dados e a necessidade de processamento em tempo real.

Nas seções do capítulo, foram expostos os principais desafios de Agrupamento e Classificação em FCDs, destacando conceitos fundamentais como mudança de conceito, evolução de conceito e esquecimento de conceito. No contexto deste trabalho, o foco está exclusivamente no tratamento de mudanças de conceito, uma vez que o problema envolve quatro classes fixas. Dessa forma, não há possibilidade de surgimento de novas classes ao longo do fluxo.

No Capítulo 4 será apresentado o algoritmo SOM, que utiliza aprendizado competitivo não supervisionado para representar dados de alta dimensão de forma eficiente.

Capítulo 4

Mapas Auto-Organizáveis

Após apresentar os conceitos gerais sobre FCDs, este capítulo dedica-se a explorar os Mapas Auto-Organizáveis (SOMs), uma técnica de aprendizado não supervisionado amplamente utilizada para a redução de dimensionalidade, a visualização e o agrupamento de dados. A técnica SOM foi originalmente proposta por Kohonen (1988) e tem sido aplicado em diversas áreas, incluindo mineração de dados, reconhecimento de padrões e análise exploratória.

4.1 Conceitos fundamentais

As redes neurais, inspiradas no cérebro humano, são modelos computacionais que utilizam neurônios artificiais organizados em camadas para processar dados. Esses neurônios computam funções matemáticas com os sinais de entrada, cujo conhecimento é armazenado nos pesos das conexões, ajustados durante o aprendizado da rede (RUMELHART; HINTON; WILLIAMS, 1986).

O SOM é uma técnica de redução de dimensionalidade que busca representar dados de alta dimensão de forma eficiente, minimizando a complexidade computacional. Os SOMs são empregados na análise exploratória de dados, na resolução de problemas de agrupamento, na classificação e na visualização de conjuntos de dados complexos e multidimensionais (BRAGA; BASSANI, 2018; PONMALAI; KAMATH, 2019).

O SOM utiliza aprendizado competitivo não supervisionado para produzir mapas de características de baixa dimensão a partir de dados de entrada de alta dimensão, preservando as relações de similaridade entre os dados. Durante o treinamento

não supervisionado, o algoritmo busca encontrar o conjunto ótimo de neurônios que representam os exemplos, respeitando as restrições topológicas (MILJKOVIĆ, 2017; MELIN et al., 2020).

4.2 Estruturas dos Mapas Auto-Organizáveis

O SOM é um modelo neural que converte padrões de entrada de dimensão arbitrária em um mapa discreto, seja unidimensional, bidimensional ou até mesmo tridimensional, de forma adaptativa e topologicamente ordenada. O processo de organização adaptativa ocorre por meio de neurônios dispostos em uma grade, em que cada neurônio representa um ponto no espaço de saída do mapa. A quantidade de neurônios na grade é definida previamente e pode variar conforme as características do conjunto de dados. Na Figura 4, ilustra-se como os dados de entrada são mapeados para os neurônios da camada de saída, o que facilita a compreensão da organização dos dados no mapa de características do SOM (HAYKIN, 2009). A camada de entrada é composta por uma série de neurônios sensoriais, responsáveis por estimular a rede neural. O número de neurônios sensoriais é sempre igual ao número de atributos no conjunto de dados analisado, e todos os atributos descritivos de um conjunto de objetos são apresentados a todos os neurônios da saída da arquitetura (ROLEMBERG, 2021).

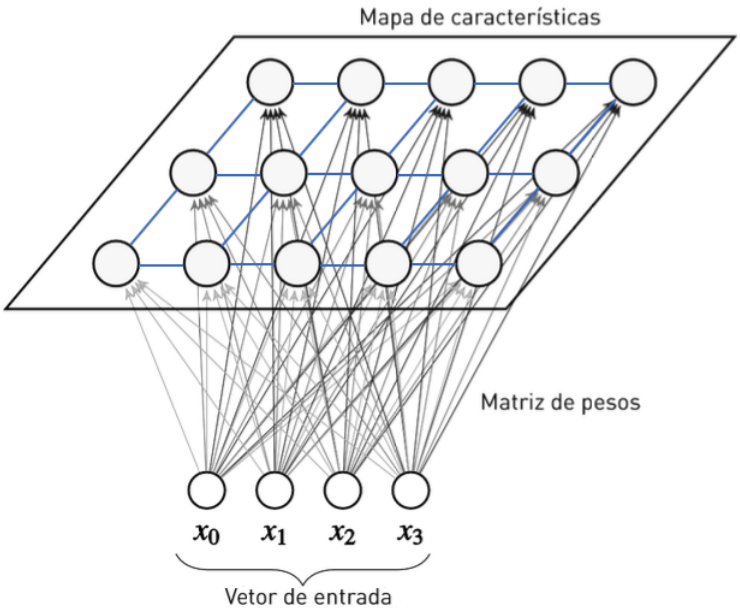


Figura 4 – Arquitetura de funcionamento do SOM

Fonte: Adaptado de Barnawi et al. (2023)

Nos SOMs, os neurônios de entrada, também conhecidos como neurônios sensoriais, requerem que suas entradas sejam normalizadas. Essa etapa de pré-processamento

é essencial para garantir que todas as características contribuam de forma equilibrada para o cálculo da distância na rede, reescalando seus valores para um intervalo comum. À medida que esses valores normalizados são introduzidos na rede, desencadeiam respostas nas camadas de saída (HEATON, 2008).

A estrutura do SOM é exemplificada na Figura 5, na qual o vetor de entrada passa pelo processo de normalização. Esse processo envolve várias etapas essenciais para o funcionamento da rede SOM. Primeiramente, o vetor de entrada é normalizado para garantir que todos os dados estejam na mesma escala. Em seguida, na primeira camada da rede, os processamentos consistem em calcular a similaridade entre o vetor de entrada e os vetores de pesos sinápticos de cada neurônio utilizando medidas de distância, como a distância Euclidiana. O neurônio com a menor distância, ou seja, o que mais se assemelha ao vetor de entrada, é identificado como o neurônio vencedor, ou Best Matching Unit (BMU). Os resultados deste processo são apresentados na camada de saída, e os neurônios vencedores e seus vizinhos passam por ajustes de pesos para melhorar o reconhecimento e a classificação dos dados de entrada.

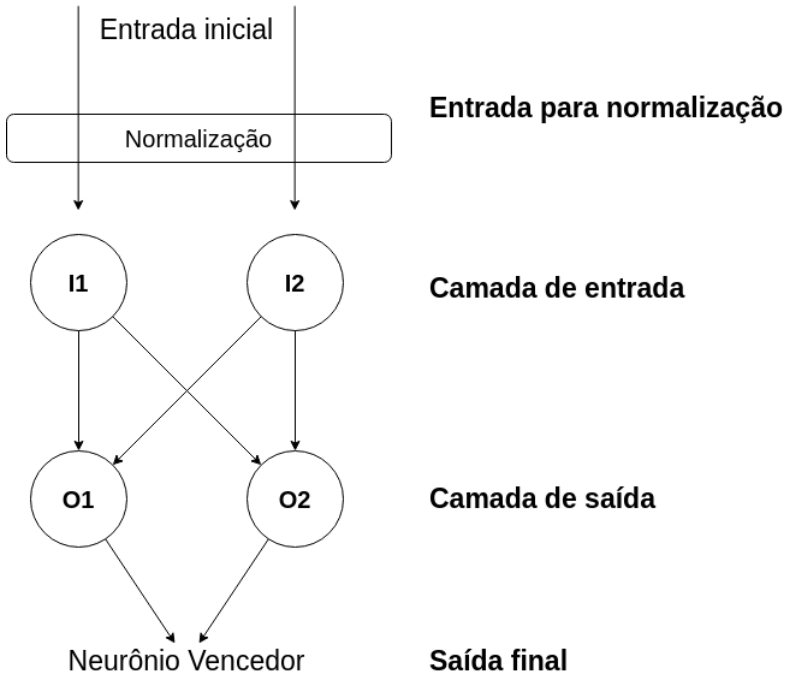


Figura 5 – Estrutura dos Mapas Auto-Organizáveis

Fonte: Adaptado de Heaton (2008).

No diagrama esquemático representado na Figura 6, o vetor de entrada contém três neurônios sensoriais, e cada neurônio sensorial está conectado a todos os neurônios da rede, recebendo as mesmas informações de entrada. Isso significa que todos os neurônios compartilham o vetor de entrada completo. Dessa forma, todos os neurônios no arranjo são representados por vetores de pesos sinápticos que recebem o mesmo dado de entrada (HAYKIN, 2009).

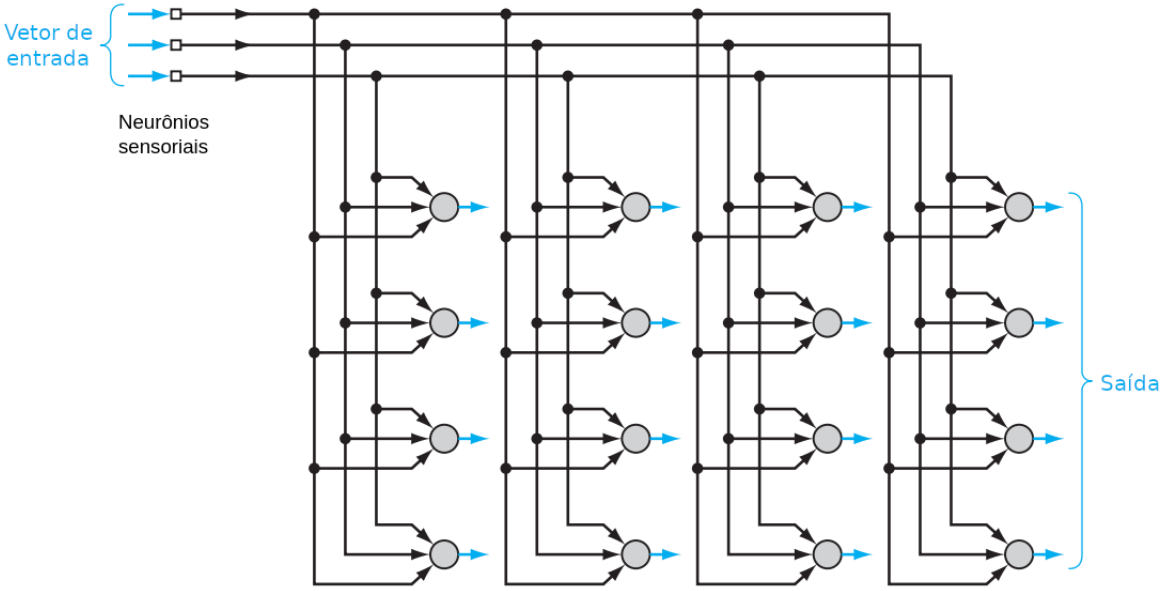


Figura 6 – Diagrama esquemático de uma rede bidimensional de neurônios

Fonte: Adaptado de Haykin (2009)

O algoritmo encarregado da formação do SOM inicia o processo inicializando os pesos sinápticos na rede. Esta etapa é realizada atribuindo pequenos valores, escolhidos a partir de um gerador de números aleatórios, aos pesos sinápticos, o que garante que nenhuma ordem prévia seja imposta ao mapa de características. Uma vez que a rede tenha sido devidamente inicializada, o processo de formação do SOM prossegue por meio de três processos essenciais: competitivo, cooperativo e de adaptação sináptica (HAYKIN, 2009).

O processo competitivo será abordado na Seção 4.2.1, o processo cooperativo na Seção 4.2.2 e a adaptação sináptica na Seção 4.2.3.

4.2.1 Processo Competitivo

O processo competitivo é responsável por identificar, para cada vetor de entrada, o neurônio vencedor da rede. Nesse processo, cada neurônio calcula sua similaridade ao exemplo apresentado, geralmente por meio de uma medida de distância entre o vetor de entrada e seus pesos sinápticos. O neurônio que apresentar a menor distância em relação ao vetor de entrada é eleito vencedor para aquele exemplo (KOHONEN, 2012).

Para quantificar essa similaridade, diferentes medidas de distância podem ser utilizadas para comparar o vetor de entrada aos vetores de pesos dos neurônios. Segundo Miljković (2017), algumas das principais métricas empregadas nos SOMs incluem:

– A distância Euclidiana entre dois vetores $\mathbf{x}$ e $\mathbf{w}$ dada por:

$$d(\mathbf{x}, \mathbf{w}) = \sqrt{\sum_{i=1}^{n}(x_i - w_i)^2} \tag{1}$$

– A Distância do Cosseno entre dois vetores $\mathbf{x}$ e $\mathbf{w}$ dada por:

$$d(\mathbf{x}, \mathbf{w}) = 1 - \frac{\mathbf{x} \cdot \mathbf{w}}{\|\mathbf{x}\|\|\mathbf{w}\|} \tag{2}$$

– A distância de Manhattan (Distância do Bloco) entre dois vetores $\mathbf{x}$ e $\mathbf{w}$ dada por:

$$d(\mathbf{x}, \mathbf{w}) = \sum_{i=1}^{n}|x_i - w_i| \tag{3}$$

Em uma aplicação real, a distância Euclidiana, conforme mostrado na Equação 1, é frequentemente utilizada como função discriminante para identificar o neurônio vencedor, aquele que apresenta a menor distância entre o vetor de entrada e seus pesos (HAYKIN, 2001).

Quando um neurônio é identificado como o vencedor, os neurônios localizados em sua vizinhança na grade também têm seus pesos ajustados. Esse processo de ajuste dos neurônios permite que a rede aprenda e se adapte aos padrões de entrada ao longo do tempo, refinando sua capacidade de reconhecimento e classificação de dados (HULLE, 2000).

Na Figura 7 é ilustrado o ajuste dos neurônios após a identificação da BMU. Os círculos escuros representam os neurônios antes do ajuste, enquanto os círculos claros representam os neurônios após o ajuste.

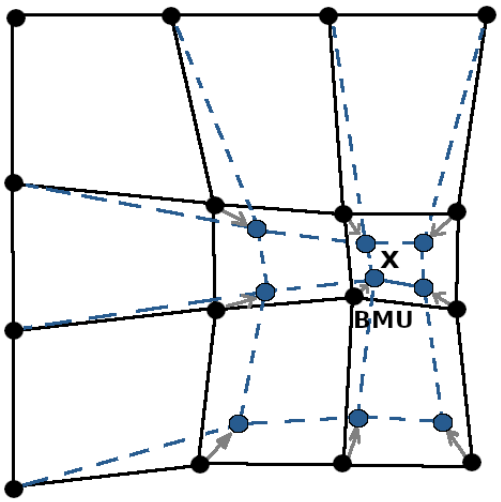


Figura 7 – Ajuste de pesos do neurônio vencedor e dos seus vizinhos

Fonte: Adaptado de Vesanto e Alhoniemi (2000).

4.2.2 Processo Cooperativo

O processo cooperativo define quais neurônios, além do vencedor, terão seus pesos sinápticos alterados.

Segundo Haykin (2009), neurobiologicamente, existe uma interação lateral entre os neurônios, de forma que o neurônio ativado tende a excitar mais fortemente os neurônios na sua vizinhança imediata do que aqueles distantes dele. Uma função de vizinhança topológica, h_{ji}, deve satisfazer duas exigências distintas, considerando d_{ij} como a distância lateral entre o neurônio vencedor e o neurônio excitado j:

- Deve ser simétrica em relação ao ponto máximo definido por $d_{ij} = 0$; em outras palavras, ela alcança o seu valor máximo no neurônio vencedor i para o qual a distância d_{ii} é zero;
- Sua amplitude deve decrescer monotonicamente com o aumento da distância lateral d_{ij}, decaindo a zero para $d_{ij} \to \infty$; esta é uma condição necessária para a convergência.

Geralmente, utiliza-se a função gaussiana, conforme a Equação 4, pois atende a esses requisitos e a amplitude da vizinhança tende a zero à medida que a distância lateral aumenta, conforme ilustrado na Figura 8.

$$h_{ij} = \alpha \exp\left(\frac{-d_{ij}^2}{2\sigma^2}\right), \tag{4}$$

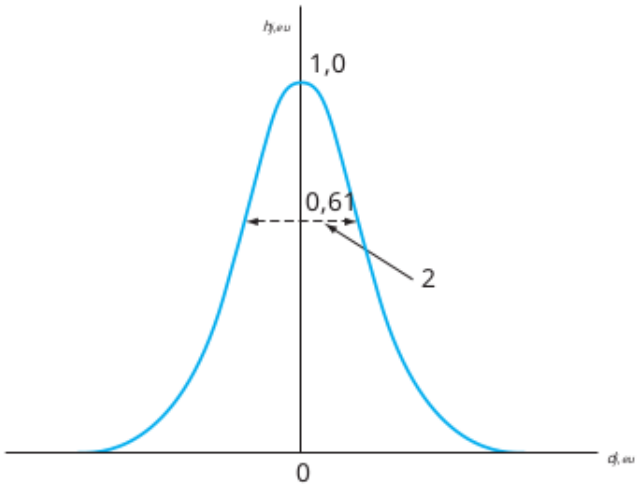


Figura 8 – Função de vizinhança gaussiana

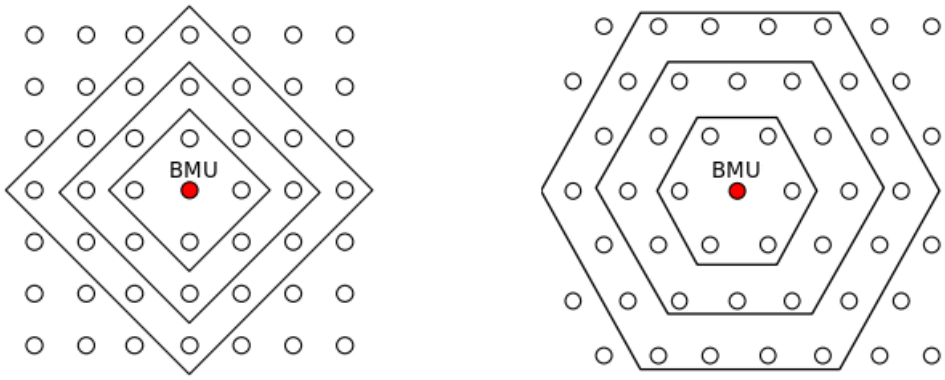
Fonte: Adaptado de Haykin (2009).

A escolha da topologia de uma rede pode influenciar diretamente o desempenho e a capacidade de representação do modelo. Entre as topologias mais utilizadas na literatura, destacam-se as formas quadradas e hexagonais (KOHONEN, 2001a).

A topologia quadrada é a mais simples de implementar, composta por neurônios organizados em linhas e colunas. É também a mais utilizada devido à sua simplicidade. Sua principal vantagem é a facilidade de visualização e indexação dos neurônios. No entanto, uma limitação dessa topologia é que cada neurônio possui apenas quatro vizinhos imediatos, o que pode dificultar o aprendizado de detalhes e de relações complexas nos dados.

Por outro lado, a topologia hexagonal apresenta cada neurônio conectado a seis vizinhos, permitindo uma interação mais equilibrada e uniforme entre os elementos da rede. Essa configuração tende a produzir mapas mais suaves e representações mais precisas, especialmente em dados com distribuição irregular. A desvantagem principal é que a implementação é mais complexa e a indexação dos neurônios não é tão direta quanto na topologia quadrada, o que pode aumentar a dificuldade de manipulação e de visualização.

Na Figura 9 é ilustrado um exemplo comparativo entre essas duas topologias.



(a) Vizinhança retangular (b) Vizinhança hexagonal

Figura 9 – Topografia da grade de neurônios

Fonte: Adaptado de Zuchini (2003)

4.2.3 Processo Adaptativo

O ajuste dos pesos sinápticos é o último passo na formação auto-organizada de um mapa de características. Para que esse mapa seja auto-organizável, é necessário que o vetor de peso sináptico w_j do neurônio j da grade se modifique em relação ao vetor de entrada x. Os pesos dos neurônios (vencedor e seus vizinhos) serão atualizados a partir da Equação 5:

$$w_j(n+1) = w_j(n) + \eta(n)h_{ij}(n)(x(n) - w_j(n)) \tag{5}$$

em que $w_j(n)$ é o vetor de pesos no instante n, $w_j(n+1)$ é o vetor de pesos atualizado, x é o padrão de entrada e $\eta(n)$ é a taxa de aprendizado no instante n. A taxa de

aprendizado segue as mesmas regras de decaimento do raio de vizinhança; isto é, pode ser calculada por decaimento exponencial ou ainda variar de acordo com um valor predeterminado fixo a cada iteração. Logo, o processo adaptativo é dividido, segundo Haykin (2009), em dois momentos: fase de ordenação e fase de convergência.

- **Fase de ordenação:**

  Esta etapa tem como objetivo promover a ordenação topológica dos vetores de pesos, independentemente de terem sido inicializados linearmente ou aleatoriamente. Normalmente, são realizadas cerca de 1000 iterações, durante as quais os neurônios são organizados conforme a distribuição dos padrões no espaço de entrada.

  A taxa de aprendizagem é geralmente alta no início, próxima de 1, o que permite ajustes rápidos nos pesos para acompanhar a organização dos padrões. Com o avanço da fase, essa taxa é reduzida gradualmente até valores próximos de 0,1, tornando os ajustes mais finos e evitando oscilações excessivas (HAYKIN, 2009).

  O raio de vizinhança deve englobar inicialmente todos ou quase todos os neurônios, possibilitando ampla exploração do espaço de pesos. Esse raio vai sendo progressivamente diminuído ao longo das iterações, até atingir um valor que permita a influência de apenas um ou nenhum neurônio vizinho (KOHONEN, 2013).

- **Fase de convergência:**

  Essa etapa tem como papel principal o ajuste fino do mapa, buscando uma quantização estatisticamente precisa do espaço de entrada (HAYKIN, 2009).

  Recomenda-se que o número de iterações nesta fase seja pelo menos 500 vezes o número de neurônios da grade.

  A taxa de aprendizagem permanece baixa ao longo de toda a fase, tipicamente em torno de 0,01 ou menos.

4.3 Configuração dos hiperparâmetros da rede

De acordo com (SILVA, 2013), a vizinhança e a taxa de aprendizagem devem ser reduzidas monotonicamente conforme o avanço do tempo durante o treinamento do algoritmo. Sendo assim, ao definir e treinar uma rede, é fundamental definir os seguintes parâmetros:

- A função de vizinhança (bolha, degrau, gaussiana, *etc.*);
- O tamanho do mapa e a sua topologia (hexagonal ou retangular);
- O tipo de treinamento (sequencial ou em lote);

- A função da taxa de aprendizagem caso o treinamento seja sequencial (decaimento linear, exponencial, *etc.*);
- O número de épocas para o treinamento;
- Inicialização das posições dos neurônios no espaço de entrada.

A inicialização das posições dos neurônios no espaço de dados pode ser aleatória ou linear.

4.4 Análise de agrupamento

Estudos em análise de agrupamento revelam que existem diversos algoritmos que buscam identificar padrões e estruturas em conjunto de dados (JAIN; DUBES, 1988; HAN; KAMBER; PEI, 2012). Muitos desses algoritmos realizam o processo de forma simples, com o uso de técnicas estatísticas, como média, mediana, etc. Essas técnicas são eficazes quando há um pequeno número de dimensões. Porém, em espaços de alta dimensionalidade, a identificação de grupos torna-se mais complexa, pois a distinção entre as amostras tende a se tornar menos evidente. Em diversos trabalhos, é comum o uso de técnicas, como a *Principal Component Analysis* (PCA), para reduzir a dimensionalidade dos dados antes da aplicação do algoritmo de agrupamento (JAIN; DUBES, 1988; HRUSCHKA; EBECKEN, 2003).

A quantidade de dimensões em um conjunto de dados pode ser considerada um problema em análise de agrupamento (BERKHIN, 2006). Além de aumentar a complexidade computacional, compromete a visualização e a análise das relações entre os vetores do conjunto (HAN; KAMBER; PEI, 2012). Métodos tradicionais não realizam redução de dimensionalidade e, quando o número de atributos é elevado, a visualização dos dados torna-se praticamente incompreensível (CASTRO; FERRARI, 2016).

O SOM baseia-se no cálculo de distâncias entre vetores em espaços de alta dimensionalidade, processo que pode perder capacidade discriminativa devido à chamada maldição da dimensionalidade (AGGARWAL; HINNEBURG; KEIM, 2001). Apesar dessa limitação, o algoritmo projeta os dados em um espaço de menor dimensão, preservando as relações de vizinhança e permitindo uma melhor visualização e interpretação das estruturas do conjunto de dados (KOHONEN, 2001b).

No SOM, a camada de entrada possui tantos neurônios quanto as dimensões do vetor de atributos do conjunto de dados. Cada neurônio do mapa de saída é conectado a todos os neurônios da entrada por vetores de pesos, que são ajustados durante o treinamento. Assim, o mapa organiza seus neurônios de saída em uma grade bidimensional, possibilitando a identificação de grupos e padrões nos dados (KOHONEN, 2012).

4.5 Classificação em Mapas Auto-Organizáveis

O SOM organiza os vetores de entrada de modo que padrões semelhantes sejam mapeados em neurônios vizinhos, preservando a topologia do espaço de dados. Esse processo é guiado pela minimização do *erro de quantização*:

$$E_q = \frac{1}{N} \sum_{i=1}^{N} \|x_i - w_{c(i)}\|^2, \tag{6}$$

em que x_i representa o vetor de entrada, $w_{c(i)}$ é o vetor de peso do neurônio vencedor e N é o número de instâncias (KOHONEN, 1990; VESANTO; ALHONIEMI, 2000).

No contexto da classificação, o processo pode ser compreendido em duas etapas principais. Na primeira, durante o treinamento não supervisionado, o SOM organiza os exemplos de entrada em grupos, de modo que padrões semelhantes sejam posicionados em regiões próximas do mapa. Na segunda etapa, após o treinamento, cada neurônio pode ser rotulado de acordo com a classe mais frequente entre os exemplos de treinamento que foram mapeados para ele, utilizando o método do "voto majoritário" (BUNGUM; GAMBÄCK, 2015). Assim, para classificar uma nova instância, basta identificar o neurônio vencedor e atribuir-lhe o rótulo previamente associado.

Esse processo de rotulação permite visualizar a distribuição das classes no mapa, facilitando a interpretação da separação entre diferentes regiões do espaço de dados. A Figura 10 ilustra esse resultado, evidenciando quatro regiões distintas que correspondem às classes do conjunto de dados analisado.

