PEDRO HORCHULHACK

ATUALIZAÇÃO CONFIÁVEL DOS MODELOS DE DETECÇÃO DE INTRUSÃO BASEADA EM APRENDIZAGEM DE MÁQUINA

CURITIBA

2023

PEDRO HORCHULHACK

ATUALIZAÇÃO CONFIÁVEL DOS MODELOS DE DETECÇÃO DE INTRUSÃO BASEADA EM APRENDIZAGEM DE MÁQUINA

Projeto de Dissertação de Mestrado apresentado ao Programa de Pós-Graduação em Informática da Pontifícia Universidade Católica do Paraná como requisito parcial para obtenção do título de Mestre em Informática.

Orientador: Dr. Altair Olivo Santin
Coorientador: Dr. Eduardo Kugler Viegas

Pontifícia Universidade Católica do Paraná
Programa de Pós-Graduação em Informática

CURITIBA
2023

Dados da Catalogação na Publicação
Pontifícia Universidade Católica do Paraná
Sistema Integrado de Bibliotecas – SIBI/PUCPR
Biblioteca Central
Luci Eduarda Wielganczuk – CRB 9/1118

Horchulhack, Pedro
H811a Atualização confiável dos modelos de detecção de intrusão baseada em
2023 aprendizagem de máquina / Pedro Horchulhack ; orientador: Altair Olivo Santin ;
coorientador: Eduardo Kugler Viegas. – 2023.
75 f. : il. ; 30 cm

Dissertação (mestrado) – Pontifícia Universidade Católica do Paraná, Curitiba, 2023
Bibliografia: f. 69-75

1. Informática. 2. Aprendizado do computador. 3. Computadores – Medidas de segurança. I. Santin, Altair Olivo. II. Viegas, Eduardo Kugler. III. Pontifícia Universidade Católica do Paraná. Programa de Pós-Graduação em Informática. IV. Título.

CDD. 20. ed. – 004

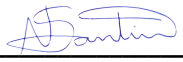
Pontifícia Universidade Católica do Paraná
Escola Politécnica
Programa de Pós-Graduação em Informática

ATA DA SESSÃO PÚBLICA DE DEFESA DE DISSERTAÇÃO DE MESTRADO DO PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA, DA PONTIFÍCIA UNIVERSIDADE CATÓLICA DO PARANÁ

DEFESA DE DISSERTAÇÃO N.º 366

Em sessão pública realizada a distância, às 09h no dia 17 de fevereiro de 2023, ocorreu a defesa intitulada **"ATUALIZAÇÃO CONFIÁVEL DOS MODELOS DE DETECÇÃO DE INTRUSÃO BASEADA EM APRENDIZAGEM DE MÁQUINA"** elaborada pelo aluno **PEDRO HORCHULHACK** como requisito parcial para a obtenção do título de **Mestre em Informática**, na área de concentração **Ciência da Computação.** Após a apresentação da dissertação pelo aluno e correspondente arguição, a banca examinadora emitiu o seguinte parecer sobre a dissertação:

Membros da Banca Examinadora:

Prof. Dr. Altair Olivo Santin (orientador) – PUCPR/PPGIa
DELIBERAÇÃO: aprovado **ASSINATURA:** 

Prof. Dr. Eduardo Kugler Viegas – PUCPR/PPGIa
DELIBERAÇÃO: aprovado **ASSINATURA: À DISTÂNCIA**

Prof. Dr. André Ricardo Abed Grégio - UFPR
DELIBERAÇÃO: aprovado **ASSINATURA: À DISTÂNCIA**

Portanto, conforme as normas regimentais do PPGIa e da PUCPR, a dissertação foi considerada:

(x) APROVADA
(aprovação condicionada ao atendimento integral das correções e melhorias recomendadas pela banca examinadora, conforme anexo, dentro do prazo regimental)
( ) REPROVADA


Prof. Dr. Emerson Cabrera Paraiso
Coordenador do Programa de Pós-Graduação em Informática

AGRADECIMENTOS

O desenvolvimento do projeto foi uma tarefa árdua e gratificante e, com certeza, o caminho até aqui só foi possível devido às pessoas que estiveram comigo durante o processo. Em primeiro lugar, agradeço a Laura dos Santos Rocha, Laurinha, meu amor, por sempre ter acreditado em mim e por ter permanecido ao meu lado. Sem você, sem dúvida, não seria a pessoa e o profissional que sou hoje. Uma frase de agradecimento nunca será suficiente para demonstrar toda a minha gratidão por tê-la em minha vida. Agradeço aos meus pais, que me apoiaram incondicionalmente durante o trabalho e ao longo de toda a minha vida. À Carla Alessandra Fontanetto Horchulhack, minha amada mãe, que sempre me deu apoio emocional e conselhos ao longo do projeto. Ao Márcio José Horchulhack, meu querido pai, que sempre me ensinou o valor da disciplina e do comprometimento com o trabalho e o estudo. Agradeço também aos meus queridos irmãos, Lucas Horchulhack e Mariana Horchulhack, que sempre me apoiaram.

Agradeço aos meus orientadores, Dr. Altair Olivo Santin e Dr. Eduardo Kugler Viegas, por todo o apoio, confiança e por terem me dado a oportunidade de me desenvolver pessoal e profissionalmente, e pelos inúmeros conselhos durante todo o processo. Agradeço ao Dr. Vilmar de Abreu Júnior, que me introduziu à pesquisa através da iniciação científica durante a graduação. Também expresso minha gratidão ao Jhonatan Geremias, que se tornou um grande amigo e companheiro durante minha trajetória no mestrado, e com certeza aprendi muito com a sua experiência.

À Laurinha, com muito amor.

LISTA DE ALGORITMOS

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AM	Aprendizagem de Máquina
CVE	Common Vulnerabilities and Exposures
HIDS	Host Intrusion Detection System
IDS	Intrusion Detection System
NIDS	Network Intrusion Detection System
PQ	Pergunta de Pesquisa

SUMÁRIO

RESUMO

Este trabalho propõe um novo método de realização de atualizações dos modelos usando aprendizagem de fluxo para sistema de detecção de intrusão. Isto facilita as atualizações por meio da redução de instâncias e redução de custos computacionais referente ao treino e atualização. Instâncias rejeitadas na classificação são armazenadas para posteriormente serem utilizadas na atualização incremental. Assim, as instâncias rejeitadas podem ser facilmente rotuladas a partir de repositórios públicos de ataques, sem intervenção humana. Os experimentos empregaram uma base de dados de fluxos de rede de 2,6TB. A proposta foi capaz de manter taxas de acerto por até três meses, mesmo os modelos estando desatualizados, sendo capaz de diminuir as taxas de falsos positivos em até 12% enquanto rejeita 8% das instâncias. Quando atualizações periódicas são feitas, há uma melhora na acurácia de detecção de até 6%, enquanto rejeita 8% dos eventos. As taxas de acerto podem ser mantidas por até três meses, tempo para obter um rótulo apropriado para um determinado evento. A proposta consumiu somente 3,2% do tempo de processamento e 2% de novas instâncias para serem rotuladas, em comparação com as técnicas tradicionais.

Palavras-chave: Aprendizagem de Máquina; Detecção de Intrusão; Aprendizagem por Fluxo; Opção de Rejeição.

ABSTRACT

The current work proposes a new method of updating stream learning models for intrusion detection systems that reduces the required number of instances and computational costs for training and updating. The method stores rejected classification instances for future use in incremental updates. Hence, rejected events may be labeled easily through public attack repositories without human intervention. The experiments used a database comprised of 2.6TB of network flows. The proposed approach maintains its accuracy rates for up to three months even if the underlying models are outdated, reducing the false-positive rates by up to 12% while rejecting 8% of the instances. When executing periodic updates, the accuracy rates increase by up to 6% while neglecting 8% of events. Furthermore, the accuracy rates can be maintained for up to three months, which is the time to obtain an appropriate label for a given event. Finally, the proposal consumed only 3.2% of the processing time and 2% of new instances to be labeled compared to traditional techniques.

Keywords: Machine Learning; Intrusion Detection; Stream Learning; Reject Option.

Capítulo 1

Introdução

O capítulo em questão irá contextualizar o escopo do projeto, bem como as motivações, os objetivos gerais e específicos do trabalho, e ao final será apresentada a estrutura do documento. Em síntese, essa produção visa explorar a utilização de uma abordagem de opção de rejeição para melhorar a confiabilidade das classificações dos eventos de rede, e, em paralelo, reduzir custos computacionais em relação ao tempo de treinamento e quantidade de instâncias para atualização.

1.1. Contexto

Em 2022 cerca de 505 milhões de ataques realizados através da Internet foram bloqueados (KASPERSKY SECURITY BULLETIN, 2022). Administradores de rede frequentemente utilizam de Sistemas de Detecção de Intrusão baseados em Rede (Network Intrusion Detection Systems, *NIDS*) para bloquear fluxos de rede que são potencialmente considerados como ataque, existindo duas principais abordagens: as baseadas em assinatura e as baseadas no comportamento ou anomalia (MOLINA-CORONADO *et al.*, 2020).

A primeira estratégia foca principalmente na criação de regras para tratar ataques já conhecidos (por exemplo, SYN Flood), porém é dependente do conhecimento das características do ataque para a sua detecção, como por exemplo com qual frequência são realizadas requisições de um determinado cliente.

A outra abordagem consiste na coleta de pacotes e criar modelos estatísticos que representem o comportamento do tráfego da rede, para fluxos normais e de potenciais ataques (GATES; TAYLOR, 2006). Com isso, presume-se que a abordagem baseada em comportamento é capaz de identificar novos ataques, já que, em comparação com eventos previamente analisados, se comportam de maneira significativamente diferente (SOMMER; PAXSON, 2010).

Devido ao aumento de ataques na rede, várias soluções de detecção de intrusão baseadas em comportamento foram utilizadas e aplicadas principalmente através de algoritmos de Aprendizagem de Máquina (AM) (CASSALES *et al.*, 2019).

Um algoritmo de AM necessita de um conjunto de dados para realizar três etapas básicas: treinamento, validação e teste. Além disso, no contexto de detecção de intrusão baseado em rede, os fluxos devem ser rotulados em normais ou ataques. Em ambientes de produção tais algoritmos são utilizados para identificar similaridades no comportamento entre novos fluxos na rede com o comportamento apresentado anteriormente na etapa de treinamento (KILINCER; ERTAM; SENGUR, 2021).

Como o comportamento da rede pode sofrer alterações ao longo do tempo, o modelo AM treinado anteriormente se torna desatualizado, pois o comportamento atual não corresponde ao comportamento dos fluxos onde o algoritmo foi previamente treinado (VIEGAS *et al.*, 2019). Como resultado, o antigo modelo de AM apresentará altas taxas de erro em comparação com as taxas obtidas na fase de testes. Por isso, os administradores de redes devem tomar medidas a fim de mitigar a crescente taxa de erro do NIDS assim que ela começa a aumentar (SOMMER; PAXSON, 2010).

Identificar o aumento da taxa de erro em ambientes de produção é uma tarefa desafiadora, principalmente porque os rótulos nestes cenários ainda são desconhecidos (GATES; TAYLOR, 2006).

1.2. Motivação

O tráfego da rede varia significativamente ao longo do tempo, situação causada pelo surgimento de novos ataques, novos serviços e novas conexões (VIEGAS *et al.*, 2019). Nesse contexto, para manter um *NIDS* com altas taxas de acerto, os modelos de AM devem ser frequentemente atualizados (GAO *et al.*, 2019). Dessa forma, o treinamento de modelos com novos dados permitirá que o algoritmo aprenda o atual comportamento da rede, mantendo as taxas de erro baixas. No entanto, em ambientes de produção é uma tarefa difícil, pois o tráfego da rede deve ser coletado e rotulado corretamente (NISIOTI *et al.*, 2018).

O processo de treinamento de modelos de AM é uma tarefa custosa em termos de tempo e armazenamento, especialmente considerando milhões de fluxos de redes. A rotulagem dos fluxos de rede exige mão de obra especializada e tempo, podendo levar de

dias até semanas para que esteja completo, pois depende da assistência humana (INJADAT *et al.*, 2021). Consequentemente, *NIDS* em ambientes de produção ficarão desprotegidos até que os dados atualizados não sejam fornecidos.

A rotulagem de fluxos de rede em tempo real não é viável, pois ela depende de assistência humana, o que pode ser escasso ou demandar um alto custo temporal e financeiro (FONTUGNE *et al.*, 2010). Tal processo torna-se humanamente inviável devido a gigante quantidade de dados, porque é necessário que seja realizado de forma manual, assim os operadores de rede comumente recorrem a técnicas baseadas em assinatura para a rotulagem, pois, posteriormente, podem ser utilizados para a atualização de *NIDS* baseados em AM (GATES; TAYLOR, 2006).

Os rótulos de ataques recentes só estarão disponíveis publicamente após um tempo consideravelmente longo desde a sua primeira exploração, o que pode levar semanas ou até meses (BLAISE *et al.*, 2020). Isso significa que as atualizações dos *NIDS* baseados em AM só podem ser realizadas com base em informações desatualizadas, sendo crucial que o sistema implantado em produção mantenha taxas de acerto altas, mesmo se o modelo de AM subjacente estiver desatualizado (GATES; TAYLOR, 2006, SOMMER; PAXSON, 2010).

Neste contexto, as abordagens atuais para atualizar modelos de AM descartam o modelo antigo e treinam um novo modelo com os novos dados coletados, assim a tarefa de atualização requer uma quantidade maior de fluxos de rede rotulados, o que demanda mais custo computacional no que tange ao tempo de treinamento e maior capacidade de armazenamento para as instâncias (MOLINA-CORONADO *et al.*, 2020).

A atualização de *NIDS* baseado em AM ainda é um problema pouco tratado na literatura, apesar de já mencionado. Os autores supõem que as atualizações periódicas são realizadas, sem considerar os desafios que acompanham, como o custo temporal e de armazenamento, atraso na disponibilização dos rótulos e alto custo financeiro para mão de obra especializada para analisar e rotular os fluxos de rede em seus esquemas propostos.

Em cenários onde há mudanças no comportamento dos dados ao longo do tempo, os autores recorrem a técnica de aprendizado por fluxo (KRAWCZYK *et al.*, 2017). A motivação para adotar tal técnica é devido à incorporação de novos dados ao modelo, facilitando as atualizações em situações em que ocorram mudanças no comportamento dos dados (UD DIN *et al.*, 2020). Apesar dessa característica, supõe-se que os rótulos

estão sempre disponíveis, o que não corresponde à realidade em ambientes de produção já que a grande quantidade de fluxos torna inviável a rotulagem ao longo do tempo.

1.3. Objetivos

Objetivo geral

O trabalho proposto busca detectar intrusões em rede utilizando a aprendizagem de máquina por fluxo, sendo facilitada pela incorporação e rejeição de instâncias ao longo do tempo.

Objetivos Específico

Com o objetivo de facilitar a atualização de NIDS baseados em AM, as seguintes demandas devem ser atendidas:

I. Atualizações periódicas devem ser feitas, de modo a manter as taxas de acerto ao longo do tempo;

II. Através da opção de rejeição, baseadas em um limiar pré-estabelecido, realizar as atualizações ao modelo;

III. Reduzir os custos computacionais tangentes ao armazenamento de instâncias para atualização, bem como a quantidade a ser utilizada no processo de atualização;

IV. Utilizar modelos de AM por fluxo para a tarefa de classificação e atualização, tendo em vista sua característica de aprendizado incremental;

V. Diminuir necessidade de assistência humana em relação ao processo de rotulagem, já que menos instâncias são armazenadas.

1.4. Contribuições

As contribuições do trabalho podem ser resumidas na facilidade de atualizações de modelos de AM, redução de custos computacionais tangentes ao armazenamento de instâncias e tempo de retreino dos modelos, e a diminuição de instâncias a serem rotuladas ao longo do tempo. Desta forma, de maneira mais detalhada, as contribuições são apresentadas a seguir:

I. Classificadores de fluxo e em lotes bastante utilizados na literatura são avaliados considerando suas taxas de acerto ao longo do tempo. Experimentos foram realizados em uma base de dados abrangendo um ano de tráfego de rede real, mostrando que as abordagens atuais são incapazes de detectar intrusões em cenários onde o comportamento da rede se altera, exigindo atualizações periódicas inviáveis, bem como uma quantidade significativa de dados rotulados para o treinamento.

II. Um novo esquema baseado em aprendizagem de fluxo para detecção de intrusão que conduz uma classificação com opção de rejeição e pode manter sua precisão por longos períodos sem atualizações. Nesse contexto, o esquema proposto, sem atualizações, forneceu taxas de acurácia similares às das técnicas tradicionais com atualização mensal.

III. Uma abordagem que viabiliza a atualização do modelo pode realizar atualizações incrementais ao modelos sobre esquemas de detecção de intrusão implantadas durante ambientes de produção. As atualizações dos modelos são realizadas em um pequeno subconjunto de instâncias previamente rejeitadas, diminuindo significativamente os custos computacionais, de tempo e armazenamento, enquanto não exigindo a intervenção humana para a rotulagem, que pode ser aplicada de forma autônoma por meio do uso tradicional de técnicas baseadas em assinatura. O esquema proposto pode armazenar as instâncias rejeitadas por longos períodos antes de usá-los como entrada para atualização sem um impacto significativo nas taxas de acurácia.

1.5. Estrutura do Documento

O restante do documento está organizado de forma que apresenta a fundamentação teórica no Capítulo 2, tratando sobre os desafios da detecção de intrusão em redes, a aplicação de AM para tal tarefa e a relevância da AM por fluxo. O Capítulo 3 aborda os trabalhos relacionados e como eles tratam o problema proposto no trabalho. O Capítulo 4 apresenta a proposta do trabalho. Depois, o Capítulo 5 demonstra os experimentos realizados que elucidam as dificuldades relacionadas à detecção de intrusão e como podem ser solucionados, em seguida apresentando demais experimentos voltados à

proposta, comparando com os resultados da literatura e os obtidos no capítulo anterior. Finalmente, o Capítulo 6 conclui o trabalho.

Capítulo 2
Fundamentação Teórica

O crescimento desenfreado da tecnologia, junto à adoção das redes de computadores, apresenta uma série de problemas voltados à segurança da informação (LIAO *et al.*, 2013). Por exemplo, através da análise de chamadas de sistema ou por meio da contagem da frequência de pacotes de redes em uma janela de tempo é possível determinar se um sistema está sendo atacado ou não. A primeira abordagem foca em Detecção de Intrusão baseada no Hospedeiro (Host-based Intrusion Detection System, *HIDS*), já a segunda objetiva a Detecção de Intrusão baseada na Rede (Network-based Intrusion Detection System, *NIDS*).

