Um modelo de detecção de intrusão baseado em *deep autoencoders* e *transfer learning*

Roger R. dos Santos[1], **Eduardo K. Viegas**[1], **Altair O. Santin**[1]

[1]Programa de Pós-Graduação em Informática (PPGIa)
Pontifícia Universidade Católica do Paraná (PUCPR)
80.215-901 - Curitiba - PR

{roger.robson, eduardo.viegas, santin}@ppgia.pucpr.br

Abstract. *Machine learning techniques for network-based intrusion detection often assumes that network traffic does not change over time, or that model updates can be easily performed. In this paper we propose a novel reminiscent intrusion detection model based on deep autoencoders and transfer learning to easiness the model update burden. Experiments carried out have shown that approaches in the literature are unable to deal with changes in network traffic over time. The proposed approach is able to decrease the false positive rate by up to 23.9%, and provide accuracy rates similar to traditional techniques, requiring only 22% of training data and 28% of computational costs.*

Resumo. *As técnicas de aprendizado de máquina para detecção de intrusão baseada na rede geralmente pressupõem que o tráfego da rede não muda com o tempo ou que as atualizações do modelo podem ser realizadas facilmente. Neste artigo, propomos um novo modelo de detecção de intrusão baseado em deep autoencoders e transfer learning para facilitar a atualização do modelo. Experimentos realizados mostraram que as abordagens na literatura são incapazes de lidar com mudanças de tráfego de rede ao longo do tempo. A abordagem proposta é capaz de melhorar a taxa de falso positivo em até 23,9%, e fornecer taxas de precisão semelhantes às técnicas tradicionais, exigindo apenas 22% de dados de treinamento e 28% dos custos computacionais.*

1. Introdução

Sistemas de detecção de intrusão baseados em rede (*Network-based Intrusion Detection Systems - NIDS*) têm sido amplamente usados para detectar ameaças de rede, através de abordagens *baseadas em assinatura* ou abordagens *baseadas em comportamento* [Molina-Coronado et al. 2020]. Abordagens *baseadas em assinatura* só conseguem identificar ataques conhecidos anteriormente durante o treinamento. Contudo, as abordagens *baseadas em comportamento* analisam o comportamento do evento para detecção de intrusão, portanto, esta técnica é capaz de detectar ataques através do comportamento esperado do ambiente [Gates and Taylor 2006]. Contudo, nas últimas décadas várias abordagens foram propostas para detecção de intrusão *baseada em comportamento* utilizando técnicas de aprendizagem de máquina através de reconhecimento de padrões produzindo resultados promissores [Molina-Coronado et al. 2020]. Neste caso, um modelo comportamental é criado através de um conjunto de dados de treinamento, normalmente composto por milhões de amostras rotuladas de eventos como *normal* e *ataque*

[Viegas et al. 2019]. O conjunto de dados deve conter um número representativo de amostras do ambiente de produção, levando em consideração que o modelo será construído contendo o comportamento do ambiente. O modelo construído pode então ser usado na produção para a classificação de outros eventos.

O comportamento dos ambientes de rede varia diariamente e muda ao longo do tempo, seja devido ao surgimento de novos serviços ou à descoberta de novos ataques de rede [Sommer and Paxson 2010]. Como consequência, o modelo de aprendizagem de máquina ficará desatualizado com o passar do tempo, tornando a detecção de intrusão ineficaz ao longo do tempo e aumentando consideravelmente as taxas de erro do modelo [Viegas et al. 2019]. Para abordar tal cenário, o modelo desatualizado de NIDS baseado em aprendizagem de máquina, deve ser atualizado utilizando um conjunto de dados que represente o comportamento de uma rede real de produção. Contudo, a tarefa de atualização do modelo não é facilmente realizada [Ramos et al. 2021]. Nesse caso, os dados de rede do ambiente de produção devem ser monitorados por um longo período de tempo, resultando em um conjunto de dados com milhões de fluxos de rede, onde cada evento deve ser previamente rotulado antes que o procedimento de treinamento possa ser executado [Tomio et al. 2021]. No entanto, a rotulagem de eventos de rede é uma tarefa desafiadora, pois muitas vezes exige assistência de um especialista para a avaliação dos eventos coletados, que nem sempre está disponível ou estará a um custo elevado [Fontugne et al. 2010].