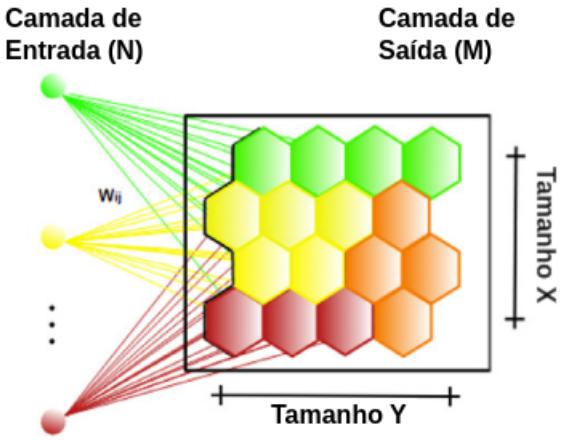


Figura 10 – Visualização da auto-organização através do mapeamento dos dados de entrada para o espaço de neurônios

Fonte: Adaptado de Melin et al. (2020)

Além de possibilitar a interpretação visual, essa abordagem também torna o SOM aplicável como classificador, mesmo sem o uso explícito de rótulos durante o treinamento. Isso o torna especialmente útil em cenários de rótulo escasso ou de aprendi-

zado semissupervisionado. Ademais, como neurônios vizinhos frequentemente representam padrões de classes semelhantes, o SOM é capaz de identificar relações entre classes e facilitar a análise de dados complexos, além de fornecer uma representação intuitiva das fronteiras de decisão (BUNGUM; GAMBÄCK, 2015; KOHONEN, 2012).

Diversos estudos demonstram a aplicação do SOM para classificação em diferentes domínios, como a análise de documentos multirrótulo (BUNGUM; GAMBÄCK, 2015), a classificação de imagens de sensoriamento remoto (SILVA et al., 2022) e problemas de reconhecimento de padrões em dados de alta dimensionalidade (MILJKOVIĆ, 2017). Entre as vantagens do uso do SOM para classificação, destacam-se:

- **Capacidade de lidar com dados não rotulados ou parcialmente rotulados:** O algoritmo SOM é, por definição, não supervisionado, o que o torna adequado para análise de dados sem rótulos ou com rótulos escassos. Isso é útil em estudos exploratórios e no aprendizado semissupervisionado;
- **Preservação da topologia dos dados:** Uma das características do SOM é o mapeamento topológico, permitindo que estruturas e relacionamentos do conjunto de dados original sejam mantidos de modo visualizável e interpretável;
- **Robustez a ruídos e dados de alta dimensionalidade:** A dinâmica de competição e adaptação do SOM permite que ele seja resistente a dados ruidosos, sendo amplamente adotado em aplicações que envolvem grandes volumes de dados;
- **Facilidade de adaptação a diferentes tarefas de classificação:** O SOM pode ser expandido para múltiplas tarefas de classificação, como problemas binários, multiclasse e multirrótulo.

No entanto, é importante ressaltar que o desempenho do SOM em tarefas de classificação pode ser impactado por fatores como o desbalanceamento das classes, a escolha dos hiperparâmetros (tamanho do mapa, função de vizinhança, taxa de aprendizado) e a estratégia de rotulação dos neurônios (MILJKOVIĆ, 2017; VESANTO; ALHONIEMI, 2000). Em cenários de FCDs, variantes como o Mapa Auto-Organizável Crescente (GSOM) podem ser empregadas para permitir a expansão adaptativa da rede e melhor acomodação de novos padrões.

4.6 Considerações Finais

Este capítulo apresentou o algoritmo SOM, mostrando sua aplicação na análise de agrupamento de dados e na classificação. Foram discutidos a estrutura e o funcionamento do SOM, destacando os processos competitivo, cooperativo e adaptativo.

No Capítulo 5 será apresentado o GSOM, que estende o SOM com capacidade de crescimento dinâmico, permitindo adaptação contínua a novos padrões em fluxos de dados, característica essencial para a classificação de *logs* de *firewall* em ambientes de rede dinâmicos.

Capítulo 5

Mapas Auto-Organizáveis Crescentes

Após a apresentação do SOM, que oferece uma abordagem para a redução de dimensionalidade e visualização de dados em uma grade de neurônios de tamanho fixo, este capítulo introduz o Mapa Auto-Organizável Crescente (GSOM), uma extensão dinâmica do SOM. Originalmente desenvolvido para superar a limitação da topologia fixa do SOM, o GSOM foi posteriormente reconhecido na literatura como uma arquitetura promissora para ambientes dinâmicos, tornando-se uma escolha robusta para a análise em FCDs.

5.1 Conceitos fundamentais

Os SOMs têm se mostrado eficazes em diversas aplicações, oferecendo uma representação compacta e visual das relações topológicas em conjuntos de dados complexos. No entanto, sua topologia fixa apresenta limitações ao lidar com dados dinâmicos, especialmente em cenários que demandam flexibilidade, como na análise do tráfego de redes ou de sistemas com distribuição de dados em constante mudança. Nessas situações, a necessidade de uma abordagem mais adaptativa permite uma representação eficiente de padrões que evoluem ao longo do tempo.

Para superar essas limitações, foi desenvolvido o GSOM. Diferentemente do SOM tradicional, o GSOM possui uma estrutura dinâmica que começa com uma configuração mínima de quatro neurônios, expandindo-se adaptativamente conforme a complexidade dos dados de entrada (ALAHAKOON; HALGAMUGE; SRINIVASAN,

1998b; ALAHAKOON; HALGAMUGE; SRINIVASAN, 1998a). Essa capacidade de crescimento controlado permite ao GSOM ajustar sua topologia de modo a capturar melhor os padrões subjacentes, mesmo em cenários em que as características dos dados são altamente heterogêneas ou evolutivas.

No GSOM, o crescimento da rede é governado por um parâmetro-chave, o fator de espalhamento (SF), que define a granularidade do mapa final. O SF determina o limite de crescimento da rede (GT), controlando a formação de novos neurônios. À medida que novos neurônios são inseridos, o GSOM reorganiza os pesos dos neurônios existentes, de forma semelhante ao processo de ajuste do SOM. Esse mecanismo dinâmico permite que o GSOM capture tanto padrões gerais quanto detalhes específicos dos dados.

Além disso, o GSOM mantém a robustez característica do SOM na preservação de relações topológicas, ao mesmo tempo em que se adapta às demandas de problemas que exigem maior flexibilidade e escalabilidade. O processo do GSOM será detalhado nas seções seguintes, destacando suas etapas.

5.2 Fase de Inicialização

A fase de inicialização define a estrutura básica da rede GSOM, incluindo a grade inicial de neurônios e os pesos atribuídos a eles. Também é nesse momento que se calcula o GT, o valor que determina quando novos neurônios devem ser inseridos. Esse cálculo depende diretamente do SF, parâmetro definido pelo usuário que controla a granularidade do mapa.

O SF varia entre 0 e 1: valores mais baixos geram mapas mais compactos e generalistas, enquanto valores mais altos favorecem um crescimento mais detalhado (ZHENG; WANG; BLACK, 2008). A relação entre o SF e o GT é dada pela equação:

$$GT = -D \cdot \ln(\mathrm{SF}) \tag{7}$$

em que D representa a dimensionalidade do espaço de entrada. Quanto menor o SF, maior o GT, limitando o crescimento da rede; já valores mais altos de SF reduzem o GT, permitindo uma expansão mais granular do mapa (ALAHAKOON; HALGAMUGE; SRINIVASAN, 2000).

A configuração inicial do GSOM consiste em quatro neurônios dispostos em uma grade bidimensional 2×2, conforme mostrado na Figura 11. Essa escolha inicial oferece:

1. Uma estrutura básica e simétrica, que facilita o crescimento dinâmico em todas as direções e permite que o mapa se adapte à distribuição dos dados;

2. Um ponto de partida estratégico, com cada neurônio atuando como unidade de fronteira, preservando a coerência topológica durante a expansão.

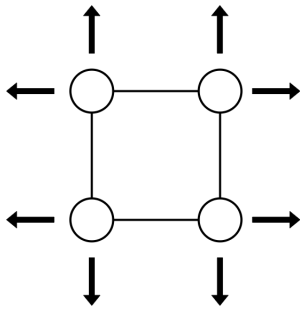


Figura 11 – Configuração inicial da rede GSOM com quatro neurônios.

Fonte: Adaptado de Alahakoon, Halgamuge e Srinivasan (2000)

Os vetores de peso desses neurônios são inicializados aleatoriamente no intervalo dos dados normalizados (entre 0 e 1). Essa normalização assegura a compatibilidade com os dados de entrada e a flexibilidade no treinamento inicial, permitindo que o mapa cresça conforme a distribuição dos dados, enquanto mantém a robustez necessária para capturar padrões emergentes.

5.3 Fase de Crescimento

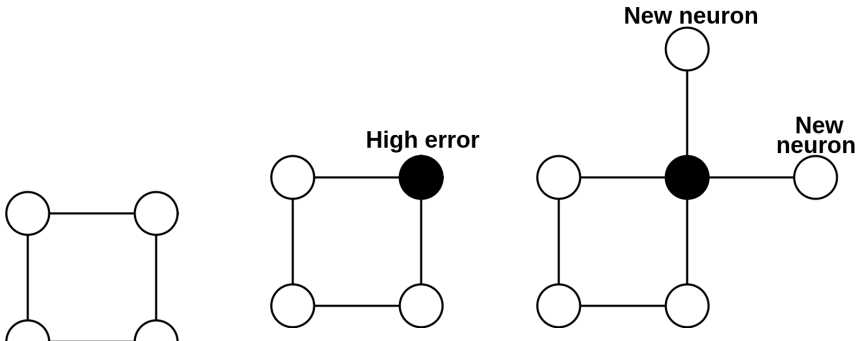
A fase de crescimento é responsável pela expansão dinâmica do mapa, adicionando novos neurônios quando necessário. Essa expansão é regulada pelo GT, que define o ponto em que o erro acumulado de um neurônio justifica a inserção de novos neurônios (ALAHAKOON; HALGAMUGE; SRINIVASAN, 2000). O GT é calculado com base na dimensionalidade do espaço de entrada D e no fator de espalhamento (SF), que ajusta a propagação da rede à complexidade dos dados, independentemente de sua dimensionalidade.

Durante o treinamento, os neurônios acumulam erro de quantização, medido pela distância entre os vetores de entrada e os respectivos vetores de peso. O neurônio com menor erro relativo é identificado como o vencedor, e seus pesos, assim como os de seus vizinhos, são ajustados. Caso o erro acumulado do neurônio vencedor ultrapasse o GT, ele é considerado insuficiente para representar adequadamente o espaço de entrada. Nesse cenário, novos neurônios são adicionados ao mapa. Se o neurônio vencedor for uma unidade de borda, o crescimento ocorre na borda; caso contrário, o erro é redistribuído entre os neurônios vizinhos, e o novo neurônio é inicializado com pesos derivados dos neurônios adjacentes. Esse mecanismo permite que o mapa cresça em áreas mais complexas, adaptando-se melhor à distribuição dos dados.

5.3.1 Geração de Novos Neurônios

Novos neurônios são gerados a partir de neurônios de fronteira, como ilustrado na Figura 12. Um neurônio de fronteira é aquele que possui pelo menos uma posição vizinha desocupada, o que permite a expansão da grade (ALAHAKOON; HALGAMUGE; SRINIVASAN, 2000). Cada neurônio pode possuir até quatro vizinhos imediatos, o que significa que um neurônio de fronteira pode ter de 1 a 3 posições vizinhas livres. Quando selecionados para crescimento, novos neurônios são gerados em todas as posições vizinhas disponíveis. Embora isso possa gerar neurônios redundantes, estes são removidos após algumas iterações, pois não acumulam acertos e permanecem com 0.

A Figura 12 mostra o processo de geração de novos neurônios: a Figura 12a apresenta a grade inicial; a Figura 12b demonstra a superação do GT por um neurônio; e a Figura 12c ilustra as possíveis posições de novos neurônios no mapa.



(a) Grade inicial.

(b) Erro excede o limite de crescimento.

(c) Posições possíveis para um novo neurônio.

Figura 12 – Geração de novos neurônios na fronteira da rede.

Fonte: Adaptado de Alahakoon, Halgamuge e Srinivasan (2000)

5.3.2 Inicialização de Peso de Novos Neurônios

Os neurônios recém-gerados recebem valores iniciais de peso para garantir continuidade com o mapa existente. Como os neurônios antigos já estão parcialmente organizados, a inicialização aleatória dos novos neurônios poderia introduzir inconsistências. Para evitar isso, os pesos dos novos neurônios são derivados dos valores dos neurônios vizinhos, o que garante suavidade na transição. Para a inicialização de peso dos novos neurônios, existem quatro situações a serem consideradas. A Figura 13 apresenta os casos mais comuns, descritos a seguir.

Caso A: O novo neurônio tem dois neurônios antigos consecutivos de um de seus lados, conforme Figura 13a:

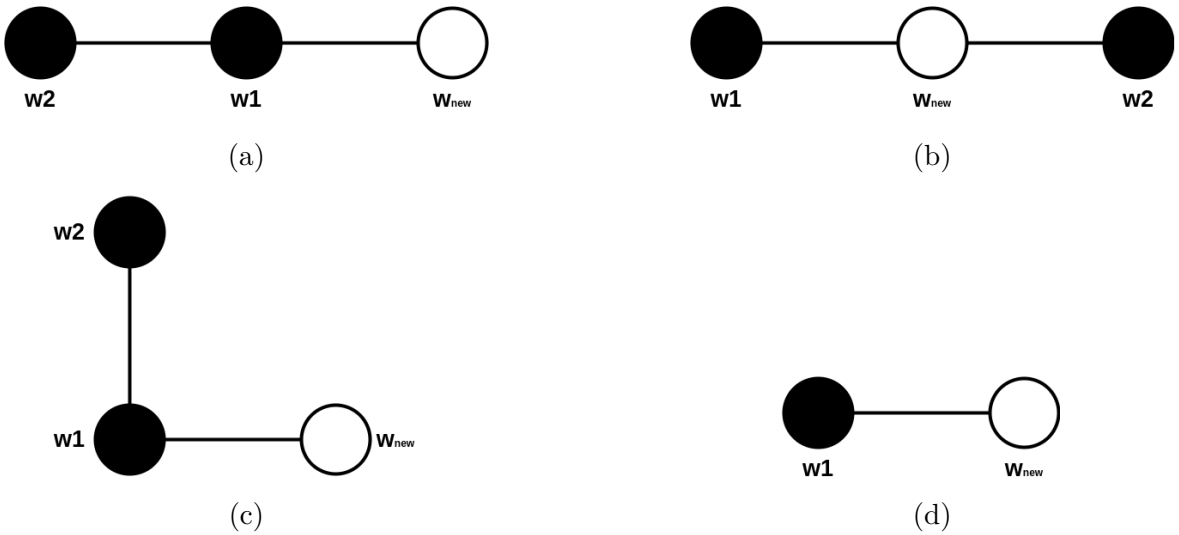


Figura 13 – Inicialização de peso de novos neurônios

Fonte: Adaptado de Alahakoon, Halgamuge e Srinivasan (2000)

$$\texttt{Se}\ \ w_2 > w_1\texttt{:}$$
$$\texttt{Entao}\ \ w_{\texttt{new}} = w_1 - (w_2 - w_1)$$
$$\texttt{Se}\ \ w_1 > w_2\texttt{:}$$
$$\texttt{Entao}\ \ w_{\texttt{new}} = w_1 + (w_1 - w_2)$$

Caso B: O novo neurônio está entre dois neurônios mais antigos, conforme Figura 13b:

$$w_{\texttt{new}} = \frac{w_1 + w_2}{2}$$

Caso C: O novo neurônio tem apenas um vizinho direto (o neurônio pai mais antigo). Mas o neurônio mais antigo tem um vizinho de um lado que não é o lado diretamente oposto ao novo neurônio, conforme Figura 13c:

$$\texttt{Se}\ \ w_2 > w_1\texttt{:}$$
$$\texttt{Entao}\ \ w_{\texttt{new}} = w_1 - (w_2 - w_1)$$
$$\texttt{Se}\ \ w_1 > w_2\texttt{:}$$
$$\texttt{Entao}\ \ w_{\texttt{new}} = w_1 + (w_1 - w_2)$$

Pode-se observar que as equações do Caso 13c são as mesmas do Caso A. Portanto, nesses dois casos, a diferença reside apenas na posição do vizinho considerado. Quando ambas as opções estão disponíveis, o Caso A é utilizado.

Caso D: O novo neurônio tem apenas um neurônio vizinho (w_1) conforme Figura 13d:

Se o valor estiver fora do intervalo, um valor intermediário é atribuído (0,5 para dados binários), permitindo que a auto-organização ajuste o peso.

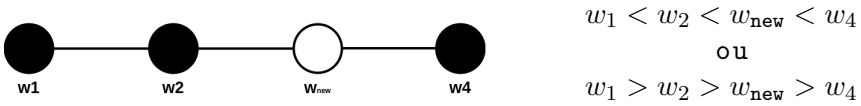


Figura 14 – Fluxo dos vetores de peso

Fonte: Adaptado de Alahakoon, Halgamuge e Srinivasan (2000)

5.4 Fase de Suavização

Após a fase de crescimento, a fase de suavização tem como objetivo refinar a estrutura do mapa, ajustando os pesos dos neurônios para minimizar o erro de quantização. Nessa etapa, não há a adição de novos neurônios, mas sim o aperfeiçoamento da topologia existente, garantindo que as regiões formadas durante o crescimento sejam representadas com maior precisão. Essa fase contribui para reduzir distorções, estabilizar a estrutura da rede e aprimorar sua capacidade de adaptação às distribuições de dados.

Além de melhorar a robustez e a representação da rede, a suavização aumenta a estabilidade ao longo do tempo, permitindo uma adaptação mais rápida a novos dados ou a mudanças nas distribuições. Assim, essa fase é essencial para garantir o desempenho máximo do GSOM, tornando-o mais eficaz na captura da complexidade e da dinâmica dos dados.

Após a fase de suavização, o GSOM utiliza os neurônios vencedores identificados para cada amostra de treinamento, permitindo que as regiões do mapa sejam associadas aos padrões aprendidos durante o treinamento. Assim, cada neurônio passa a representar um conjunto de dados semelhantes, facilitando a análise exploratória e a identificação de agrupamentos naturais no conjunto de dados.

5.5 Considerações finais

Ao longo deste capítulo, foram apresentados os principais conceitos e características do GSOM, destacando sua adaptabilidade e aplicabilidade na análise de dados em FCDs, como os *logs* de *firewall* em redes de computadores. A flexibilidade do GSOM em expandir sua estrutura conforme novos padrões são identificados o torna uma alternativa promissora para cenários dinâmicos de segurança em redes.

No próximo capítulo, serão apresentados e discutidos os trabalhos presentes na literatura, abordando diferentes técnicas e abordagens para a análise de logs de *firewall* e a classificação em FCDs, o que permitirá situar o uso do GSOM no contexto das pesquisas recentes da área.

Capítulo 6

Trabalhos Relacionados

O estudo da classificação de *logs* de *firewall* concentra-se na análise das ações adotadas pelo *firewall* em cada sessão de tráfego de rede, visando otimizar a gestão das políticas de segurança de rede. Essa abordagem difere da detecção de intrusões, pois foca na categorização das ações do *firewall* e no suporte à tomada de decisão, e não apenas na identificação de ameaças ou anomalias.

Esse capítulo apresenta uma revisão dos principais trabalhos relacionados a duas frentes: a classificação de logs de firewall, restrita a abordagens em lote, e a classificação em sistemas de detecção de intrusão, explorando o que se tem na literatura sobre FCDs.

O objetivo é apresentar os principais resultados já obtidos, destacando técnicas, limitações e lacunas, de forma a contextualizar como este trabalho avança na análise de tráfego de rede ao propor a classificação de *logs* de *firewall* em cenários de FCDs.

6.1 Classificação de logs de firewall

A literatura recente tem focado na classificação de ações em *logs* de *firewall*. Esses trabalhos, no entanto, são desenvolvidos exclusivamente em cenários de processamento em lote, com conjuntos de dados estáticos, o que limita sua aplicação em ambientes dinâmicos.

Ertam e Kaya (2018) realizaram a coleta de dados a partir dos *logs* do *firewall* Palo Alto 5020, instalado na *Firat University*, na Turquia. O conjunto de dados abrange um intervalo de aproximadamente 30 segundos de tráfego de rede real, totalizando

65.532 instâncias distintas. Para a tarefa de classificação, foram utilizados os seguintes 11 atributos principais extraídos dos *logs*: *Source Port*, *Destination Port*, *NAT Source Port*, *NAT Destination Port*, *Elapsed Time*, *Bytes*, *Bytes Sent*, *Bytes Received*, *Packets*, *Packets Sent*, *Packets Received*. Além desses atributos, o atributo *Action* foi utilizado como atributo-alvo, categorizando cada registro como *Allow*, *Deny*, *Drop* ou *Reset-Both*. Todos os atributos selecionados são de natureza numérica, exceto o de classe, e foram escolhidos por sua relevância para caracterizar o comportamento do tráfego e as decisões do *firewall*. Esses atributos permitiram a construção e avaliação de modelos de classificação multiclasse para automatizar a análise das ações do *firewall* em ambientes reais.

Foram aplicadas técnicas de aprendizado de máquina supervisionado, utilizando *Support Vector Machine* (SVM) para classificar as ações. Foram avaliadas diferentes funções de ativação para o SVM: linear, polinomial, sigmoide e RBF. O melhor resultado de *recall* geral foi obtido com a função sigmoide (98,5%), enquanto o melhor F1-Score geral foi obtido com a função RBF (76,4%). Os autores também destacam que a classe *Reset-Both* apresentou maior dificuldade de classificação devido à sua baixa frequência nos dados, mas, de modo geral, o modelo apresentou desempenho consistente em todas as classes.

Aljabri et al. (2022) utilizaram um conjunto de dados coletado a partir dos registros de tráfego de rede de uma organização privada, no período de 18 a 27 de maio de 2021. As entradas foram extraídas diretamente dos *logs* gerados por um *firewall* corporativo, totalizando, inicialmente, 1.048.576 instâncias. Com base nesse conjunto de dados, os autores desenvolveram um modelo de classificação multiclasse para *logs* de *firewall*, utilizando algoritmos de AM e de Aprendizagem Profunda (AP). No entanto, foi identificado um forte desbalanceamento no atributo-alvo: 88,23% das instâncias correspondiam à ação *Allow*, 5,08% à ação *Reset-Both*, 4,01% à ação *Drop* e apenas 2,68% à ação *Deny*. Para mitigar esse problema, foi aplicada a técnica de subamostragem aleatória, selecionando-se 28.133 instâncias para cada uma das quatro classes, resultando em um conjunto balanceado de 112.532 instâncias.

Os registros de tráfego originalmente continham 25 atributos, dos quais dois subconjuntos foram selecionados para fins experimentais. Os atributos utilizados nos testes foram: *Destination Port*, *Source Port*, *NAT Destination Port*, *NAT Source Port*, *Elapsed Time*, *Bytes*, *Bytes Sent*, *Bytes Received*, *Packets*, *Packets Sent*, *Packets Received*, *Application* e *Category*. No primeiro experimento, utilizaram-se 11 atributos comumente adotados na literatura especializada. No segundo, foram adicionados os atributos *Application* e *Category*, escolhidos por apresentarem elevada correlação com o atributo-alvo, totalizando 13 atributos. A escolha excluiu variáveis diretamente relacionadas à ação final, como as regras do *firewall*, para evitar resultados que indicariam a decisão.

Foram avaliados cinco algoritmos: KNN, NB, J48, RF e RNAs. Os modelos foram treinados com validação cruzada (10 *folds*) e os hiperparâmetros foram ajustados por *grid search*. O RF obteve o melhor desempenho em ambos os experimentos, com acurácia de 99,11% no primeiro e de 99,64% no segundo.

Apesar dos avanços, até o momento não foram encontrados estudos que abordem a classificação de *logs* de *firewall* em cenários de FCDs. Todas as abordagens identificadas utilizam conjuntos de dados estáticos e processamento em lote, o que restringe sua aplicação em contextos que demandam análise contínua e em tempo real.

6.2 Classificação em Sistemas de Detecção de Intrusão

Diversos trabalhos abordam a classificação em Sistemas de Detecção de Intrusão, com foco na distinção entre tráfego legítimo e malicioso, ou na detecção de eventos anômalos em grandes volumes de dados de rede. Diferentemente dos estudos sobre *firewall*, essas pesquisas exploram algoritmos voltados a cenários de FCDs.

Labib e Vemuri (2002) propuseram o Mapa Auto-Organizável de Rede (NSOM), um IDS em tempo real baseado em SOM. Para a coleta de dados, o NSOM utilizou um *host* conectado a uma rede *Ethernet*, no qual o tráfego foi capturado diretamente na interface de rede por meio da ferramenta *tcpdump*, configurada em modo promíscuo para monitorar todos os pacotes que passavam pelo segmento de rede. Os pacotes eram coletados em janelas de 50 pacotes, extraíam atributos, como partes dos endereços IP e o tipo de protocolo, e utilizavam o SOM para agrupar o tráfego em padrões normais e anômalos. O sistema apresenta os resultados em uma interface gráfica, permitindo a visualização dinâmica dos grupos formados. Nos testes, o NSOM conseguiu separar claramente o tráfego rotineiro dos padrões associados a ataques, como Negação de Serviço (DoS), demonstrando a eficácia da abordagem para identificar comportamentos fora do padrão sem depender de rótulos prévios.

No artigo de Abdualrahman e Ibrahem (2021), é apresentado um IDS em redes baseado em classificação em FCDs. Utilizando o conjunto CIC-IDS 2017, composto por tráfego real de rede e múltiplos tipos de ataques modernos (DoS, DDoS, *Brute Force*, *Botnet*, infiltração, *Web Attacks*, etc.), os autores demonstram que métodos tradicionais de aprendizado em *batch* não são adequados para ambientes em que os dados chegam continuamente em grande volume. Para superar esse desafio, o estudo emprega algoritmos de classificação incremental, como NB, ARF (GOMES et al., 2017), MLP, OzaBoost (OZA; RUSSELL, 2001) e *Stochastic Gradient Descent* (SGD) (BOTTOU, 2010), avaliando-os em cenários de mudança de conceito e de desbalanceamento de classes.

O processo metodológico inclui etapas de pré-processamento e seleção de atributos relevantes, além do uso do método "*test-then-train*", em que cada novo dado é testado antes de atualizar o modelo. Os resultados mostram que o ARF alcançou acurácia de até 99,9%, o SGD de 99,7%, o OzaBoost de 97,9%, o MLP de 88,5% e o NB de 61,4% na média dos principais cenários testados.

No artigo de Seth, Singh e Chahal (2021), apresenta-se uma abordagem para IDS em FCDs, com foco em lidar com mudanças de conceito ao longo do tempo, comuns em ambientes de segurança de redes. O estudo utiliza o classificador ARF combinado com o detector de mudanças ADWIN, permitindo que o sistema identifique rapidamente alterações na distribuição dos dados e adapte o modelo em tempo real, sem a necessidade de re-treinamento completo.