Dessa forma, este capítulo visa apresentar os conceitos básicos de detecção de intrusão baseada em rede, a aprendizagem de máquina aplicada ao contexto e a aprendizagem por fluxo. Para fins de escopo do trabalho, será abordado somente o conceito de NIDS.

O capítulo está organizado de forma que a Seção 2.1 introduz sobre o conceito de Aprendizagem de Máquina (AM). Na Seção 2.2 estabelece um fundamento teórico sobre detecção de intrusão baseado em rede. Já a Seção 2.3 apresenta os desafios e a aplicação de aprendizagem de máquina no contexto de *NIDS*. Por fim, a Seção 2.4 denota as características ideais para um *NIDS* de fácil atualização.

2.1. Aprendizagem de Máquina

Por muitos anos, a estatística tem sido utilizada para entender o comportamento de uma amostra de dados. No entanto, com a redução de custos relacionados aos computadores e o aumento da capacidade de processamento, a aprendizagem de máquina (AM) está cada vez mais sendo adotada para tarefas de reconhecimento de padrões em dados.

A aprendizagem de máquina consiste em aprender o comportamento de um conjunto de dados. Por exemplo, um algoritmo (modelo) pode ser treinado para reconhecer a presença de câncer de mama com base em um conjunto de características (WOLBERG *et al.*, 1993).

Esse conjunto de características pode ser representado logicamente por um vetor, onde cada elemento corresponde a um atributo. A Tabela 1. Amostra de tamanho 4 da base de dados de câncer de mama. pode representar a relação entre a instância e o vetor de características, onde cada linha representa uma instância e cada coluna representa um atributo específico. É importante ressaltar que a tabela mencionada contém algumas características (três) do conjunto de dados mencionado anteriormente, juntamente com o rótulo de cada instância para fins didáticos sendo 1 quando há presença de câncer e 0 quando não existe a presença.

Tabela 1. Amostra de tamanho 4 da base de dados de câncer de mama.

ID	***Raio médio***	***Concavidade média***	***Área média***	***Diagnóstico***
1	11,08	0,02363	372,70	1
2	20,47	0,15230	1299,00	0
3	12,87	0,01797	509,20	1
4	10,26	0,07542	321,60	1

De modo geral, existem três abordagens principais: aprendizagem supervisionada, não supervisionada e semi-supervisionada (SAHANI *et al.*, 2023). A primeira abordagem enfoca o uso de dados rotulados (por exemplo, se um vetor de características de um conjunto de dados indica a presença ou ausência de câncer de mama) para aprender a prever a saída com base no vetor de características fornecido como entrada. Por outro lado, a abordagem não supervisionada visa encontrar uma representação dos dados sem o uso de rótulos, dividindo-os em grupos ou clusters. Por fim, a aprendizagem semi-supervisionada é uma combinação das duas primeiras técnicas, na qual busca-se melhorar o desempenho de uma das tarefas, seja classificação ou agrupamento, utilizando os dados das outras (VAN ENGELEN e HOOS, 2020).

Além das abordagens de aprendizado, existem duas formas de processamento consideradas: aprendizado em lote e aprendizado por fluxo (SCHNEIDER *et al.*, 2016). A estratégia de aprendizado em lote pressupõe que todos os dados estejam disponíveis para o treinamento do modelo e que sejam utilizados simultaneamente. Por outro lado, o

aprendizado por fluxo ocorre de forma incremental, com uma instância sendo fornecida de cada vez, de maneira sequencial. Além disso, os autores consideram que as observações mais recentes contêm informações mais relevantes do que as mais antigas, permitindo que um modelo se adapte melhor aos dados atuais e alcance melhores taxas de acurácia.

Considerando o cenário de aprendizado por fluxo, existe a ideia de desvio de conceito, onde as propriedades estatísticas dos dados mudam ao longo do tempo de forma inesperada. Esse desvio pode ocorrer devido a uma variedade de fatores, como mudanças nas condições do ambiente, alterações nas preferências dos usuários ou até mesmo devido a erros ou anomalias nos dados de entrada (LU *et al.*, 2018).

Essas mudanças repentinas podem impactar negativamente o desempenho dos modelos de aprendizado de máquina, uma vez que os padrões que foram aprendidos anteriormente podem não ser mais relevantes ou aplicáveis aos novos dados. Além disso, nos últimos anos vários estudos têm aplicado técnicas de aprendizado por fluxo em cenários em que o comportamento muda ao longo do tempo (ZHONG *et al.*, 2020). Uma possível aplicação para a detecção de desvios de conceito e da aprendizagem por fluxo é a detecção de intrusão, onde apresenta um cenário com alta variabilidade estatística dos dados.

Em resumo, a aprendizagem de máquina foca no reconhecimento de padrões sobre dados e, tendo em vista o contexto de detecção de intrusão baseado em rede, torna-se uma técnica bastante pertinente, o qual será apresentado na seção a seguir. Além disso, a aprendizagem por fluxo permite com que, de modo geral, exista menor consumo de recursos computacionais, no que tange ao tempo de treinamento e armazenamento de instâncias, viabilizando ainda mais seu emprego no contexto a seguir.

2.2. Detecção de Intrusão Baseado em Rede

Um sistema de detecção de intrusão (Intrusion Detection System, *IDS*) busca identificar atividades maliciosas em software ou hardware, a fim de planejar e aplicar contramedidas onde um firewall comum não seria capaz de detectar (LIAO *et al.*, 2013). A arquitetura básica de um *IDS* é composta por quatro módulos sequenciais (MOLINA-CORONADO *et al.,* 2020): *aquisição de dados*, *extração de características*, *classificação* e *alerta*. O primeiro módulo é responsável por coletar os dados do ambiente

monitorado, o que é comumente realizado através da coleta de pacotes de uma interface de rede ou chamadas de sistema. Os dados coletados, então, são fornecidos ao módulo de *extração de características*, extraindo um conjunto de características a respeito do comportamento dos dados. Já no módulo de classificação são comparados com uma base de conhecimento previamente armazenada (seja um conjunto de assinaturas ou um padrão de comportamento dos dados), classificando se cada evento é um *ataque* ou um evento *normal*. Por fim, quando um fluxo é identificado como ataque, então o módulo alerta reporta ao administrador da rede.

Em um *IDS* existem duas abordagens de detecção, a primeira baseada em assinatura e a segunda baseada em comportamento (ou anomalia) (KHRAISAT *et al.*, 2019). A primeira foca principalmente em identificar regras que compõem um determinado ataque como, por exemplo, analisar um conjunto de chamadas de sistema ou uma frequência temporal de pacotes com as mesmas *flags* ativadas. Já a baseada em comportamento foca em extrair um padrão estatístico de um grande conjunto de dados, sejam eles de *host* ou de rede. Ainda, todos os fluxos que se diferem de um comportamento definido como padrão são considerados como ataque. Em um NIDS os dados da rede são, em sua maioria, representados por um fluxo de rede, sendo resumido na comunicação entre equipamentos e serviços em uma dada janela de tempo.

No contexto de *NIDS*, a Tabela 2 lista um exemplo de características de rede que podem ser utilizadas para a tarefa de classificação (MOORE; ZUEV, 2005). Neste caso, os pacotes da rede que são trocados entre os equipamentos foram agrupados em origem, destino e hosts, totalizando 66 características extraídas, sendo 22 para cada subconjunto. Os fluxos de rede denotam o comportamento da comunicação entre entidades na rede, medido pela quantidade de troca de pacotes na rede ao longo do tempo, então construindo um vetor de características, podendo classificar um fluxo de rede como normal ou ataque, que posteriormente é fornecido de entrada para o módulo de classificação.

Tabela 2. Exemplo de conjunto de características que pode ser extraído no nível de rede, considerando que cada agrupamento de características está dentro de uma janela de tempo de 15 segundos.

Agrupamento de Rede	**#**	***Características Coletadas***
Comunicação de *hosts,* Origem para Destino, Destino para Origem	1	Primeiro Quartil do Tempo de Inter-Chegada
	2	Mediana do Tempo de Inter-Chegada
	3	Média do Tempo de Inter-Chegada
	4	Terceiro Quartil do Tempo de Inter-chegada

	5	Tempo Máximo de Inter-chegada
	6	Variância do Tempo de Inter-Chegada
	7	Tempo Mínimo de Inter-Chegada
	8	Tamanho Mínimo do Pacote
	9	Primeiro Quartil do Tamanho do Pacote
	10	Média do Tamanho do Pacote
	11	Mediana do Tamanho do Pacote
	12	Terceiro Quartil do Tamanho do Pacote
	13	Tamanho Máximo do Pacote
	14	Variância do Tamanho do Pacote
	15	Quantidade Total de Pacotes
	16	Total de Pacotes com a Opção TCP ACK
	17	Total de Pacotes Somente com a Opção TCP ACK
	18	Total de Pacotes com a opção TCP SYN
	19	Total de Pacotes com a opção TCP FIN
	20	Total de Pacotes com a opção TCP PSH
	21	Total de Pacotes com a opção TCP URG
	22	Taxa de Transferência da Rede

Diversas técnicas foram propostas para a classificação de fluxo de rede, onde os autores frequentemente recorrem a abordagens baseadas em AM, comumente pelo reconhecimento de padrões em lotes (MOLINA-CORONADO *et al.,* 2020). Tal esquema consiste em três etapas sequenciais: treinamento, validação e teste (VIEGAS; SANTIN; OLIVEIRA, 2017).

Na primeira etapa, um algoritmo de AM é construído através da avaliação dos dados da rede disponíveis em um subconjunto separado para testes, onde o algoritmo procura aprender o padrão de comportamento dos dados, assim otimizando a separação das classes recebidas de entrada em normal e ataque (MISHRA *et al.,* 2019). Portanto, o subconjunto de treino deve ser composto de uma quantidade significante de eventos rotulados, ambos normal e ataque.

A etapa de validação procura identificar possíveis melhorias ao algoritmo, como por exemplo realizar ajustes finos dos parâmetros e a seleção de características. É importante ressaltar que para tal etapa é separado um subconjunto de dados composto de amostras

diferentes do subconjunto de treinamento. Finalmente, a taxa de acerto do modelo otimizado é estimada durante a fase de testes, utilizando um subconjunto dos dados específicos para esse propósito. Assim, o modelo resultante pode ser implantado em ambiente de produção para classificar novos eventos na rede (GATES; TAYLOR, 2006).

2.3. Desafios de Aprendizagem de Máquina para NIDS

Ambientes de rede apresentam vários desafios comparados a outras áreas onde a AM foi aplicada (GATES; TAYLOR, 2006; SOMMER; PAXSON, 2010; VIEGAS; SANTIN; OLIVEIRA, 2017). O comportamento do tráfego da rede é altamente variável e construir uma base realista se torna uma tarefa desafiadora, devido à necessidade de a base de treino ser composta por uma quantidade realista de eventos de rede, que também seja capaz de retratar o ambiente de produção esperado, para finalmente criar um modelo de AM confiável.

Devido à possibilidade de a rede sofrer mudanças significativas em curtos períodos, se torna dificultoso construir uma base de dados realista que capture todas as variações (VIEGAS; SANTIN; OLIVEIRA, 2017). A variabilidade no tráfego da rede, uma situação que pode ser causada tanto pela ocorrência de novos ataques, pelo provisionamento de novos serviços ou de novos conteúdos dos serviços sendo requisitados (VIEGAS; SANTIN; OLIVEIRA, 2017; VIEGAS *et al.,* 2019) que, por consequência, mesmo se um modelo de AM "perfeito" for construído com uma base de dados realista, irá apresentar aumento na taxa de erro ao longo do tempo, exigindo que atualizações periódicas sejam realizadas para manter as taxas de acerto do sistema (VIEGAS *et al.,* 2019).

As atualizações em NIDS baseados em AM não é uma tarefa fácil, considerando que uma nova base de treino deve ser construída e os dados coletados devem ser propriamente rotulados (GATES; TAYLOR, 2006). O processo de rotulagem pode ser executado manualmente ou de forma autônoma.

O primeiro processo seria inviável, considerando a vasta quantidade de dados que devem ser inspecionados ou executados por técnicas baseadas em assinatura (MOLINA-CORONADO *et al.,* 2020).

No segundo processo os eventos de rede podem ser rotulados de forma autônoma, contanto que os dados coletados dos ataques sejam conhecidos publicamente, se tornando

disponível, por exemplo, ao ser reportado em uma base de dados de vulnerabilidades e exposições comuns (*Common Vulnerability and Exposures*, CVEs). No entanto, a divulgação de tais ataques ocorre somente depois de um longo período (SOMMER; PAXSON, 2010), com alguns estudos sugerindo até 300 dias depois a sua divulgação (BILGE; DUMITRAŞ, 2012).

Para aplicar técnicas baseadas em assinatura, os eventos da rede que foram coletados devem ser armazenados por longos períodos antes que eles sejam utilizados adequada e autonomamente rotuladas, deixando os sistemas desprotegidos contra novos ataques devido a desatualização do *NIDS* baseado em AM não ter sido prontamente atualizado.

2.4. Características Ideais para Facilitar Atualizações

As características ideais para a detecção de intrusão baseada em AM, que seja confiável, foram bastante discutidas em diversos trabalhos na literatura (VIEGAS; SANTIN; OLIVEIRA, 2017).

De modo geral, é desejado que os esquemas propostos sejam capazes de generalizar o comportamento dos eventos de uma base de treino, assim, os esquemas propostos devem detectar ataques e serviços similares e/ou novos, independente do ambiente atual. No entanto, apesar da capacidade de generalização, o comportamento do ambiente muda com o passar do tempo, exigindo atualizações.

A tarefa de atualização de algoritmos de AM dos *NIDS* ainda permanece negligenciada na literatura e os autores frequentemente assumem que essa pode ser realizada rapidamente quando necessário (GATES; TAYLOR, 2006; SOMMER; PAXSON, 2010), porém, na prática tal tarefa apresenta um desafio significante para os administradores da rede (VIEGAS; SANTIN; OLIVEIRA, 2017). No entanto, até mesmo identificar modelos de AM desatualizados é negligenciado pelos autores. Um modelo de detecção de intrusão baseado em AM deve ser capaz de performar atualizações facilmente, então o sistema se torna capaz de lidar com mudanças no tráfego da rede.

Idealmente um *NIDS* baseado em AM deve fornecer as seguintes características em relação à atualização dos algoritmos:

- ***Pequena amostra dos novos eventos da rede***. Uma tarefa de atualização deve ser executada somente com poucos eventos da rede. Neste caso, a quantidade de

instâncias usadas para atualizações do modelo deve ser reduzida, devido aos desafios com relação a coleta de novos dados e o conhecimento de que o tráfego da rede muda ao longo do tempo.

- ***Procedimento rotulagem de fácil uso***. O processo de rotulagem de novos eventos de rede deve ser realizado de maneira fácil de utilizar, pois intervenção humana é tipicamente necessária para tal processo. Tal intervenção inviabiliza a periodicidade, principalmente quando quantidades gigantes de dados devem ser rotuladas. Idealmente, a tarefa de rotulagem pode fazer uso de técnicas tradicionais baseadas em assinatura, o que permite a rotulagem autônoma.

- ***Baixo consumo de recursos computacionais***. Atualizações são tipicamente executadas offline, então os custos computacionais podem ser utilizados à vontade. No entanto, devido à natureza variável do comportamento do tráfego da rede, as atualizações devem ser realizadas frequentemente, tendo um alto consumo de recursos computacionais, e esse alto custo computacional pode se tornar um obstáculo no que tange a administração da rede.

- ***Realizadas dentro de um curto intervalo***. As atualizações periódicas tipicamente demandam vários dias ou semanas para serem realizadas, e, enquanto isso, o modelo desatualizado executa em ambiente de produção, deixando o sistema desprotegido contra novos tipos de ataques. Portanto, a tarefa de atualização deve providenciar um esquema que execute em uma curta janela de tempo.

- ***Esquema com alta durabilidade***. O comportamento do tráfego da rede varia significativamente no decorrer do tempo, assim, os modelos de AM subjacentes devem lidar com tais mudanças sem que sejam realizadas atualizações frequentes. Ainda, tais atualizações não devem ser executadas com tanta frequência, principalmente devido aos desafios relacionados à tal tarefa.

- ***Manter as acurácias mesmo com os algoritmos desatualizados***. Algoritmos de AM atualizados não podem ser adquiridos frequentemente, considerando o tempo necessário para as atualizações e quão frequente um administrador de rede pode atualizá-los. O modelo de AM deve garantir que ele consiga manter sua acurácia durante um tempo, mesmo frente a um novo tráfego de rede que não foi experienciado durante a etapa de treino.

Capítulo 3
Trabalhos Relacionados

Diferentes abordagens de detecção de intrusão baseadas em rede já foram consideradas na literatura. Este capítulo procura apresentar os trabalhos relacionados ao projeto, dando ênfase na aplicação de aprendizagem de máquina para *NIDS* na Seção 3.1. Já na Seção 3.2 é abordada o mecanismo de atualização em *NIDS* baseados em AM e qual a sua relevância para o contexto tratado. Na Seção 3.3 é apresentada a questão de utilização de taxas de confiança para classificação e como elas são úteis para o trabalho. Por fim, a Seção 3.4 apresenta uma discussão e comparação acerca dos trabalhos ao projeto proposto e quais problemas ele procura resolver que ainda são existentes na literatura.

3.1. Aprendizagem de Máquina para *NIDS*

As técnicas de Aprendizado de Máquina (AM) têm sido aplicadas com sucesso em diversas áreas, com o objetivo de realizar classificações precisas (GATES; TAYLOR, 2006; VIEGAS *et al.*, 2019). No entanto, muitos dos esquemas propostos na literatura buscam alcançar altas taxas de acerto, frequentemente considerando configurações de ambiente de produção que não são realistas, diferente deste trabalho, que busca utilizar uma base de dados de eventos de rede realística. A seguir serão apresentados trabalhos relacionados a este que tratam de bases de dados irrealistas, justificando o porquê de tal adversidade ser tratada.