O conjunto de dados utilizado foi o CIC-IDS 2018, que originalmente apresentava múltiplos rótulos correspondentes a diferentes tipos de tráfego e ataques, como *Bot*, *Brute Force*, DDoS, DoS, *Infiltration*, *SQL Injection*, entre outros. Após o pré-processamento, todos os rótulos referentes a ataques foram agrupados na classe *Attack*, enquanto o tráfego legítimo permaneceu na classe *Benign*, reduzindo o problema a uma classificação binária. Os resultados mostram que o ARF com ADWIN alcançou acurácia de 99,5%, *recall* de 99,8% e precisão de 99,9%, superando outros métodos incrementais e modelos tradicionais de aprendizado profundo treinados em *batch*. O trabalho evidencia que técnicas de aprendizado incremental com detecção de mudança de conceito são altamente eficazes para detecção de intrusões em tempo real, tornando-se especialmente relevantes para ambientes dinâmicos de rede que exigem adaptação contínua e resposta a novos padrões de ataque.

Embora o uso de FCDs já seja explorado em IDS para identificação de anomalias em tempo real, não foram encontrados estudos que aplicam esse tipo de abordagem à classificação de ações em *logs* de *firewall*. As pesquisas se concentram em distinguir comportamentos normais e anômalos, sem abordar a categorização entre ações permitidas e bloqueadas pelo *firewall*.

6.3 Tabela comparativa dos trabalhos relacionados

A Tabela 1 apresenta uma comparação dos principais trabalhos relacionados, destacando as técnicas empregadas, a natureza dos dados utilizados e os resultados obtidos. Os trabalhos que focam em *logs* de *firewall* empregam abordagens supervisionadas em conjuntos de dados estáticos, enquanto os trabalhos que operam em FCDs concentram-se em detecção de intrusões, utilizando dados de tráfego de rede ou IDS.

Nota-se também uma diferença significativa na formulação do problema entre as abordagens. Enquanto a classificação de *logs* de *firewall* prioriza a distinção entre as ações de segurança (*Allow*, *Deny*, *Drop*, *Reset-Both*), os sistemas baseados em FCDs para segurança de redes tendem a reduzir o problema a uma classificação binária (tráfego normal vs. tráfego malicioso). Esta simplificação, embora eficaz para a detecção de anomalias, não captura as decisões de *firewall*, limitando sua aplicabilidade à análise detalhada de políticas de segurança.

A ausência de trabalhos que combinem a classificação multiclasse de *logs* de *firewall* com processamento em FCDs configura a lacuna identificada, justificando a proposta do presente estudo.

Tabela 1 – Resumo dos principais trabalhos relacionados

Trabalho	**Técnicas**	**Cenário / Dados**	**Principais Resultados**
Ertam et al. (2018)	SVM (linear, polinomial, sigmoide, RBF)	Logs de firewall – Batch; 65 mil instâncias	Recall de até 98,5%; dificuldade na classe pouco frequente
Aljabri et al. (2022)	RF, KNN, NB, J48, RNAs	Logs de firewall – Batch; 112 mil instâncias balanceadas	Acurácia até 99,64% com RF
Labib et al. (2002)	SOM	IDS – FCDs; tráfego de rede via tcpdump	Separação clara entre tráfego normal e ataques em tempo real
Abdualrahman e Ibrahem (2021)	ARF, OzaBoost, MLP, NB, SGD	IDS – FCDs; CIC-IDS 2017	ARF com acurácia até 99,9% em cenários de mudança de conceito
Setha et al. (2021)	ARF + ADWIN	IDS – FCDs; CIC-IDS 2018	Acurácia 99,5%, recall 99,8%, adaptação eficaz a mudanças de conceito

Fonte: Elaborado pelo autor.

6.4 Considerações finais

A maioria dos estudos sobre classificação de *logs* de *firewall* utiliza processamento em lote, geralmente empregando métodos supervisionados para prever ou analisar as ações do *firewall*. Nos últimos anos, houve avanços relevantes em técnicas de análise de dados em FCDs, mas essas abordagens têm sido direcionadas principalmente para IDS no tráfego de rede. Até o momento, não foram encontrados trabalhos que abordem especificamente a classificação de *logs* de *firewall* em FCDs.

Diante dessa lacuna, este trabalho propõe a aplicação da classificação de regras de

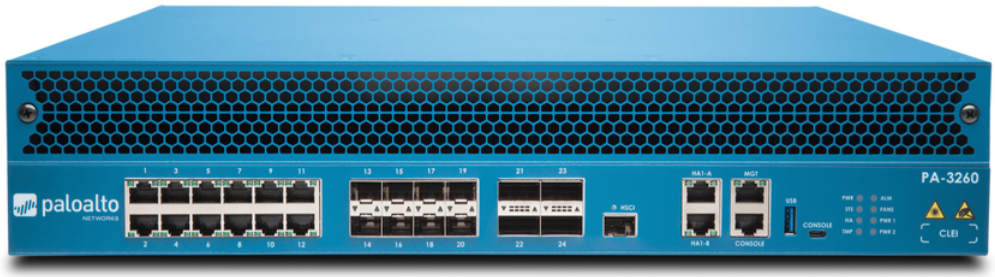
firewall em FCDs, explorando um tema inédito na literatura e contribuindo para o aprimoramento dos SSR.

Capítulo 7

Materiais e Métodos

Neste capítulo, são apresentados os materiais e os métodos empregados na condução dos experimentos desta pesquisa. Os dados analisados foram coletados a partir de registros de log gerados por um *firewall* de próxima geração, do modelo Palo Alto PA-3260, conforme a Figura 15, instalado em um ambiente acadêmico com tráfego real. Este equipamento é amplamente utilizado em redes corporativas e educacionais de médio a grande porte devido à sua capacidade de inspeção profunda de pacotes, controle granular de políticas de segurança e geração de logs detalhados com contexto, como a identidade do usuário, o tipo de ameaça e a ação tomada.

Figura 15 – Palo Alto PA-3260 Firewall



Fonte: Palo Alto Networks (2023)

Embora o *firewall* de próxima geração ofereça várias funções avançadas, este trabalho foca na filtragem de pacotes, a funcionalidade mais consolidada na literatura. Essa opção permite alinhar a pesquisa às práticas amplamente adotadas na lite-

ratura, além de garantir a comparabilidade dos resultados com estudos anteriores sobre classificação e análise de logs de tráfego de rede. Dessa forma, a investigação concentra-se em um mecanismo que segue padrões consolidados na área e facilita a avaliação dos resultados em relação ao estado da arte nos SSR.

O *firewall* em questão atua na borda da rede de uma instituição de ensino superior composta por aproximadamente 19.000 usuários ativos, entre estudantes, docentes, pesquisadores e técnicos-administrativos. A infraestrutura protegida abrange cerca de 6.000 dispositivos conectados simultaneamente, distribuídos em várias zonas de segurança, como redes administrativas, acadêmicas e laboratórios de informática. O tráfego médio diário registrado ultrapassa 435 milhões de sessões, gerando aproximadamente 265 GB de logs, o que evidencia o elevado volume dos fluxos monitorados.

A arquitetura da rede é segmentada para garantir a segurança e o controle do tráfego interno. Cada departamento da instituição conta com uma VLAN própria, promovendo uma separação lógica entre áreas administrativas, acadêmicas, laboratórios e outros setores, permitindo o estabelecimento de políticas específicas para cada segmento. Já nas redes sem fio, todos os prédios contam com pelo menos três redes *Wi-Fi*: a eduroam, destinada à comunidade acadêmica com autenticação federada; a *Wi-Fi* Visitantes, voltada a acessos temporários de convidados; e a *Wi-Fi* Instruções, uma rede aberta utilizada exclusivamente para o cadastro inicial. Caso haja necessidade, cada prédio pode ainda dispor de uma rede adicional ("Departamento-Dispositivos"), destinada a demandas locais.

Esse conjunto de redes distintas possibilita a aplicação de controles de acesso à rede, reforçando a segurança por meio do isolamento de segmentos, do controle de autenticação e do gerenciamento individualizado dos recursos de TI. A operação do *firewall* é gerida por uma equipe específica que administra o *firewall*, composta por cinco profissionais responsáveis pela administração, manutenção e atualização das políticas de segurança da rede. Essa equipe garante a operação do *firewall*, configurando as regras de filtragem de pacotes e monitorando continuamente o tráfego para identificar possíveis incidentes ou anomalias.

A Figura 16 apresenta a interface gráfica utilizada para monitorar e gerenciar as regras do *firewall*, ferramenta fundamental para a equipe de segurança. Esta tela oferece uma visão consolidada e em tempo real do tráfego de rede, permitindo visualizar sessões ativas, as regras aplicadas e as ações executadas pelo equipamento.

Por meio desse painel, os administradores podem acompanhar o comportamento do *firewall*, identificar discrepâncias nas políticas, realizar auditorias e exportar registros para análise. A interface é intuitiva, com filtros e ordenações dinâmicas que agilizam a investigação de eventos específicos. Essa capacidade é fundamental para

manter a integridade, a confidencialidade e a disponibilidade da rede, especialmente em ambientes complexos e de alto volume.

Figura 16 – Interface de monitoramento de tráfego de rede

RECEIVE TIME	TYPE	FROM ZONE	TO ZONE	SOURCE	DESTINATION	TO PORT	APPLICATION	ACTION	RULE	SESSION END REASON	BYTES
07/21 15:00:49	drop	internet	hospedag...			5038	not-applicable	drop	blacklist-edl_serpro-in	policy-deny	60
07/21 15:00:49	drop	internet	intranet			53283	not-applicable	deny	Deny_All	policy-deny	60
07/21 15:00:49	drop	internet	intranet			81	not-applicable	deny	Deny_All	policy-deny	60
07/21 15:00:49	end	autonomos	internet			443	ssl	allow	fai-web	tcp-fin	11.9k
07/21 15:00:49	end	intranet	intranet			53	dns-base	allow	dns_interno	tcp-fin	808
07/21 15:00:49	end	intranet	internet			443	non-syn-tcp	allow	acesso_internet	tcp-rst-from-server	336
07/21 15:00:49	end	rede_inst... L2	rede_inst... L2			80	whatsapp-base	allow	instrucoes-L2	tcp-rst-from-client	1.7k
07/21 15:00:49	end	intranet	internet			443	ssl	allow	acesso_internet	tcp-rst-from-client	7.8k
07/21 15:00:49	end	intranet	internet			443	ssl	allow	acesso_internet	tcp-rst-from-client	16.5k
07/21 15:00:49	end	intranet	internet			80	web-browsing	allow	acesso_internet	tcp-fin	897
07/21 15:00:49	end	intranet	internet			80	web-browsing	allow	acesso_internet	tcp-fin	2.6k
07/21 15:00:49	end	intranet	intranet			53	dns-base	allow	dns_interno	tcp-fin	832
07/21 15:00:49	end	intranet	servidores			137	netbios-ns	allow	ad_servidores	aged-out	2.3k
07/21 15:00:49	end	intranet	internet			6881	incomplete	allow	acesso_internet	aged-out	140
07/21 15:00:49	end	intranet	internet			6881	incomplete	allow	acesso_internet	aged-out	140
07/21 15:00:49	end	intranet	internet			60874	incomplete	allow	acesso_internet	aged-out	140
07/21 15:00:49	end	vpn-GP	intranet			62078	incomplete	allow	vpn-gp_citi	aged-out	140
07/21 15:00:49	end	internet	autonomos			1016	incomplete	allow	autonomos	aged-out	60
07/21 15:00:49	end	internet	autonomos			52131	incomplete	allow	autonomos	aged-out	60

Fonte: Elaborado pelo autor.

Nesse contexto, os logs gerados pelo *firewall* configuram-se como a principal fonte de dados deste estudo. Cada linha corresponde a uma sessão de tráfego de rede entre dois dispositivos, contendo informações detalhadas sobre a conexão e a ação executada pelo *firewall* . Em especial, o atributo-alvo (*action*) pode assumir quatro valores: *Allow*, *Deny*, *Reset-Both* e *Drop*, conforme descrito na Tabela 2.

Tabela 2 – Descrição das classes de ação do firewall

Ação	Descrição
Allow	Autoriza o tráfego conforme as regras do *firewall*;
Deny	Bloqueia o tráfego e reinicia a conexão TCP;
Drop	Descarta pacotes sem resposta ao remetente;
Reset-Both	Encerra a conexão enviando pacotes de *reset* para ambos os lados.

Fonte: Elaborado pelo autor.

Embora as ações *Allow*, *Deny*, *Drop* e *Reset-Both* possam ser definidas de forma objetiva, cada uma produz um efeito distinto na comunicação. A ação *Allow* permite o tráfego conforme a política do *firewall*. Já *Deny* bloqueia a comunicação de forma explícita, informando ao emissor que a conexão não foi permitida.

A ação *Drop* também bloqueia o tráfego, porém de forma silenciosa, sem enviar resposta ao remetente. Assim, embora *Deny* e *Drop* resultem em bloqueio, a principal diferença é que *Deny* torna a rejeição explícita, enquanto *Drop* apenas descarta os

pacotes. Por sua vez, a ação *Reset-Both* encerra ativamente uma conexão TCP, enviando pacotes de *reset* para os dois lados da sessão. Dessa forma, diferentemente de *Deny* e *Drop*, que representam formas de bloqueio da comunicação, *Reset-Both* caracteriza uma intervenção ativa sobre uma sessão já estabelecida ou em andamento.

Essas diferenças são importantes neste trabalho porque as classes analisadas representam decisões reais do *firewall*. Assim, variações na frequência dessas ações ao longo do tempo podem refletir mudanças nas regras de segurança, no perfil do tráfego observado ou nas estratégias de proteção adotadas na rede.

Com isso, o treinamento do modelo é realizado com base nos logs, que registram as decisões tomadas pelo *firewall* a partir das regras configuradas no ambiente monitorado. Dessa forma, os modelos aprendem padrões presentes nos registros associados a cada ação observada, e não apenas características isoladas do tráfego. Posteriormente, o modelo é avaliado na tarefa de prever, para novos registros de tráfego, qual ação tende a ser associada àquela observação, permitindo analisar sua capacidade de classificar corretamente o tratamento aplicado a fluxos ainda não vistos.

Quando os logs são exportados a partir do *firewall*, eles são disponibilizados em arquivos estruturados que abrangem um total de 113 atributos distintos, conforme a Tabela 3. Esses atributos incluem desde informações fundamentais, como endereços e portas de origem e destino, até detalhes mais avançados sobre aplicações, dispositivos conectados, zonas de segurança, métricas temporais e volumétricas, além dos motivos de encerramento das sessões. A abrangência desses dados é relevante para esta pesquisa, pois fornece diferentes perspectivas sobre o contexto em que cada decisão do *firewall* foi tomada.

7.1 Análise Temporal dos Dados

Para investigar as mudanças de conceito nos registros do *firewall*, foram realizadas três coletas em momentos estratégicos: agosto de 2024, fevereiro de 2025 e outubro de 2025, totalizando 500.000 instâncias em cada uma. Essa abordagem permitiu capturar o comportamento do sistema em diferentes estágios e analisar o impacto de possíveis alterações nas regras de filtragem, nos padrões de tráfego e nas políticas de segurança implementadas ao longo de um período superior a um ano. A Tabela 4 resume a distribuição quantitativa e percentual das principais classes nos três períodos.

A evolução temporal das classes confirma a presença de mudanças de conceito. A classe *Allow*, que era majoritária na primeira coleta, apresentou uma redução significativa até a terceira coleta, enquanto a classe *Drop* aumentou de forma consistente,

Tabela 3 – Atributos utilizados na base de dados.

Domain	*Virtual System*	*Action*
DG *Hierarchy Level* 1	SCTP *Chunks*	*Source Device Profile*
Container ID	*Technology of app*	*Receive Time*
Source Zone	*Bytes*	*DG Hierarchy Level* 2
SCTP Chunks Sent	*Source Device Model*	POD *Namespace*
Risk of app	*Serial #*	*Destination Zone*
Bytes Sent	DG *Hierarchy Level* 3	SCTP *Chunks Received*
Source Device Vendor	POD *Name*	*Characteristic of app*
Type	*Inbound Interface*	*Bytes Received*
DG *Hierarchy Level* 4	UUID *for rule*	*Source Device OS Family*
Source External Dynamic List	*Container of app*	Threat/Content Type
Outbound Interface	*Packets*	*Virtual System Name*
HTTP/2 *Connection*	*Source Device* OS *Version*	*Destination External Dynamic List*
Tunneled app	*Config Version*	*Log Action*
Start Time	*Device Name*	*link_change_count*
Source Hostname	*Host* ID	SaaS *of app*
Generate Time	*Time Logged*	*Elapsed Time* (sec)
Action Source	policy_id	*Source Mac Address*
Serial Number	*Sanctioned State of app*	*Source address*
Session ID	*Category*	*Source VM* UUID
link_switches	*Destination Device Category*	*Source Dynamic Address Group*
offloaded	*Destination address*	*Repeat Count*
Sequence Number	*Destination* VM UUID	sdwan_cluster
Destination Device Profile	*Destination Dynamic Address Group*	NAT *Source* IP
Source Port	*Action Flags*	*Tunnel* ID/IMSI
sdwan_device_type	*Destination Device Model*	session_owner
NAT *Destination* IP	*Destination Port*	*Source Country*
Monitor Tag/IMEI	sdwan_cluster_type	*Destination Device Vendor*
High Res Timestamp	Rule	NAT *Source Port*
Destination Country	*Parent Session* ID	sdwan_site
Destination Device OS Family	nssai_sst	*Source User*
NAT *Destination Port*	*Packets Sent*	*Parent Session Start Time*
dynusergroup_name	*Destination Device OS Version*	nssai_sd
Destination User	*Flags*	*Packets Received*
Tunnel	XFF *address*	*Destination Hostname*
Subcategory of app	*Application*	IP *Protocol*
Session End Reason	SCTP *Association* ID	*Source Device Category*
Destination Mac Address	*Category of app*	

Fonte: Elaborado pelo autor.

especialmente entre fevereiro e outubro de 2025. A classe *Deny* também apresentou

Tabela 4 – Distribuição em valores absolutos e porcentagem das classes em três coletas

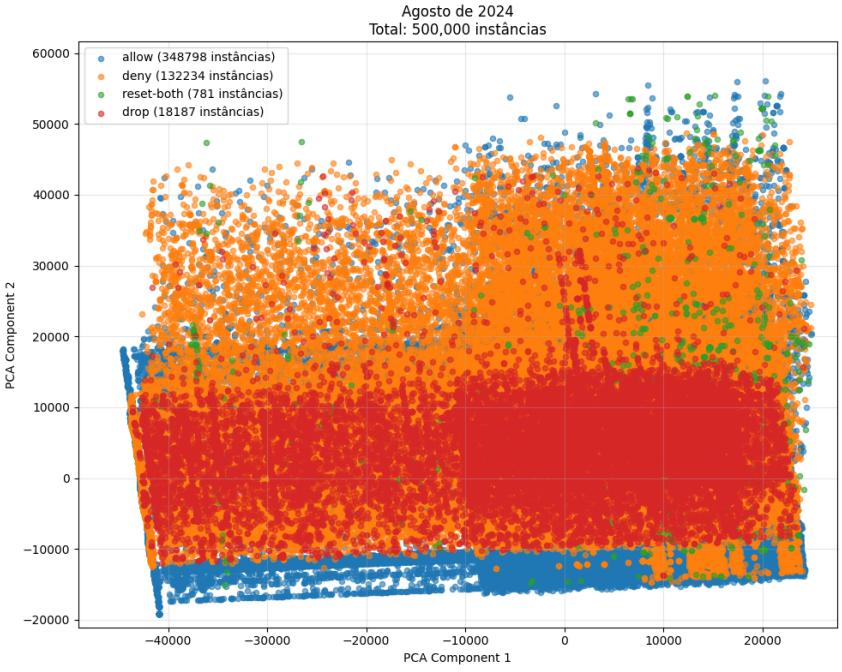
Classe	**Agosto/2024**		**Fevereiro/2025**		**Outubro/2025**	
	Qtd	**%**	**Qtd**	**%**	**Qtd**	**%**
Allow	348.798	69,76%	322.082	64,42%	206.624	41,32%
Deny	132.234	26,45%	85.352	17,07%	161.339	32,27%
Reset-Both	781	0,16%	531	0,11%	589	0,12%
Drop	18.187	3,64%	92.035	18,41%	131.448	26,29%

Fonte: Elaborado pelo autor.

variações na distribuição entre as duas primeiras coletas, seguidas de um aumento significativo na terceira. Esses resultados indicam modificações nas políticas de bloqueio ou nos padrões de tráfego, possivelmente decorrentes de ajustes operacionais ou de novas demandas de segurança.

A Figura 17 apresenta a distribuição dos dados referentes a agosto de 2024. Observa-se a predominância da classe *Allow* e a menor ocorrência da classe *Drop*, o que indica um padrão de filtragem mais permissivo nesse período.

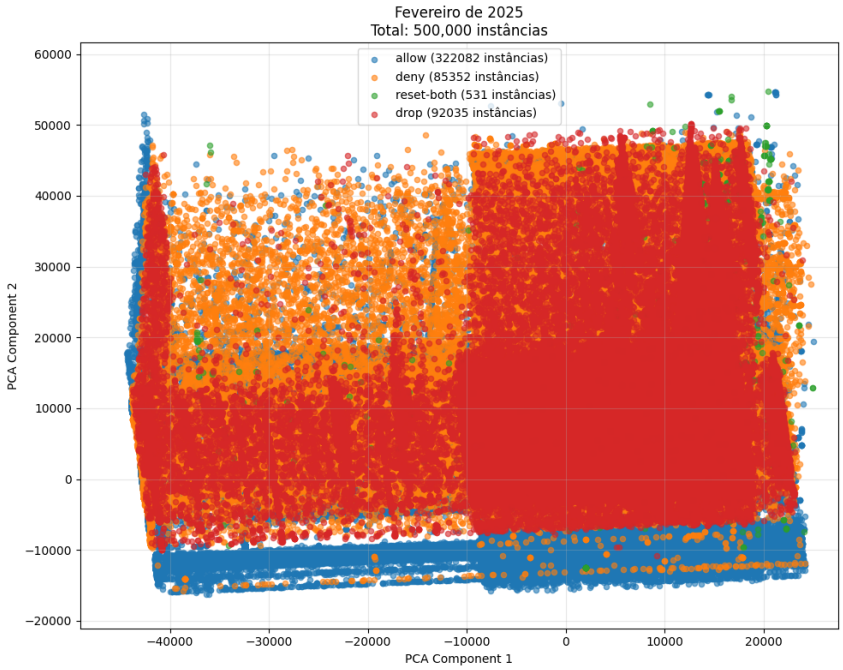
Figura 17 – Projeção PCA dos dados coletados em agosto de 2024



Fonte: Elaborado pelo autor.

A Figura 18 apresenta os dados de fevereiro de 2025; observa-se uma redistribuição significativa entre as classes. A classe *Drop* apresenta um crescimento expressivo, indicando a intensificação das ações de bloqueio em relação ao período anterior.Em contrapartida, as participações das classes *Allow* e *Deny* apresentam uma redução proporcional no conjunto de dados.

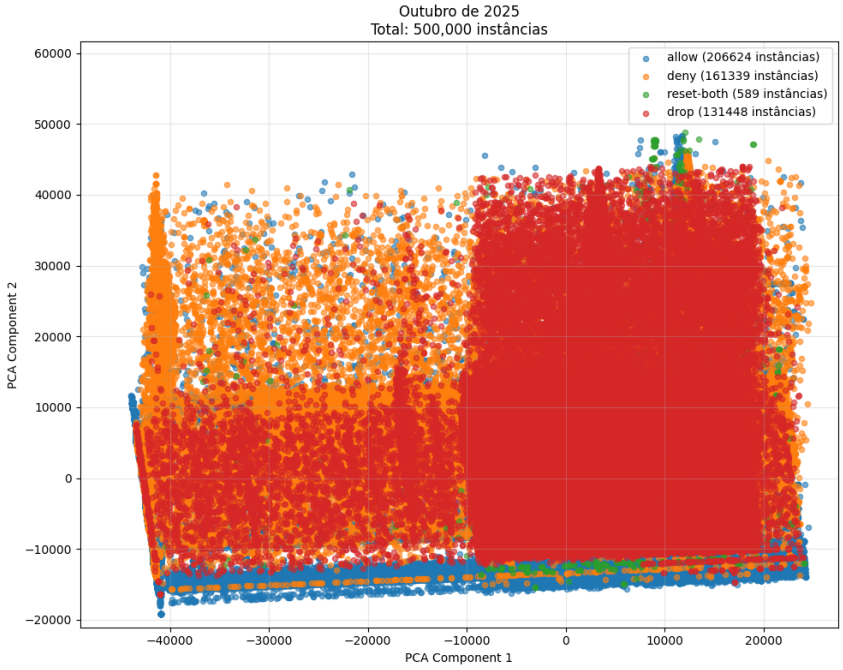
Figura 18 – Projeção PCA dos dados coletados em fevereiro de 2025



Fonte: Elaborado pelo autor.

Por fim, a Figura 19 apresenta o cenário de outubro de 2025. A classe *Drop* torna-se dominante, enquanto a classe *Allow* reduz-se a menos da metade do volume observado inicialmente. Já a classe *Deny* volta a crescer de forma expressiva, refletindo alterações nas políticas do *firewall*.

Figura 19 – Projeção PCA dos dados coletados em outubro de 2025



Fonte: Elaborado pelo autor.

Essas transformações, evidenciadas tanto nas métricas quantitativas quanto nas projeções em PCA, refletem o impacto das revisões nas políticas de segurança, a adaptação a novos contextos de uso e a evolução dos padrões de tráfego.

7.1.1 Agosto de 2024

Foi realizada uma análise detalhada da distribuição das classes em blocos de 50.000 instâncias, totalizando 10 blocos por conjunto de dados. No conjunto de dados de agosto de 2024, observa-se que a classe *Allow* se mantém majoritária ao longo do período, enquanto a classe *Deny* apresenta crescimento gradual. As classes *Reset-Both* e *Drop* permanecem minoritárias, com pequenas oscilações, sem indícios de alterações abruptas.

Esses resultados indicam que não houve alterações significativas na distribuição dos dados em agosto de 2024, o que se alinha com as projeções em PCA, que mostram padrões semelhantes entre os blocos. A Tabela 5 apresenta a variação percentual das classes em cada bloco.