O trabalho de Kilincer *et al*. (2021), por exemplo, comparou as taxas de acerto de diferentes classificadores em lote, como Máquinas de Vetores de Suporte (Support Vector Machines, SVM), k-Vizinhos Mais Próximos (k-Nearest Neighbors, KNN) e Árvores de Decisão (Decision Trees, DT), utilizando cinco bases de dados de intrusão. Os classificadores alcançaram altas taxas de acerto, independentemente dos conjuntos de dados utilizados. No entanto, é importante considerar que a avaliação desses classificadores em cenários reais pode apresentar desafios adicionais, como mudanças no comportamento do tráfego de rede e a necessidade de atualizações periódicas do modelo.

Portanto, é fundamental desenvolver abordagens que levem em conta esses aspectos e permitam uma detecção eficiente de intrusões em ambientes reais.

Outra abordagem de aprendizagem em lote foi proposta por Sangkatsanee *et al.* (2011), onde realizam uma detecção de intrusão em tempo real utilizando árvores de decisão. Apesar do baixo tempo de treinamento e de ter atingido altas taxas de acerto, os autores não consideraram atualizações periódicas.

A técnica de Ensemble de classificadores também é uma abordagem popular para aumentar as taxas de acerto. A título de exemplo, os autores Gao *et al.* (2019) mostraram que com a detecção de intrusão é possível aumentar as taxas de acerto do sistema, através de um conjunto de classificadores e um esquema de voto majoritário, contudo, o modelo proposto não considera atualizações periódicas. Fatemeh *et al.* (2020) também propuseram um sistema de classificação com múltiplos algoritmos, baseado em AdaBoost e redes neurais, nesse, os autores atingiram altas taxas de acerto em uma única base de dados através da seleção de características e múltiplos classificadores, enquanto as mudanças de tráfego não foram tratadas.

Jie Gu *et al.* (2019) utilizaram técnicas de aumento de dados, concomitantemente à ensemble, para melhorar as taxas de acerto, porém o esquema proposto não considerou a característica não-estacionária do tráfego da rede ou os desafios oriundos à não atualização dos algoritmos. Otoum *et al.* (2020) propuseram um sistema de detecção de intrusão também baseado em ensemble em um campo de rede sem fio, e da mesma forma apresentaram baixas taxas de erro, no entanto, o seu esquema deixou o sistema desprotegido contra mudanças no tráfego da rede.

Os pesquisadores Das *et al.* (2022) também propuseram um procedimento baseado em AM utilizando ensemble de classificadores para melhorar as taxas de acerto de sistemas de detecção de intrusão. Além disso, sua abordagem também focou na engenharia de características para otimizar ainda mais os resultados, que são selecionadas através da votação majoritária entre vários selecionadores de características. Desta forma, os autores foram capazes de atingir altas taxas de acerto e baixas taxas de alarmes falsos, apesar de tais resultados, sua solução não trata de ambientes onde há mudança no comportamento da rede e os custos computacionais relacionados à seleção de características.

Ainda, os autores Pu *et al.* (2021) propuseram uma técnica para identificar anomalias combinando Máquinas de Vetores de Suporte de Uma Classe (One-Class Support Vector

Machines, OCSVM) e Clusterização de Subespaço (Sub-Space Clustering, SSC), onde cada subespaço é usado como entrada para uma instância do OCSVM e finalmente para a detecção de anomalias. Apesar de sua técnica rotular os eventos de forma autônoma, ela é computacionalmente custosa para aplicar em tempo real.

Para lidar com as dificuldades resultantes das atualizações periódicas em um ambiente dinâmico, os pesquisadores frequentemente recorrem a técnicas de aprendizagem por fluxo (KRAWCZYK *et al.,* 2017), tais técnicas podem atualizar incrementalmente o algoritmo com novas instâncias rotuladas. Apesar de ser utilizada em várias áreas, a detecção de intrusão baseada em aprendizagem por fluxo ainda está no seu início. Assim, Adhikari *et al.* (2018) propuseram um esquema de aprendizagem por fluxo para tratar tentativas de intrusão que evoluem. Apesar da sua técnica facilitar a atualização do esquema, a proposta assume que o rótulo do evento está sempre disponível.

Outra abordagem para detecção de intrusão utilizando aprendizagem por fluxo foi proposta por Martindale *et al.* (2020), onde a junção de múltiplos classificadores melhorou as taxas de acerto, no entanto, os autores também assumiram que os rótulos dos eventos podem ser providenciados quando necessário, sendo irrealista em ambientes de produção.

Um esquema proposto por Horchulhack *et al.* (2022a) foi baseado em aprendizagem por fluxo para tratar do problema relacionado ao custo computacional das atualizações periódicas. Além disso, a proposta dos autores utiliza de um detector de mudança de conceito para identificar mudanças no comportamento da rede e por fim atualizar os modelos de AM de fluxo. Desta forma, assim que um evento é detectado como fora do padrão comportamental da rede, ele é armazenado para eventuais atualizações, desse modo, foram capazes de demonstrar a efetividade de se realizar atualizações periódicas utilizando modelos de aprendizagem por fluxo, porém não é aplicável em ambientes de produção devido à alta quantidade de eventos a serem rotulados e os autores assumirem que os rótulos estão sempre disponíveis.

3.2. Atualizações Periódicas dos Modelos de AM para *NIDS*

Conforme mencionado na Seção 3.1, a utilização de Aprendizagem de Máquina (AM) para detecção de intrusão tem sido amplamente explorada na literatura. No entanto, é importante destacar que os trabalhos geralmente se concentram em altas taxas de acerto,

deixando uma lacuna em relação as atualizações periódicas. À medida que o tempo passa, o comportamento dos fluxos da rede sofre alterações, o que requer a realização de atualizações concomitantes. A seguir, serão apresentados trabalhos que abordam especificamente esse assunto.

As atualizações periódicas foram consideradas pelos autores Alebachew *et al.* (2021), onde uma base de dados para treino é continuamente atualizada com novas instâncias. A cada atualização dos modelos, a quantidade usada para treino é aumentada, demandando mais tempo de treinamento ao longo do tempo. Além disso, outros trabalhos como Alshammari e Zincir-Heywood (2012) já consideravam nocivas as mudanças no tráfego da rede e a avaliaram através da exposição de classificadores já treinados em ambientes com tráfego totalmente diferente, por consequência, os pesquisadores identificaram como tais mudanças afetam significativamente a segurança de sistemas de detecção de intrusão.

Liang e Ma (2021) também consideraram as mudanças no tráfego da rede e o problema de considerar que os rótulos dos ataques estão disponíveis na medida do necessário. Os pesquisadores argumentam que a velocidade emergente de ataques é muito maior que à da disponibilização de assinaturas que representam tais ataques, ou seja, rótulos públicos, então sua proposta se baseou em uma técnica de *blockchain* para construir uma base de dados confiável sendo mantida por instituições e universidades, com o objetivo de aumentar a vida útil de sistemas de detecção de intrusão. Porém, sua abordagem não considerou a frequência das atualizações realizadas, bem como o custo computacional de tais atualizações.

Como tentativa de reduzir o custo computacional no tocante ao treinamento e atualização de modelos de AM, Li *et al.* (2020) propuseram um esquema que utiliza aprendizagem federada, esta pode realizar a tarefa de treinamento de forma colaborativa e distribuída, onde os dados de treino podem ser armazenados em dispositivos terminais com o objetivo de treinamento. No entanto, a técnica pode incorrer em problemas de privacidade principalmente por enviar dados sensíveis à uma unidade centralizada.

Horchulhack *et al.* (2022b) apresentaram uma técnica de AM para detecção de intrusão baseada no aumento de dados e transferência de aprendizado para mitigar os problemas relacionados a escassez de rótulos. Através de uma Rede Adversária Generativa Tabular, os pesquisadores recriaram a distribuição original da base de dados, assim utilizando as instâncias geradas como entrada para eventuais atualizações dos

modelos. Para reduzir os custos computacionais do treinamento, eles se apoiaram na técnica de transferência de aprendizagem, os reduzindo em até 14%, contudo, os pesquisadores não consideraram a confiabilidade das classificações ao longo do tempo, assim levando a interpretações potencialmente errôneas de um evento da rede.

3.3. Confiabilidade das Classificações

Ainda que os classificadores apresentem altas taxas de acerto ao longo do tempo, podem existir inconsistências nos resultados, como por exemplo um vetor de características ser classificado como falso positivo ou falso negativo. No contexto de *NIDS*, a confiabilidade das classificações, ou seja, qual a probabilidade de um evento da rede ser efetivamente um ataque ou não, são cruciais para manter a segurança do sistema no longo prazo. Os trabalhos a seguir elucidarão a necessidade de implementar estratégias para analisar a confiabilidade das classificações, bem como o impacto de se rejeitar eventos com base em um limiar pré-definido.

Os esquemas propostos devem manter sua confiança ao longo do tempo, mesmo estando desatualizados, ainda lidando com os principais desafios relacionados a atualizações de modelos em *NIDS*. De modo geral, para garantir a confiabilidade das classificações, diferentes autores avaliam as taxas de confiança dos classificadores (HANCZAR, 2019). Por exemplo, Lin *et al.* (2018) avaliaram os valores de confiança dos classificadores para rejeitar potenciais classificações errôneas em análises de imagens biomédicas. MARINHO *et al.* (2017) também utilizaram da mesma estratégia de rejeição para a classificação de imagens de mapas. VIEGAS *et al.* (2020) usaram os valores de confiança das classificações para avaliar a confiança da classificação na detecção de intrusão, nesse, o modelo proposto é capaz de manter a acurácia do sistema, porém, os autores tratam das atualizações do modelo ao longo do tempo assumindo que o rótulo de um evento está disponível quando necessário.

A classificação com opção de rejeição tem apresentado resultados significantes em diversos campos onde uma classificação errada leva à um alto risco, contudo, o uso não é regular em detecção de intrusão, apesar dos danos que um falso negativo pode causar à um sistema computacional. Devido as dificuldades relacionadas à falta de confiança dos esquemas propostos, com frequência causada pela mudança no comportamento da rede ao longo do tempo, as abordagens são raramente utilizadas em produção. No entanto, os trabalhos relacionados dificilmente consideram uma avaliação eficiente em suas

propostas, onde abordagens baseadas em AM constantemente tomam conta, descartando as propriedades de um ambiente de produção. MAGÁN-CARRIÓN *et al.* (2020) propuseram um framework para uma avaliação confiável de abordagens de detecção de intrusão que considera uma metodologia estruturada para replicar todos os passos necessários, desde a engenharia de características até as métricas de performance. Desta forma, os autores providenciaram uma abordagem para avaliação bem definida que trabalhos relacionados possam replicar. O principal objetivo foi que sua abordagem fosse capaz de fornecer a replicação dos resultados reportados na literatura, sem desafiar o senso comum usados em demais trabalhos.

Por exemplo, os autores Viegas, Santin e Oliveira (2017) introduziram as características esperadas para um modelo de detecção de intrusão baseado em AM: que seja confiável, reprodutível e com uma base de dados que permita a avaliação utilizando o framework proposto por MAGÁN-CARRIÓN *et al.* (2020). As propriedades desejadas são inclusas na detecção de ataques e serviços similares e/ou novos, enquanto mantém a medição da taxa de acerto quando implantado em novos ambientes. No entanto, o impacto causado pelas mudanças no tráfego da rede ao longo do tempo não foi tratado nos trabalhos citados anteriormente.

3.4. Discussão

Após os trabalhos apresentados no presente capítulo nota-se que a maioria possui um foco em melhorar as taxas de acurácia dos sistemas de detecção de intrusão que, apesar desses resultados, são dificilmente aplicados em ambientes reais. Ainda que tais soluções fossem implantadas em ambientes de produção, ainda estariam sujeitas à alta variabilidade do comportamento do tráfego da rede, prejudicando os classificadores no longo prazo.

Quando o comportamento da rede muda, os classificadores não interpretam corretamente o novo padrão dos dados, necessitando que atualizações periódicas sejam realizadas. Além disso, segundo os trabalhos listados nas seções anteriores, os autores dificilmente consideraram os aspectos voltados ao consumo de recursos computacionais que, em um ambiente real, são cruciais para uma resposta imediata à uma mudança de comportamento. Ainda que os autores se preocupassem com tal consumo, a aplicação dos modelos propostos não seria tão efetiva no longo prazo, devido ao alto tempo de

treinamento, atualização, espaço de armazenamento e rotulagem dos eventos, que precisa ser avaliado.

Mesmo que existam soluções capazes de manter as taxas de acerto no longo prazo sem demandar atualizações e que também sejam capazes de realizar atualizações periódicas, outro ponto importante a considerar é a confiança das classificações. Alguns trabalhos relacionados que foram apresentados trataram das confianças de classificação, porém em outros contextos e, ainda que no cenário de sistemas de detecção de intrusão baseado em rede, não consideraram as mudanças do comportamento da rede. Através de tal análise é possível que se diminua as taxas de falsos-positivos e falsos-negativos, consequentemente aumentando a confiabilidade do sistema.

Com os pontos apresentados nos últimos parágrafos, é possível notar as lacunas ainda não exploradas na literatura, assim oportunizando que o presente trabalho as preencha. Por fim, a solução de tais adversidades podem alavancar a implementação de esquemas de detecção de intrusão baseados em AM em ambientes reais, sendo estes problemas: manter as taxas de acurácia ao longo do tempo, tratar de mudanças do tráfego da rede através de atualizações periódicas, avaliação das classificações e o custo computacional do esquema. Desta forma, os esquemas tornam-se de fácil implantação e com alta confiabilidade para administradores de rede. A

Tabela 3 lista todos os trabalhos relacionados em comparação com a proposta.

Tabela 3. Trabalhos relacionados comparados ao trabalho proposto.

Trabalho	**Ensemble de Classificadores**	**Aprendizagem por Fluxo**	**Atualizações Periódicas**	**Avaliação de Classificação**	**Avaliação de Custo Computacional**
Kilincer *et al.* (2021)	Não	Não	Não	Não	Não
Sangkatsanee *et al.* (2011)	Não	Não	Sim	Não	Não
Gao *et al.* (2019)	Sim	Não	Não	Não	Não
Fatemeh *et al.* (2020)	Sim	Não	Não	Não	Não
Jie Gu *et al.* (2019)	Sim	Não	Não	Não	Não

Otoum *et al.* (2020)	Sim	Não	Não	Não	Não
Das *et al.* (2022)	Sim	Não	Não	Não	Não
Pu *et al.* (2021)	Sim, porém é realizado a combinação entre OCSVM e SSC	Não	Não	Não	Não
Adhikari *et al.* (2018)	Não	Sim	Sim	Não	Não
Martindale *et al*. (2020)	Sim	Sim	Sim	Não	Não
Horchulhack *et al.* (2022a)	Não	Sim	Sim	Não	Sim
Liang e Ma (2021)	Não	Não	Sim	Não	Não
Li *et al*. (2020)	Não	Não	Sim	Não	Sim
Horchulhack *et al.* (2022b)	Não	Não	Sim	Não	Sim
Viegas *et al.* (2020)	Não	Sim	Sim	Sim	Não
Viegas, Santin e Oliveira (2017)	Não	Não	Sim, porém as atualizações têm enfoque na base de dados	Sim	Não

Capítulo 4
Processo de Atualização de Modelo de Aprendizagem de Máquina para Detecção de Intrusão em Redes

A detecção de intrusão em redes por meio de Aprendizado de Máquina (AM) tem sido amplamente explorada e discutida na literatura científica. No entanto, as soluções atuais carecem de considerar a dinâmica do comportamento da rede, além da dificuldade de obter rótulos para os fluxos em tempo hábil, o que compromete a atualização dos modelos. Diante desses desafios, este trabalho apresenta uma abordagem que trata tanto a mudança do comportamento da rede quanto a atualização do modelo, combinando técnicas de AM de fluxo e uma opção de rejeição.

Nesse contexto, este capítulo está organizado da seguinte forma: a Seção 4.1 apresenta a proposta desenvolvida; a Seção 4.2 contextualiza a detecção de intrusão com base em classificadores de fluxo; a Seção 4.3 apresenta o procedimento de atualizações *offline*; e, por fim, a Seção 4.4 discute os desafios de NIDS em relação aos aspectos computacionais, como o tempo de processamento e o armazenamento das instâncias ao longo do tempo.

4.1. Proposta

A proposta tem o objetivo de manter as taxas de acurácia, juntamente facilitando a atualização do modelo, ao longo do tempo, conforme detalhado na Seção 1.3. Com tal finalidade, existem duas etapas a serem implementadas, denominadas: *Detecção de Intrusão com Aprendizagem por Fluxo* e *Atualizações offline*.

A etapa *Detecção de Intrusão com Aprendizagem por Fluxo* busca manter as taxas de acurácia ao longo do tempo, mesmo que atualizações não sejam realizadas. A proposta

considera que as atualizações podem ser feitas com atrasos ou nem mesmo serem aplicadas, como ocorre em ambientes de produção. Como consequência, o algoritmo de classificação implantado deve lidar com o novo comportamento do tráfego da rede, mesmo quando os modelos de AM estejam desatualizados.

Para evitar o impacto negativo sobre a acurácia causado pelo novo comportamento da rede, são avaliados os valores de confiança da classificação durante uma classificação baseada em uma lógica de opção de rejeição. Os valores de confiança podem ser representados pela probabilidade de um vetor de características avaliado ser um ataque ou não. Como resultado, somente classificações com alta confiança e com maior probabilidade de estarem corretas são aceitas pelo modelo, permitindo assim a manutenção da acurácia do sistema ao longo do tempo, mesmo com classificadores desatualizados.

Já a segunda etapa da proposta, *Atualizações Offline*, foca em facilitar as atualizações dos modelos de AM. O esquema aplica atualizações incrementais ao modelo utilizando instâncias que foram rejeitadas pelos classificadores implantados durante o ambiente de produção, diminuindo assim o número de instâncias usadas para atualização. Além disso, para facilitar a tarefa de rotulagem das instâncias, o esquema armazena as instâncias rejeitadas por um período prolongado antes de utilizá-las para atualização.

Como resultado, o administrador da rede pode rotulá-las facilmente utilizando a técnica tradicional baseada em assinatura de sistemas de detecção de intrusão. Isso ocorre porque, quando o sistema é atualizado com as instâncias rejeitadas, o rótulo apropriado do evento estará publicamente disponível, por exemplo, em uma base de dados de CVE. A proposta sugere que a tarefa de atualização do modelo pode ser facilmente realizada usando instâncias que não puderam ser classificadas de forma confiável pelo modelo de AM subjacente durante a produção, conforme avaliado por meio de uma classificação com abordagem de opção de rejeição. As seções seguintes irão descrever os módulos implementados.