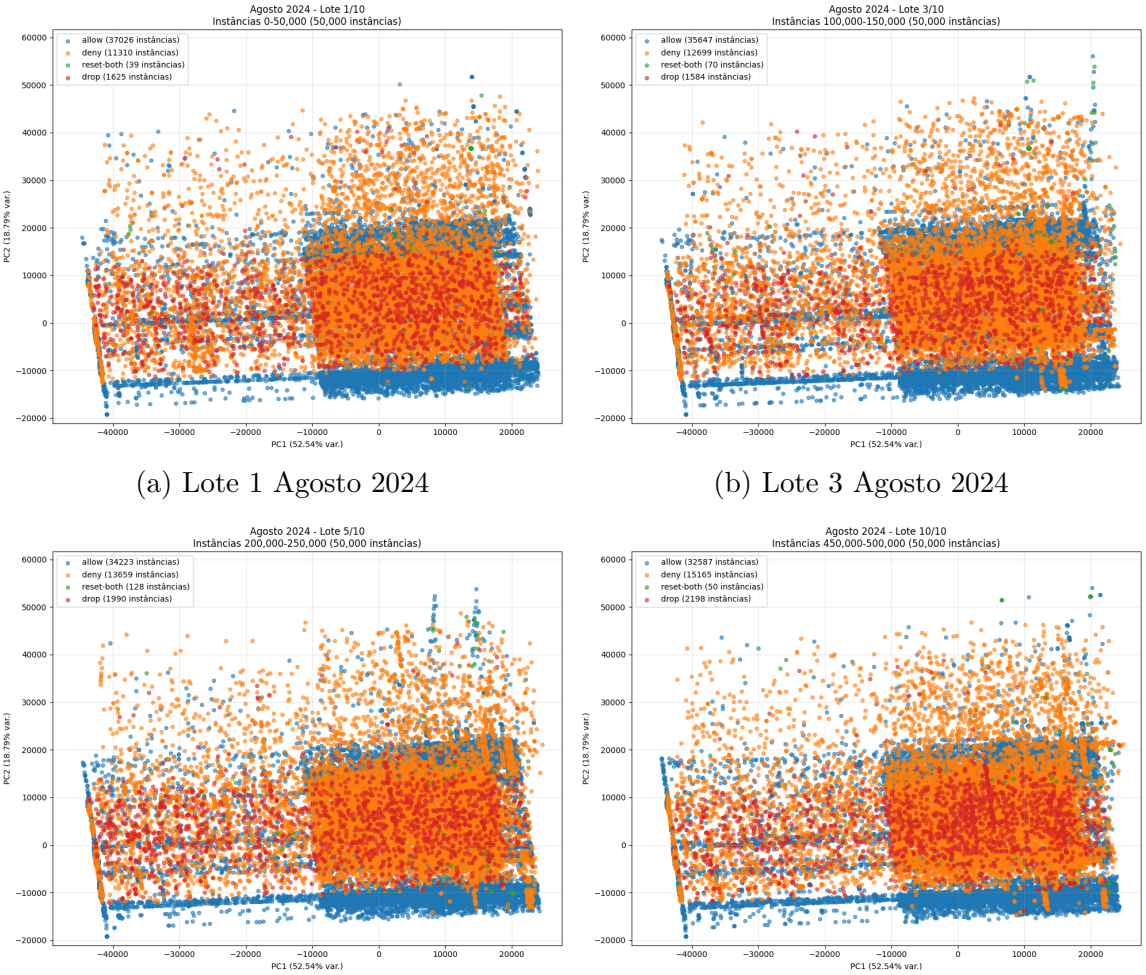
Tabela 5 – Distribuição das classes nos blocos em agosto de 2024.

Bloco	***Allow***	***Deny***	***Reset-Both***	***Drop***
1	74,05%	22,62%	0,08%	3,25%
2	70,84%	25,87%	0,16%	3,14%
3	71,29%	25,40%	0,14%	3,17%
4	72,33%	24,52%	0,13%	3,02%
5	68,45%	27,32%	0,26%	3,98%
6	68,76%	26,08%	0,16%	5,01%
7	69,70%	27,00%	0,19%	3,11%
8	69,75%	26,89%	0,22%	3,14%
9	67,25%	28,45%	0,13%	4,17%
10	65,17%	30,33%	0,10%	4,40%

Fonte: Elaborado pelo autor.

As projeções em PCA dos diferentes lotes do conjunto de agosto de 2024, conforme as imagens da Figura 20, mostram uma distribuição bastante estável ao longo do tempo. A organização dos pontos mantém-se semelhante entre os lotes, sem indícios de mudanças de conceito abruptas ou de deslocamentos significativos nas regiões ocupadas pelas instâncias.

Figura 20 – Projeções PCA por lotes selecionados do conjunto de dados de agosto de 2024



(a) Lote 1 Agosto 2024

(b) Lote 3 Agosto 2024

(c) Lote 5 Agosto 2024

(d) Lote 10 Agosto 2024

Fonte: Elaborado pelo autor.

7.1.2 Fevereiro de 2025

Durante fevereiro de 2025, a distribuição das classes apresentou maior variabilidade em relação ao período anterior. A classe *Allow* permaneceu predominante na maior parte dos blocos, embora com variações mais amplas ao longo do tempo. As classes *Deny* e *Reset-Both* mantiveram-se relativamente estáveis, com participação reduzida no conjunto.

Em contrapartida, a classe *Drop* apresenta um crescimento expressivo em alguns blocos, indicando um ajuste gradual nas decisões de bloqueio do *firewall*. Embora essas alterações não configurem uma mudança de conceito abrupta, sugerem uma evolução gradual, possivelmente associada a modificações no tráfego ou nas políticas de filtragem adotadas ao longo do período.

Na Tabela 6 é apresentada a distribuição percentual das classes nos blocos analisados. Embora o comportamento geral permaneça estável, há indícios de maior

variação em comparação a agosto de 2024, especialmente na classe *Drop*.

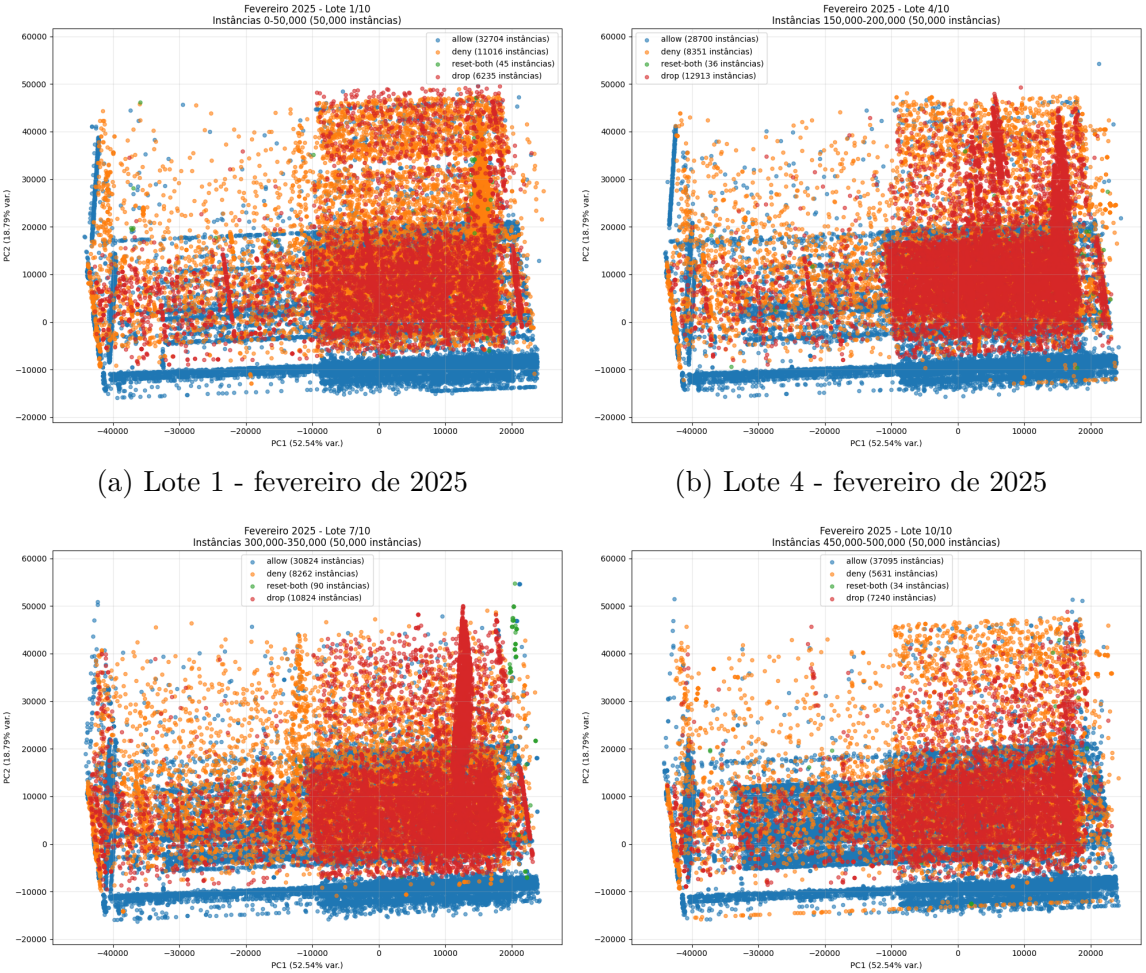
Tabela 6 – Distribuição das classes nos blocos de 50.000 instâncias em fevereiro de 2025.

Bloco	***Allow***	***Deny***	***Reset-Both***	***Drop***
1	65,41%	22,03%	0,09%	12,47%
2	68,73%	17,25%	0,19%	13,83%
3	65,33%	16,44%	0,08%	18,14%
4	57,40%	16,70%	0,07%	25,83%
5	56,76%	18,60%	0,04%	24,60%
6	69,47%	15,31%	0,09%	15,13%
7	61,65%	16,52%	0,18%	21,65%
8	57,06%	21,10%	0,14%	21,70%
9	68,16%	15,48%	0,11%	16,25%
10	74,19%	11,26%	0,07%	14,48%

Fonte: Elaborado pelo autor.

As projeções em PCA dos diferentes lotes do conjunto de fevereiro de 2025, apresentadas na Figura 21, reforçam essa observação. Visualmente, os agrupamentos das classes mantêm uma estrutura semelhante entre os blocos, porém, com maior variação da classe *Drop* em determinadas regiões do espaço de projeção. Esse comportamento indica uma possível transição nas características do tráfego, sem mudanças de conceito abruptas.

Figura 21 – Projeções PCA de lotes selecionados do conjunto de dados de fevereiro de 2025.



(a) Lote 1 - fevereiro de 2025

(b) Lote 4 - fevereiro de 2025

(c) Lote 7 - fevereiro de 2025

(d) Lote 10 - fevereiro de 2025

Fonte: Elaborado pelo autor.

7.1.3 Outubro de 2025

Durante outubro de 2025, a distribuição das classes mudou em relação aos períodos anteriores. A classe *Allow* apresentou queda consistente, enquanto *Deny* ganhou representatividade nos blocos intermediários, tornando-se predominante nas etapas finais. A classe *Drop* manteve participação relevante e comportamento oscilante, enquanto *Reset-Both* teve participação mínima ao longo de todo o período.

Essas alterações refletem uma mudança nas decisões do *firewall*, possivelmente em resposta a variações no tráfego ou a ajustes nas políticas de segurança. O fenômeno mais relevante é a inversão entre *Allow* e *Deny*, que caracteriza uma mudança de conceito abrupta, quando comparada com períodos anteriores. A Tabela 7 apresenta a distribuição percentual das classes nos blocos analisados, evidenciando as tendências observadas.

Tabela 7 – Distribuição das classes nos blocos de 50.000 instâncias durante outubro de 2025.

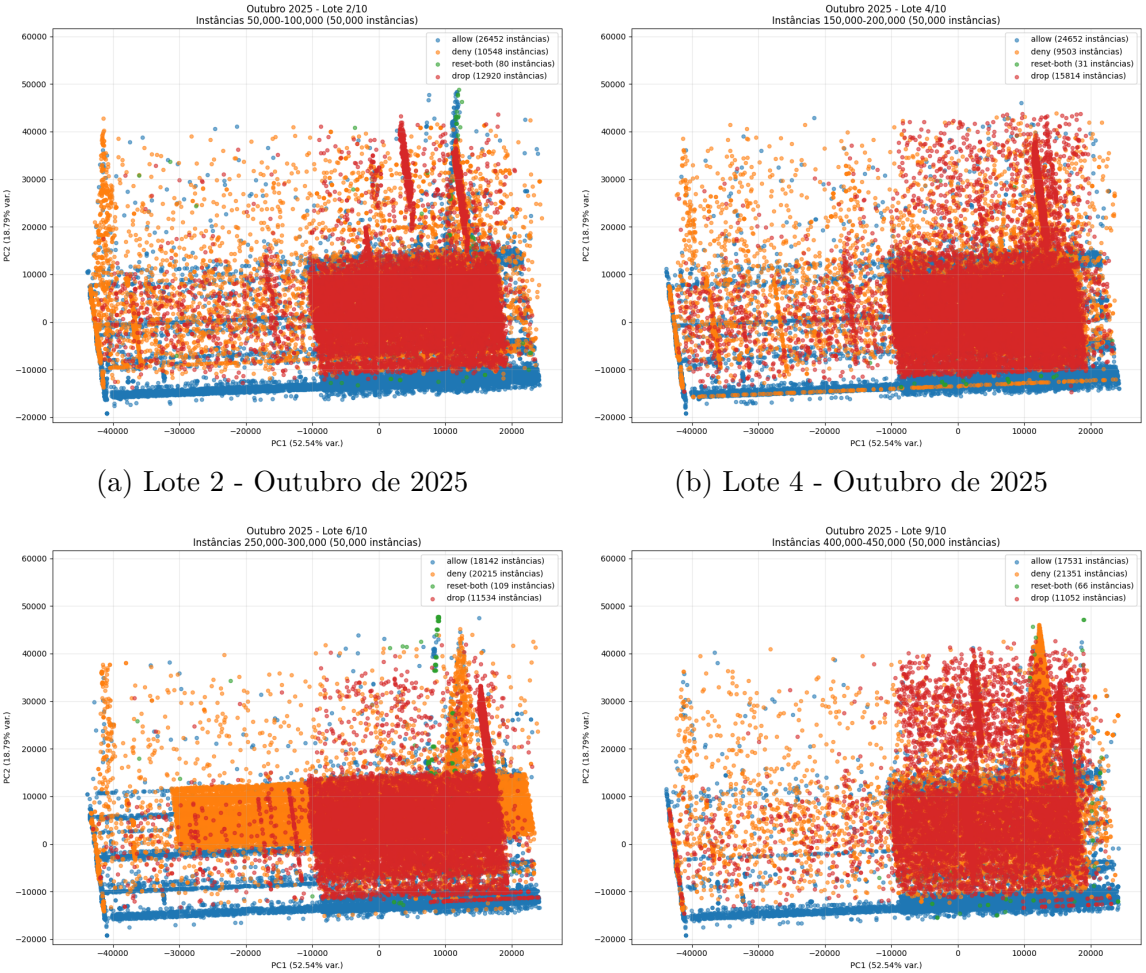
Bloco	***Allow***	***Deny***	***Reset-Both***	***Drop***
1	47,95%	16,60%	0,11%	35,34%
2	52,90%	21,10%	0,16%	25,84%
3	50,71%	19,71%	0,13%	29,44%
4	49,30%	19,01%	0,06%	31,63%
5	41,84%	30,27%	0,10%	27,80%
6	36,28%	40,43%	0,22%	23,07%
7	35,40%	45,76%	0,08%	18,76%
8	32,49%	47,00%	0,07%	20,45%
9	35,06%	42,70%	0,13%	22,10%
10	31,31%	40,11%	0,12%	28,46%

Fonte: Elaborado pelo autor.

As projeções em PCA para outubro de 2025, apresentadas na Figura 22, confirmam essa análise. Visualmente, os grupos das classes permanecem separados, porém, destaca-se uma mudança no comportamento da classe *Deny* nos lotes intermediários: ela apresenta maior densidade e dispersão, o que corresponde ao aumento de sua predominância. Já nos blocos finais, o padrão retorna à configuração original.

As diferenças observadas entre as três coletas não se restringem à variação na proporção das classes. As projeções em PCA também sugerem deslocamentos nas regiões ocupadas pelas ações do *firewall* ao longo do tempo, indicando mudanças na relação entre os atributos de entrada e as classes associadas a cada registro. Em outras palavras, a relação entre os dados de entrada e a decisão do *firewall* mudou ao longo do tempo, o que sugere alteração de $p(y|x)$ e caracteriza um cenário de Mudança de Conceito.

Figura 22 – Projeções PCA por lotes selecionados do conjunto de dados de outubro de 2025.



(a) Lote 2 - Outubro de 2025

(b) Lote 4 - Outubro de 2025

(c) Lote 6 - Outubro de 2025

(d) Lote 9 - Outubro de 2025

Fonte: Elaborado pelo autor.

7.2 Pré-processamento e Particionamento dos Dados

O particionamento dos dados neste estudo foi realizado de modo a refletir cenários reais de Fluxo Contínuo de Dados (FCDs), respeitando a ordem temporal de chegada das instâncias. Inicialmente, os dados coletados a partir dos logs do *firewall* foram divididos em duas fases principais: a fase *offline* e a fase *online*. Na fase *offline*, um conjunto estático de instâncias foi utilizado para o treinamento inicial e a validação dos modelos, contendo dados rotulados que permitiram a configuração e a adaptação preliminar dos parâmetros do modelo escolhido. Já na fase *online*, o restante das instâncias foi processado em sequência, simulando um ambiente de fluxo real em que novas sessões de tráfego chegam continuamente e são classificadas de forma incremental, sem repetição e sem acesso aos rótulos durante o processamento. Essa

divisão garante que a avaliação dos modelos reflita condições reais de operação, nas quais o sistema precisa lidar com a chegada de dados em tempo real e adaptar-se a possíveis mudanças no padrão de tráfego.

Em ambientes reais, a mudança no comportamento dos dados não decorre apenas de variações estatísticas espontâneas do tráfego, mas também de alterações deliberadas na política de segurança. Novas regras podem ser criadas, regras antigas podem ser ajustadas, e critérios de bloqueio podem se tornar mais ou menos restritivos em resposta a novos ataques, mudanças de infraestrutura ou revisão administrativa. Nesse contexto, uma mesma combinação de atributos pode passar a receber decisões diferentes ao longo do tempo. Isso é relevante pois a classe prevista representa a ação aplicada pelo *firewall*. Assim, alterações nas regras podem impactar diretamente a associação entre os atributos observados e a classe alvo.

Um ponto importante do pré-processamento foi a estratégia adotada para a normalização dos atributos. Especificamente, a técnica de normalização Min-Max foi aplicada em blocos de 1.000 registros. Isso significa que o sistema acumula temporariamente um lote de 1.000 registros, normaliza os valores com base no mínimo e no máximo identificados nesse bloco e, só então, encaminha os dados normalizados para as etapas de classificação. Essa abordagem, além de garantir maior eficiência computacional e adaptabilidade a diferentes padrões dinâmicos presentes no fluxo, mantém a consistência da escala dos dados em cada bloco analisado. Outro benefício relevante é que ela dispensa a necessidade de atualização contínua dos parâmetros ao longo de todos os FCDs, simplificando a implementação em cenários reais de processamento em tempo real.

Para enfrentar o problema do desbalanceamento das classes, foram realizados experimentos *offline* utilizando estratégias de *oversampling* sintético, especificamente ADASYN e SMOTE (BRANDT; LANZÉN, 2021), além da criação de um conjunto de treinamento totalmente balanceado, com 125.000 instâncias para cada classe. Apesar dos esforços, observou-se que tais técnicas não geraram ganhos relevantes para as classes minoritárias *Reset-Both* e *Drop*. Em todos os cenários testados, as métricas de avaliação, como F1-Score, permaneciam próximas de zero para essas classes, indicando a dificuldade em classificar corretamente as classes minoritárias, mesmo com o balanceamento artificial dos dados. Este resultado evidencia os limites das técnicas de reamostragem em contextos de desbalanceamento extremo, em que classes muito esparsas podem não oferecer padrões generalizáveis.

Foram realizadas três coletas de dados distintas, com o objetivo de capturar padrões variados de tráfego de rede em diferentes períodos acadêmicos e operacionais da instituição. A primeira coleta ocorreu entre os dias 19/08/2024 e 23/08/2024, totalizando 5 dias consecutivos de monitoramento em tempo real. Durante esse

período, os logs foram coletados em diferentes horários do dia, abrangendo tanto os picos de utilização, como os horários de aula e de expediente administrativo, quanto períodos de menor atividade, com o intuito de garantir uma amostragem representativa do comportamento da rede. Essa coleta resultou em um total de 1.563.940 instâncias, das quais 545.365 foram utilizadas na fase *offline*, dedicada à construção inicial do modelo, e 1.018.575 foram destinadas à fase *online*, que simula um cenário de classificação em FCDs.

A segunda coleta de dados foi realizada posteriormente, entre 18/02/2025 e 11/03/2025, totalizando 22 dias consecutivos. Essa etapa adicional teve como objetivo oferecer uma comparação direta entre diferentes momentos operacionais da rede, refletindo possíveis mudanças nos padrões de uso e na dinâmica do tráfego ao longo do tempo. Os registros foram distribuídos em vários dias e horários, totalizando 1.654.433 instâncias, sendo 605.862 destinadas à fase *offline* e 1.048.571 à fase *online*.

A terceira coleta de dados ocorreu entre 13/10/2025 e 27/10/2025, por 15 dias consecutivos. Esta etapa final teve como objetivo capturar padrões de tráfego em um período acadêmico distinto, permitindo observar a evolução temporal das políticas e do comportamento da rede. Assim como nas coletas anteriores, os registros foram distribuídos em diversos horários ao longo dos dias, assegurando a diversidade dos fluxos capturados. O conjunto resultante totalizou 1.707.415 instâncias, sendo 624.482 destinadas à fase *offline* e 1.082.933 à fase *online*.

Para as três coletas, os dados passaram por um processo de pré-processamento, fundamental para garantir a comparabilidade entre os experimentos realizados em diferentes períodos temporais.

A definição dos atributos utilizados nos experimentos foi realizada a partir de avaliações preliminares sobre o conjunto de logs exportado pelo *firewall*, originalmente composto por 113 atributos. Esse conjunto incluía atributos relacionados à identificação das aplicações e de suas categorias, aos protocolos de comunicação, às zonas de segurança, à origem e ao destino do tráfego, a informações temporais dos registros, a métricas de volume e duração das sessões, bem como a dados operacionais, como a regra aplicada, a ação executada e o motivo de encerramento da sessão.

A seleção dos atributos foi orientada por dois critérios complementares: o desempenho observado nos experimentos preliminares e a aderência dos atributos ao contexto operacional do problema. Assim, buscou-se selecionar atributos capazes de representar características relevantes para a decisão do *firewall*, preservando informações sobre o tipo de aplicação envolvida, o contexto da comunicação e o domínio de segurança associado ao tráfego analisado.

Durante essas avaliações, observou-se que os atributos numéricos não contribuíram de forma relevante para o desempenho do modelo nos cenários analisados. Em con-

trapartida, combinações formadas apenas por atributos categóricos apresentaram resultados mais consistentes. Com base nesse comportamento, foram mantidos os atributos que preservaram melhor desempenho ao longo dos experimentos: *application*, *subcategory of app*, *source country*, *destination zone*, *category*, *IP protocol* e *technology of app*.

A escolha desses atributos também se justifica por sua relevância no contexto de segurança de rede. Os atributos *application*, *subcategory of app*, *category* e *technology of app* descrevem o serviço ou a aplicação associada ao fluxo, fornecendo informação importante para distinguir diferentes comportamentos de tráfego. O atributo *destination zone* representa o contexto de segmentação da rede e está diretamente relacionado às políticas de segurança entre diferentes domínios. O atributo *source country* adiciona uma informação contextual sobre a origem do tráfego, podendo refletir padrões recorrentes observados no ambiente monitorado. Já o atributo *IP protocol* contribui para diferenciar comportamentos associados a distintos mecanismos de transporte e comunicação.

Por outro lado, atributos diretamente associados à decisão já tomada pelo *firewall*, como a regra aplicada, a ação executada e o motivo de encerramento da sessão, não foram utilizados como variáveis de entrada. A exclusão desses campos foi necessária para evitar vazamento de informação para o modelo, uma vez que tais atributos incorporam, de forma direta ou indireta, o resultado da decisão que se deseja prever.

Após a seleção dos atributos, os campos categóricos escolhidos foram convertidos para representação numérica por meio de um mapeamento fixo entre cada categoria observada e um valor inteiro correspondente. Esse mapeamento foi definido na fase *offline* e posteriormente reaplicado na fase *online*, de modo a preservar a consistência da representação dos atributos ao longo de todo o fluxo de dados. Assim, uma mesma categoria manteve sempre o mesmo código numérico em todas as etapas dos experimentos.

Após a etapa de codificação, os dados foram normalizados para o intervalo $[0, 1]$ por meio da técnica Min-Max. Na fase *offline*, a normalização foi ajustada sobre o conjunto de treinamento inicial, permitindo uniformizar a escala dos atributos antes da formação da topologia do GSOM. Na fase *online*, a transformação foi aplicada em blocos de 1.000 instâncias, respeitando a natureza sequencial do fluxo e evitando o uso de informação futura durante o processamento.

A normalização foi necessária porque o GSOM baseia seu processo de competição e adaptação em medidas de distância no espaço de entrada. Dessa forma, diferenças de escala entre atributos poderiam influenciar indevidamente a seleção do neurônio vencedor e o ajuste dos pesos, comprometendo a organização do mapa e a qualidade da classificação.

Para oferecer uma visão consolidada da composição dos conjuntos de dados, a Tabela 8 apresenta a distribuição completa das três coletas, detalhada por ação do *firewall* (*Allow*, *Deny*, *Reset-Both* e *Drop*) e pelo total de instâncias obtidas em cada período.

Tabela 8 – Distribuição detalhada das três coletas por fase e ação

Coleta	***Allow***	***Deny***	***Drop***	***Reset-Both***	**Total**
Coleta 1	972.652	540.592	78.587	2.109	**1.564.940**
Coleta 2	944.150	401.031	307.960	1.292	**1.654.433**
Coleta 3	602.502	736.839	366.581	1.493	**1.707.415**

Os quatro experimentos foram definidos com base nas três coletas de logs de *firewall*, seguindo as distribuições de ações e fases apresentadas na Tabela 9. No Experimento 1, foram utilizados exclusivamente os dados da Coleta 1, enquanto o Experimento 2 considerou apenas a Coleta 2 e o Experimento 3 apenas a Coleta 3. Por fim, o Experimento 4 foi composto pela combinação das Coletas 1, 2 e 3, permitindo avaliar o desempenho dos modelos em um cenário mais abrangente e heterogêneo.

Tabela 9 – Distribuição detalhada dos experimentos por fase e ação

Experimento	**Fase**	***Allow***	***Deny***	***Drop***	***Reset-Both***	**Total**
Experimento 1	*offline*	276.490	229.104	39.208	563	545.365
	online	696.162	311.488	39.379	1.546	1.018.575
	Total	**972.652**	**540.592**	**78.587**	**2.109**	**1.564.940**
Experimento 2	*offline*	387.351	104.689	113.182	640	605.862
	online	556.799	296.342	194.778	652	1.048.571
	Total	**944.150**	**401.031**	**307.960**	**1.292**	**1.654.433**
Experimento 3	*offline*	219.620	176.360	145.629	609	542.218
	online	382.882	560.479	220.952	884	1.165.197
	Total	**602.502**	**736.839**	**366.581**	**1.493**	**1.707.415**
Experimento 4	*offline*	348.798	132.234	18.187	781	500.000
	online	528.706	246.691	223.483	1.120	1.000.000
	Total	**877.504**	**378.925**	**241.670**	**1.901**	**1.500.000**

Os códigos-fonte utilizados neste trabalho foram disponibilizados em repositório público no GitHub.[1] Em relação aos dados, foi disponibilizada uma versão tratada do conjunto utilizado nos experimentos, contendo apenas os atributos empregados no modelo. Essa decisão foi adotada para preservar a privacidade e evitar a exposição de informações sensíveis presentes nos logs originais do *firewall*. Detalhes adicionais sobre essa disponibilização são apresentados no Apêndice A.