4.2. Detecção de Intrusão Com Aprendizagem Por Fluxo

O comportamento do tráfego de rede muda independentemente do *NIDS* baseado em AM implantado. Ele deve manter suas taxas de acurácia mensuradas na etapa de teste, mesmo que o modelo de AM subjacente esteja desatualizado, contudo, *NIDS* baseados

em AM tradicionais aplicam uma decisão em todas as instâncias avaliadas, mesmo que elas sejam desconhecidas para o modelo de AM, incorrendo no aumento das taxas de erro ao longo do tempo. Dessa forma, os atuais esquemas de detecção frequentemente aumentam suas taxas de erro mensuradas na fase de testes ao longo do tempo, causada principalmente pela mudança do comportamento do tráfego da rede.

Para tratar tais lacunas a classificação é realizada de duas formas. Primeiro, para facilitar as atualizações do modelo, a classificação é realizada através de um *pool* (conjunto) de classificadores de fluxo e, como resultado, as atualizações dos modelos podem ser realizadas incrementalmente, diminuindo os custos computacionais, de tempo e armazenamento, atrelados ao processo de atualização tradicional.

Segundo, para manter as taxas de acurácia por períodos mais extensos, mesmo que atualizações periódicas não sejam realizadas, os eventos são classificados através de opção de rejeição. Para atingir tal objetivo, foi avaliado as taxas de confiança de classificação de cada classificador e foram aceitas apenas aquelas classificações que ultrapassaram um limiar de aceitação de classificação pré-definido.

Finalmente, o rótulo do evento da rede é atribuído através de um sistema de voto majoritário de todas as classificações aceitas de cada classificador. Em contrapartida, se nenhum classificador aceitar uma classificação, o evento é rejeitado e um alerta é emitido. A confiança na classificação independe do algoritmo de classificação, por exemplo, o classificador Random Forest produz seus valores de confiança de acordo com a relação de confiança individual entre árvores de decisão que atribuem um determinado rótulo ao evento avaliado.

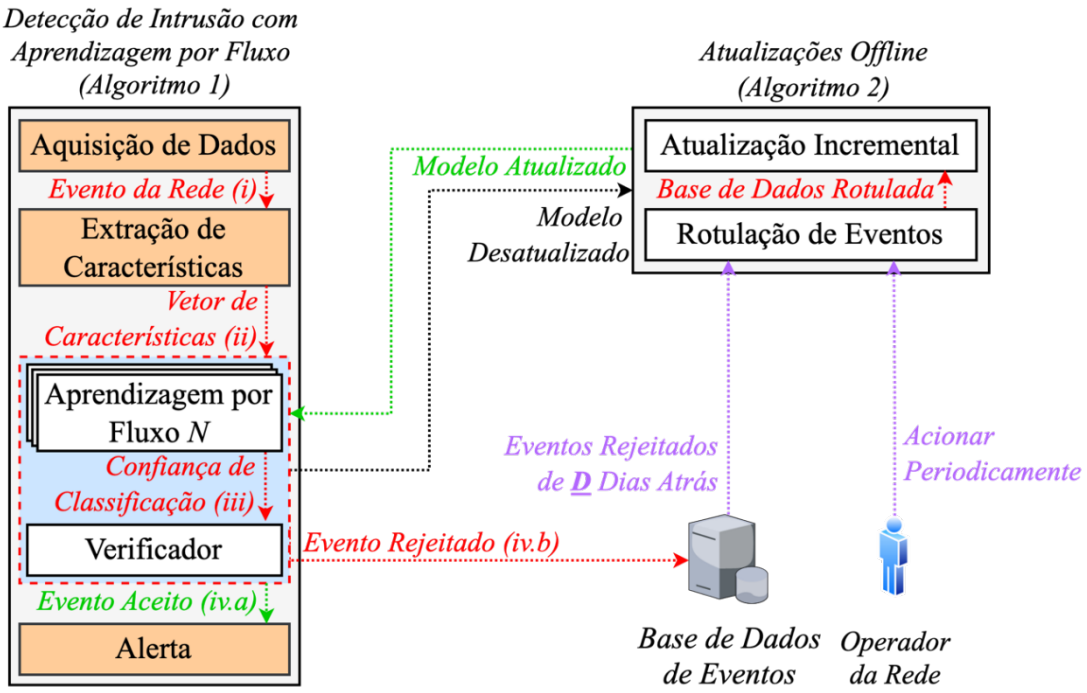
É importante observar que o limiar de rejeição deve ser definido de acordo com o critério do operador porque um limite mais alto produzirá maior confiabilidade, mas com diferentes eventos sendo rejeitados. Em contrapartida, um limiar mais baixo rejeitará menos instâncias, mas produzirá taxas de erro mais altas durante a classificação ao longo do tempo.

Todo o processo de classificação é ilustrado na Figura 1 (*Detecção de Intrusão com Aprendizagem por Fluxo*). Ele inicia com eventos da rede a serem coletados por um módulo *Aquisição de Dados* (Figura 1, Evento *i*). O comportamento dos dados coletados é extraído através do módulo *Extração de Características*, compondo um vetor de características (Figura 1, Evento *ii*). O vetor extraído é classificado por um *pool*

(conjunto) de classificadores de fluxo, em que cada modelo retorna um valor de confiança atrelado à classificação (Figura 1, *Aprendizagem por Fluxo N*).

Os valores de confiança são avaliados através do módulo *Verificador*, onde avalia tais valores de cada classificador de acordo com o limiar de aceitação de classificação (Figura 1, Evento *iii*). Cada classificação aceita é usada para estabelecer o rótulo final de cada evento através do processo de voto majoritário. Se todos os classificadores aceitarem o evento avaliado, ele é redirecionado para o módulo *Alerta*, onde são reportados propriamente ao administrador da rede (Figura 1, Evento *iv.a*). Por outro lado, os eventos rejeitados pelo *Verificador* são redirecionados para o armazenamento para eventuais atualizações do modelo (Figura 1, Evento *iv.b*).

Figura 1. Proposta de detecção de intrusão baseada em AM por fluxo com atualizações atrasadas. O procedimento de classificação é executado continuamente através de um pool de classificadores de fluxo. As atualizações do modelo são realizadas *offline* e periodicamente com antigas instâncias rejeitadas.



O Algoritmo 1 descreve o processo de classificação da proposta. Ele recebe como entrada uma instância (*inst)* a ser classificada, composta de um conjunto de características de fluxo ($x_1, \ldots, x_n$), um *pool* de classificadores com vários classificadores de fluxo e um limiar de aceitação das classificações para os rótulos *ataque* e *normal* para cada classificador. Cada classificador realiza a classificação e o valor de confiança (*conf*) correspondente é avaliado para aceitar ou rejeitar uma decisão de um único classificador. As decisões aceitas são usadas por um processo de voto majoritário (*vote*). Finalmente,

se nenhum classificador aceitar a classificação, o evento é rejeitado (*rejeitar*); senão, um alerta é disparado (*alertar*).

Algoritmo 1. Proposta do Classificador de Fluxo de Rede.

Algoritmo 1: Proposta do Classificador de Fluxo de Rede

Requer:
Instância *inst* = $\{x_1, \dots, x_n\}$
Pool de classificadores *pool* = $\{c_1, \dots, c_n\}$
Limiares *limiares* = $\{(t_a^1, t_n^1), \dots, (t_a^N, t_n^N)\}$
Procedimento CLASSIFICAÇÃO(*inst, pool, limiares*)
Para cada *classificador, t* ∈ {*pool, limiares*} **faça**
classe, conf ← *classificar(classificador, inst)*
Se *classe = normal* **e** *conf* $\geq t_n$, **então**
vote(inst, normal)
Senão se *classe = ataque* **e** *conf* $\geq t_a$, **então**
vote(inst, ataque)
Fim se
Fim para
Se *retornaQuantidadeVotos(inst)* = 0, **então**
rejeitar(inst)
Senão
alertar(retornaVotoMajoritário(inst))
Fim se
Fim procedimento

4.3. Atualizações O*ffline*

Atualizações do modelo são realizadas *offline*, considerando que o modelo de AM desatualizado ainda estará implantado em ambiente de produção. Para diminuir o número de eventos a serem rotulados ao longo do tempo, a tarefa de atualização é feita somente com os eventos que foram previamente rejeitados pela proposta. Em razão disso, as atualizações podem ser realizadas somente através de instâncias que não foram classificadas de forma confiável pelo modelo desatualizado implantado em produção.

Isso é devido ao fato de que o modelo de AM subjacente será melhorado somente com as instâncias que ele foi incapaz de classificar corretamente, e o resultado é que a proposta pode diminuir a quantidade de instâncias a serem rotuladas, armazenadas e usadas para as atualizações do modelo. Ademais, as atualizações são feitas somente após uma janela de tempo pré-definida desde a rejeição de um dado evento (Figura 1, *Eventos Rejeitados de **D** Dias Atrás*) porque, se eventos antigos são usados para a atualização, eles podem ser rotulados de forma autônoma usando ferramentas tradicionais baseadas em assinatura. Consequentemente, a partir do momento que o evento rejeitado é usado

para atualização do modelo, o rótulo será disponibilizado publicamente, por exemplo, em bancos de dados de ataques à cibersegurança, tornando a tarefa de rotulagem viável em um ambiente de produção. Assim, as atualizações podem ser aplicadas autonomamente sem intervenção humana.

O processo de atualização do modelo é ilustrado na Figura 1. Ele é ativado periodicamente pelo administrador da rede como, por exemplo, todo mês. Neste caso, os eventos rejeitados mais antigos que uma janela de tempo pré-definida que foram armazenados (por exemplo, 30 dias depois da sua rejeição) são coletados de um banco de dados de eventos que armazena tais eventos rejeitados do ambiente de produção.

Os eventos selecionados então são rotulados através do módulo *Rotulação de Eventos*, que pode ser realizada tanto por assistência especializada ou por técnicas baseadas em assinatura para a rotulagem autônoma dos eventos. Assim, tais eventos são usados para atualizações incrementais do modelo do *pool* de classificadores de fluxo implantado no ambiente de produção.

O Algoritmo 2 descreve a tarefa de atualização do modelo da proposta. Ele recebe como entrada um intervalo de armazenamento *d* que representa a quantidade de dias que um evento rejeitado deve ser armazenado antes que ele seja utilizado para atualização, um *dataset* composto pelas instâncias rejeitadas mais velhas que *d*, um *pool* de classificadores e um provedor de rótulos *l*.

O provedor de rótulos é usado para a rotulagem dos eventos de rede corretamente, este pode ser o emprego de assistência especializada, ferramentas baseadas em assinatura ou até mesmo algoritmos de AM não-supervisionados. Sendo assim, as atualizações do modelo são realizadas através da requisição do rótulo do evento para cada instância na base de dados. Com tal rótulo, o evento é utilizado para a atualização incremental do modelo para cada classificador no *pool*. Finalmente, o *pool* de classificadores atualizado é redirecionado para o módulo de classificação para atualizar os modelos de AM subjacentes (Figura 1, *Modelo Atualizado*).

Algoritmo 2. Proposta de Atualização *Offline*.

Algoritmo 2: Proposta de Atualização *Offline*
Requer:
Intervalo de Armazenamento d = *diasParaArmazenarInstRejeitadas()*
Base de dados *dataset* = retornaInstRejeitadasAntigas()
Pool de classificadores *pool* = $\{c_1, \ldots, c_n\}$
Provedor de rótulos *l* = *retornaProvedorRótulos()*
Procedimento ATUALIZAÇÃOMODELO(*dataset, pool, l*)
Para cada *instância* ∈ *dataset* **faça**
Rótulo = providenciarRótulo(l, instância)
Para cada *classificador* ∈ *pool* **faça**
atualizaçãoIncremental(classificador, instância, rótulo)
Fim para
Fim para
Fim procedimento

Como resultado, o processo de atualização dos modelos é significativamente facilitado. Menos instâncias são necessárias para as atualizações, considerando somente as rejeitadas para tal. As instâncias selecionadas podem ser autonomamente rotuladas através de técnicas tradicionais baseadas em assinatura, considerando que as instâncias podem ser armazenadas por um extenso período e que seu rótulo será publicamente conhecido quando forem utilizados com o propósito de atualização do modelo. Não obstante, o custo temporal e de armazenamento necessário durante as atualizações do modelo também é significativamente reduzido, principalmente porque o modelo aproveita da característica incremental dos classificadores de fluxo. Isso também é resultado da quantidade reduzida de instâncias aplicadas para as atualizações, pois são utilizadas somente as instâncias previamente rejeitadas pelo esquema.

4.4. Discussão

A quantidade necessária de eventos de rede para atualização diminui na medida que o esquema proposto é capaz de selecionar quais instâncias devem ser utilizadas com o propósito de atualização do modelo através da classificação com opção de rejeição, assim, as instâncias são armazenadas por um tempo determinado antes que sejam utilizadas para atualização. Desta forma, novos eventos de rede coletados podem ser facilmente rotulados à medida que são armazenados antes de serem utilizado como entrada dos classificadores para a atualização dos modelos.

Os custos computacionais são reduzidos, sejam de tempo de processamento e armazenamento, já que é considerado que somente um subconjunto das instâncias seja utilizado para as atualizações do modelo e os modelos desatualizados são atualizados ao longo do tempo.

Finalmente, considerando que o modelo aceita somente classificações com alta taxa de confiança, as taxas de acurácia e a vida útil do modelo são melhoradas durante o ambiente de produção mesmo com modelos desatualizados, considerando que somente classificações com alta confiança são aceitas pela proposta. Como resultado, o modelo proposto pode tratar os principais desafios relacionados à atualização de modelos de *NIDS* baseados em AM, ainda melhorando a confiabilidade na detecção de intrusão ao longo do tempo.

Capítulo 5
Experimentos

O presente capítulo procura investigar como as mudanças no tráfego da rede podem afetar as abordagens de detecção de intrusão baseadas em AM e como atualizações do modelo podem ser utilizadas para resolver tal desafio. Além disso, o capítulo apresentará os experimentos realizados aplicando o esquema proposta no trabalho, detalhado no capítulo anterior.

A organização do capítulo está de forma que a Seção 5.1 descreve o ambiente de testes, informações sobre a base de dados e o software utilizado. Na Seção 5.2 são realizadas análises sobre como a mudança no comportamento do tráfego da rede pode prejudicar a segurança dos NIDS. Por fim, na Seção 5.3 são realizados experimentos que buscam validar o esquema proposto neste trabalho.

5.1. Descrição do Ambiente de Testes

As técnicas de detecção de intrusão construídas sobre tais bases de dados não foram avaliadas considerando um cenário onde mudanças no comportamento da rede estejam presentes, principalmente com relação as taxas de acerto. Um conjunto de dados de detecção de intrusão realista (VIEGAS; SANTIN; OLIVEIRA, 2017) deve proporcionar um tráfego de rede real que possa ser observado em um ambiente de produção. Os dados da rede devem conter dados válidos e um tráfego de rede altamente diversificado. Os eventos coletados devem ser previamente rotulados como eventos *normais* ou *ataque* através de assistência especializada ou abordagens autônomas, como por exemplo através de ferramentas baseadas em assinatura ou algoritmos de AM não-supervisionados. O tráfego da rede deve ser coletado para intervalos prolongados, permitindo assim uma avaliação do impacto das mudanças no comportamento do tráfego da rede ao longo do tempo. Finalmente, o conjunto de dados construído deve estar disponível ao público para um benchmark adequado dos resultados obtidos.

Atender as características acima em um conjunto de dados para detecção de intrusão é uma tarefa desafiadora que pode ser alcançada de duas maneiras (VIEGAS; SANTIN;

OLIVEIRA, 2017). Primeiro, os autores podem coletar tráfego real da rede a partir do ambiente de produção, assim, embora os dados coletados sejam realistas, altamente diversos e válidos, o compartilhamento é muitas vezes impossível devido a preocupações com a privacidade (WU *et al.*, 2020). Apesar disso, a coleta do tráfego real da rede por intervalos prolongados pode representar um desafio significativo por causa das dificuldades relacionadas a rotulagem dos eventos (WANG; BAH; HAMMAD, 2019). Em segundo lugar, os autores podem montar uma bateria de testes controlada que pode facilitar a tarefa de rotulagem, assim como a coleta de dados para longos intervalos (VIEGAS; SANTIN; OLIVEIRA, 2017), porém, o tráfego de rede coletado é muitas vezes irrealista, apresentando uma baixa diversidade na comunicação cliente/servidor (TAVALLAEE; STAKHANOVA; GHORBANI, 2010). Por exemplo, a Tabela 4 lista as estatísticas da base de dados que foi utilizada no trabalho, sendo mais detalhada na sequência.

Tabela 4. Estatísticas da base de dados de intrusão.

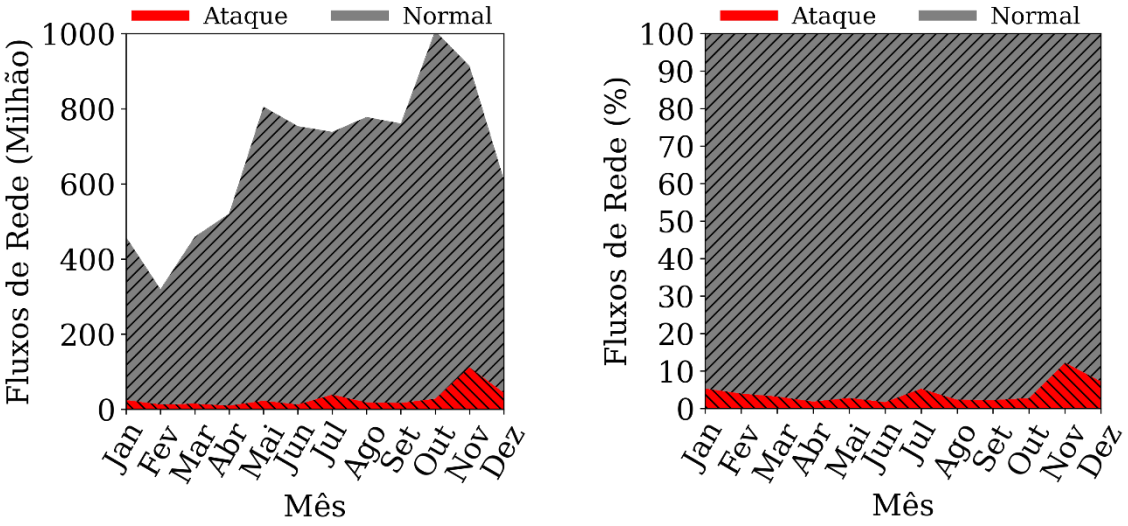
Propriedade	***Valor***
Média Diária de Pacotes de Rede	110 Milhões
Média Diária de Fluxos de Rede	22 Milhões
Média Diária de Fluxos Anômalos	0,9 Milhões
Média Diária da Taxa de Transferência	420 Mbps
Média Diária do Tamanho da Base	7,1GB
Total de Pacotes de Rede	40,1 Bilhões
Total de Fluxo de Rede	8,1 Bilhões
Tamanho Total da Base	2,6TB

A proposta do trabalho é testada utilizando a base de dados *MAWIFlow* (VIEGAS *et al.*, 2019). Os dados são compostos por tráfego de rede que passa através do Samplepoint-F do arquivo MAWI (MAWI, 2012). Ainda, é composto por tráfego real, válido e altamente diverso. Todos os dias é realizada uma coleta, tendo uma duração de 15 minutos, de uma interconexão entre o Japão e os Estados Unidos da América. Para a avaliação, foi selecionado o tráfego da rede que ocorreu durante o ano de 2014. A base consiste em mais de 2.6TB de dados, composta de aproximadamente 40 bilhões de pacotes de rede. Devido a imensa quantidade de dados, para viabilizar a tarefa de rotulagem, tal etapa foi realizada uma técnica de AM não-supervisionada do MAWILab (FONTUGNE *et al.*, 2010), que permite a rotulagem autônoma das instâncias, portanto

sem assistência humana. As anomalias identificadas são rotuladas como *ataque*, enquanto o restante é rotulado como *normal.* Para a extração de características, a ferramenta BigFlow (VIEGAS *et al.*, 2019) foi utilizada, que agrupa eventos em intervalos de 15 segundos enquanto extrai 66 características de fluxo da abordagem de Moore (MOORE; ZUEV, 2005), como listado na Tabela 2 (Seção 2.2).

Além disso, a Figura 2 denota as distribuições da base de dados considerada. A Figura 2a mostra as quantidades de cada classe de fluxo (ataque ou normal), enquanto a Figura 2b mostra a distribuição percentual de cada classe (ataque ou normal).

Figura 2. Distribuição dos fluxos de rede da base de dados.



(a) Quantidade de fluxos de rede ao longo do tempo.

(b) Distribuição dos fluxos ao longo do tempo.

É importante ressaltar que a base de dados está desbalanceada, ou seja, existem mais exemplos de uma classe do que outra. Para tratar essa situação foi realizada uma subamostragem aleatória da classe majoritária, sem repetição, deixando a proporção entre as classes igual. O tratamento do desbalanceamento permitirá melhores taxas de acerto, pois não existirá um viés sobre a classe majoritária. Na seção a seguir será mais bem detalhado o procedimento realizado sobre a base de dados.

5.2. Tratando Mudanças no Comportamento da Rede

Esta seção busca responder as seguintes Perguntas de Pesquisa (PQ):

- (**PQ1**) Como alterações no comportamento do tráfego da rede impacta técnicas tradicionais baseadas em AM?

- (**PQ2**) Como atualizações periódicas ao modelo afetam a acurácia ao longo do tempo?
- (**PQ3**) Quais os custos temporais e de armazenamento de se realizar atualizações periódicas ao modelo?

Para fins de avaliação, duas técnicas de AM bastante utilizadas foram consideradas: aprendizado por *lote* e aprendizado por *fluxo*. A aprendizagem por *lote* se refere ao reconhecimento de padrões tradicional, onde o retreino do modelo descarta o antigo a cada atualização. Por outro lado, o aprendizado por *fluxo* se refere a abordagens que aplicam atualizações incrementais nos modelos desatualizados, ou seja, não há descarte do modelo antigo.

Aprendizado por Lote. Quatro algoritmos de aprendizado por lote, amplamente utilizados na detecção de intrusão, foram avaliados, nomeadamente Florestas Aleatórias (Random Forest, RF), *Gradient Boosting* (GBT), AdaBooting (Ada) e *Ensemble*. O RF foi avaliado considerando 100 árvores de decisão como seu estimador de base, cada uma configurada com *gini* como medidor de qualidade de nós, sem um valor máximo da profundidade de cada árvore. O GBT também foi avaliado considerando 100 árvores de decisão como estimador base, com uma taxa de aprendizagem de 0,1 e *friedman mse* como a medida de qualidade de poda. Ada foi avaliado usando o algoritmo de *boosting* com 100 arvores de decisão como os estimadores base e uma taxa de aprendizagem de 1,0. Finalmente, o *ensemble* foi implementado através do sistema de voto majoritário para os três classificadores apresentados. Os algoritmos foram implementados sobre a API scikit-learn v.0.24.

Aprendizado por Fluxo. De forma semelhante, outros quatro classificadores de fluxo, amplamente utilizados na literatura, foram avaliados sobre a base de dados do trabalho: Árvore de Hoeffding (Hoeffding Tree, HT), Leveraging Bag (Bag), OzaBagging (Oza) e *Ensemble*. O HT foi avaliado usando o ganho de informação como o critério de separação de nós, um período de carência de 200 e Naive Bayes adaptativo para a predição dos nós folha. O Bag foi avaliado com 3 HT como estimador base e com o ADWIN como algoritmo de detecção de mudança, com 0,002 no parâmetro delta. Para o Oza também foram consideradas 3 HT como estimadores base. O *Ensemble* foi implementado através do sistema de voto majoritário para os três classificadores apresentados. Os classificadores de aprendizado por fluxo foram implementados sobre a API scikit-multiflow v.0.5.3. Considerando a alta disparidade entre as classes *normal* e

ataque, foi realizada uma subamostragem aleatória, sem reposição, em cada dia da base de dados (Figura 2).

Os classificadores foram avaliados de acordo com suas taxas de falsos positivos, falsos negativos e F1 scores. Para tanto, foram utilizadas as métricas:

- Verdadeiro-Positivo (VP): Quantidade de amostras de *ataque* corretamente classificadas como *ataque*.
- Verdadeiro-Negativo (VN): Quantidade de amostras de *normal* corretamente classificadas como *normal*.
- Falso-Positivo (FP): Quantidade de amostras *normal* erroneamente classificadas como *ataque*.
- Falso-Negativo (FN): Quantidade de amostras *ataque* erroneamente classificadas como *normal*.

O F1-Score foi calculado como a média harmônica entre a precisão e a revocação, considerando amostras de *ataque* como positivas e amostras *normal* como negativas (TAHERI *et al.*, 2020), como listado a seguir.

$$\text{Precisão} = \frac{\text{VP}}{\text{VP+FP}}$$

$$\text{Revocação} = \frac{\text{VP}}{\text{VP+FN}}$$

$$\text{F1-Score} = 2\frac{\text{Precisão} * \text{Revocação}}{\text{Precisão} + \text{Revocação}}$$

O primeiro experimento foi focado em responder a *PQ1* e avaliar a performance da acurácia de ambas os classificadores, em lote e em fluxo, quando atualização não são realizadas. Os classificadores foram treinados usando os primeiros 30 dias do mês de *janeiro* e avaliados com o restante da base de dados, ao longo do ano, sem realizar atualizações aos algoritmos. O objetivo é medir o impacto nas taxas de acerto causado pela mudança de comportamento da rede ao longo do tempo.

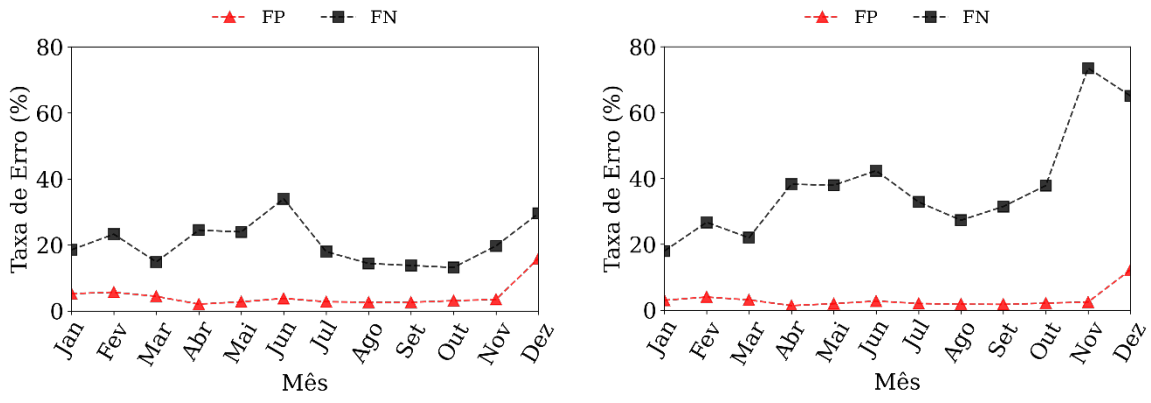
As Figuras Figura 3 e

Figura 4 mostram a performance da acurácia dos classificadores em lote e em fluxo, respectivamente. É possível notar um impacto significante na acurácia e o tempo de vida do classificador ao longo do tempo. Por exemplo, o classificador de fluxo HT aumenta sua taxa de FN em 4,7% no mês de fevereiro, somente um mês após o período de treinamento (Figura 3a). Junho foi o mês com pior performance, atingindo 34% na taxa de FN, o que é um acréscimo de 15,4% quando comparado com as taxas obtidas em janeiro. No tocante aos classificadores em lote, eles também são afetados pela mudança no tráfego da rede. Neste caso, considerando o classificador RF (

Figura 4a), ele aumenta sua taxa de FN no mês de fevereiro para 11%. Ainda, ele apresenta pior performance no mês de novembro, atingindo 53% na taxa de FN. Em contrapartida, as taxas de FP não são significativamente afetadas no decorrer do tempo, porém as altas taxas de FN demandam que atualizações sejam feitas. Independente do esquema de classificação (em lotes ou em fluxo) utilizada, as mudanças do comportamento da rede afetam significativamente a acurácia se nenhuma atualização é realizada, mesmo quando um conjunto de classificadores é utilizado (Figuras Figura 3d e

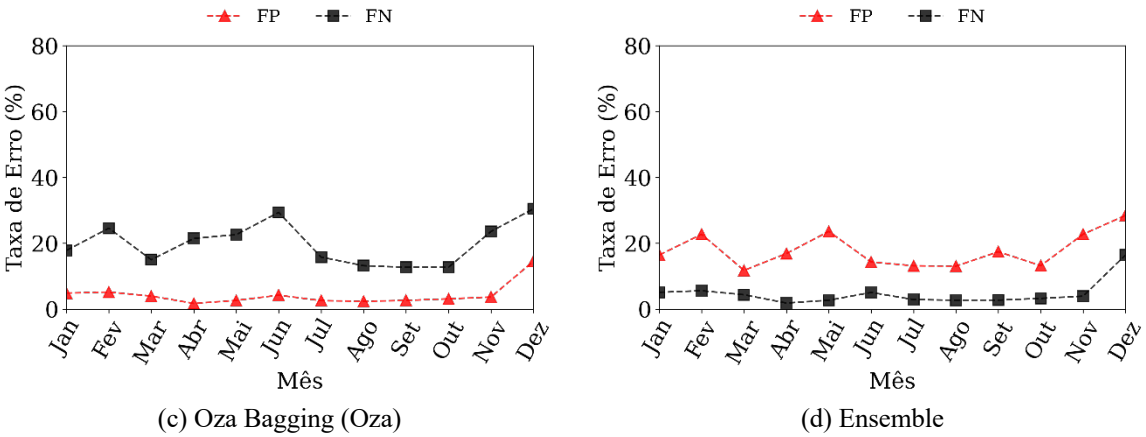
Figura 4d).

Figura 3. Performance da acurácia dos classificadores de *fluxo* avaliados sem que sejam realizadas atualizações periódicas aos algoritmos.



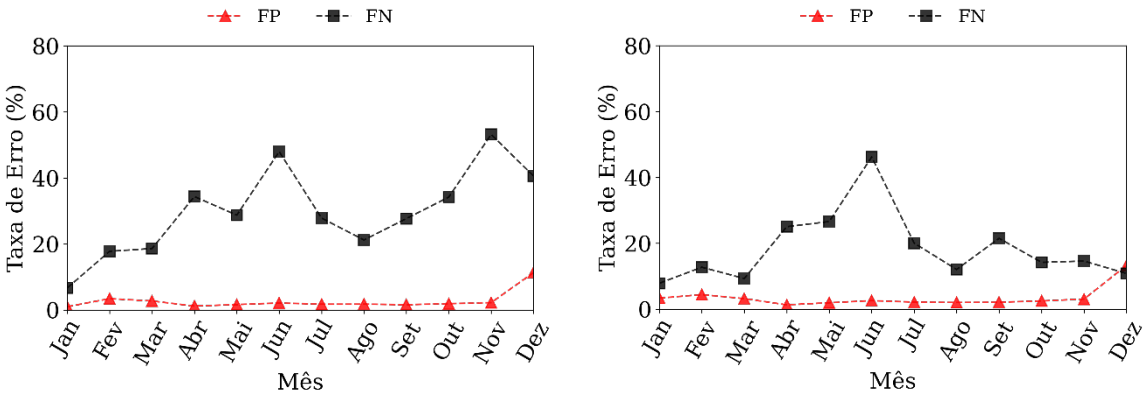
(a) Hoeffding Tree (HT)

(b) Leveraging Bag (Bag)

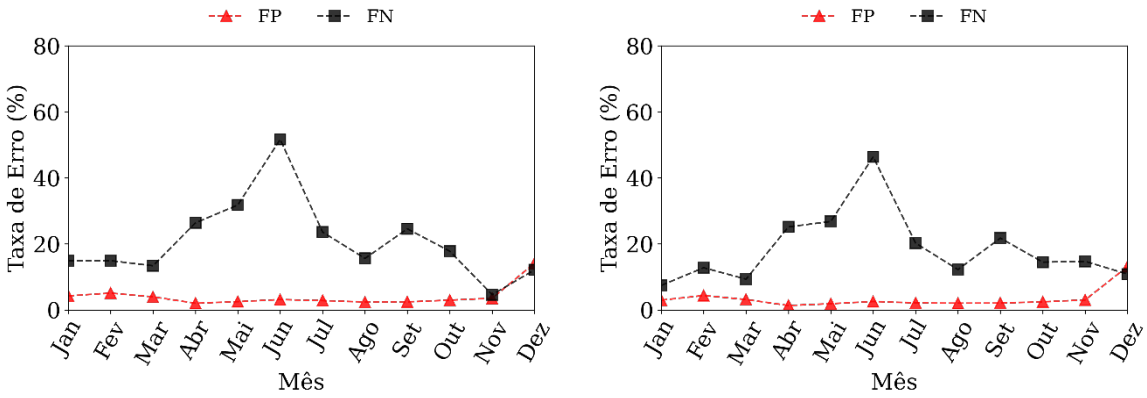


(c) Oza Bagging (Oza) (d) Ensemble

Figura 4. Performance da acurácia dos classificadores de *lotes* avaliados sem que sejam realizadas atualizações periódicas aos algoritmos.



(a) Random Forest (RF) (b) Gradient Boosting (GBT)



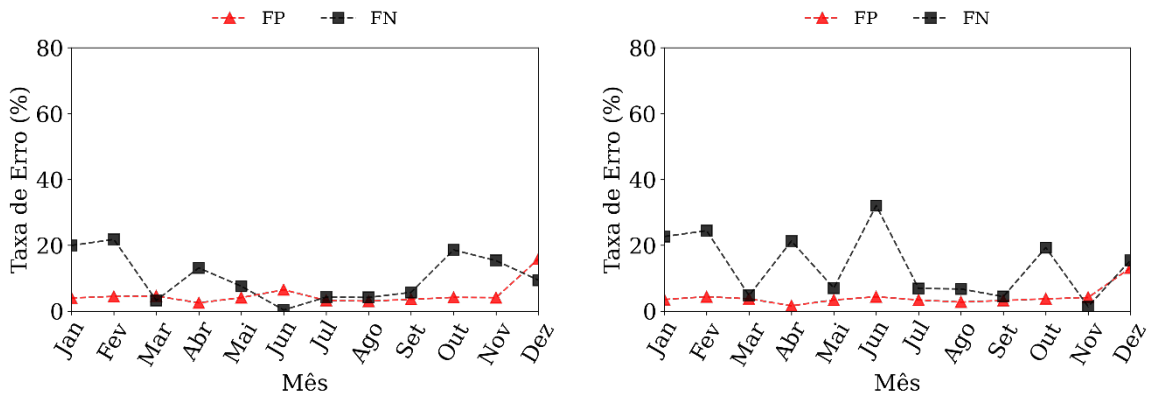
(c) AdaBoosting (Ada) (d) Ensemble

O segundo experimento é voltado em responder a *PQ2* e avaliar a performance das taxas de acerto quando performando atualizações periódicas ao modelo. Tais

atualizações são feitas mensalmente, usando os dados com um mês de atraso. Por exemplo, em primeiro de fevereiro, o modelo de AM subjacente é atualizado com os dados que ocorreram nos últimos 30 dias (1º de janeiro até 31º de janeiro). Ambas as classificações em lote e em fluxo foram treinadas do começo. Assim, os classificadores de aprendizagem de fluxo não são incrementalmente atualizados, como é comumente assumido em estudos relacionados. Em outras palavras, a avaliação aborda as mudanças na rede comportamento do tráfego através de atualizações mensais do modelo, uma suposição que ainda precisa ser avaliada na literatura.