[1] <https://github.com/wgiarini/gsom-firewall>

7.3 Treinamento e Operação Contínua do GSOM

O algoritmo GSOM foi adotado neste trabalho devido à sua capacidade de expansão dinâmica da topologia e de capturar padrões emergentes em cenários de dados não estacionários. Diferentemente do SOM tradicional, cuja grade é fixa, o GSOM ajusta automaticamente sua estrutura à complexidade dos dados, permitindo representar regiões mais densas ou mais heterogêneas do espaço de entrada.

O processo de treinamento foi estruturado em duas fases complementares: uma fase offline, responsável pela formação inicial da topologia e pela organização global dos dados, e uma fase online, na qual o modelo passa a operar em FCDs, processando as instâncias sequencialmente e ajustando-se de forma incremental a novos padrões. Embora a lógica geral de treinamento tenha sido mantida em todos os experimentos, os hiperparâmetros da fase *offline* foram ajustados de acordo com o cenário avaliado, conforme resumido na Tabela 10.

Tabela 10 – Configuração do treinamento *offline* do GSOM nos experimentos realizados

Experimento	**Instâncias**	**SF**	**Épocas de crescimento**	**Épocas de suavização**
1	545.365	0,6	15	7
2	605.862	0,4	20	15
3	542.218	0,6	25	15
4	500.000	0,7	12	7

Fonte: Elaborado pelo autor.

A Tabela 10 mostra que os hiperparâmetros do GSOM não foram mantidos fixos entre os experimentos. Essa escolha permitiu ajustar o nível de crescimento da rede e o tempo de refinamento da topologia às características de cada conjunto de dados, preservando a mesma estrutura metodológica de treinamento e avaliação.

7.3.1 Fase *offline*

Na fase *offline*, o GSOM é treinado em modo *batch*, permitindo múltiplas passagens pelo conjunto de dados. Essa etapa tem como objetivo formar a topologia inicial do mapa e ajustar os vetores de peso dos neurônios com base nos padrões observados no conjunto de treinamento. A rede é inicializada com quatro neurônios organizados em uma grade bidimensional 2×2, com pesos iniciais atribuídos no intervalo dos dados normalizados.

O treinamento *offline* é composto por duas subfases principais:

1. **Fase de crescimento (*Growing Phase*)**
   Nesta etapa, o mapa expande sua topologia de forma dinâmica. Para cada instância apresentada, identifica-se o neurônio vencedor e acumula-se seu erro

de quantização. Quando esse erro ultrapassa o limiar de crescimento, denominado *Growth Threshold* (GT), novos neurônios podem ser inseridos nas regiões de fronteira do mapa. O valor de GT é obtido a partir do *Spread Factor* (SF), segundo a relação apresentada no Capítulo 5:

$$GT = -D \cdot \ln(SF),$$

em que D representa a dimensionalidade do vetor de entrada. Desse modo, valores mais altos de SF reduzem o GT e favorecem maior expansão da rede, enquanto valores mais baixos aumentam o GT e resultam em mapas mais compactos.

2. **Fase de suavização (*Smoothing Phase*)**
   Após a expansão inicial da topologia, os pesos dos neurônios são refinados com taxa de aprendizado reduzida, sem a inserção de novos neurônios. Essa etapa tem como finalidade melhorar a organização interna do mapa, reduzir distorções introduzidas durante o crescimento e tornar a representação topológica mais estável para a etapa posterior de operação contínua.

Ao final da fase *offline*, obtém-se uma topologia inicial capaz de representar os padrões predominantes do conjunto de treinamento. Essa topologia constitui a base de operação do modelo na fase *online*, podendo ser mantida fixa ou atualizada de forma incremental, conforme a estratégia adotada.

7.3.2 Fase *online*

Na fase *online*, o GSOM passa a operar em ambiente de FCDs, no qual cada instância é processada uma única vez, em sequência temporal, sem revisitação de exemplos anteriores. Esse cenário impõe restrições típicas de aplicações em tempo real, como limitação de memória, necessidade de resposta imediata e adaptação contínua a possíveis mudanças na distribuição dos dados.

Nessa etapa, o objetivo deixa de ser apenas organizar globalmente o espaço de entrada e passa a ser equilibrar dois requisitos concorrentes: preservar a estrutura aprendida na fase *offline* e, ao mesmo tempo, adaptar o modelo a padrões emergentes observados ao longo do fluxo. Para investigar esse comportamento, foram avaliadas três estratégias de operação *online*, com diferentes níveis de adaptação:

1. ***get_winner***
   Nesta estratégia, o GSOM atua de forma estática. Para cada nova instância, identifica-se apenas o neurônio vencedor, sem atualização de pesos e sem modificação da topologia. Essa configuração permite avaliar a capacidade de generalização da estrutura aprendida exclusivamente na fase *offline*.

2. ***growing_phase***
   Nesta estratégia, o GSOM continua apto a expandir sua topologia durante a fase *online*. Sempre que o erro acumulado de um neurônio ultrapassa o limiar definido pelo SF, novos neurônios podem ser adicionados ao mapa. Essa abordagem é particularmente relevante em cenários nos quais surgem padrões não representados adequadamente na fase inicial de treinamento.

3. ***smoothing_phase***
   Nesta estratégia, a topologia do mapa é mantida fixa, mas os pesos dos neurônios continuam sendo ajustados de forma incremental. Assim, o modelo preserva a estrutura formada na fase *offline*, ao mesmo tempo em que realiza um refinamento contínuo da representação, o que favorece a adaptação a variações graduais no fluxo de dados.

A comparação entre essas três estratégias permitiu analisar como diferentes formas de atualização impactam o desempenho do GSOM em FCDs. De modo geral, a configuração adotada neste trabalho procura refletir condições realistas de operação em sistemas de monitoramento contínuo: cada instância é observada apenas uma vez, as atualizações são locais e incrementais, e o modelo precisa adaptar-se sem depender de reprocessamento completo do histórico de dados.

7.4 Tarefa de Classificação

Este trabalho adota a classificação multiclasse, distinguindo as quatro ações registradas pelo *firewall* : *Allow*, *Deny*, *Reset-Both* e *Drop*. Esta especificidade permite uma análise mais detalhada das políticas de segurança, identificando padrões associados a cada tipo de decisão.

O problema apresenta não apenas desequilíbrio entre classes, mas também variação temporal significativa em suas distribuições: *Drop* evolui de classe minoritária para majoritária entre a primeira e a terceira coleta, enquanto *Allow* segue uma trajetória inversa. Essa dinâmica caracteriza um cenário de mudança de conceito, pois a distribuição das classes e a relação entre os atributos observados e as decisões do *firewall* variam ao longo do tempo. Esse comportamento impõe aos modelos o desafio de manter desempenho diante de mudanças no padrão do tráfego e de possíveis alterações nas regras ou políticas de segurança.

A análise teve início com a interpretação das matrizes de confusão. Em seguida, o desempenho por classe foi avaliado por meio das métricas de *Precision*, *Recall* e *F1-Score*. Por fim, foram calculadas as médias *Macro* e *Weighted* dessas métricas, a fim de obter medidas agregadas de desempenho global. Essa abordagem combinada

permite uma visão mais abrangente do comportamento dos modelos em um cenário multiclasse com desbalanceamento entre as classes.

Capítulo 8

Experimentos e Resultados

Este capítulo apresenta os experimentos realizados e seus resultados para avaliar o desempenho do GSOM na classificação de logs de *firewall* em FCDs. São comparadas as três estratégias de atualização do GSOM (*get_winner*, *growing_phase* e *smoothing_phase*), com o objetivo de identificar qual configuração oferece o melhor equilíbrio entre desempenho geral e capacidade de detecção de classes minoritárias.

Para contextualizar o desempenho do GSOM, seus resultados são confrontados com os de dois métodos consagrados na literatura para classificação em FCDs: o SVM *Incremental* e o ARF. É importante ressaltar que, diferentemente do GSOM, ambos são métodos supervisionados, o que implica a necessidade de rótulos na fase *online*.

Os resultados são analisados por meio de métricas por classe (*Precision*, *Recall*, *F1-Score*), de suas médias gerais (*Macro* e *Weighted*) e de matrizes de confusão, permitindo uma avaliação completa, considerando o cenário multiclasse e o desbalanceamento dos dados. Por fim, discute-se a robustez dos modelos diante das mudanças de conceito observadas entre as coletas.

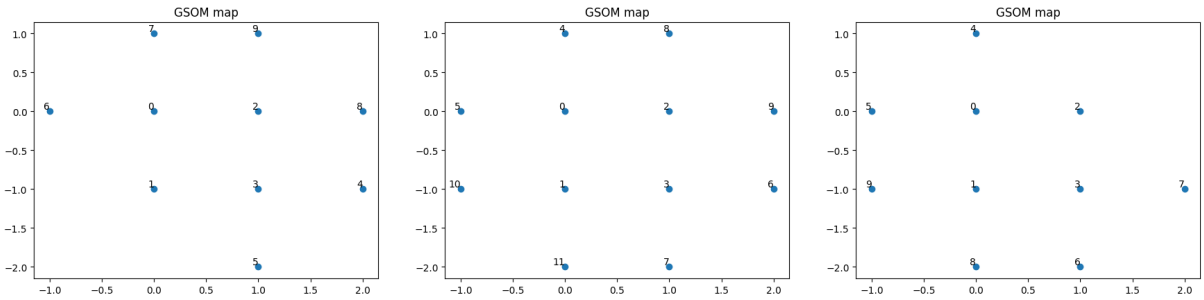
8.1 Experimento 1: agosto de 2024

O Experimento 1 avaliou o desempenho do GSOM na classificação do tráfego de rede, utilizando fluxos coletados em Agosto de 2024.

Para este experimento, utilizou-se *Spread Factor* (SF) igual a 0,6, com 15 épocas de crescimento e 7 de suavização durante a fase *offline*. Na fase *online*, os fluxos foram apresentados sequencialmente ao modelo, simulando um FCDs. Na Figura 23 é apresentada a diferença entre as três estratégias do GSOM por meio dos mapas

gerados por cada uma delas. A classificação foi realizada com base nessas estratégias, que diferem quanto à forma de atualizar o mapa.

- *get_winner*: mantém a estrutura aprendida na fase *offline*;
- *growing_phase*: expande o mapa adicionando novos neurônios;
- *smoothing_phase*: reorganiza os neurônios existentes sem expandir o mapa.



(a) *get_winner* (b) *growing_phase* (c) *smoothing_phase*

Figura 23 – Mapas das três estratégias do GSOM no Experimento 1.

O desempenho foi avaliado por meio de métricas por classe (*Precision*, *Recall*, *F1-Score*), de suas médias gerais (*Macro* e *Weighted*) e da análise das matrizes de confusão.

8.1.1 Estratégia *get_winner*

Na Tabela 11 é apresentada a matriz de confusão do GSOM para a estratégia *get_winner* na primeira coleta. Observa-se uma distribuição dos erros em que as classes *Allow* e *Deny* concentram a maioria das classificações corretas, enquanto as classes *Reset-Both* e *Drop* não obtiveram nenhum acerto.

Tabela 11 – Matriz de confusão do GSOM na primeira coleta (estratégia *get_winner*) em valores absolutos e porcentuais.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	694247	1915	0	0
Deny	79044	232444	0	0
Reset-Both	1544	2	0	0
Drop	6636	32743	0	0

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	99.73%	0.27%	0.00%	0.00%
Deny	25.38%	74.62%	0.00%	0.00%
Reset-Both	99.87%	0.13%	0.00%	0.00%
Drop	16.85%	83.15%	0.00%	0.00%

Fonte: Elaborado pelo autor.

Essa distribuição se reflete nas métricas por classe da Tabela 12, que evidenciam desempenhos muito diferentes. A classe *Allow* teve *Recall* de 99,73% e *Precision* de 88,84%, enquanto a classe *Deny* apresentou *Recall* de 74,62% e *Precision* de 87,02%. Já as classes *Reset-Both* e *Drop* zeraram todas as métricas.

Tabela 12 – Desempenho do GSOM por classe na primeira coleta (estratégia *get_winner*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	88,84%	99,73%	93,98%	696.162
Deny	87,02%	74,62%	80,35%	311.488
Reset-Both	0,00%	0,00%	0,00%	1.546
Drop	0,00%	0,00%	0,00%	39.379

Fonte: Elaborado pelo autor.

Na Tabela 13 é mostrada uma grande diferença entre as médias *Macro* e *Weighted*. Essa diferença ocorre porque a média *Macro* trata todas as classes igualmente, o que reflete o baixo desempenho nas classes minoritárias. Por outro lado, a média *Weighted* atribui mais peso às classes com mais instâncias, sendo assim influenciada pelo bom desempenho das classes majoritárias.

Tabela 13 – Médias de desempenho do GSOM na primeira coleta (estratégia *get_winner*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	43,97%	43,59%	43,58%
Weighted	84,85%	88,38%	86,31%

Fonte: Elaborado pelo autor.

O método *get_winner* mostrou boa capacidade de identificar padrões das classes majoritárias, com *Recall* de 99,73% para a classe *Allow* e *Precision* de 87,02% para a classe *Deny*. No entanto, não conseguiu classificar as classes minoritárias, que tiveram todas as métricas zeradas. Essa disparidade mostra que, devido ao número muito baixo de instâncias das classes *Reset-Both* e *Drop*, o GSOM teve grande dificuldade em aprender padrões dessas classes.

8.1.2 Estratégia *growing_phase*

Na Tabela 14 é apresentada a matriz de confusão do GSOM para a estratégia *growing_phase* na primeira coleta. Em comparação com *get_winner*, observa-se que a estratégia *growing_phase* aumentou as classificações corretas da classe *Allow*, mas reduziu as da classe *Deny*.

Na Tabela 15 são apresentadas as métricas por classe, evidenciando as mudanças no modelo. A classe *Allow* manteve um *Recall* alto, de 95,55%, mas perdeu em *Precision*, ficando em 77,96%. Já a classe *Deny* perdeu em *Recall*, caindo para 45,53%, e

Tabela 14 – Matriz de confusão do GSOM na primeira coleta (estratégia *growing_phase*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	665142	31020	0	0
Deny	169681	141807	0	0
Reset-Both	1544	2	0	0
Drop	16869	22510	0	0

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	95,54%	4,46%	0,00%	0,00%
Deny	54,46%	45,54%	0,00%	0,00%
Reset-Both	99,87%	0,13%	0,00%	0,00%
Drop	42,84%	57,16%	0,00%	0,00%

Fonte: Elaborado pelo autor.

ganhou em *Precision*, chegando a 72,59%. As classes minoritárias continuaram com todas as métricas zeradas.

Tabela 15 – Desempenho do GSOM por classe na primeira coleta (estratégia *growing_phase*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	77,96%	95,55%	85,87%	696.162
Deny	72,59%	45,53%	55,96%	311.488
Reset-Both	0,00%	0,00%	0,00%	1.546
Drop	0,00%	0,00%	0,00%	39.379

Fonte: Elaborado pelo autor.

Na Tabela 16 são apresentadas as médias de desempenho, nas quais se observa que a estratégia *growing_phase*, em comparação com a *get_winner*, obteve médias *Weighted* mais baixas e continuou sem classificar corretamente as classes minoritárias.

Tabela 16 – Médias de desempenho do GSOM na primeira coleta (estratégia *growing_phase*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	37,64%	35,27%	35,46%
Weighted	73,28%	76,96%	73,66%

Fonte: Elaborado pelo autor.

A estratégia *growing_phase*, que permite a expansão contínua do mapa, apresentou um problema. Ela piorou o equilíbrio entre *Precision* e *Recall* nas classes majoritárias e não consegue reconhecer as classes minoritárias.

8.1.3 Estratégia *smoothing_phase*

Na Tabela 17 é apresentada a matriz de confusão da estratégia *smoothing_phase*. Enquanto o *growing_phase* favoreceu a classe *Allow*, o *smoothing_phase* trouxe mais equilíbrio entre *Allow* e *Deny*.

Tabela 17 – Matriz de confusão do GSOM na primeira coleta (estratégia *smoothing_phase*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	523822	172340	0	0
Deny	30468	281020	0	0
Reset-Both	1511	35	0	0
Drop	2500	36879	0	0

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	75,24%	24,76%	0,00%	0,00%
Deny	9,78%	90,22%	0,00%	0,00%
Reset-Both	97,74%	2,26%	0,00%	0,00%
Drop	6,35%	93,65%	0,00%	0,00%

Fonte: Elaborado pelo autor.

Na Tabela 18 é apresentado um padrão nas métricas por classe. A classe *Allow* obteve *Precision* alta de 93,82% e *Recall* de 75,24%, enquanto a classe *Deny* alcançou *Recall* de 90,22% e *Precision* baixa de 57,32%.

Tabela 18 – Desempenho do GSOM por classe na primeira coleta (estratégia *smoothing_phase*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	93,82%	75,24%	83,52%	696.162
Deny	57,32%	90,22%	70,10%	311.488
Reset-Both	0,00%	0,00%	0,00%	1.546
Drop	0,00%	0,00%	0,00%	39.379

Fonte: Elaborado pelo autor.

Na Tabela 19 são apresentadas as médias de desempenho da estratégia *smoothing_phase*. As médias *Weighted* ficaram em uma posição intermediária entre as outras duas estratégias, enquanto as médias *Macro* permaneceram baixas, indicando que as classes minoritárias continuaram a apresentar problemas.

A estratégia *smoothing_phase*, que ajusta os pesos sem alterar a topologia, mostrou que cada classe majoritária se destacou em uma métrica distinta. *Allow* teve bom desempenho em *Precision*, e *Deny* em *Recall*. No entanto, não resolveu o problema das classes minoritárias, o que indica que ajustar apenas os pesos é insuficiente para lidar com o desequilíbrio entre as classes.

Tabela 19 – Médias de desempenho do GSOM na primeira coleta (estratégia *smoothing_phase*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	37,79%	41,37%	38,41%
Weighted	79,31%	76,76%	76,27%

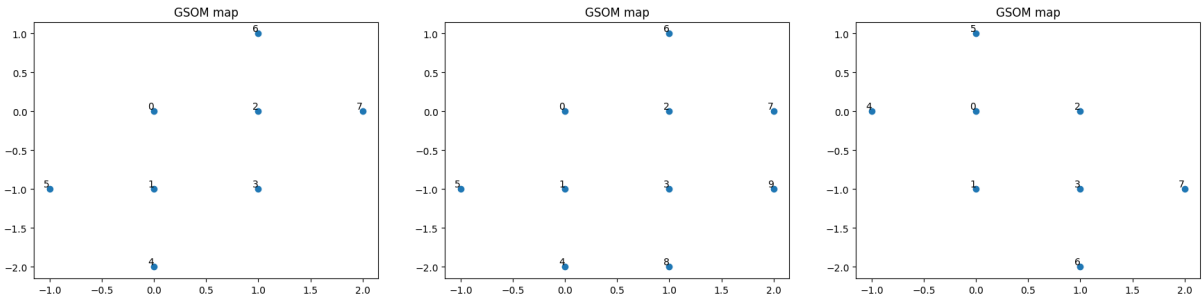
Fonte: Elaborado pelo autor.

8.2 Experimento 2: fevereiro de 2025

O Experimento 2 avaliou a estabilidade temporal do GSOM usando dados coletados em fevereiro de 2025. O objetivo foi verificar se os padrões do Experimento 1 se mantinham em outro período, testando a capacidade do modelo de lidar com mudanças na distribuição do tráfego.

Para este experimento, usou-se SF igual a 0,4, com 20 épocas de crescimento e 15 de suavização na fase *offline*. Na fase *online*, os fluxos foram apresentados sequencialmente ao modelo, simulando um FCDs. Na Figura 24 é apresentada a diferença entre as três estratégias do GSOM por meio dos mapas gerados por cada uma delas. A classificação foi realizada com base nessas estratégias, que diferem quanto à forma de atualizar o mapa.

- *get_winner*: mantém a estrutura aprendida na fase *offline*;
- *growing_phase*: expande o mapa adicionando novos neurônios;
- *smoothing_phase*: reorganiza os neurônios existentes sem expandir o mapa.



(a) *get_winner* (b) *growing_phase* (c) *smoothing_phase*

Figura 24 – Mapas das três estratégias do GSOM no Experimento 2.

O desempenho foi avaliado por meio de métricas por classe (*Precision*, *Recall*, *F1-Score*), de suas médias gerais (*Macro* e *Weighted*) e por meio da análise das matrizes de confusão.

8.2.1 Estratégia *get_winner*

Na Tabela 20 é apresentada a matriz de confusão da estratégia *get_winner* na segunda coleta. Com o aumento da classe *Drop* para 18,41%, o modelo acertou 2.154 instâncias, diferente do Experimento 1, em que não acertou nenhuma. Ainda assim, a maior parte dessas instâncias foi classificada incorretamente como *Deny*.

Tabela 20 – Matriz de confusão do GSOM na segunda coleta (estratégia *get_winner*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	545545	11254	0	0
Deny	9882	286458	0	2
Reset-Both	652	0	0	0
Drop	1586	191038	0	2154

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	97,98%	2,02%	0,00%	0,00%
Deny	3,33%	96,67%	0,00%	0,00%
Reset-Both	100,00%	0,00%	0,00%	0,00%
Drop	0,81%	98,09%	0,00%	1,11%

Fonte: Elaborado pelo autor.

Na Tabela 21 são apresentadas as métricas por classe, nas quais a classe *Drop* apresentou *Precision* alta de 99,91%, mas *Recall* de apenas 1,11%, resultando em *F1-Score* de 2,19%. Isso indica que, quando o modelo classifica uma instância como *Drop*, a classificação é quase sempre correta, mas ele detecta muito poucos casos reais dessa classe. Já a classe *Deny*, cuja proporção foi reduzida, apresentou *Recall* de 96,67% e *Precision* baixa de 58,60%.

Tabela 21 – Desempenho do GSOM por classe na segunda coleta (estratégia *get_winner*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	97,83%	97,98%	97,90%	556.799
Deny	58,60%	96,67%	72,95%	296.342
Reset-Both	0,00%	0,00%	0,00%	652
Drop	99,91%	1,11%	2,19%	194.778

Fonte: Elaborado pelo autor.

As médias da Tabela 22 indicam que o desempenho geral piorou. O *F1-Score Weighted* caiu para 72,62% contra 86,31% no Experimento 1. O *F1-Score Macro* de 43,26% indica que, considerando todas as classes igualmente, o desempenho médio permanece baixo.

O *get_winner* no Experimento 2 apresentou desempenho inferior ao do Experimento 1. O *F1-Score Weighted* caiu para 72,62%. A detecção de *Drop* melhorou em relação

Tabela 22 – Médias de desempenho do GSOM na segunda coleta (estratégia *get_winner*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	64,09%	48,94%	43,26%
Weighted	70,35%	79,38%	72,62%

Fonte: Elaborado pelo autor.

ao Experimento 1, passando de 0% para 1,11% de *Recall*, mas ainda ficou muito baixa, mesmo com mais instâncias dessa classe.

8.2.2 Estratégia *growing_phase*

Na Tabela 23 é apresentada a matriz de confusão do GSOM para a estratégia *growing_phase* na segunda coleta. Observa-se que a classe *Allow* não teve falsos negativos para as classes *Deny* e *Reset-Both*, mas apresentou 1.542 instâncias classificadas incorretamente como *Drop*. Para a classe *Drop*, que passou de 3,64% para 18,41% conforme a Tabela 4, o modelo obteve 38.489 acertos. Este resultado representa uma melhora considerável em relação aos 2.154 acertos da estratégia *get_winner*.

Tabela 23 – Matriz de confusão do GSOM na segunda coleta (estratégia *growing_phase*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	555257	0	0	1542
Deny	6855	256208	0	33279
Reset-Both	652	0	0	0
Drop	100	156189	0	38489

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	99,72%	0,00%	0,00%	0,28%
Deny	2,31%	86,45%	0,00%	11,23%
Reset-Both	100,00%	0,00%	0,00%	0,00%
Drop	0,05%	80,19%	0,00%	19,77%

Fonte: Elaborado pelo autor.

Na Tabela 24 é apresentado o desempenho por classe da estratégia *growing_phase* na segunda coleta. A classe *Allow* apresentou bom desempenho, com *F1-Score* de 99,17%. A classe *Deny* também apresentou bom desempenho, com *F1-Score* de 82,49%. A classe *Drop* apresentou alguma melhora em relação à estratégia anterior, mas manteve um *F1-Score* baixo de 14,50%, indicando que sua detecção ainda é limitada. Já a classe *Reset-Both* continuou sem ser detectada, o que evidencia a dificuldade do modelo em detectar classes minoritárias.

Tabela 24 – Desempenho do GSOM por classe na segunda coleta (estratégia *growing_phase*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	98,62%	99,72%	99,17%	556.799
Deny	78,84%	86,45%	82,49%	296.342
Reset-Both	0,00%	0,00%	0,00%	652
Drop	11,46%	19,77%	14,50%	194.778

Fonte: Elaborado pelo autor.

As médias da Tabela 25 mostram que o *growing_phase* teve *F1-Score Weighted* de 79,46%, o que é superior às outras estratégias. Mas o *F1-Score Macro* de 49,04% indica que as classes minoritárias continuaram apresentando problemas.

Tabela 25 – Médias de desempenho do GSOM na segunda coleta (estratégia *growing_phase*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	47,23%	51,49%	49,04%
Weighted	73,56%	85,24%	79,46%

Fonte: Elaborado pelo autor.

O *growing_phase* mostrou maior adaptação do que o *get_winner* para a classe *Drop*, com *F1-Score* de 14,50% em comparação a 2,19%. A expansão do mapa trouxe melhora, mas não resolveu completamente: *Reset-Both* continuou zerado e *Drop* teve *Recall* de 19,77%.

8.2.3 Estratégia *smoothing_phase*

Na Tabela 26 é apresentada a matriz de confusão da estratégia *smoothing_phase* na segunda coleta. Esta estratégia mostrou que a classe *Allow* apresentou desempenho elevado, com 99,99% de acertos, enquanto *Drop* obteve 31.959 acertos, correspondentes a 16,41% de *Recall*. Esse resultado é melhor que o obtido pelo *get_winner*, com 2.154 acertos, mas pior que o do *growing_phase*, com 38.489 acertos.

Na Tabela 27 são apresentadas as métricas: *Allow* teve *Recall* de 99,99% e *Precision* de 95,44%, com *F1-Score* de 97,66%. A classe *Drop* teve *Recall* de 16,41% e *Precision* de 51,76%, resultando em *F1-Score* de 24,94%. A classe *Deny* teve *Recall* de 84,80% e *Precision* de 62,31%.