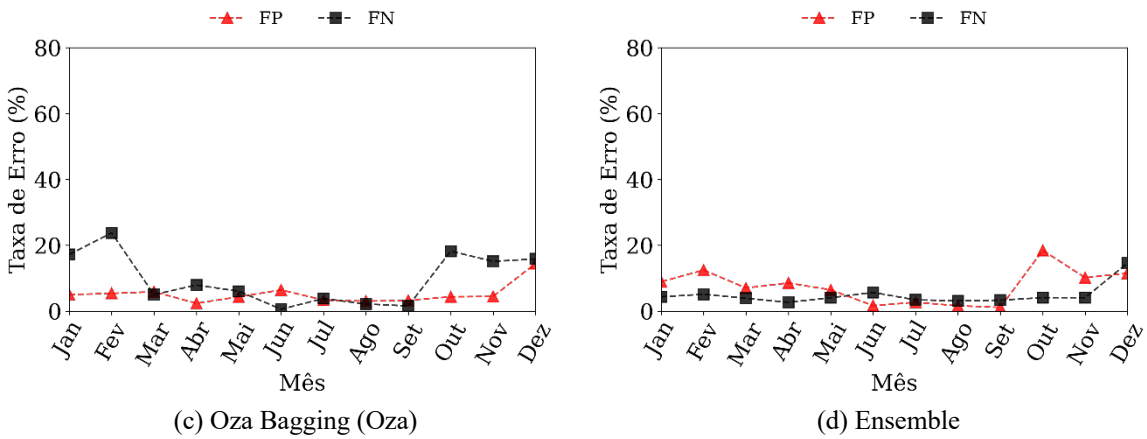
As Figuras Figura 5 e Figura 6 mostram a performance com o passar do tempo quando são feitas atualizações mensais, para as classificações em lotes e em fluxos, respectivamente. Neste caso, a maioria dos classificadores avaliados foram capazes de atingir altas taxas de acerto, ressaltando que atualizações periódicas podem ser utilizadas para tratar alterações do comportamento da rede. Por exemplo, o conjunto de classificadores de fluxo (Figura 5d) apresentaram uma média de FP de somente 7,5%, enquanto sua versão que não foi atualizada (Figura 3d) periodicamente apresenta uma média de 18%. As piores taxas de acurácias não variam significativamente quando as atualizações são aplicadas, mostrando que as mudanças no tráfego podem ser resolvidas com atualizações frequentes.

Figura 5. Performance da acurácia dos classificadores de *fluxo* avaliados quando atualizações mensais são realizadas.



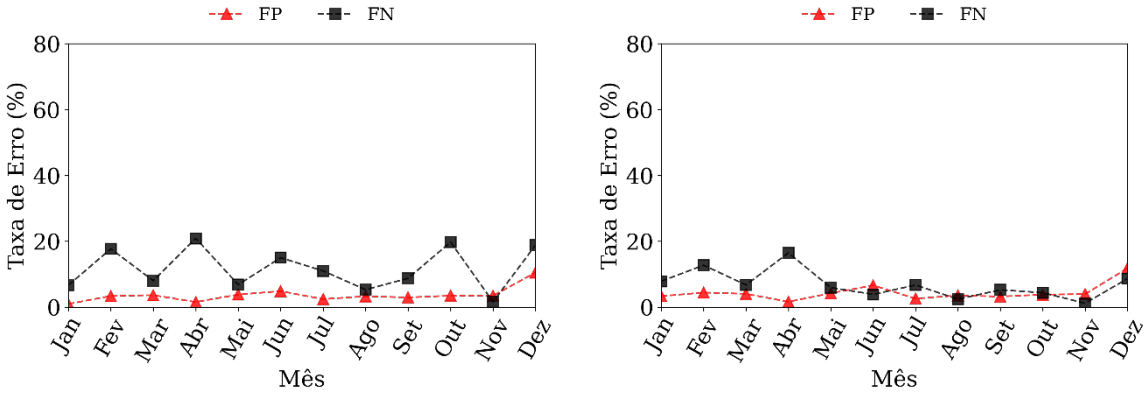
(a) Hoeffding Tree (HT)

(b) Leveraging Bag (Bag)

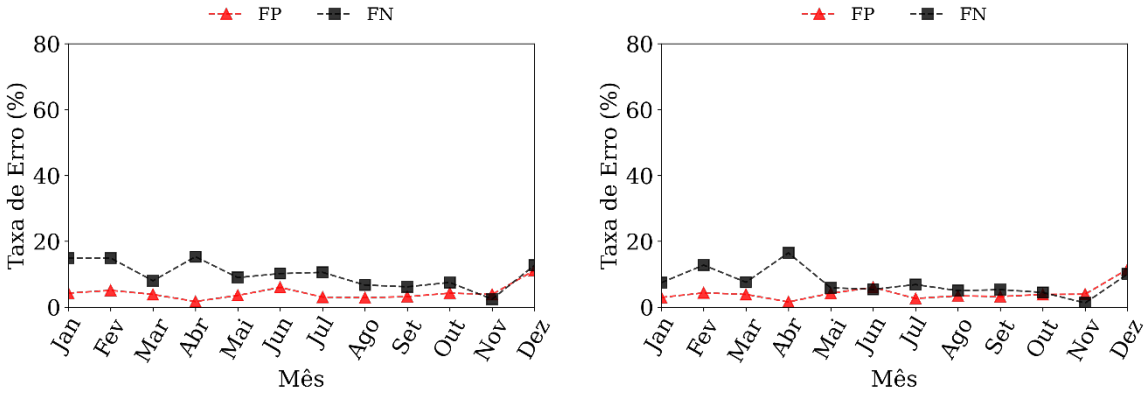


(c) Oza Bagging (Oza) (d) Ensemble

Figura 6. Performance da acurácia dos classificadores de *lotes* avaliados quando atualizações mensais são realizadas.



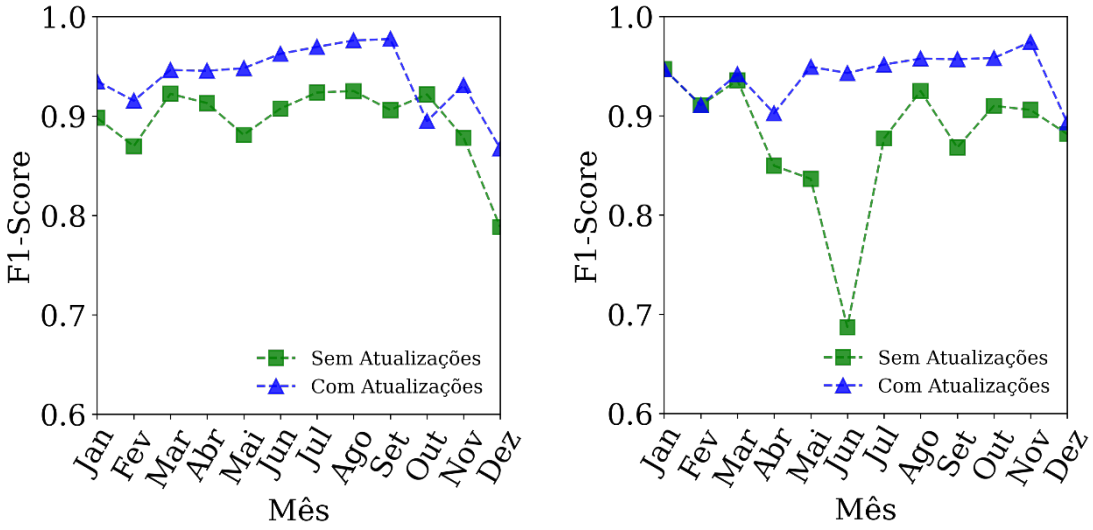
(a) Random Forest (HT) (b) Gradient Boosting (Bag)



(c) AdaBoosting (Oza) (d) Ensemble

Já a Figura 7 compara o F1-Score das abordagens com e sem atualizações periódicas, ambas em classificadores em lote e em fluxo. É possível notar uma melhora significativa no F1 quando atualizações são efetuadas, aumentando em até 0,09 e 0,24 para aprendizagem em fluxo e em lote, respectivamente. Desta forma, *NIDS* baseados em AM demandam que atualizações periódicas sejam realizadas para manter as taxas de acerto altas ao longo do tempo.

Figura 7. Comparação entre o F1-Score para aprendizagem por fluxo e em lotes para os conjuntos de classificadores, sem e com atualizações mensais.



(a) Aprendizagem por Fluxo

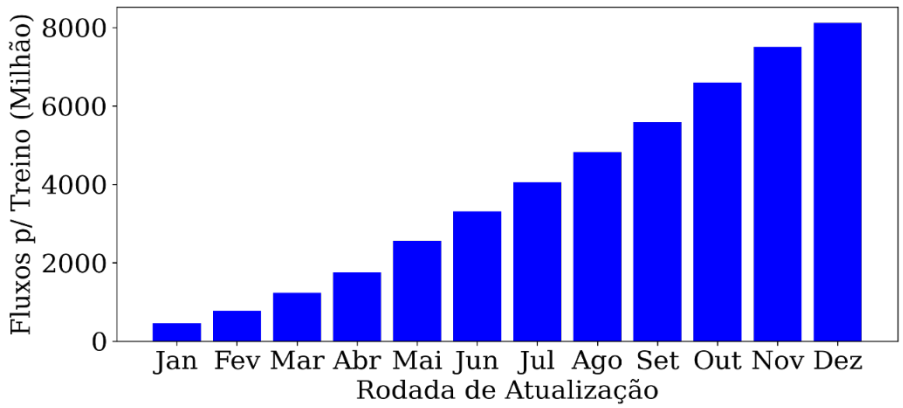
(b) Aprendizagem em Lotes

Além disso, é investigado o desafio de se efetuar atualizações nas técnicas avaliadas. Por exemplo, a Figura 8 mostra a quantidade cumulativa de instâncias demandadas pelos esquemas avaliados durante a etapa de atualização. É possível de notar que, ao passo que o tempo evolui, e mais rodadas de atualizações são efetuadas, a quantidade de instâncias que deveriam ser rotuladas também aumenta. Na prática, considerando atualizações mensais com os últimos 30 dias de dados para o treinamento, todos os eventos da rede devem ser rotulados. No entanto, a tarefa de rotulagem de eventos de rede é uma tarefa desafiadora e inviável para ser aplicada em todos os dados coletados, principalmente considerando um cenário de uma rede de alta velocidade, que gerar uma vasta quantidade de eventos em uma pequena parcela de tempo. Portanto, apesar das atualizações periódicas manterem as altas taxas de acurácia, a quantidade de

fluxos a serem rotulados no decorrer do tempo torna a tarefa inviável em um cenário do mundo real.

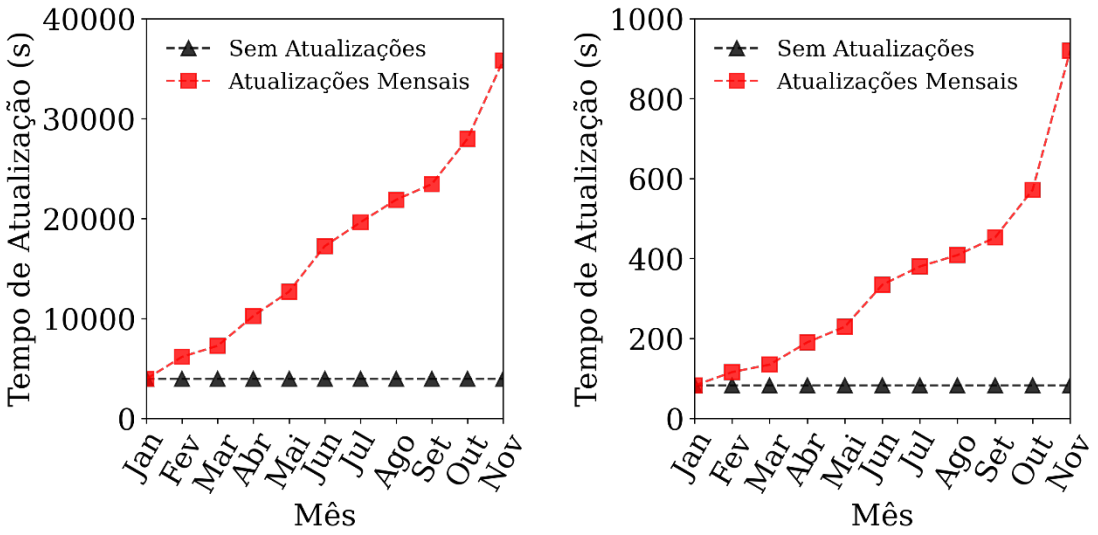
Finalmente, para responder a *PQ3*, é medido o custo computacional quando as atualizações mensais são feitas. Para atingir tal objetivo, foi comparado o tempo de processamento cumulativo dos classificadores em *ensemble* em duas situações: quando atualizações são realizadas e quando não são. Os experimentos foram realizados em uma

Figura 8. Quantidade cumulativa de instâncias que deveria ser rotulada ao longo do tempo quando são aplicadas atualizações mensais. A alta quantidade de eventos que necessitam de um rótulo durante a tarefa de atualização apresenta um desafio significante para abordagens tradicionais. Além disso, a rotulagem *baseada em assinatura* pode ser utilizada somente para um subconjunto dos eventos da rede.



máquina com hardware de *commodity* e para o custo computacional foi considerado o tempo de uso da CPU para todos os núcleos, explorando a capacidade de *multithreading* da API de AM. A Figura 9 mostra o tempo cumulativo dos classificadores *ensemble*. Atualizações periódicas do esquema demandam uma média adicional de tempo de processamento de 83 e 3.254 segundos para o aprendizado em lote e em fluxo, respectivamente. Recordando que cada técnica AM avaliada é implementada em APIs específicas e que o tempo de processamento depende da implementação subjacente da biblioteca de AM. Ademais, devido ao descarte do modelo desatualizado nos classificadores em lote, eles demandam mais processamento ao longo do tempo e mais fluxos de rede serão utilizados durante a etapa de treinamento.

Figura 9. Custo computacional cumulativo das abordagens *ensemble* com relação à tarefa de atualização.



(a) Aprendizagem por Fluxo

(b) Aprendizagem em Lotes

5.2.1. Discussão

O presente capítulo apresentou o impacto causado por mudanças naturais no tráfego da rede ao longo do tempo, seja em termos de taxa de acerto quanto por custo computacional, quando atualizações não são efetuadas. Para tratar tal problema, foi pressuposto que as atualizações são aplicadas periodicamente. Os experimentos mostraram que tais atualizações podem, de fato, serem utilizadas para manter as taxas de acerto de esquemas de detecção de intrusão, porém, o desafio relacionado com a tarefa de atualização para manter a confiabilidade do sistema torna as a tarefa inviável em ambientes de produção. Mais especificamente, a quantidade de instâncias que devem ser rotuladas durante a tarefa de atualização não pode ser providenciada em tal cenário. Não obstante, apesar de as atualizações poderem ser realizadas *offline*, o aumento do tempo de processamento no tocante à frequente execução das atualizações (Figura 9) pode também aumentar as dificuldades relacionadas a tal. Mais adiante é investigado como tais mudanças afetam as taxas de acerto da proposta de detecção de intrusão e como atualizações periódicas podem tratar tal comportamento.

5.3. Detecção de Intrusão com AM por Fluxo

Esta seção busca responder as seguintes Perguntas de Pesquisa (PQ):

- ***(PQ4***) O modelo de classificação proposto é capaz de melhorar as taxas de acurácia de classificação?
- (***PQ5***) Como o modelo se comporta quando não são realizadas atualizações periódicas?
- (***PQ6***) Como o modelo se comporta quando atualizações periódicas são efetuadas?
- (***PQ7***) Como o atraso na atualização pode afetar as taxas de acurácia de classificação ao longo do tempo?
- (***PQ8***) Quais são os custos de tempo e armazenamento da proposta do trabalho?

O modelo proposto foi implementado utilizando o mesmo conjunto de classificadores de fluxo avaliados anteriormente (Seção 5.2). Portanto, o modelo proposto depende em um *pool* de classificadores de fluxo que incluem os algoritmos HT, Bag e Oza, de maneira idêntica à avaliada na seção anterior (Seção 5.2). Ainda, os algoritmos utilizados também foram implementados sobre a API scikit-multiflow v.0.5.3, com o mesmo conjunto de parâmetros. Devido à alta desproporcionalidade entre as classes, tendo em vista que a maioria das instâncias são da classe *normal*, classe majoritária, uma subamostragem aleatória sem reposição foi aplicada em cada dia da base de dados na etapa de pré-processamento.

O primeiro experimento objetiva responder a PQ4 e avaliar como a abordagem proposta de classificação auxilia na melhora das taxas de acerto. O conjunto de classificadores de fluxo da proposta do trabalho foi treinado com os dados do mês de janeiro. Também foi avaliada a troca entre as taxas de erro e rejeição no conjunto de dados de teste, no mês de fevereiro. Tal troca é estabelecida através da abordagem de relação-classe-limiar (FUMERA; ROLI; GIACINTO, 2000). Portanto, cada classe considerada, seja *normal* e *ataque*, admitem um limiar de aceitação específico para cada classificador.

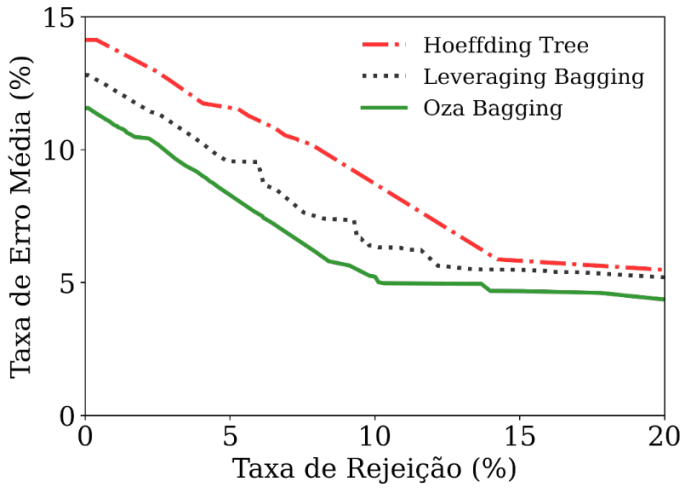
Os valores de confiança de classificação são independentes do algoritmo de classificação e foram obtidos através da função *predict_proba* da API scikit-multiflow. O objetivo é medir quando a classificação pode ser utilizada para melhorar as taxas de acerto, mesmo em um cenário onde o modelo AM subjacente esteja desatualizado.

A Figura 10 mostra a curva de Pareto entre as taxas de erro e rejeição para cada classificador no mês de fevereiro, considerando somente o conjunto ótimo de pontos de operação para cada um. Desta forma, a taxa de erro médio foi medida através das médias das taxas de FP e FN.

A taxa de rejeição foi medida de acordo com a taxa de instâncias onde o modelo proposto suprimiu o resultado da classificação. É notável que a avaliação da classificação através dos valores de confiança pode ser utilizada para melhorar as taxas de acerto do sistema, mesmo quando algoritmos de AM desatualizados são usados. Por exemplo, o classificador OzaBagging diminuiu sua taxa de erro de 12% para 5%, quando rejeitando somente 10% das instâncias. Portanto, a abordagem de avaliar as classificações pode ser utilizada para manter a acurácia do sistema, enquanto realizando atualizações periódicas.

Com relação a PQ5, o modelo foi avaliado quando nenhuma atualização é realizada, ou seja, rejeitando somente potenciais classificações erradas como definido no esquema de verificação. O objetivo da avaliação é entender como o modelo proposto pode ser aplicado, mesmo se desatualizado, através da avaliação de instâncias que foram classificadas com alta confiança. O limiar de aceitação de cada classificador é definido em uma taxa de erro de 10% (Figura 10) e a tarefa de detecção de intrusão, sem atualizações mensais, é realizada ao longo do ano. É importante ressaltar que o ponto de operação deve ser estabelecido de acordo com a necessidade do administrador da rede.

Figura 10. Troca entre a taxas de erro média e rejeição para cada classificador de fluxo, considerando os dados de fevereiro.

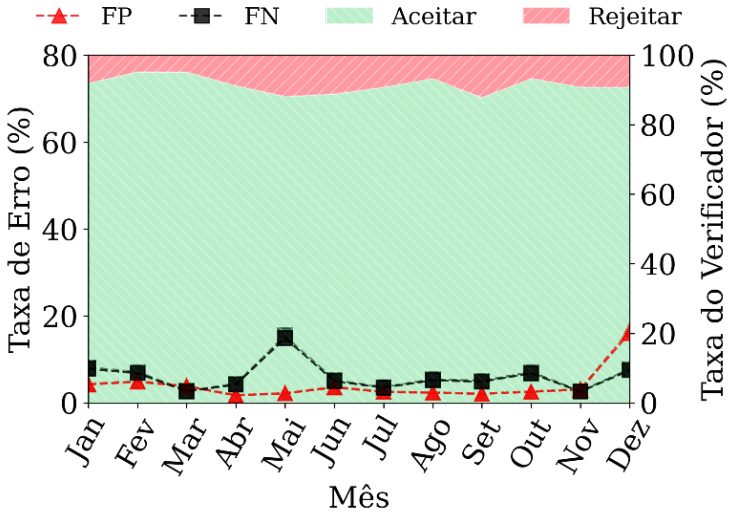


Apesar de uma alta taxa de rejeição, a proposta é capaz de fornecer altas taxas de acurácia ao longo do tempo, um número maior de eventos da rede é rejeitado, enquanto

uma baixa taxa de rejeição aceita mais eventos na rede, porém mais suscetível a produzir maiores taxas de erro. Rememorando que o esquema proposto no trabalho consiste em um conjunto de classificadores de fluxo (HT, Oza e Bag), classificando instâncias de acordo com o sistema de voto majoritário. Portanto, o modelo rejeita uma dada instância se todos os classificadores a rejeitarem, e só aceita se a maioria os classificadores aceitaram durante o processo de voto majoritário.

A Figura 11 mostra a performance da acurácia e rejeição do modelo proposto quando não são realizadas atualizações periódicas.

Figura 11. Taxas de erro e de rejeição do esquema proposto ao longo do tempo quando não são realizadas atualizações periódicas.



É notável que o modelo é capaz de manter a acurácia do sistema ao longo do tempo em comparação a janeiro, mesmo quando os classificadores subjacentes estão desatualizados. Mais especificamente, o modelo do trabalho mantém a acurácia do sistema para a maioria dos meses, apesar da taxa de rejeição, mesmo que não sejam aplicadas atualizações e usando somente a técnica de avaliação, melhorando a taxa de FP em 12% quando comparado com a técnica tradicional sem rejeição (Figura 11 vs Figura 3d), enquanto rejeitando, em média, 8,5% dos eventos.

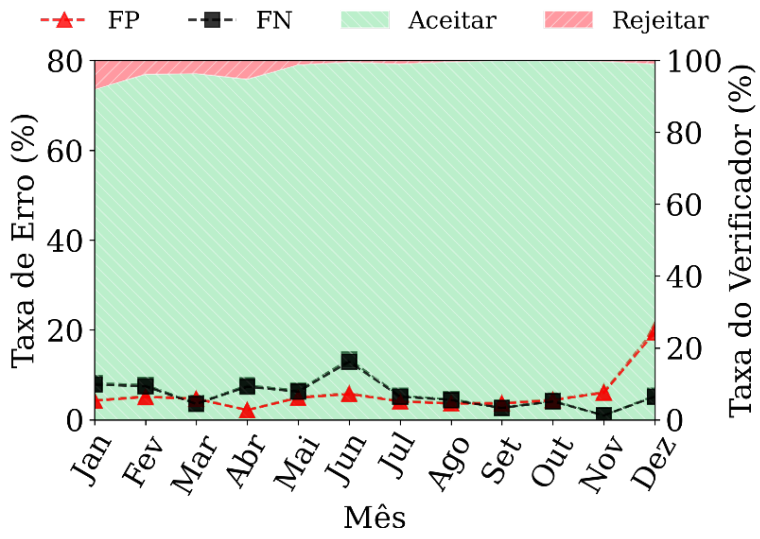
A abordagem de avaliação da classificação pode ser usada para manter a acurácia do sistema ao longo do tempo enquanto um modelo atualizado ainda não é providenciado ou até melhorar a vida útil da detecção de intrusão se nenhuma atualização é planejada.

Para responder a PQ6, foi avaliado a performance do modelo proposto, porém com atualizações periódicas e utilizando as instâncias rejeitadas e os mesmos pontos de operação da etapa anterior foram utilizados para a rejeição.

As instâncias rejeitadas foram armazenadas e usadas em um modelo de atualização incremental após 30 dias desde sua rejeição, enquanto a tarefa de atualização é executada mensalmente. Por exemplo, no dia 28 de fevereiro, o *pool* de classificadores de fluxo é atualizado com as instâncias que foram rejeitadas pelo modelo no período de 1º de janeiro até 31º de janeiro. As instâncias foram rejeitadas um mês antes das atualizações do modelo serem feitas. Por fim, a janela de tempo de armazenamento pode ser definida de acordo com as necessidades do administrador da rede, considerando que a técnica de providenciamento de rótulos é utilizada por tal.

A Figura 12 mostra a acurácia e taxas de rejeição do modelo proposto quando são efetuadas as atualizações mensais com as instâncias que foram rejeitadas nos últimos 30 dias. Neste caso, o modelo proposto pode significativamente diminuir as taxas de erro enquanto reduzindo a quantidade de instâncias rejeitadas ao longo do tempo, quando comparado com sua versão sem atualizações. Mais especificamente, em média, atualizações ao modelo melhoraram a taxa de FP em 2% enquanto rejeita 2% das instâncias, portanto reduzindo a taxa de erro em 6,5% quando comparado com sua versão sem atualizações (Figura 11 vs Figura 12). Desta forma, o modelo proposto é capaz de manter as acurácias de detecção de intrusão ao longo do tempo, enquanto facilita o processo de atualização do modelo quando necessário.

Figura 12. Taxas de erro e de rejeição do esquema proposto ao longo do tempo quando atualizações são realizadas com instâncias rejeitadas nos últimos 30 dias.



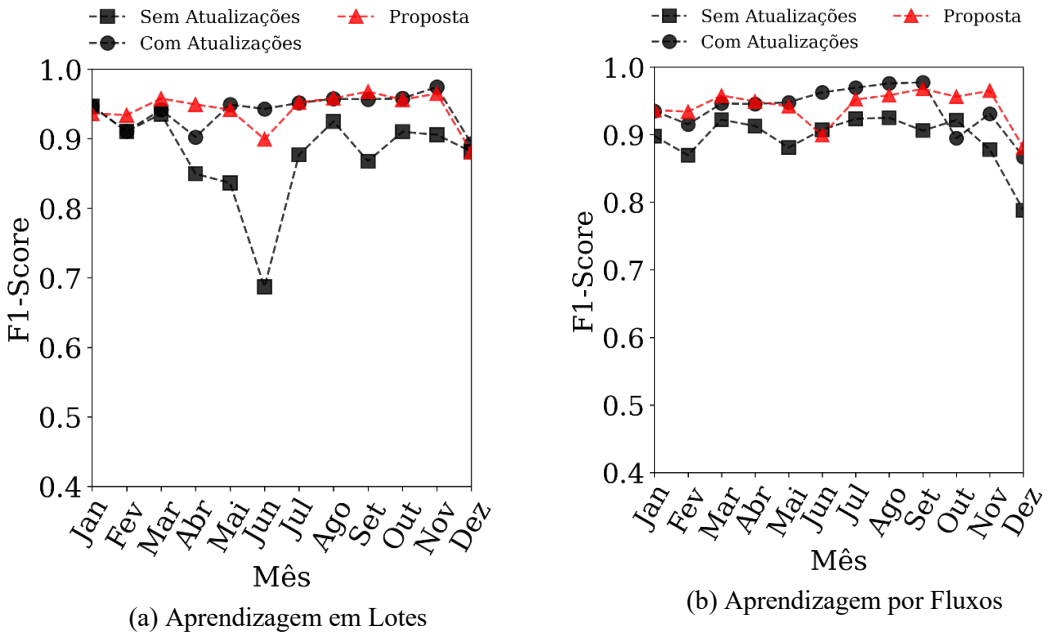
A acurácia também é investigada quando as atualizações periódicas são realizadas utilizando as instâncias rejeitadas nos últimos 30 dias. A

Figura 13 compara a performance da acurácia em relação as abordagens tradicionais, ambos de classificação em lote e em fluxo. O esquema proposto no trabalho, com

(a) Aprendizagem em Lotes (b) Aprendizagem por Fluxos

atualizações mensais, apresentou menores taxas de erro quando comparadas com as técnicas sem atualizações e com atualizações. Mais especificamente, a proposta quando comparada com a abordagem *ensemble* para a classificação- em lotes, foi capaz de melhorar o F1 Score com uma média de 0,08 e 0,03 quando comparada com as versões sem e com atualizações, respectivamente, que representa uma melhora de 5,7% e 0,6% em média na taxa de erro.

Figura 13. Comparação do F1-Score entre o modelo proposto e o conjunto de classificadores de ambas as técnicas tradicionais de aprendizagem: em lotes e por fluxos.



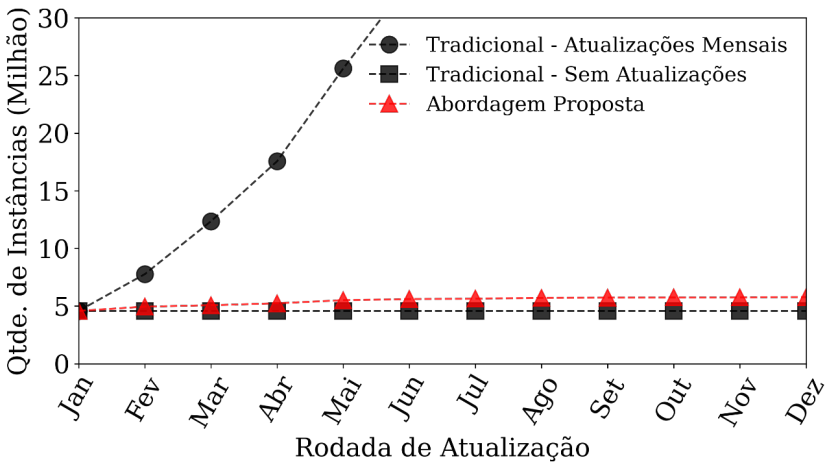
(a) Aprendizagem em Lotes (b) Aprendizagem por Fluxos

Em contraste com a abordagem de classificação em fluxo, a proposta melhorou o F1 Score em uma média de 0,06 e 0,03 comparada com as versões sem e com atualizações, portanto, também melhorando, respectivamente, em média a taxa de erro em até 6,1% e 1%, respectivamente. Lembrando que uma melhora no F1 Score foi atingida rejeitando uma média de 2% das instâncias.

Não obstante, apesar da melhora da acurácia do modelo proposto, foi possível facilitar a tarefa de atualização de maneira significante. A Figura 14 mostra a quantidade de instâncias necessárias pela proposta quando comparada com ambas as abordagens sem

e com atualizações mensais. O modelo proposto aumentou as taxas de acurácia enquanto demandava em média 2,2% das instâncias rotuladas da abordagem tradicional. Ainda, quando comparado as técnicas sem atualização, o modelo proposto incorreu somente em 2% de instâncias adicionais para ser providenciada ao decorrer do tempo. Portanto, apesar das melhorias na acurácia na proposta, como contribuição principal, a tarefa de atualização do modelo foi significativamente viabilizada já que são necessárias baixíssimas quantidades de instâncias rotuladas para serem providenciadas ao longo do

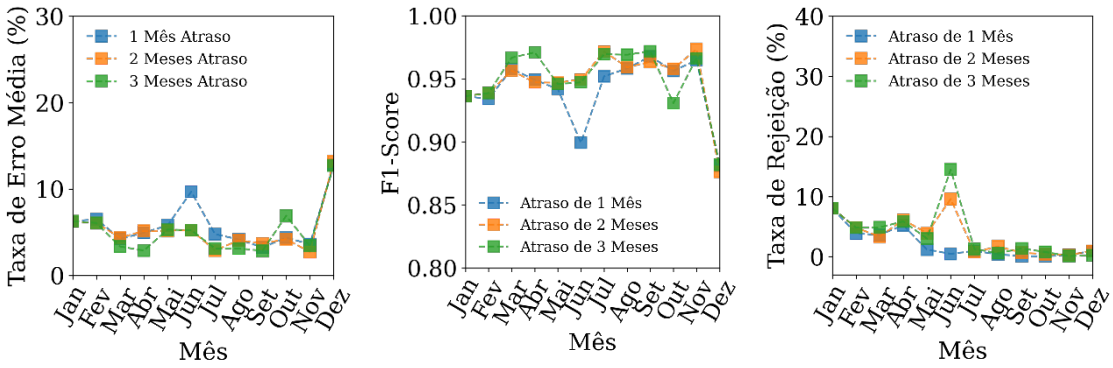
Figura 14. Quantidade cumulativa de instâncias necessárias para atualização, comparando o esquema proposto com as técnicas tradicionais.



tempo. Também, as instâncias usadas para atualização podem ser facilmente rotuladas quando considerando o grande intervalo de armazenamento durante as atualizações.

Para responder à questão PQ7 foi investigado como aumentar o atraso das atualizações, ainda utilizando as instâncias rejeitadas, sem que haja um impacto negativo na acurácia ou rejeição. Como avaliado anteriormente, o modelo foi capaz de permanecer um longo período sem atualizações periódicas sem afetar a acurácia do sistema. Portanto, mais adiante é investigado o efeito de aumentar os intervalos de armazenamento das instâncias rejeitadas em ambas as taxas de acurácia e rejeição. A Figura 15 mostra a taxa de erro média (Figura 15a), F1-Score (Figura 15b) e taxa de rejeição (Figura 15c) quando as instâncias rejeitadas são armazenadas por períodos mais extensos antes de serem usadas para atualização. É notável que maiores intervalos no armazenamento não prejudicam significativamente a acurácia ou a taxa de erro ao longo do tempo. Por exemplo, ao aumentar tal intervalo de 30 dias para 90 dias, a taxa de erro tem um acréscimo de somente 1,7%, aumentando a taxa de erro em uma média de 0,6%. Como resultado, os administradores da rede podem armazenar as instâncias rejeitadas por mais

tempo se o seu provedor de rótulos demandar, portanto facilitando as atualizações periódicas sem prejudicar significativamente a acurácia e as taxas de rejeição.



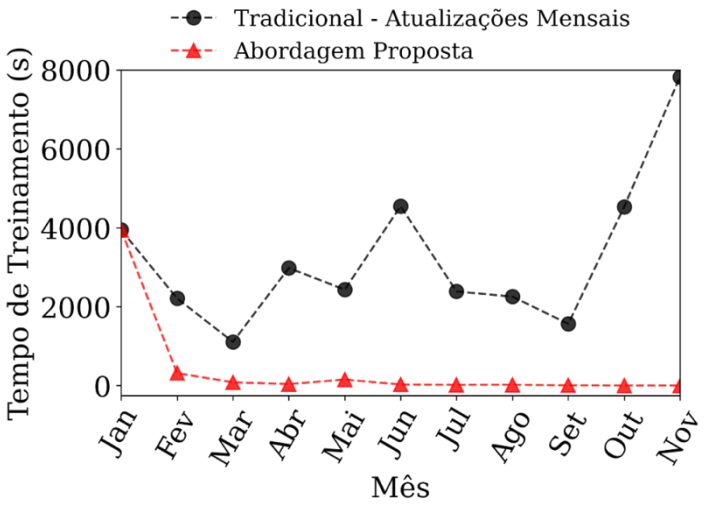
(a) Taxa de Erro (b) F1-Score (c) Taxa de Rejeição

Figura 15. Taxa média de erro, F1 Score e taxa de rejeição ao longo do tempo, enquanto variando o intervalo de armazenamento das instâncias rejeitadas.

Finalmente, é respondida a PQ8 e avaliado os custos computacionais da proposta quando comparado com as abordagens tradicionais ensemble de classificação de fluxo quando atualizações mensais são realizadas (Figura 12 vs Figura 5d). Relembrando que os experimentos foram executados em uma máquina com hardware de *commodity*, e os custos computacionais foram calculados de acordo com a soma do tempo de uso da CPU de todos os núcleos disponíveis. A Figura 16 mostra o custo computacional da proposta quando atualizações periódicas são aplicadas utilizando as instâncias rejeitadas no período de 30 dias. É possível observar que o modelo, quando comparado com à abordagem tradicional com atualizações mensais, demanda somente 3,2% do tempo computacional.

Figura 16. Custo computacional das atualizações do modelo proposto contra a abordagem tradicional de aprendizagem por fluxo.

Por fim, o modelo proposto pode significativamente melhorar a tarefa de atualização,



a vida útil e reduzir a quantidade de instâncias necessárias para serem providenciadas, ainda demandando menos recursos computacionais durante as atualizações. Por exemplo, considerando um NIDS baseado em AM implantado em um ambiente de produção para a classificação da base MAWIFlow.

Conforme a Figura 16, durante o período de 11 meses, com atualizações mensais do modelo, a proposta demandaria um total de 4,6 mil segundos de treinamento, enquanto a técnica tradicional demandaria 35,7 mil segundos de custo computacional, um aumento de 7,67 vezes. Ainda, supondo que cada instância demanda 264 bytes de armazenamento, o modelo demandaria, em média, mensalmente, uma capacidade de armazenamento de 133GB.