Na Tabela 28 são apresentadas as médias de desempenho, nas quais a estratégia *smoothing_phase* alcançou *F1-Score Weighted* de 76,81%, resultado inferior ao do *growing_phase* (79,46%) e superior ao do *get_winner* (72,62%) neste experimento. O *F1-Score Macro* de 48,62% reflete a baixa detecção de *Reset-Both* e o desempenho limitado de *Drop*.

Tabela 26 – Matriz de confusão do GSOM na segunda coleta (estratégia *smoothing_phase*) em valores absolutos e porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	556735	0	0	64
Deny	15308	251313	0	29721
Reset-Both	652	0	0	0
Drop	10677	152142	0	31959

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	99,99%	0,00%	0,00%	0,01%
Deny	5,17%	84,81%	0,00%	10,03%
Reset-Both	100,00%	0,00%	0,00%	0,00%
Drop	5,48%	78,11%	0,00%	16,41%

Fonte: Elaborado pelo autor.

Tabela 27 – Desempenho do GSOM por classe na segunda coleta (estratégia *smoothing_phase*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	95,44%	99,99%	97,66%	556.799
Deny	62,31%	84,80%	71,87%	296.342
Reset-Both	0,00%	0,00%	0,00%	652
Drop	51,76%	16,41%	24,94%	194.778

Fonte: Elaborado pelo autor.

Tabela 28 – Médias de desempenho do GSOM na segunda coleta (estratégia *smoothing_phase*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	52,38%	50,30%	48,62%
Weighted	77,90%	80,11%	76,81%

Fonte: Elaborado pelo autor.

A estratégia *smoothing_phase* melhorou significativamente a detecção de *Drop* em comparação com *get_winner*, alcançando *F1-Score* de 24,94% contra 2,19% no Experimento 2. No entanto, manteve *Recall* elevado para *Allow* (99,99%), superando o desempenho limitado da estratégia *get_winner* com 60,76%.

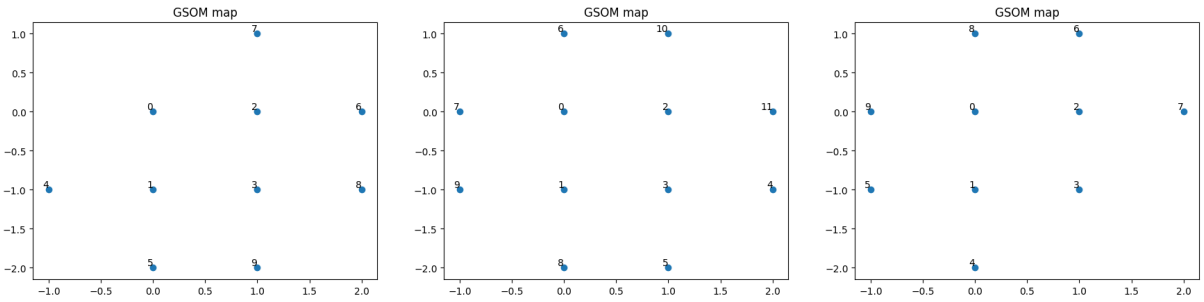
8.3 Experimento 3: Outubro de 2025

O Experimento 3 avaliou o GSOM com dados de Outubro de 2025. Na Tabela 4 é apresentada uma distribuição mais equilibrada entre as classes, onde *Allow* representou 41,32% dos dados, *Drop* 26,29%, *Deny* 32,27% e *Reset-Both* 0,12%. Pela primeira vez, nenhuma classe ultrapassou 50%, colocando o GSOM em um cenário

mais balanceado.

Para este experimento, usou-se SF igual a 0,6, com 25 épocas de crescimento e 15 de suavização na fase *offline*. Na fase *online*, os fluxos foram apresentados sequencialmente ao modelo, simulando um FCDs. Na Figura 25 é apresentada a diferença entre as três estratégias do GSOM por meio dos mapas gerados por cada uma delas. A classificação foi realizada com base nessas estratégias, que diferem quanto à forma de atualizar o mapa.

- *get_winner*: mantém a estrutura aprendida na fase *offline*;
- *growing_phase*: expande o mapa adicionando novos neurônios;
- *smoothing_phase*: reorganiza os neurônios existentes sem expandir o mapa.



(a) *get_winner* (b) *growing_phase* (c) *smoothing_phase*

Figura 25 – Mapas das três estratégias do GSOM no Experimento 3.

O desempenho foi avaliado por meio de métricas por classe (*Precision*, *Recall*, *F1-Score*), de suas médias gerais (*Macro* e *Weighted*) e da análise das matrizes de confusão.

8.3.1 Estratégia *get_winner*

Na Tabela 29 é apresentada a matriz do *get_winner* na terceira coleta. Com a distribuição mais equilibrada, em que *Drop* se tornou a classe majoritária (26,29%), todas as três classes principais apresentaram *Recall* acima de 60%. *Drop* com 85,93%, *Deny* com 65,06% e *Allow* com 60,76%.

A Tabela 30 mostra que *Allow* teve *F1-Score* de 71,89%, *Deny* 68,13% e *Drop* 62,06%. Cada classe apresentou um perfil diferente. *Allow* apresentou boa *Precision*, enquanto *Drop* apresentou bom *Recall*.

Na Tabela 31 o *F1-Score Weighted* ficou em 68,15%, abaixo dos experimentos anteriores. No entanto, a diferença entre as médias *Macro* (50,52%) e *Weighted* (68,15%) é menor do que antes. Isso reflete o maior equilíbrio entre as classes, em que a distribuição das classes faz com que a média *Weighted* fique mais próxima da média *Macro*.

Tabela 29 – Matriz de confusão do GSOM na terceira coleta (estratégia *get_winner*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	232643	122198	0	28041
Deny	23117	364639	0	172723
Reset-Both	636	247	0	1
Drop	7790	23290	0	189872

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	60,76%	31,92%	0,00%	7,32%
Deny	4,12%	65,06%	0,00%	30,82%
Reset-Both	71,95%	27,94%	0,00%	0,11%
Drop	3,53%	10,54%	0,00%	85,93%

Fonte: Elaborado pelo autor.

Tabela 30 – Desempenho do GSOM por classe na terceira coleta (estratégia *get_winner*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	88,06%	60,76%	71,89%	382.882
Deny	71,45%	65,06%	68,13%	560.479
Reset-Both	0,00%	0,00%	0,00%	884
Drop	48,58%	85,93%	62,06%	220.952

Fonte: Elaborado pelo autor.

Tabela 31 – Médias de desempenho do GSOM na terceira coleta (estratégia *get_winner*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	52,02%	52,94%	50,52%
Weighted	72,52%	67,55%	68,15%

Fonte: Elaborado pelo autor.

A estratégia *get_winner* melhorou significativamente a detecção de *Drop*, com *F1-Score* de 62,06% no Experimento 3, contra 2,19% no Experimento 2. No entanto, manteve *Recall* baixo para *Allow*, de 60,76%. O desempenho geral ficou em 68,15% de *F1-Score Weighted*, indicando que, em cenários balanceados, o GSOM apresenta melhor desempenho.

8.3.2 Estratégia *growing_phase*

Na Tabela 32 é apresentada a matriz de confusão para a estratégia *growing_phase* na terceira coleta. A classe *Allow* teve 99,91% de acertos, mas a classe *Drop*, que representa 26,29% da coleta, teve apenas 13 acertos, correspondentes a 0,01% de *Recall*.

A Tabela 33 mostra um desempenho muito desigual entre as classes. *Allow* teve

Tabela 32 – Matriz de confusão do GSOM na terceira coleta (estratégia *growing_phase*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	382548	334	0	0
Deny	123461	437010	0	8
Reset-Both	884	0	0	0
Drop	115416	105523	0	13

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	99,91%	0,09%	0,00%	0,00%
Deny	22,03%	77,97%	0,00%	0,00%
Reset-Both	100,00%	0,00%	0,00%	0,00%
Drop	52,24%	47,76%	0,00%	0,01%

Fonte: Elaborado pelo autor.

Recall de 99,91% e *Precision* de 61,47%, enquanto *Drop* teve *Recall* de apenas 0,01%. A classe *Deny* apresentou desempenho mais estável, com *F1-Score* de 79,22%.

Tabela 33 – Desempenho do GSOM por classe na terceira coleta (estratégia *growing_phase*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	61,47%	99,91%	76,12%	382.882
Deny	80,50%	77,97%	79,22%	560.479
Reset-Both	0,00%	0,00%	0,00%	884
Drop	61,90%	0,01%	0,01%	220.952

Fonte: Elaborado pelo autor.

As médias da Tabela 34 mostram que o *growing_phase* no Experimento 3 teve *F1-Score Weighted* de 63,12%, muito abaixo dos 79,46% do Experimento 2. A média *Macro* de 38,84% indica que, mesmo em um cenário balanceado, as classes não apresentaram desempenho equilibrado.

Tabela 34 – Médias de desempenho do GSOM na terceira coleta (estratégia *growing_phase*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	50,97%	44,47%	38,84%
Weighted	70,66%	70,34%	63,12%

Fonte: Elaborado pelo autor.

O *growing_phase* teve resultados variados. No Experimento 1, não detectou *Drop*; no Experimento 2, adaptou-se bem, com *F1-Score* de 14,50%, mas no Experimento 3 voltou a falhar, com *F1-Score* de 0,01%. A estratégia mostrou dificuldade em se ajustar às mudanças na distribuição.

8.3.3 Estratégia *smoothing_phase*

A classe *Drop* obteve 10.810 acertos, equivalentes a 4,89% de *Recall*, um resultado melhor que o do *growing_phase*, com 13 acertos, mas pior que o do *get_winner*, com 189.872 acertos.

Na Tabela 35 é apresentada a matriz de confusão da estratégia *smoothing_phase* na terceira coleta. Para a classe *Drop*, foram obtidos 10.810 acertos, correspondentes a um *Recall* de 4,89%, indicando um desempenho melhor do que o observado no *growing_phase*, que apresentou apenas 13 acertos. Ainda assim, o desempenho permanece inferior ao observado na estratégia *get_winner*, que alcançou 189.872 acertos nessa classe.

Tabela 35 – Matriz de confusão do GSOM na terceira coleta (estratégia *smoothing_phase*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	341423	41459	0	0
Deny	78325	474006	0	8148
Reset-Both	884	0	0	0
Drop	32305	177837	0	10810

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	89,17%	10,83%	0,00%	0,00%
Deny	13,97%	84,57%	0,00%	1,45%
Reset-Both	100,00%	0,00%	0,00%	0,00%
Drop	14,62%	80,49%	0,00%	4,89%

Fonte: Elaborado pelo autor.

Na Tabela 36 são apresentadas as métricas por classe. A técnica de *smoothing_phase* alcançou *F1-Score* de 81,68% para *Allow*, 75,60% para *Deny*, mas apenas 9,01% para *Drop*.

Tabela 36 – Desempenho do GSOM por classe na terceira coleta (estratégia *smoothing_phase*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	75,38%	89,17%	81,68%	382.882
Deny	68,37%	84,57%	75,60%	560.479
Reset-Both	0,00%	0,00%	0,00%	884
Drop	57,03%	4,89%	9,01%	220.952

Fonte: Elaborado pelo autor.

As médias da Tabela 37 apresentam *F1-Score Weighted* de 64,90% e *Macro* de 41,57%. Esses valores são semelhantes aos do *growing_phase* e indicam desempenho intermediário.

Tabela 37 – Médias de desempenho do GSOM na terceira coleta (estratégia *smoothing_phase*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	50,20%	44,66%	41,57%
Weighted	68,47%	70,91%	64,90%

Fonte: Elaborado pelo autor.

O *smoothing_phase* não se adaptou bem ao cenário, apresentando *F1-Score* de apenas 9,01% para a classe *Drop*, mesmo esta sendo majoritária. Entre as três estratégias, foi a que apresentou menor capacidade de generalização para a classe *Drop*, evidenciando limitações em cenários balanceados.

8.4 Experimento 4: Dados Combinados

O Experimento 4 avaliou o GSOM em condições de mudança temporal, combinando dados das três coletas. A Tabela 38 mostra a divisão dos dados, onde a fase *offline* utiliza 500.000 instâncias da primeira coleta e a fase *online* utiliza 1.000.000 instâncias das coletas 2 e 3 combinadas.

Tabela 38 – Divisão dos dados no Experimento 4: treinamento e teste com coletas temporais distintas

Fase	**Coleta**	***Allow***	***Deny***	***Drop***	***Reset-Both***	**Total**
Offline	1ª Coleta	348.798 69,76%	132.234 26,45%	18.187 3,64%	781 0,16%	500.000
Online	2ª + 3ª Coleta	528.706 52,87%	246.691 24,67%	223.483 22,35%	1.120 0,11%	1.000.000

Fonte: Elaborado pelo autor.

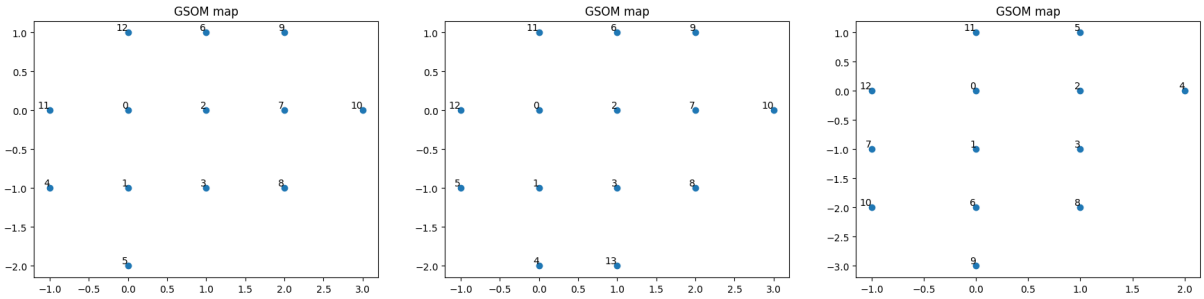
Esta configuração reproduz desafios reais de classificação de tráfego em ambientes dinâmicos. O modelo é treinado com uma distribuição histórica de regras de *firewall* e, depois, exposto a dados de períodos subsequentes, em que a aplicação das regras pode ter evoluído devido a atualizações de políticas, mudanças na infraestrutura ou ajustes operacionais.

Para este experimento, foi utilizado SF igual a 0,7, com 12 épocas de crescimento e 7 épocas de suavização na fase *offline*. Na fase *online*, os fluxos foram apresentados sequencialmente ao modelo, simulando um FCDs. Na Figura 26 é apresentada a diferença entre as três estratégias do GSOM por meio dos mapas gerados por cada uma delas. A classificação foi realizada com base nessas estratégias, que diferem quanto à forma de atualizar o mapa.

– *get_winner*: mantém a estrutura aprendida na fase *offline*;

– *growing_phase*: expande o mapa adicionando novos neurônios;

– *smoothing_phase*: reorganiza os neurônios existentes sem expandir o mapa.



(a) *get_winner* (b) *growing_phase* (c) *smoothing_phase*

Figura 26 – Mapas das três estratégias do GSOM no Experimento 4.

O desempenho foi avaliado por meio de métricas por classe (*Precision*, *Recall*, *F1-Score*), de suas médias gerais (*Macro* e *Weighted*) e da análise das matrizes de confusão.

8.4.1 Estratégia *get_winner*

Na Tabela 39 é apresentada a matriz de confusão do GSOM para a estratégia *get_winner* no Experimento 4. Observa-se que as classes *Allow* e *Deny* obtiveram a maior parte dos acertos, com 496.133 e 232.922 classificações corretas, respectivamente. Em contraste, a classe *Drop*, que representa 22,35% dos dados na fase *online*, conforme a Tabela 38, obteve apenas 3.756 acertos. A maior parte de suas instâncias, 206.470, foi classificada incorretamente como *Deny*.

Tabela 39 – Matriz de confusão do GSOM no Experimento 4 (estratégia *get_winner*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	496.133	32.573	0	0
Deny	12.624	232.922	0	1.145
Reset-Both	1.091	29	0	0
Drop	13.257	206.470	0	3.756

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	93,84%	6,16%	0,00%	0,00%
Deny	5,12%	94,42%	0,00%	0,46%
Reset-Both	97,41%	2,59%	0,00%	0,00%
Drop	5,93%	92,39%	0,00%	1,68%

Fonte: Elaborado pelo autor.

A Tabela 40 apresenta os resultados da estratégia *get_winner* no Experimento 4. As classes *Allow* e *Deny* apresentaram bom desempenho, com *Recall* de 93,84% e 94,42%, respectivamente. Já a classe *Drop*, que representa 22,35% dos dados na fase *online*, apresentou *Recall* de apenas 1,68%. Esse resultado indica que a estrutura aprendida na primeira coleta, quando *Drop* correspondia a apenas 3,64% dos dados, não se adaptou ao aumento dessa classe.

Tabela 40 – Desempenho do GSOM por classe no Experimento 4 (estratégia *get_winner*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	94,84%	93,84%	94,33%	528.706
Deny	49,35%	94,42%	64,83%	246.691
Reset-Both	0,00%	0,00%	0,00%	1.120
Drop	76,64%	1,68%	3,29%	223.483

Fonte: Elaborado pelo autor.

Na Tabela 41 são apresentadas as médias de desempenho. O *F1-Score Weighted* foi de 65,95%, indicando um desempenho geral moderado. No entanto, a grande diferença de 25,34% entre as médias *Macro* e *Weighted*, a maior entre todos os experimentos com essa estratégia, revela que o desempenho foi muito desigual entre as classes. Isso indica que o modelo teve dificuldade em se adaptar às mudanças no cenário combinado, mantendo-se excessivamente ligado à distribuição das classes aprendida na fase *offline*.

Tabela 41 – Médias de desempenho do GSOM no Experimento 4 (estratégia *get_winner*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	55,21%	47,49%	40,61%
Weighted	79,45%	72,94%	65,95%

Fonte: Elaborado pelo autor.

A estratégia *get_winner* no cenário combinado demonstrou limitações significativas na adaptação a mudanças temporais. Enquanto manteve desempenho elevado nas classes historicamente dominantes *Allow* e *Deny*, falhou completamente em captar o crescimento da classe *Drop*. Este resultado indica que o treinamento com apenas 3,64% de instâncias de *Drop* da primeira coleta não foi suficiente para que o modelo aprendesse a representação desta classe, o que prejudicou a classificação quando sua proporção aumentou para 22,35% nas coletas seguintes.

8.4.2 Estratégia *growing_phase*

Na Tabela 42 é apresentada a matriz de confusão do GSOM para a estratégia *growing_phase* no Experimento 4. Observa-se um padrão extremamente desbalanceado. A classe *Drop*, que representa 22,35% dos dados da fase *online*, obteve

apenas 11.472 acertos, com grande parte de suas instâncias, 204.241, classificadas incorretamente como *Deny*. As classes *Allow* e *Deny* mantiveram a maioria das classificações corretas.

Tabela 42 – Matriz de confusão do GSOM no Experimento 4 (estratégia *growing_phase*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	495.469	33.237	0	0
Deny	15.920	228.311	0	2.460
Reset-Both	1.054	66	0	0
Drop	7.770	204.241	0	11.472

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	93,71%	6,29%	0,00%	0,00%
Deny	6,45%	92,55%	0,00%	1,00%
Reset-Both	94,11%	5,89%	0,00%	0,00%
Drop	3,48%	91,39%	0,00%	5,13%

Fonte: Elaborado pelo autor.

Na Tabela 43 é apresentado o desempenho por classe. A classe *Allow* alcança *F1-Score* de 94,49% enquanto *Deny* apresenta *Recall* elevado de 92,55% mas *Precision* baixa de 49,01%. Diferentemente da classe *Drop*, que tem apenas 5,13% de *Recall* e 9,66% de *F1-Score*.

Tabela 43 – Desempenho do GSOM por classe no Experimento 4 (estratégia *growing_phase*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	95,24%	93,71%	94,49%	528.706
Deny	49,01%	92,55%	64,08%	246.691
Reset-Both	0,00%	0,00%	0,00%	1.120
Drop	82,35%	5,13%	9,66%	223.483

Fonte: Elaborado pelo autor.

As médias apresentadas na Tabela 44 indicam um *F1-Score Weighted* de 65,98%. Este valor é próximo ao *F1-Score* da estratégia *get_winner*, que foi de 65,95%, mas inferior ao desempenho do Experimento 2, que atingiu 79,46%. A diferença entre as médias *Macro* e *Weighted* é de 23,92%.

A estratégia *growing_phase* apresentou desempenho próximo ao *get_winner get_winner* nas classes historicamente dominantes. No entanto, a classe *Drop* obteve uma melhora significativa em relação à estratégia anterior, com o *F1-Score* aumentando de 3,29% para 9,66%.

Tabela 44 – Médias de desempenho do GSOM no Experimento 4 (estratégia *growing_phase*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	56,65%	47,85%	42,06%
Weighted	80,85%	72,49%	65,98%

Fonte: Elaborado pelo autor.

8.4.3 Estratégia *smoothing_phase*

Na Tabela 45 é apresentada a matriz de confusão do GSOM para a estratégia *smoothing_phase* no Experimento 4. O padrão observado é semelhante ao da estratégia *growing_phase*. A classe *Drop*, que representa 22,35% dos dados da fase *online*, obteve apenas 3.756 acertos, tendo a maior parte de suas instâncias, 204.284, classificada incorretamente como *Deny*. As classes *Allow* e *Deny* mantiveram desempenho elevado, ambas com mais de 90% de acertos.

Tabela 45 – Matriz de confusão do GSOM no Experimento 4 (estratégia *smoothing_phase*) em Valores Absolutos e em Porcentagem.

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	495.574	33.132	0	0
Deny	17.188	228.358	0	1.145
Reset-Both	1.079	41	0	0
Drop	15.443	204.284	0	3.756

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	93,73%	6,27%	0,00%	0,00%
Deny	6,97%	92,57%	0,00%	0,46%
Reset-Both	96,34%	3,66%	0,00%	0,00%
Drop	6,91%	91,41%	0,00%	1,68%

Fonte: Elaborado pelo autor.

Na Tabela 46 é apresentado o desempenho por classe da estratégia *smoothing_phase*. A classe *Drop* teve *Recall* de 1,68% e *F1-Score* de 3,29%, valores idênticos aos obtidos pela estratégia *get_winner*. As classes *Allow* e *Deny* mantiveram desempenho semelhante à estratégia *growing_phase*, com *Deny* apresentando *Recall* elevado de 92,57% e *Precision* de 49,02%.

Na Tabela 47 são apresentadas as médias de desempenho, com *F1-Score Weighted* de 65,42%. Este valor é ligeiramente inferior ao da estratégia *growing_phase* com 65,98% e inferior ao da estratégia *get_winner* com 65,95%. A diferença entre as médias *Macro* e *Weighted* é de 25,15%.

A estratégia *smoothing_phase* demonstrou menor capacidade de adaptação às mudanças temporais combinadas. Com *F1-Score* de 3,29% para *Drop*, apresentou de-

Tabela 46 – Desempenho do GSOM por classe no Experimento 4 (estratégia *smoothing_phase*)

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	93,63%	93,73%	93,68%	528.706
Deny	49,02%	92,57%	64,10%	246.691
Reset-Both	0,00%	0,00%	0,00%	1.120
Drop	76,64%	1,68%	3,29%	223.483

Fonte: Elaborado pelo autor.

Tabela 47 – Médias de desempenho do GSOM no Experimento 4 (estratégia *smoothing_phase*)

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	54,82%	47,00%	40,27%
Weighted	78,72%	72,43%	65,42%

Fonte: Elaborado pelo autor.

sempenho idêntico ao da estratégia *get_winner* e inferior ao da estratégia *growing_phase*, que atingiu 9,66%. Este resultado indica que o ajuste de pesos sem expansão topológica não resultou em melhoria no reconhecimento da classe *Drop*.

8.5 Comparação entre os Experimentos

A análise dos quatro experimentos mostra que o desempenho do GSOM na classificação *online* depende principalmente da distribuição das instâncias por classe e da estratégia adotada. No Experimento 1, todas as estratégias apresentaram baixo desempenho nas classes minoritárias *Reset-Both* e *Drop*. Isso demonstra que o GSOM não conseguiu capturar padrões representativos dessas classes quando elas continham poucas instâncias.

No Experimento 2, o aumento da proporção da classe *Drop*, que passou de 3,64% para 18,41% melhorou o desempenho, especialmente na estratégia *growing_phase*. No entanto, o *Recall* desta classe permaneceu baixo.

No Experimento 3, com a maior proporção da classe *Drop* (26,29%), todas as estratégias apresentaram melhor desempenho. A estratégia *get_winner* alcançou *F1-Score* de 62,06% nesta classe. Este resultado indica que a quantidade de instâncias disponíveis para uma classe influencia diretamente o desempenho do GSOM.

No Experimento 4, a classe *Drop* representou 3,64% na fase *offline* e 22,35% na fase *online*. Na fase *online*, a estratégia *growing_phase* alcançou *F1-Score* de 9,66% para esta classe, valor superior ao de 3,29% das outras estratégias. Este resultado mostra que o aumento da proporção de uma classe na fase *online* não garante bom

desempenho quando sua representação na fase *offline* é baixa.

De forma geral, os resultados indicam que o desempenho do GSOM é sensível à distribuição das classes. As três estratégias testadas apresentaram desempenhos distintos, dependendo da proporção de cada classe, com melhores resultados quando a distribuição é mais equilibrada. Além disso, a análise temporal revela que mudanças nessa distribuição ao longo do tempo impactam significativamente a efetividade do modelo.

8.6 Comparação com Métodos Supervisionados

Para contextualizar o desempenho do algoritmo GSOM, que opera com latência infinita de rótulos, realizou-se uma análise comparativa com dois métodos incrementais de latência nula de rótulos. Esta comparação estabelece um *Upper Bound* (limite superior de desempenho), uma vez que os métodos comparados utilizam informações de rotulação completas e instantâneas, que não estão disponíveis ao GSOM. Os algoritmos supervisionados selecionados para este fim, ambos amplamente reconhecidos na literatura em FCDs, foram o SVM *Incremental* (CAUWENBERGHS; POGGIO, 2000) e o *Adaptive Random Forest* (ARF) (GOMES et al., 2017).

8.6.1 SVM *Incremental*

Experimentos adicionais foram realizados utilizando um algoritmo supervisionado de latência nula, no qual os rótulos são fornecidos imediatamente. Foi utilizado um algoritmo SVM *Incremental* para prever as ações do *firewall*. O algoritmo foi uma adaptação do `SGDClassifier` do *Scikit-learn* (DEVELOPERS, 2025), projetado para aprendizado incremental, com atualização contínua dos hiperparâmetros a cada instância, processada individualmente, na fase *online*.

Na implementação, a maioria dos hiperparâmetros manteve os valores padrão da biblioteca. A função de perda foi ajustada para `loss="hinge"`, o que configura o modelo como uma máquina de vetores de suporte linear com capacidade de atualização incremental. Essa abordagem permite que o modelo se adapte continuamente a novos dados, sem necessidade de retreinamento completo ou de reprocessamento do conjunto de dados, característica essencial para a operação em FCDs e para a classificação de logs de *firewall* em tempo real.