Por outro lado, a abordagem tradicional demandaria, mensalmente, aproximadamente 958GB, um aumento de 7,2 vezes. Portanto, o modelo proposto no trabalho facilita a atualização do modelo, levando em conta o período de armazenamento das instâncias rejeitadas enquanto o rótulo se torna publicamente disponível e significativamente reduz o custo computacional no que tange ao tempo de processamento e capacidade de armazenamento das atualizações do modelo.

Capítulo 6
Conclusão

6.1. Considerações Finais

De forma geral, a literatura científica negligencia os danos causados pelas mudanças no comportamento do tráfego de rede em seus esquemas, mesmo que atualizações periódicas sejam necessárias na prática. No entanto, a tarefa de atualização é custosa para os Sistemas de Detecção de Intrusão em Redes (NIDS), principalmente devido à grande quantidade de tráfego de rede que deve ser avaliado, rotulado e utilizado durante o treinamento.

Este trabalho propõe uma abordagem que torna a tarefa de atualização viável por meio de um conjunto de classificadores de fluxo, onde ela permite a avaliação e atualização periódica com atraso das classificações. Os algoritmos de aprendizado por fluxo facilitam a incorporação do novo comportamento do tráfego de rede, enquanto a avaliação das classificações fornece resultados mais confiáveis, mesmo quando os algoritmos subjacentes estão desatualizados.

Como resultado, mesmo sem a necessidade de atualizações periódicas, o modelo proposto demonstrou a capacidade de manter sua acurácia ao longo do tempo e até mesmo melhorar suas taxas de Falso Positivo (FP) em até 12% em comparação com a abordagem tradicional de classificação baseada em aprendizado por fluxo. A utilização de atualizações com atraso permitiu uma tarefa de rotulagem mais eficiente do novo tráfego de rede, especialmente após a disponibilização pública de um ataque.

Portanto, o esquema proposto é capaz de realizar atualizações mensais sem intervenção humana, considerando um atraso de três meses antes que uma atualização seja efetuada e os eventos sejam rotulados corretamente, sem causar um impacto significativo na acurácia do sistema. Além disso, o modelo proposto demandou apenas 3,2% dos custos computacionais, de tempo e armazenamento, e 2% das novas instâncias a serem rotuladas ao longo do tempo, o que torna as atualizações em um NIDS uma tarefa viável e de baixo custo.

6.2. Publicações e Prêmios

O presente trabalho resultou em um total de nove publicações, sendo que quatro delas estão diretamente relacionadas ao tópico central do trabalho, enquanto as demais abrangem áreas correlatas. A Tabela 5 apresenta os títulos das publicações, os locais de publicação e a lista de autores. Vale ressaltar que os trabalhos estão ordenados de forma que os relacionados diretamente ao presente trabalho são listados primeiramente, seguidos pelos relacionados a áreas correlatas.

Além das publicações realizadas durante o período de desenvolvimento do trabalho, é importante destacar que houve a conquista do prêmio de melhor apresentação oral na área de ciências exatas durante o XXIX Seminário de Iniciação Científica da PUCPR.

Tabela 5. Lista de publicações realizadas durante o trabalho, para trabalhos de áreas diretamente relacionadas ou correlatas.

Título	***Qualis***	***Local de Publicação***	***Autores***
Toward feasible machine learning model updates in network-based intrusion detection	Q1	Periódico - Computer Networks	**Pedro Horchulhack**, Eduardo K. Viegas, Altair O. Santin
Intrusion Detection Model Updates Through GAN Data Augmentation and Transfer Learning	A1	Conferência - IEEE Global Communications Conference	**Pedro Horchulhack**, Eduardo K. Viegas, Altair O. Santin, Jhonatan Geremias
Atualização de Modelo baseado em Aumento de Dados e Transferência de Aprendizagem para Detecção de Intrusão em Redes	A4	Conferência - XXII Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais	**Pedro Horchulhack**, Eduardo K. Viegas, Altair O. Santin, Jhonatan Geremias
A Stream Learning Intrusion Detection System for Concept Drifting Network Traffic	-	Conferência – 6th Cyber Security in Networking Conference	**Pedro Horchulhack**, Eduardo K. Viegas, Martin Andreoni Lopez
Detection of Service Provider Hardware Over-commitment in Container Orchestration Environments	A1	Conferência - IEEE Global Communications Conference	**Pedro Horchulhack**, Eduardo K. Viegas, Altair O. Santin

Detecção de Overbooking em Aplicações Baseadas em Docker Através de Aprendizagem de Máquina	A4	Conferência - XL Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos	**Pedro Horchulhack**, Eduardo K. Viegas, Altair O. Santin
Towards a Reliable Hierarchical Android Malware Detection Through Image-based CNN	A2	Conferência - Consumer Communications & Networking Conference	Jhonatan Geremias, Eduardo K Viegas, Altair O Santin, Alceu Britto, **Pedro Horchulhack**
Towards multi-view android malware detection through image-based deep learning	A2	Conferência - International Wireless Communications and Mobile Computing	Jhonatan Geremias, Eduardo K Viegas, Altair O Santin, Alceu Britto, **Pedro Horchulhack**
A machine learning model for detection of docker-based APP overbooking on kubernetes	A1	Conferência - IEEE International Conference on Communications	Felipe Ramos, Eduardo Viegas, Altair Santin, **Pedro Horchulhack**, Roger R. dos Santos, Allan Espindola

Referências Bibliográficas

ADHIKARI, Uttam; MORRIS, Thomas H.; PAN, Shengyi. Applying Hoeffding Adaptive Trees for Real-Time Cyber-Power Event and Intrusion Classification. **IEEE Transactions on Smart Grid**, v. 9, n. 5, p. 4049–4060, 2018.

ALSHAMMARI, Riyad; ZINCIR-HEYWOOD, A. Nur. The Impact of Evasion on the Generalization of Machine Learning Algorithms to Classify VoIP Traffic. *In*: **2012 21st International Conference on Computer Communications and Networks (ICCCN)**. Munich, Germany: IEEE, 2012, p. 1–8.

BILGE, Leyla; DUMITRAŞ, Tudor. Before we knew it: an empirical study of zero-day attacks in the real world. *In*: **Proceedings of the 2012 ACM conference on Computer and communications security**. Raleigh North Carolina USA: ACM, 2012, p. 833–844.

BLAISE, Agathe; BOUET, Mathieu; CONAN, Vania; *et al*. Detection of zero-day attacks: An unsupervised port-based approach. **Computer Networks**, v. 180, p. 107391, 2020.

BOSE, K. S.; SARMA, R. H. Delineation of the intimate details of the backbone conformation of pyridine nucleotide coenzymes in aqueous solution. **Biochemical and Biophysical Research Communications**, v. 66, n. 4, p. 1173–1179, 1975.

CASSALES, Guilherme Weigert; SENGER, Hermes; DE FARIA, Elaine Ribeiro; *et al*. IDSA-IoT: An Intrusion Detection System Architecture for IoT Networks. *In*: **2019 IEEE Symposium on Computers and Communications (ISCC)**. Barcelona, Spain: IEEE, 2019, p. 1–7.

CHICHE, Alebachew; MESHESHA, Million. Towards a Scalable and Adaptive Learning Approach for Network Intrusion Detection. **Journal of Computer Networks and Communications**, v. 2021, p. 1–9, 2021.

DAS, Saikat; SAHA, Sajal; PRIYOTI, Annita Tahsin; *et al*. Network Intrusion Detection and Comparative Analysis Using Ensemble Machine Learning and Feature Selection. **IEEE Transactions on Network and Service Management**, v. 19, n. 4, p. 4821–4833, 2022.

FONTUGNE, Romain; BORGNAT, Pierre; ABRY, Patrice; *et al*. MAWILab: combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. *In*: **Proceedings of the 6th International COnference**. Philadelphia Pennsylvania: ACM, 2010, p. 1–12.

FUMERA, Giorgio; ROLI, Fabio; GIACINTO, Giorgio. Reject option with multiple thresholds. **Pattern Recognition**, v. 33, n. 12, p. 2099–2101, 2000.

GAO, Xianwei; SHAN, Chun; HU, Changzhen; *et al*. An Adaptive Ensemble Machine Learning Model for Intrusion Detection. **IEEE Access**, v. 7, p. 82512–82521, 2019.

GATES, Carrie; TAYLOR, Carol. Challenging the anomaly detection paradigm: a provocative discussion. *In*: **Proceedings of the 2006 workshop on New security paradigms**. Germany: ACM, 2006, p. 21–29.

GU, Jie; WANG, Lihong; WANG, Huiwen; *et al*. A novel approach to intrusion detection using SVM ensemble with feature augmentation. **Computers & Security**, v. 86, p. 53–62, 2019.

HANCZAR, Blaise. Performance visualization spaces for classification with rejection option. **Pattern Recognition**, v. 96, p. 106984, 2019.

HORCHULHACK, Pedro; VIEGAS, Eduardo K.; LOPEZ, Martin Andreoni. A Stream Learning Intrusion Detection System for Concept Drifting Network Traffic. *In*: **2022 6th Cyber Security in Networking Conference (CSNet)**. Rio de Janeiro, Brazil: IEEE, 2022a, p. 1–7.

HORCHULHACK, Pedro; VIEGAS, Eduardo K.; SANTIN, Altair O.; *et al*. Intrusion Detection Model Updates Through GAN Data Augmentation and Transfer Learning. *In*: **GLOBECOM 2022 - 2022 IEEE Global Communications Conference**. Rio de Janeiro, Brazil: IEEE, 2022b, p. 2668–2673.

INJADAT, MohammadNoor; MOUBAYED, Abdallah; NASSIF, Ali Bou; *et al*. Multi-Stage Optimized Machine Learning Framework for Network Intrusion Detection. **IEEE Transactions on Network and Service Management**, v. 18, n. 2, p. 1803–1816, 2021.

KHRAISAT, Ansam; GONDAL, Iqbal; VAMPLEW, Peter; *et al*. Survey of intrusion detection systems: techniques, datasets and challenges. **Cybersecurity**, v. 2, n. 1, p. 20, 2019.

KILINCER, Ilhan Firat; ERTAM, Fatih; SENGUR, Abdulkadir. Machine learning methods for cyber security intrusion detection: Datasets and comparative study. **Computer Networks**, v. 188, p. 107840, 2021.

KRAWCZYK, Bartosz; MINKU, Leandro L.; GAMA, João; *et al*. Ensemble learning for data stream analysis: A survey. **Information Fusion**, v. 37, p. 132–156, 2017.

LI, Tian; SAHU, Anit Kumar; TALWALKAR, Ameet; *et al*. Federated Learning: Challenges, Methods, and Future Directions. **IEEE Signal Processing Magazine**, v. 37, n. 3, p. 50–60, 2020.

LIANG, Junwei; MA, Maode. Co-Maintained Database Based on Blockchain for IDSs: A Lifetime Learning Framework. **IEEE Transactions on Network and Service Management**, v. 18, n. 2, p. 1629–1645, 2021.

LIAO, Hung-Jen; RICHARD LIN, Chun-Hung; LIN, Ying-Chih; *et al*. Intrusion detection system: A comprehensive review. **Journal of Network and Computer Applications**, v. 36, n. 1, p. 16–24, 2013.

LIN, Dongyun; SUN, Lei; TOH, Kar-Ann; *et al*. Biomedical image classification based on a cascade of an SVM with a reject option and subspace analysis. **Computers in Biology and Medicine**, v. 96, p. 128–140, 2018.

LU, Jie; LIU, Anjin; DONG, Fan; *et al*. Learning under Concept Drift: A Review. **IEEE Transactions on Knowledge and Data Engineering**, p. 1–1, 2018.

MAGÁN-CARRIÓN, Roberto; URDA, Daniel; DÍAZ-CANO, Ignacio; *et al*. Towards a Reliable Comparison and Evaluation of Network Intrusion Detection Systems Based on Machine Learning Approaches. **Applied Sciences**, v. 10, n. 5, p. 1775, 2020.

MARINHO, Leandro B.; ALMEIDA, Jefferson S.; SOUZA, João Wellington M.; *et al*. A novel mobile robot localization approach based on topological maps using classification with reject option in omnidirectional images. **Expert Systems with Applications**, v. 72, p. 1–17, 2017.

MARTINDALE, Nathan; ISMAIL, Muhammad; TALBERT, Douglas A. Ensemble-Based Online Machine Learning Algorithms for Network Intrusion Detection Systems Using Streaming Data. **Information**, v. 11, n. 6, p. 315, 2020.

MAWI, Working Group. **MAWI Working Group traffic archive**. Disponível em: <https://mawi.wide.ad.jp/mawi>. Acesso em: 7 fev. de 2023.

MISHRA, Preeti; VARADHARAJAN, Vijay; TUPAKULA, Uday; *et al*. A Detailed Investigation and Analysis of Using Machine Learning Techniques for Intrusion Detection. **IEEE Communications Surveys & Tutorials**, v. 21, n. 1, p. 686–728, 2019.

MOLINA-CORONADO, Borja; MORI, Usue; MENDIBURU, Alexander; *et al*. Survey of Network Intrusion Detection Methods From the Perspective of the Knowledge Discovery in Databases Process. **IEEE Transactions on Network and Service Management**, v. 17, n. 4, p. 2451–2479, 2020.

MOORE, Andrew W.; ZUEV, Denis. Internet traffic classification using bayesian analysis techniques. *In*: **Proceedings of the 2005 ACM SIGMETRICS international conference on Measurement and modeling of computer systems**. Banff Alberta Canada: ACM, 2005, p. 50–60.

NGUYEN, Hai-Long; WOON, Yew-Kwong; NG, Wee-Keong. A survey on data stream clustering and classification. **Knowledge and Information Systems**, v. 45, n. 3, p. 535–569, 2015.

NISIOTI, Antonia; MYLONAS, Alexios; YOO, Paul D.; *et al*. From Intrusion Detection to Attacker Attribution: A Comprehensive Survey of Unsupervised Methods. **IEEE Communications Surveys & Tutorials**, v. 20, n. 4, p. 3369–3388, 2018.

OTOUM, Safa; KANTARCI, Burak; MOUFTAH, Hussein T. A Novel Ensemble Method for Advanced Intrusion Detection in Wireless Sensor Networks. *In*: **ICC 2020 - 2020 IEEE International Conference on Communications (ICC)**. Dublin, Ireland: IEEE, 2020, p. 1–6.

PU, Guo; WANG, Lijuan; SHEN, Jun; *et al*. A hybrid unsupervised clustering-based anomaly detection method. **Tsinghua Science and Technology**, v. 26, n. 2, p. 146–153, 2021.

SAFARA, Fatemeh; SOURI, Alireza; SERRIZADEH, Masoud. Improved intrusion detection method for communication networks using association rule mining and artificial neural networks. **IET Communications**, v. 14, n. 7, p. 1192–1197, 2020.

SAHANI, Nitasha; ZHU, Ruoxi; CHO, Jin-Hee; *et al*. Machine Learning-based Intrusion Detection for Smart Grid Computing: A Survey. **ACM Transactions on Cyber-Physical Systems**, v. 7, n. 2, p. 1–31, 2023.

SANGKATSANEE, Phurivit; WATTANAPONGSAKORN, Naruemon; CHARNSRIPINYO, Chalermpol. Practical real-time intrusion detection using machine learning approaches. **Computer Communications**, v. 34, n. 18, p. 2227–2235, 2011.

SCHNEIDER, Markus; ERTEL, Wolfgang; RAMOS, Fabio. Expected similarity estimation for large-scale batch and streaming anomaly detection. **Machine Learning**, v. 105, n. 3, p. 305–333, 2016.

SOMMER, Robin; PAXSON, Vern. Outside the Closed World: On Using Machine Learning for Network Intrusion Detection. *In*: **2010 IEEE Symposium on Security and Privacy**. Oakland, CA, USA: IEEE, 2010, p. 305–316.

TAHERI, Rahim; GHAHRAMANI, Meysam; JAVIDAN, Reza; *et al*. Similarity-based Android malware detection using Hamming distance of static binary features. **Future Generation Computer Systems**, v. 105, p. 230–247, 2020.

TAVALLAEE, Mahbod; STAKHANOVA, Natalia; GHORBANI, Ali Akbar. Toward Credible Evaluation of Anomaly-Based Intrusion-Detection Methods. **IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)**, v. 40, n. 5, p. 516–524, 2010.

UD DIN, Salah; SHAO, Junming; KUMAR, Jay; *et al*. Online reliable semi-supervised learning on evolving data streams. **Information Sciences**, v. 525, p. 153–171, 2020.

VAN ENGELEN, Jesper; HOOS, Holger. A survey on semi-supervised learning. **Machine Learning**, v. 109, n. 2, p. 373–440, 2020.

VIEGAS, Eduardo K.; SANTIN, Altair O.; COGO, Vinicius V.; *et al*. Facing the Unknown: A Stream Learning Intrusion Detection System for Reliable Model Updates. *In*: BAROLLI, Leonard; AMATO, Flora; MOSCATO, Francesco; *et al* (Orgs.). **Advanced Information Networking and Applications**. Cham: Springer International Publishing, 2020, v. 1151, p. 898–909. (Advances in Intelligent Systems and Computing).

VIEGAS, Eduardo K.; SANTIN, Altair O.; OLIVEIRA, Luiz S. Toward a reliable anomaly-based intrusion detection in real-world environments. **Computer Networks**, v. 127, p. 200–216, 2017.

VIEGAS, Eduardo; SANTIN, Altair; BESSANI, Alysson; *et al*. BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. **Future Generation Computer Systems**, v. 93, p. 473–485, 2019.

WANG, Hongzhi; BAH, Mohamed Jaward; HAMMAD, Mohamed. Progress in Outlier Detection Techniques: A Survey. **IEEE Access**, v. 7, p. 107964–108000, 2019.

WOLBERG, William; MANGASARIAN, Olvi; STREET, Nick; *et al*. Breast Cancer Wisconsin (Diagnostic).

WU, Hua; YU, Zhenhua; CHENG, Guang; *et al*. Identification of Encrypted Video Streaming Based on Differential Fingerprints. *In*: **IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)**. Toronto, ON, Canada: IEEE, 2020, p. 74–79.

ZHONG, Ying; CHEN, Wenqi; WANG, Zhiliang; *et al*. HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning. **Computer Networks**, v. 169, p. 107049, 2020.