Assim como nos experimentos com o GSOM, a avaliação do SVM *Incremental* seguiu a mesma estrutura de três experimentos, comparando as predições aos rótulos reais fornecidos ao longo da operação.

8.6.1.1 Experimento 1: agosto de 2024

Na Tabela 48 é apresentada a matriz de confusão do SVM *Incremental* para a primeira coleta. Em comparação com as estratégias do GSOM, o método supervisionado conseguiu detectar as classes minoritárias, embora com desempenho ainda limitado.

Tabela 48 – Matriz de confusão do SVM *Incremental* na primeira coleta em Valores Absolutos e Porcentagem

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	625499	69464	0	1199
Deny	6409	305048	0	31
Reset-Both	1532	14	0	0
Drop	3280	36043	0	56

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	89,85%	9,98%	0,00%	0,17%
Deny	2,06%	97,93%	0,00%	0,01%
Reset-Both	99,09%	0,91%	0,00%	0,00%
Drop	8,33%	91,53%	0,00%	0,14%

Fonte: Elaborado pelo autor.

Na Tabela 49 são apresentadas as métricas por classe, o que mostra um padrão complementar entre as classes majoritárias. A classe *Allow* teve *Precision* alta de 98,24% com *Recall* de 89,85%, enquanto a classe *Deny* apresentou *Recall* de 97,93% com *Precision* moderada de 74,29%. Para as classes minoritárias, a classe *Reset-Both* teve todas as métricas zeradas, enquanto *Drop* teve um *Recall* de 0,14%.

Tabela 49 – Desempenho do SVM *Incremental* por classe na primeira coleta

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	98,24%	89,85%	93,84%	696.162
Deny	74,29%	97,93%	84,46%	311.488
Reset-Both	0,00%	0,00%	0,00%	1.546
Drop	4,35%	0,14%	0,27%	39.379

Fonte: Elaborado pelo autor.

Na Tabela 50 são apresentadas as médias gerais de desempenho. A diferença entre as médias *Macro* e *Weighted* reflete o desequilíbrio extremo do conjunto. Enquanto a média *Weighted* é elevada, com *F1-Score* de 87,46%, a média *Macro* cai para 44,64%, indicando o colapso nas classes minoritárias.

A disponibilidade imediata de rótulos permitiu ao SVM *Incremental* um pequeno progresso na detecção da classe *Drop*, com 56 instâncias, equivalente a 0,14% de *Recall*, o que se mantém ausente em todas as estratégias do GSOM. No entanto, o

Tabela 50 – Médias de desempenho do SVM *Incremental* na primeira coleta

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	44,22%	46,98%	44,64%
Weighted	87,46%	88,76%	87,46%

Fonte: Elaborado pelo autor.

método supervisionado não resolveu o problema fundamental do desbalanceamento, com as classes minoritárias permanecendo praticamente não detectadas.

8.6.1.2 Experimento 2: fevereiro de 2025

Na Tabela 51 é apresentada a matriz de confusão do SVM *Incremental* para a segunda coleta. Neste período, a classe *Drop* aumentou significativamente para 18,41% do total de instâncias, como evidenciado na Tabela 4, criando um cenário ideal para testar a capacidade adaptativa do método supervisionado diante de mudanças na distribuição.

Tabela 51 – Matriz de confusão do SVM *Incremental* na segunda coleta em Valores Absolutos e Porcentagem

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	553720	3078	0	1
Deny	11085	285158	0	99
Reset-Both	652	0	0	0
Drop	19582	175145	0	51

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	99,45%	0,55%	0,00%	0,00%
Deny	3,74%	96,23%	0,00%	0,03%
Reset-Both	100,00%	0,00%	0,00%	0,00%
Drop	10,05%	89,91%	0,00%	0,03%

Fonte: Elaborado pelo autor.

Na Tabela 52 são apresentadas as métricas de desempenho por classe. A classe *Allow* apresentou desempenho elevado, com *Precision* de 94,65%, *Recall* de 99,45% e *F1-Score* de 96,99%, indicando estabilidade em relação ao experimento anterior. A classe *Deny* manteve *Recall* elevado de 96,23%, porém com redução expressiva de *Precision* para 61,54%, o que reflete um aumento nas falsas predições dessa classe, especialmente oriundas da classe *Drop*. A classe *Drop* apresentou *Precision* de 33,77% e *Recall* muito baixo de 0,03%, resultando em um *F1-Score* de apenas 0,05%. Para a classe *Reset-Both*, todas as métricas foram de 0,00%.

Na Tabela 53 são apresentadas as médias gerais de desempenho. A média *Weighted* alcançou *F1-Score* de 72,73%, o que reflete bom desempenho nas classes majori-

Tabela 52 – Desempenho do SVM *Incremental* por classe na segunda coleta

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	94,65%	99,45%	96,99%	556.799
Deny	61,54%	96,23%	75,07%	296.342
Reset-Both	0,00%	0,00%	0,00%	652
Drop	33,77%	0,03%	0,05%	194.778

Fonte: Elaborado pelo autor.

tárias. Em contrapartida, a média *Macro* permaneceu significativamente inferior, com *F1-Score* de 43,03%, o que evidencia a incapacidade do modelo de tratar adequadamente classes de menor representatividade. A diferença de aproximadamente 29,7% entre as duas médias indica uma forte dependência do desempenho das classes dominantes.

Tabela 53 – Médias de desempenho do SVM *Incremental* na segunda coleta

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	47,49%	48,92%	43,03%
Weighted	73,92%	80,01%	72,73%

Fonte: Elaborado pelo autor.

Quando comparado às estratégias baseadas em GSOM na mesma coleta, o SVM *Incremental* apresenta um comportamento oposto. Embora apresente desempenho superior nas métricas globais ponderadas, não demonstra capacidade de adaptação diante do crescimento da classe *Drop*. Enquanto o SVM *Incremental* identificou apenas 51 instâncias dessa classe com *Recall* de 0,03%, a estratégia GSOM *growing_phase* detectou 38.489 instâncias, alcançando *Recall* de 19,77% e *F1-Score* de 14,50%.

8.6.1.3 Experimento 3: Outubro de 2025

Na Tabela 54 é apresentada a matriz de confusão do SVM *Incremental* para a terceira coleta. Na Tabela 4 observa-se que este período apresenta a distribuição mais equilibrada entre as classes, com *Allow* reduzindo para 41,32%, *Deny* aumentando para 32,27% e *Drop* atingindo 26,29%. Este cenário testa definitivamente a capacidade adaptativa do método supervisionado diante de mudanças na distribuição dos dados.

Na Tabela 55 são apresentadas as métricas por classe, mostrando um avanço significativo para a classe *Reset-Both*, que alcançou pela primeira vez um *Recall* de 50,79% e *F1-Score* de 17,83%. A classe *Drop* também apresentou progresso, com *Recall* de 10,39%, correspondente a 22.963 instâncias corretas, e *F1-Score* de 16,05%. As clas-

Tabela 54 – Matriz de confusão do SVM *Incremental* na terceira coleta em Valores Absolutos e Porcentagem

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	282217	96060	3708	897
Deny	22739	495159	718	41863
Reset-Both	435	0	449	0
Drop	7377	190612	0	22963

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	73,72%	25,09%	0,97%	0,23%
Deny	4,06%	88,34%	0,13%	7,47%
Reset-Both	49,21%	0,00%	50,79%	0,00%
Drop	3,34%	86,27%	0,00%	10,39%

Fonte: Elaborado pelo autor.

ses majoritárias mantiveram desempenho sólido, com *Allow* apresentando *F1-Score* de 80,96% e *Deny* de 77,03%.

Tabela 55 – Desempenho do SVM *Incremental* por classe na terceira coleta

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	90,24%	73,72%	81,13%	382.882
Deny	63,34%	88,34%	73,91%	560.479
Reset-Both	9,21%	50,79%	15,59%	884
Drop	34,93%	10,39%	16,02%	220.952

Fonte: Elaborado pelo autor.

Na Tabela 56 são apresentadas as médias gerais de desempenho. A diferença entre as médias *Macro*, de 47,97%, e *Weighted*, de 69,03%, reduziu para 21,06%, o que reflete um maior equilíbrio na distribuição das classes em comparação aos experimentos anteriores. Esta redução indica uma distribuição mais uniforme do desempenho entre todas as classes.

Tabela 56 – Médias de desempenho do SVM *Incremental* na terceira coleta

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	49,43%	55,81%	46,66%
Weighted	66,74%	69,34%	69,02%

Fonte: Elaborado pelo autor.

Comparado às estratégias do GSOM no mesmo período, o SVM *Incremental* demonstrou capacidades distintas. Embora tenha obtido médias gerais inferiores à da estratégia *get_winner* do GSOM, com *F1-Score Weighted* de 69,03% contra 68,15%, foi o único método a detectar significativamente a classe *Reset-Both*, com 50,79%

de *Recall*. Este resultado contrasta com todas as estratégias do GSOM, que mantiveram esta classe com 0% de detecção.

8.6.1.4 Experimento 4: Dados Combinados

Na Tabela 57 é apresentada a matriz de confusão do SVM *Incremental* no cenário combinado, com a fase *offline* utilizando dados históricos de agosto de 2024 e a fase *online* processando dados posteriores de fevereiro e outubro de 2025. Esta configuração simula desafios reais de mudanças de conceito, com alterações significativas na distribuição, conforme documentado na Tabela 38, em que *Allow* reduz de 69,76% para 52,87% e *Drop* aumenta de 3,64% para 22,35%.

Tabela 57 – Matriz de confusão do SVM *Incremental* no Experimento 4 em Valores Absolutos e Porcentagem

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	518605	0	1	10100
Deny	6875	99078	0	140738
Reset-Both	1120	0	0	0
Drop	206	64557	0	158720

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	98,09%	0,00%	0,00%	1,91%
Deny	2,79%	40,16%	0,00%	57,05%
Reset-Both	100,00%	0,00%	0,00%	0,00%
Drop	0,09%	28,89%	0,00%	71,02%

Fonte: Elaborado pelo autor.

Na Tabela 58 são apresentadas as métricas por classe, o que revela um comportamento adaptativo assimétrico. A classe *Allow* apresentou bom desempenho, com *F1-Score* de 98,27% e *Recall* de 98,09%. Já a classe *Deny* apresentou *Recall* de 40,16% e *F1-Score* de 48,35%. A classe *Drop*, que cresceu significativamente, alcançou *Recall* de 71,02% e *F1-Score* de 59,52%. Na classe *Reset-Both* não houve detecção.

Tabela 58 – Desempenho do SVM *Incremental* por classe no Experimento 4

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	98,44%	98,09%	98,27%	528.706
Deny	60,55%	40,16%	48,35%	246.691
Reset-Both	0,00%	0,00%	0,00%	1.120
Drop	51,27%	71,02%	59,52%	223.483

Fonte: Elaborado pelo autor.

Na Tabela 59 são apresentadas as médias gerais de desempenho. O *F1-Score Weighted* de 77,53% supera todas as estratégias do GSOM neste experimento. A diferença

entre as médias *Macro* (52,13%) e *Weighted* (77,53%) é de 25,40%, refletindo o desequilíbrio no desempenho entre as classes.

Tabela 59 – Médias de desempenho do SVM *Incremental* no Experimento 4

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	52,57%	52,32%	51,53%
Weighted	77,24%	77,55%	77,32%

Fonte: Elaborado pelo autor.

Comparado às estratégias do GSOM no mesmo cenário combinado, o SVM *Incremental* demonstrou capacidades de adaptação distintas. Embora tenha superado significativamente o GSOM na detecção da classe *Drop*, com *F1-Score* de 61,91% contra, no máximo, 9,66% do GSOM *growing_phase*, apresentou desempenho inferior na classe *Deny*, com 48,35% contra aproximadamente 64% das estratégias GSOM.

8.6.2 Adaptive Random Forest

O *Adaptive Random Forest* (ARF) é um método de ensemble desenvolvido para FCDs. Diferentemente do Random Forest tradicional, o ARF atualiza e substitui árvores continuamente, mantendo a diversidade do modelo sem necessidade de retreinamento completo. Essa característica o torna adequado para cenários com dados não estacionários.

A implementação utilizada foi a disponibilizada no framework CapyMOA. Nesta pesquisa, o ARF foi configurado e avaliado nos mesmos quatro experimentos definidos para o GSOM e para o SVM *Incremental*, utilizando o mesmo conjunto de atributos. Isso permitiu uma comparação direta de desempenho entre os métodos.

Entretanto, diferentemente do GSOM e do SVM *Incremental*, o ARF não foi avaliado sob uma separação entre fases *offline* e *online*. Seu funcionamento é incremental, com atualização contínua do modelo ao longo do fluxo de dados. Assim, neste trabalho, o ARF foi tratado como um método que opera integralmente em regime *online*, utilizando os rótulos disponíveis durante o processamento para ajustar suas árvores de forma adaptativa.

Um aspecto relevante dessa abordagem é o custo computacional. Devido à sua complexidade, o ARF demandou tempo de processamento significativamente superior ao dos demais métodos, levando aproximadamente quatro dias para a conclusão de cada experimento.

8.6.2.1 Experimento 1: agosto de 2024

Na Tabela 60 é apresentada a matriz de confusão do ARF para a primeira coleta. Em comparação aos métodos anteriores, o ARF demonstrou maior capacidade de detecção das classes minoritárias (*Reset-Both* e *Drop*).

Tabela 60 – Matriz de confusão do ARF na primeira coleta em Valores Absolutos e Porcentagem

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	695063	1067	19	13
Deny	2150	303361	1	5976
Reset-Both	194	2	1350	0
Drop	87	22963	0	16329

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	99,84%	0,15%	0,00%	0,00%
Deny	0,69%	97,39%	0,00%	1,92%
Reset-Both	12,55%	0,13%	87,32%	0,00%
Drop	0,22%	58,29%	0,00%	41,47%

Fonte: Elaborado pelo autor.

Na Tabela 61 são apresentadas as métricas por classe, evidenciando avanços marcantes. A classe *Reset-Both*, não detectada no GSOM e no SVM *Incremental*, alcançou *Recall* de 87,32% e *F1-Score* de 92,59% no ARF. A classe *Drop* também apresentou desempenho significativo, com *Recall* de 41,47% e *F1-Score* de 52,87%. As classes majoritárias mantiveram desempenho elevado, com *Allow* apresentando *F1-Score* de 99,75% e *Deny* de 95,07%.

Tabela 61 – Desempenho do ARF por classe na primeira coleta

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	99,65%	99,84%	99,75%	696.162
Deny	92,84%	97,39%	95,07%	311.488
Reset-Both	98,54%	87,32%	92,59%	1.546
Drop	72,87%	41,47%	52,87%	39.379

Fonte: Elaborado pelo autor.

Na Tabela 62 são apresentadas as médias gerais de desempenho. O *F1-Score Weighted* de 96,56% é significativamente superior aos resultados do GSOM e do SVM *Incremental*. Mais importante ainda, a diferença entre as médias *Macro* de 85,06% e *Weighted* de 96,56% é de apenas 11,5%, indicando um equilíbrio entre o desempenho geral e a detecção das classes minoritárias.

Comparado às estratégias do GSOM e ao SVM *Incremental* no mesmo período, o ARF demonstrou superioridade consistente em todas as métricas. Sua capacidade

Tabela 62 – Médias de desempenho do ARF na primeira coleta

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	91,00%	81,51%	85,06%
Weighted	96,58%	96,90%	96,56%

Fonte: Elaborado pelo autor.

de detectar a classe *Reset-Both* com 87,32% de *Recall*, quando tanto o GSOM quanto o SVM *Incremental* obtiveram 0%, representa um avanço significativo. Da mesma forma, para a classe *Drop*, o ARF alcançou *F1-Score* de 52,87%, enquanto o melhor resultado do GSOM foi 0% e do SVM *Incremental* foi 0,27%.

Este resultado inicial estabelece o ARF como o método mais eficaz para classificação multiclasse em cenários de desbalanceamento extremo, confirmando que seu mecanismo de ensemble adaptativo oferece uma vantagem decisiva na detecção de classes minoritárias, quando comparado a métodos não supervisionados, como o GSOM, e mesmo a outros métodos supervisionados, como o SVM *Incremental*.

8.6.2.2 Experimento 2: fevereiro de 2025

Na Tabela 63 é apresentada a matriz de confusão do ARF para a segunda coleta. Neste período, a classe *Drop* aumentou para 18,41% do total de instâncias, o que criou um cenário de teste mais desafiador para a adaptação do modelo.

Tabela 63 – Matriz de confusão do ARF na segunda coleta em Valores Absolutos e Porcentagem

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	551606	3796	55	1342
Deny	4407	244922	2	47011
Reset-Both	498	11	142	1
Drop	904	72971	0	120903

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	99,07%	0,68%	0,01%	0,24%
Deny	1,49%	82,66%	0,00%	15,87%
Reset-Both	76,38%	1,69%	21,78%	0,15%
Drop	0,46%	37,46%	0,00%	62,07%

Fonte: Elaborado pelo autor.

Na Tabela 64 são apresentadas as métricas por classe, mostrando que o ARF manteve desempenho elevado na classe *Allow*, com *F1-Score* de 98,90%. Para a classe *Drop* em crescimento, alcançou *Recall* de 62,07% e *F1-Score* de 66,65%. A classe *Reset-Both* manteve detecção com *F1-Score* de 33,39% e a classe *Deny* apresentou *F1-Score* de 79,29%.

Tabela 64 – Desempenho do ARF por classe na segunda coleta

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	98,73%	99,07%	98,90%	556.799
Deny	76,18%	82,66%	79,29%	296.342
Reset-Both	71,72%	21,78%	33,39%	652
Drop	71,94%	62,07%	66,65%	194.778

Fonte: Elaborado pelo autor.

Na Tabela 65 são apresentadas as médias gerais de desempenho. O *F1-Score Weighted* de 89,29% indica um bom desempenho geral. A diferença entre as médias *Macro* (69,56%) e *Weighted* (89,29%) é de 19,73%, o que reflete os desafios da mudança na distribuição das classes.

Tabela 65 – Médias de desempenho do ARF na segunda coleta

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	79,39%	66,40%	69,56%
Weighted	89,29%	89,30%	89,29%

Fonte: Elaborado pelo autor.

Comparado às estratégias do GSOM no mesmo período, o ARF demonstrou capacidade adaptativa significativamente superior. Para a classe *Drop* em crescimento, o ARF alcançou *F1-Score* de 66,65%, enquanto a melhor estratégia do GSOM, o *growing_phase*, obteve apenas 14,50%. Para a classe *Reset-Both*, o ARF manteve detecção com 33,39% de *F1-Score*, enquanto todas as estratégias do GSOM apresentaram 0% de detecção.

Em termos de desempenho geral, o ARF também superou as estratégias do GSOM, com *F1-Score Weighted* de 89,29% em comparação com 79,46% da melhor estratégia GSOM, o *growing_phase*. Este experimento confirma a vantagem do método supervisionado adaptativo em cenários em que a distribuição das classes muda ao longo do tempo.

8.6.2.3 Experimento 3: Outubro de 2025

Na Tabela 66 é apresentada a matriz de confusão do ARF para a terceira coleta. Este período apresenta a distribuição mais equilibrada do estudo, com as classes principais em proporções semelhantes: *Allow* com 41,32%, *Deny* com 32,27% e *Drop* com 26,29%. Este cenário testa a capacidade do ARF em ambientes balanceados.

Na Tabela 67 são apresentadas as métricas por classe, o que indica bom desempenho em todas as categorias. A classe *Allow* alcançou *F1-Score* de 99,49%, a classe *Deny* de 90,68% e a classe *Drop* de 76,71%. A classe *Reset-Both* manteve detecção

Tabela 66 – Matriz de confusão do ARF na terceira coleta em Valores Absolutos e Porcentagem

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	381327	1332	53	170
Deny	2340	517599	3	40537
Reset-Both	413	5	465	1
Drop	87	58024	0	162841

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	99,59%	0,35%	0,01%	0,04%
Deny	0,42%	92,35%	0,00%	7,23%
Reset-Both	46,72%	0,57%	52,60%	0,11%
Drop	0,04%	26,27%	0,00%	73,70%

Fonte: Elaborado pelo autor.

significativa com *F1-Score* de 66,37%, recuperando-se do desempenho reduzido no Experimento 2.

Tabela 67 – Desempenho do ARF por classe na terceira coleta

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	99,38%	99,59%	99,49%	382.882
Deny	89,07%	92,35%	90,68%	560.479
Reset-Both	89,58%	52,60%	66,37%	884
Drop	79,94%	73,70%	76,71%	220.952

Fonte: Elaborado pelo autor.

Na Tabela 68 são apresentadas as médias gerais de desempenho. O *F1-Score Weighted* de 91,31% representa o melhor desempenho geral observado nos experimentos. A diferença entre as médias *Macro* (83,31%) e *Weighted* (91,31%) é de apenas 8,00%, indicando um equilíbrio quase perfeito entre o desempenho geral e a detecção equitativa de todas as classes.

Tabela 68 – Médias de desempenho do ARF na terceira coleta

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	89,49%	79,56%	83,31%
Weighted	91,31%	91,32%	91,31%

Fonte: Elaborado pelo autor.

Comparado às estratégias do GSOM no mesmo período, o ARF demonstrou superioridade absoluta em todas as métricas. Seu desempenho geral de 91,31% no *F1-Score Weighted* superou em mais de 20% a melhor estratégia do GSOM, o *get_winner*, com 68,15%. Para a classe *Reset-Both*, o ARF alcançou 66,37% de *F1-Score*, enquanto

todas as estratégias do GSOM obtiveram 0% de detecção. Nas classes majoritárias, o ARF também mostrou vantagem significativa, com *F1-Scores* superiores de aproximadamente 28% para *Allow* e 23% para *Deny* em relação à melhor estratégia GSOM.

Este experimento confirma que o ARF atinge seu desempenho máximo em cenários balanceados, demonstrando capacidade adaptativa superior ao método não supervisionado. Enquanto o GSOM mostrou dificuldades em se ajustar à nova distribuição equilibrada, a estratégia *growing_phase* praticamente colapsou na detecção de *Drop*, o ARF manteve desempenho elevado em todas as classes simultaneamente.

8.6.2.4 Experimento 4: Dados Combinados

Na Tabela 69 é apresentada a matriz de confusão do ARF no cenário combinado mais desafiador, em que o modelo foi submetido a uma fase *offline* com dados históricos de agosto de 2024 e a uma fase *online* com dados posteriores de fevereiro e outubro de 2025. Esta configuração simula um cenário real de mudança de conceito, com alterações significativas na distribuição das classes.

Tabela 69 – Matriz de confusão do ARF no Experimento 4 em Valores Absolutos e Porcentagem

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	523209	4118	94	1285
Deny	3596	185102	3	57990
Reset-Both	599	19	498	4
Drop	534	48337	0	174612

	Allow	*Deny*	*Reset-Both*	*Drop*
Allow	98,96%	0,78%	0,02%	0,24%
Deny	1,46%	75,04%	0,00%	23,51%
Reset-Both	53,48%	1,70%	44,46%	0,36%
Drop	0,24%	21,63%	0,00%	78,13%

Fonte: Elaborado pelo autor.

Na Tabela 70 são apresentadas as métricas por classe, o que evidencia a capacidade de generalização temporal. A classe *Allow* apresentou bom desempenho, com *F1-Score* de 98,97%, apesar de sua proporção ter diminuído de 69,76% para 52,87% entre treino e teste. A classe *Drop*, que passou de 3,64% para 22,35%, alcançou *F1-Score* de 76,42% e *Recall* de 78,13%. A classe *Reset-Both* obteve 58,07% de *F1-Score*, um resultado relevante considerando sua baixa representação nos dados da fase *offline*.

Na Tabela 71 são apresentadas as médias gerais de desempenho. O *F1-Score Weighted* de 88,58% representa um bom desempenho geral neste cenário. A diferença entre

Tabela 70 – Desempenho do ARF por classe no Experimento 4

Classe	***Precision***	***Recall***	***F1-Score***	**Suporte**
Allow	98,99%	98,96%	98,97%	528.706
Deny	78,17%	75,04%	76,57%	246.691
Reset-Both	83,69%	44,46%	58,07%	1.120
Drop	74,79%	78,13%	76,42%	223.483

Fonte: Elaborado pelo autor.

as médias *Macro* (77,51%) e *Weighted* (88,58%) é de 11,07%, indicando um bom balanceamento, mesmo com mudanças significativas na distribuição.

Tabela 71 – Médias de desempenho do ARF no Experimento 4

Métrica	***Precision***	***Recall***	***F1-Score***
Macro	83,91%	74,15%	77,51%
Weighted	88,58%	88,59%	88,58%

Fonte: Elaborado pelo autor.

Comparado às estratégias do GSOM no mesmo cenário combinado, o ARF bom desempenho em todas as métricas avaliadas. Seu desempenho geral de 88,58% no *F1-Score Weighted* superou em mais de 20% as estratégias do GSOM, que variaram entre 65,42% e 65,98%. Para a classe *Drop* em crescimento, o ARF alcançou 76,42% de *F1-Score*, enquanto a melhor estratégia do GSOM, o *growing_phase*, obteve apenas 9,66%. A classe *Reset-Both*, não detectada por nenhuma estratégia do GSOM (0% de *F1-Score*), foi identificada pelo ARF com 58,07% de *F1-Score*.

Este experimento demonstra que o ARF possui capacidade de generalização temporal e de adaptação a mudanças na distribuição dos dados. Enquanto as estratégias do GSOM apresentaram dificuldades para detectar a classe *Drop* em crescimento, com *F1-Scores* inferiores a 10%, o ARF mostrou-se eficaz nesse cenário, mantendo bom desempenho em todas as classes simultaneamente.

8.6.3 Análise Comparativa com Métodos Supervisionados

A comparação com os métodos supervisionados de latência nula de rótulos (SVM *Incremental* e ARF) permitiu estabelecer um limite superior de desempenho (*upper bound*) e, assim, avaliar as capacidades e limitações do GSOM, que opera com latência infinita de rótulos.

– **Comparação entre GSOM e SVM *Incremental*:** Em sua melhor configuração, o GSOM demonstrou maior eficácia na detecção de classes que cresceram ao longo do tempo. Para a classe *Drop* no Experimento 3, o GSOM alcançou *F1-Score* de 62,06%, enquanto o SVM *Incremental* obteve 16,05%.

Este resultado sugere uma vantagem na identificação de padrões emergentes sem necessidade de supervisão. Por outro lado, o SVM *Incremental* mostrou-se mais eficaz no balanceamento geral e na detecção de classes minoritárias como *Reset-Both*, evidenciando o benefício direto do aprendizado supervisionado.

– **Comparação entre GSOM e ARF:** O ARF apresentou os melhores resultados em todos os experimentos. Sua arquitetura de *ensemble* adaptativa mostrou-se mais robusta ao lidar com o desbalanceamento dos dados e com mudanças de conceito. Contudo, este desempenho superior acarreta um custo computacional significativo, com tempos de processamento na ordem de dias, em contraste com a maior eficiência computacional do GSOM.

Esta análise revela os diferentes compromissos entre as abordagens. Os métodos supervisionados estabelecem uma referência de desempenho que o GSOM não atinge, pois operam em cenários opostos quanto à latência dos rótulos. Contudo, essa referência tem limitações práticas, pois depende de rotulação imediata, o que é inviável em ambientes de monitoramento, e também exige alto poder computacional.

O GSOM, por outro lado, oferece uma solução prática para as limitações reais do monitoramento de *firewall*. Ele demonstra que é possível identificar padrões importantes por meio de uma abordagem não supervisionada. Assim, seu desempenho não deve ser avaliado em cenários de latência zero de rótulos, mas sim como uma solução para o cenário de monitoramento contínuo, em que os rótulos não estão disponíveis.

8.6.4 Considerações Finais do Capítulo

Os quatro experimentos conduzidos permitem concluir que o GSOM apresenta desempenho adequado para a classificação de logs de *firewall* em cenários com três características principais: (1) distribuição estável das classes; (2) indisponibilidade de rótulos durante a operação; e (3) prioridade na detecção de classes em crescimento, como observado para a classe *Drop*.

Ao comparar as três estratégias de atualização, observou-se que cada uma apresenta vantagens distintas. A estratégia *get_winner* manteve maior robustez em relação a classes históricas, a estratégia *growing_phase* apresentou flexibilidade para capturar classes emergentes, e a estratégia *smoothing_phase* ocupou uma posição intermediária. Essa diferença permite selecionar a estratégia de acordo com a dinâmica de mudanças do ambiente monitorado.

A comparação com métodos supervisionados (SVM *Incremental* e ARF) mostrou o melhor desempenho possível quando há rótulos imediatos. No entanto, na prática

do monitoramento de *firewall*, não é viável obter rótulos instantâneos para todo o tráfego, o que limita a aplicação desses métodos.

Diante disso, o GSOM se consolida como uma solução viável para operações com latência infinita de rótulos. Seu desempenho demonstrou capacidade prática de capturar padrões recorrentes e de adaptar-se a mudanças na distribuição, oferecendo uma abordagem efetiva para o monitoramento contínuo em que a supervisão completa não está disponível.

Capítulo 9

Conclusão

Neste capítulo, são apresentadas as conclusões deste trabalho, incluindo as principais contribuições, as limitações identificadas e as sugestões para trabalhos futuros.

9.1 Principais Contribuições

As principais contribuições deste trabalho são:

- **Aplicação de uma abordagem não supervisionada em FCDs:** Este trabalho aplica o GSOM, um algoritmo não supervisionado, para classificação de logs de *firewall* em FCDs. A abordagem permite o processamento em tempo real de grandes volumes de dados de tráfego, com capacidade de adaptação contínua às mudanças nas distribuições. Diferentemente de métodos supervisionados tradicionais, não requer retreinamento completo nem disponibilidade imediata de rótulos, sendo adequada para cenários com latência infinita de rótulos;
- **Avaliação do GSOM em diferentes cenários temporais:** Condução de uma análise abrangente do GSOM utilizando coletas temporais distintas, evidenciando seu comportamento frente a mudanças nas regras de *firewall* e na distribuição dos dados em FCDs;
- **Identificação de conjuntos de atributos adequados para cenários não supervisionados com latência infinita de rótulos:** A avaliação de diferentes combinações de atributos, extraídas a partir dos 113 atributos disponíveis

nos logs do *firewall*, indicou que o uso de atributos categóricos resultou no melhor desempenho para a classificação nos cenários avaliados;

– **Comparação com métodos supervisionados:** Os resultados do GSOM foram comparados com o SVM *Incremental* e o ARF, métodos supervisionados que operam com latência nula de rótulos e, por isso, representam um limite superior de desempenho. Apesar disso, tais abordagens são inviáveis no cenário prático de classificação de logs de *firewall*, sendo utilizadas neste trabalho como referência comparativa para contextualizar os resultados obtidos com o GSOM.

9.2 Limitações

As principais limitações identificadas são:

– **Sensibilidade à distribuição das classes:** O desempenho do GSOM mostrou forte dependência da distribuição das classes. Classes bem representadas nos dados foram classificadas com alta precisão. Já as classes minoritárias tiveram desempenho significativamente inferior, sendo frequentemente confundidas com classes majoritárias;

– **Restrição de generalização em ambientes específicos:** Os experimentos deste trabalho utilizaram logs coletados a partir de um único *firewall*, cujas regras e formatos de log refletem características próprias desse dispositivo. Assim, os resultados obtidos não devem ser generalizados para *firewalls* de outros fabricantes, podendo ser necessárias adaptações nos algoritmos para obter resultados comparáveis;

– **Sensibilidade à escolha de hiperparâmetros:** O desempenho do GSOM mostrou-se fortemente dependente da configuração dos hiperparâmetros, especialmente do Fator de Espalhamento, do número de Épocas de Crescimento e das Épocas de Suavização. A definição desses valores exigiu ajustes manuais e experimentos prévios. Valores inadequados podem resultar em mapas grandes e fragmentados, com muitos neurônios dispersos, ou em mapas pequenos e pouco representativos, com pouca capacidade de discriminação, impactando diretamente a estabilidade e a precisão do modelo.

9.3 Trabalhos Futuros

Como direcionamentos para pesquisas futuras, sugere-se:

– **Estratégias de balanceamento para fluxos contínuos de dados:** Investigar abordagens de balanceamento de dados adaptadas a cenários de FCDs,

avaliando se técnicas dinâmicas ou sensíveis ao tempo podem reduzir os efeitos da desproporção entre classes;

- **Avaliação em ambientes heterogêneos de segurança de rede:** Estender a avaliação do GSOM para logs provenientes de diferentes tipos de *firewalls* e outros dispositivos de segurança de rede, com o objetivo de analisar a robustez do modelo frente a variações de formato, regras e padrões de tráfego, bem como identificar eventuais adaptações necessárias para sua aplicação em contextos distintos;
- **Avaliação em cenários de mudanças de conceito abruptas:** Realizar experimentos em cenários com mudanças de conceito abruptas nas políticas de *firewall*, de modo a analisar a capacidade de adaptação do GSOM e seus efeitos sobre a estabilidade e o desempenho do modelo;
- **Análise da evolução estrutural do mapa ao longo do fluxo**: como continuidade deste trabalho, pretende-se investigar de forma mais detalhada o crescimento do GSOM durante a fase *online*, registrando os momentos em que novos neurônios são inseridos e relacionando essa expansão com mudanças na distribuição dos dados. Essa análise pode contribuir para compreender melhor a capacidade adaptativa do modelo diante de mudanças de conceito e padrões emergentes no tráfego de rede.
- **Arquiteturas híbridas e semissupervisionadas:** Investigar arquiteturas que combinem o GSOM com métodos supervisionados ou semissupervisionados, explorando o uso de rótulos quando disponíveis. Essa abordagem pode contribuir para melhorar a precisão da classificação, especialmente em classes pouco frequentes, sem comprometer a operação em cenários com latência infinita de rótulos.
- **Uso do GSOM como Sistema de Detecção de Intrusão:** investigar o uso do GSOM não apenas para classificar ações do *firewall*, mas também como base para um Sistema de Detecção de Intrusão, capaz de identificar padrões incomuns de tráfego, comportamentos emergentes e mudanças no perfil da rede. Nessa perspectiva, o modelo também pode apoiar ajustes nas políticas de filtragem e auxiliar a administração do *firewall*.

Apêndice A

Disponibilidade do código e dos dados

Os códigos-fonte desenvolvidos neste trabalho foram disponibilizados em repositório público no GitHub, com o objetivo de favorecer a reprodutibilidade dos experimentos apresentados.[1]

Os logs originais exportados do *firewall* continham atributos com sensibilidade operacional e informacional, incluindo campos capazes de revelar detalhes do ambiente monitorado. Por essa razão, optou-se por não disponibilizar integralmente os arquivos originais.

Para fins de reprodutibilidade, foi disponibilizada uma versão tratada do conjunto de dados, contendo apenas os atributos utilizados nos experimentos descritos nesta dissertação. Dessa forma, foram removidas colunas não empregadas pelo modelo.

Essa estratégia busca preservar a confidencialidade do ambiente monitorado e, ao mesmo tempo, permitir a compreensão da estrutura dos dados e a reprodução das etapas de pré-processamento, treinamento e avaliação descritas neste trabalho.

[1] <https://github.com/wgiarini/gsom-firewall>

Referências

ABDALLAH, Z. S. et al. Anynovel: Detection of novel concepts in evolving data streams: An application for activity recognition. **Evolving Systems**, Springer, v. 7, p. 73–93, 2016.

ABDUALRAHMAN, A. A.; IBRAHEM, M. K. Intrusion detection system using data stream classification. **Iraqi Journal of Science**, p. 319–328, 2021.

AGGARWAL, C. C. **Data streams: models and algorithms**. [S.l.]: Springer, 2007. v. 31.

AGGARWAL, C. C.; HINNEBURG, A.; KEIM, D. A. On the surprising behavior of distance metrics in high dimensional space. In: SPRINGER. **International conference on database theory**. [S.l.], 2001. p. 420–434.

AGGARWAL, C. C. et al. A framework for clustering evolving data streams. In: ELSEVIER. **Proceedings 2003 VLDB conference**. [S.l.], 2003. p. 81–92.

AL-MALIKI, S. et al. Addressing data distribution shifts in online machine learning powered smart city applications using augmented test-time adaptation. **arXiv preprint arXiv:2211.01315**, 2022.

ALAHAKOON, D.; HALGAMUGE, S. K.; SRINIVASAN, B. A self-growing cluster development approach to data mining. In: IEEE. **SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No. 98CH36218)**. [S.l.], 1998. v. 3, p. 2901–2906.

ALAHAKOON, D.; HALGAMUGE, S. K.; SRINIVASAN, B. A structure adapting feature map for optimal cluster representation. In: OHMSHA LTD. **International Conference on Neural Information Processing 1998**. [S.l.], 1998. p. 809–812.

ALAHAKOON, D.; HALGAMUGE, S. K.; SRINIVASAN, B. Dynamic self-organizing maps with controlled growth for knowledge discovery. **IEEE Transactions on neural networks**, IEEE, v. 11, n. 3, p. 601–614, 2000.

ALJABRI, M. et al. Classification of firewall log data using multiclass machine learning models. **Electronics**, MDPI, v. 11, n. 12, p. 1851, 2022.

ALSAQOUR, R.; MOTMI, A.; ABDELHAQ, M. A systematic study of network firewall and its implementation. **International Journal of Computer Science & Network Security**, International Journal of Computer Science & Network Security, v. 21, n. 4, p. 199–208, 2021.

ASLAN, Ö. et al. A comprehensive review of cyber security vulnerabilities, threats, attacks, and solutions. **Electronics**, MDPI, v. 12, n. 6, p. 1333, 2023.

BABCOCK, B. et al. Models and issues in data stream systems. In: **Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems**. [S.l.: s.n.], 2002. p. 1–16.

BARNAWI, A. et al. A systematic analysis of deep learning methods and potential attacks in internet-of-things surfaces. **Neural Computing and Applications**, Springer, v. 35, n. 25, p. 18293–18308, 2023.

BASILE, C.; LIOY, A. Analysis of application-layer filtering policies with application to http. **IEEE/ACM Transactions On Networking**, IEEE, v. 23, n. 1, p. 28–41, 2013.

BERKHIN, P. A survey of clustering data mining techniques. In: **Grouping multidimensional data: Recent advances in clustering**. [S.l.]: Springer, 2006. p. 25–71.

BOTTOU, L. Large-scale machine learning with stochastic gradient descent. In: SPRINGER. **Proceedings of COMPSTAT'2010: 19th International Conference on Computational StatisticsParis France, August 22-27, 2010 Keynote, Invited and Contributed Papers**. [S.l.], 2010. p. 177–186.

BRAGA, P. H.; BASSANI, H. F. A semi-supervised self-organizing map for clustering and classification. In: IEEE. **2018 International Joint Conference on Neural Networks (IJCNN)**. [S.l.], 2018. p. 1–8.

BRANDT, J.; LANZÉN, E. A comparative review of smote and adasyn in imbalanced data classification. 2021.

BREIMAN, L. Random forests. **Machine learning**, Springer, v. 45, n. 1, p. 5–32, 2001.

BUNGUM, L.; GAMBÄCK, B. Self-organizing maps for classification of a multi-labeled corpus. International Institute of Information Technology Trivandrum, India, 2015.

CAO, F. et al. Density-based clustering over an evolving data stream with noise. In: SIAM. **Proceedings of the 2006 SIAM international conference on data mining**. [S.l.], 2006. p. 328–339.

CASTRO, L. de; FERRARI, D. Introdução à mineração de dados: Conceitos básicos. **Algoritmos e Aplicações, Saraiva**, 2016.

CAUWENBERGHS, G.; POGGIO, T. Incremental and decremental support vector machine learning. **Advances in neural information processing systems**, v. 13, 2000.

CHEHRI, A.; FOFANA, I.; YANG, X. Security risk modeling in smart grid critical infrastructures in the era of big data and artificial intelligence. **Sustainability**, MDPI, v. 13, n. 6, p. 3196, 2021.

CORTES, C.; VAPNIK, V. Support-vector networks. **Machine learning**, Springer, v. 20, n. 3, p. 273–297, 1995.

COVER, T.; HART, P. Nearest neighbor pattern classification. **IEEE transactions on information theory**, IEEE, v. 13, n. 1, p. 21–27, 1967.

DAS, M. et al. A skip-connected evolving recurrent neural network for data stream classification under label latency scenario. In: **Proceedings of the AAAI Conference on artificial intelligence**. [S.l.: s.n.], 2020. v. 34, n. 04, p. 3717–3724.

DENNY; WILLIAMS, G. J.; CHRISTEN, P. Visualizing temporal cluster changes using relative density self-organizing maps. **Knowledge and Information Systems**, Springer, v. 25, n. 2, p. 281–302, 2010.

DEVELOPERS, S. learn. **Stochastic Gradient Descent (SGD) - Scikit-learn**. 2025. Scikit-learn. Disponível em: <https://scikit-learn.org/stable/modules/sgd.html>.

DONTIREDDY, S. R. et al. Enhancing transparency: Ai applications for detecting cheating and predicting player attrition in online gaming. In: IEEE. **2024 International Conference on Signal Processing, Computation, Electronics, Power and Telecommunication (IConSCEPT)**. [S.l.], 2024. p. 1–6.

DUA, M. et al. Machine learning approach to ids: A comprehensive review. In: IEEE. **2019 3rd International conference on Electronics, Communication and Aerospace Technology (ICECA)**. [S.l.], 2019. p. 117–121.

DUTT, S.; KALRA, A. A scalable and robust framework for intelligent real-time video surveillance. In: IEEE. **2016 International Conference on Advances in Computing, Communications and Informatics (ICACCI)**. [S.l.], 2016. p. 212–215.

DYER, K. B.; CAPO, R.; POLIKAR, R. Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. **IEEE transactions on neural networks and learning systems**, IEEE, v. 25, n. 1, p. 12–26, 2013.

D'ELIA, G. et al. Concept drift mitigation in low-cost air quality monitoring networks. **Sensors**, MDPI, v. 24, n. 9, p. 2786, 2024.

ELWELL, R.; POLIKAR, R. Incremental learning of concept drift in nonstationary environments. **IEEE Transactions on Neural Networks**, IEEE, v. 22, n. 10, p. 1517–1531, 2011.

ERTAM, F.; KAYA, M. Classification of firewall log files with multiclass support vector machine. In: IEEE. **2018 6th International symposium on digital forensic and security (ISDFS)**. [S.l.], 2018. p. 1–4.

FAYYAD, U. M. et al. Advances in knowledge discovery and data mining. In: AMERICAN ASSOCIATION FOR ARTIFICIAL INTELLIGENCE. [S.l.], 1996.

GALETE, L. B. Um método para agrupamento em fluxo de dados utilizando o algoritmo som. **Curitiba-PR: Pontifícia Universidade Católica do Paraná**, 2012.

GAMA, J. **Knowledge discovery from data streams**. [S.l.]: CRC Press, 2010.

GHESMOUNE, M.; LEBBAH, M.; AZZAG, H. State-of-the-art on clustering data streams. **Big Data Analytics**, Springer, v. 1, n. 1, p. 13, 2016.

GOEL, R.; KUMAR, D.; RAJA, A. A packet filtering firewall. **International Journal of Emerging Technology and Advanced Engineering**, Citeseer, v. 4, p. 362–363, 2014.

GOMES, H. M. et al. Adaptive random forests for evolving data stream classification. **Machine Learning**, Springer, v. 106, n. 9, p. 1469–1495, 2017.

GUHA, S.; MISHRA, N. Clustering data streams. In: **Data stream management: processing high-speed data streams**. [S.l.]: Springer, 2016. p. 169–187.

GUPTA, P. Review of next-generation firewalls. **Available at SSRN 5159425**, 2024.

HAN, J.; KAMBER, M.; PEI, J. Data mining: concepts and techniques, waltham, ma. **Morgan Kaufman Publishers**, v. 10, p. 978–1, 2012.

HARIPRIYA, D. et al. A comparative study on online machine learning techniques for network traffic streams analysis. **International Journal of Intelligent Systems and Applications in Engineering**, v. 12, n. 13s, p. 09–19, 2024.

HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman Editora, 2001.

HAYKIN, S. **Neural networks and learning machines, 3/E**. [S.l.]: Pearson Education India, 2009.

HEATON, J. **Introduction to neural networks with Java**. [S.l.]: Heaton Research, Inc., 2008.

HOFFMANN, H. Kernel pca for novelty detection. **Pattern recognition**, Elsevier, v. 40, n. 3, p. 863–874, 2007.

HRUSCHKA, E. R.; EBECKEN, N. F. A genetic algorithm for cluster analysis. **Intelligent data analysis**, IOS Press, v. 7, n. 1, p. 15–25, 2003.

HULLE, M. V. Faithful representations and topographic maps: From distortion-to information-based self-organization. Wiley; New York, 2000.

JAIN, A. K.; DUBES, R. C. **Algorithms for clustering data**. [S.l.]: Prentice-Hall, Inc., 1988.

JHA, B. K.; SIVASANKARI, G.; VENUGOPAL, K. Fraud detection and prevention by using big data analytics. In: IEEE. **2020 Fourth international conference on computing methodologies and communication (ICCMC)**. [S.l.], 2020. p. 267–274.

KASHPRUK, N.; PISKOR-IGNATOWICZ, C.; BARANOWSKI, J. Time series prediction in industry 4.0: A comprehensive review and prospects for future advancements. **Applied sciences**, MDPI, v. 13, n. 22, p. 12374, 2023.

KHAN, Z. Real-time ai systems for icu risk monitoring. **Int J Multidiscip Res.**, v. 7, n. 2, p. 1–10, 2025.

KHRAISAT, A. et al. Survey of intrusion detection systems: techniques, datasets and challenges. **Cybersecurity**, Springer, v. 2, n. 1, p. 1–22, 2019.

KIZZA, J. M. **Computer network security and cyber ethics**. [S.l.]: McFarland, 2014.

KOHONEN, T. Self-organization and associative memory. **Self-Organization and Associative Memory**, 1988.

KOHONEN, T. The self-organizing map. **Proceedings of the IEEE**, IEEE, v. 78, n. 9, p. 1464–1480, 1990.

KOHONEN, T. The basic som. In: **Self-organizing maps**. [S.l.]: Springer, 2001. p. 105–176.

KOHONEN, T. Self-organizing maps, ser. **Information Sciences. Berlin: Springer**, v. 30, 2001.

KOHONEN, T. **Self-organization and associative memory**. [S.l.]: Springer Science & Business Media, 2012. v. 8.

KOHONEN, T. Essentials of the self-organizing map. **Neural networks**, Elsevier, v. 37, p. 52–65, 2013.

LABIB, K.; VEMURI, R. Nsom: A real-time network-based intrusion detection system using self-organizing maps. **Networks and Security**, v. 21, n. 1, 2002.

LANGLEY, P. et al. An analysis of bayesian classifiers. In: **Aaai**. [S.l.: s.n.], 1992. v. 90, p. 223–228.

LIAO, Y.; VEMURI, V. R. Use of k-nearest neighbor classifier for intrusion detection. **Computers & security**, Elsevier, v. 21, n. 5, p. 439–448, 2002.

LICEN, S.; ASTEL, A.; TSAKOVSKI, S. Self-organizing map algorithm for assessing spatial and temporal patterns of pollutants in environmental compartments: A review. **Science of The Total Environment**, Elsevier, v. 878, p. 163084, 2023.

MAHDIRAJI, A. R. Clustering data stream: A survey of algorithms. **International Journal of Knowledge-based and Intelligent Engineering Systems**, IOS Press, v. 13, n. 2, p. 39–44, 2009.

MAIMON, O. Z.; ROKACH, L. **Data mining with decision trees: theory and applications**. [S.l.]: World scientific, 2014. v. 81.

MARKOU, M.; SINGH, S. Novelty detection: a review—part 1: statistical approaches. **Signal processing**, Elsevier, v. 83, n. 12, p. 2481–2497, 2003.

MARKOU, M.; SINGH, S. Novelty detection: a review—part 2:: neural network based approaches. **Signal processing**, Elsevier, v. 83, n. 12, p. 2499–2521, 2003.

MARSLAND, S.; SHAPIRO, J.; NEHMZOW, U. A self-organising network that grows when required. **Neural networks**, Elsevier, v. 15, n. 8-9, p. 1041–1058, 2002.

MASUD, M. et al. Classification and novel class detection in concept-drifting data streams under time constraints. **IEEE Transactions on knowledge and data engineering**, IEEE, v. 23, n. 6, p. 859–874, 2010.

MELIN, P. et al. Analysis of spatial spread relationships of coronavirus (covid-19) pandemic in the world using self organizing maps. **Chaos, Solitons & Fractals**, Elsevier, v. 138, p. 109917, 2020.

MICHAEL, K. et al. Privacy, data rights and cybersecurity: Technology for good in the achievement of sustainable development goals. In: IEEE. **2019 IEEE International Symposium on Technology and Society (ISTAS)**. [S.l.], 2019. p. 1–13.

MILJKOVIĆ, D. Brief review of self-organizing maps. In: IEEE. **2017 40th international convention on information and communication technology, electronics and microelectronics (MIPRO)**. [S.l.], 2017. p. 1061–1066.

MITCHELL, T. M. **Machine learning**. 1997.

MOHAMMED, R.; AKAY, F. Anomaly detection in network traffic using machine learning. **Cukurova University Journal of Natural and Applied Sciences**, Cukurova University, v. 2, n. 3, p. 5–12, 2023.

NGUYEN, H.-L.; WOON, Y.-K.; NG, W.-K. A survey on data stream clustering and classification. **Knowledge and information systems**, Springer, v. 45, p. 535–569, 2015.

OZA, N. C.; RUSSELL, S. J. Online bagging and boosting. In: PMLR. **International workshop on artificial intelligence and statistics**. [S.l.], 2001. p. 229–236.

PAIVA, E. R. d. F. **Detecção de novidade em fluxos contínuos de dados multiclasse**. Tese (Doutorado) — Universidade de São Paulo, 2014.

Palo Alto Networks. **Palo Alto PA-3260 Firewall**. 2023. <https://www.indiamart.com/proddetail/palo-alto-pa-3260-firewall-24188225388.html>. Acessado em: 14 de Abril de 2025. Disponível em: <https://www.indiamart.com/proddetail/palo-alto-pa-3260-firewall-24188225388.html>.

PATGIRI, R. et al. An investigation on intrusion detection system using machine learning. In: IEEE. **2018 IEEE Symposium Series on Computational Intelligence (SSCI)**. [S.l.], 2018. p. 1684–1691.

PONMALAI, R.; KAMATH, C. **Self-organizing maps and their applications to data analysis**. [S.l.], 2019.

POPESCU, M.-C. et al. Multilayer perceptron and neural networks. **WSEAS transactions on circuits and systems**, v. 8, n. 7, p. 579–588, 2009.

QU, X. et al. A survey on the development of self-organizing maps for unsupervised intrusion detection. **Mobile networks and applications**, Springer, v. 26, p. 808–829, 2021.

ROECKL, C.; DIRECTOR, C. M. Stateful inspection firewalls. **Juniper Networks White Paper**, v. 5, 2004.

ROLEMBERG, T. M. Aplicação de conceitos de redes complexas para a descoberta de formação de grupos em mapas auto-organizáveis. Universidade Presbiteriana Mackenzie, 2021.

RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. **nature**, Nature Publishing Group UK London, v. 323, n. 6088, p. 533–536, 1986.

SCARFONE, K.; MELL, P. Guide to intrusion detection and prevention systems (idps). **NIST Special Publication**, v. 800, n. 94, p. 1–127, 2007.

SCHOLKOPF, B. et al. Support vector method for novelty detection. **Advances in neural information processing systems**, MIT Press Cambridege, Mass, USA, v. 12, n. 3, p. 582–588, 2000.

SETH, S.; SINGH, G.; CHAHAL, K. Drift-based approach for evolving data stream classification in intrusion detection system. In: **Proceedings of the Workshop on Computer Networks & Communications, Goa, India**. [S.l.: s.n.], 2021. p. 23–30.

SILVA, B. et al. Spatiotemporal segmentation of satellite image time series using self-organizing map. **ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences**, Copernicus GmbH, v. 3, p. 255–261, 2022.

SILVA, J. A. et al. Data stream clustering: A survey. **ACM Computing Surveys (CSUR)**, ACM New York, NY, USA, v. 46, n. 1, p. 1–31, 2013.

SILVA, L. E. B. d. **Contribuições a técnicas de agrupamento e visualização de dados multivariados utilizando mapas auto-organizáveis**. Dissertação (Mestrado) — Universidade Federal do Rio Grande do Norte, 2013.

SOUZA, V. M. et al. Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: SIAM. **Proceedings of the 2015 SIAM international conference on data mining**. [S.l.], 2015. p. 873–881.

UCAR, E.; OZHAN, E. The analysis of firewall policy through machine learning and data mining. **Wireless Personal Communications**, Springer, v. 96, p. 2891–2909, 2017.

VESANTO, J.; ALHONIEMI, E. Clustering of the self-organizing map. **IEEE Transactions on neural networks**, Ieee, v. 11, n. 3, p. 586–600, 2000.

ZHENG, H.; WANG, H.; BLACK, N. Human activity detection in smart home environment with self-adaptive neural networks. In: IEEE. **2008 IEEE International conference on networking, sensing and control**. [S.l.], 2008. p. 1505–1510.

ZUCHINI, M. H. Aplicações de mapas auto-organizáveis em mineração de dados e recuperação de informação. **Trabalho de Mestrado apresentado a Faculdade de Engenharia Elétrica e de Computação (FEEC-UNICAMP), Universidade de Campinas**, 2003.