As atualizações do modelo encontradas na literatura com NIDS baseado em aprendizagem de máquina permanecem quase sempre negligenciadas [Sommer and Paxson 2010, Gates and Taylor 2006]. Em geral, as abordagens propostas negligenciam o procedimento de atualização do modelo ou assumem que pode ser facilmente realizado periodicamente, apesar dos desafios que apresenta para criação do novo modelo [Viegas et al. 2019, Bulle et al. 2020]. As técnicas tradicionais baseadas em aprendizagem de máquina descartam o modelo atual desatualizado em cada tarefa de atualização, aumentando ainda mais os custos computacionais de atualização do modelo, bem como a quantidade de dados de treinamento rotulados necessários para um treinamento de modelo adequado [Vicentini et al. 2019]. Como resultado, apesar das propostas altamente precisas na literatura do NIDS baseado em aprendizagem da máquina nas últimas décadas, ele permanece principalmente como um tópico de pesquisa, dificilmente implantado em ambientes de produção [Sommer and Paxson 2010].

Diante deste cenário, este artigo propõe um novo modelo de detecção de intrusão baseado em *autoencoders* e *transfer learning*, implementados de forma dupla. Primeiro, a detecção de intrusão é realizada por um classificador de reconhecimento de padrões, construído em cima das saídas de um codificador do *autoencoder*. O objetivo é facilitar o procedimento de atualização do modelo, já que o classificador será construído levando em consideração o conjunto de características de saída, através das camadas codificadoras, que são utilizadas como representação do comportamento da rede. Em segundo lugar, para facilitar os procedimentos de atualização do modelo, a cada atualização do modelo, emprega uma técnica de *transfer learning* aplicada no modelo do *autoencoder* desatualizado, com camadas codificadoras e decodificadoras. Consequentemente, o conhecimento prévio dos dados pode ser usado durante a atualização do modelo, diminuindo assim o número exigido de eventos rotulados necessários, bem como os custos de treinamento

computacional. Assim, nossa proposta com o *autoencoder* pode ser usada como uma representação histórica das características dos dados de rede, diminuindo ainda mais o número de eventos rotulados que devem ser fornecidos durante as atualizações do modelo.

As principais contribuições deste artigo são: uma avaliação dos NIDS tradicionais baseados em aprendizagem de máquina com relação ao seu desempenho de classificação ao longo do tempo de acordo com o tamanho do conjunto de dados de treinamento. Os experimentos mostraram que as abordagens tradicionais exigem quantidades inviáveis de dados de treinamento rotulados para uma detecção de intrusão confiável. Além disso, propomos e avaliamos um novo modelo de detecção de intrusão através de *transfer learning* em *autoencoders*. A abordagem proposta é capaz de facilitar significativamente o procedimento de atualização do modelo, diminuindo significativamente o tamanho de treinamento do conjunto de dados exigido e os custos computacionais nas atualizações do modelo.

2. Contextualização

2.1. Aprendizagem de Máquina para Detecção de intrusão

Um NIDS típico é composto por quatro módulos sequenciais [Viegas et al. 2021]. Primeiro, o módulo *Coleta de dados* que coleta continuamente eventos de rede do ambiente monitorado, por exemplo, uma placa de interface de rede. Os dados coletados são então analisados por um módulo de *Extração de Características*, que extrai um conjunto de características comportamentais para compor um vetor de características. Em geral, no NIDS, o comportamento dos eventos de rede é representado por meio de um fluxo de rede, que normalmente resume a comunicação entre hosts e serviços em uma janela de tempo especificada, por exemplo número de pacotes de rede enviados em um intervalo de 15 segundos [Viegas et al. 2019]. O vetor de características extraído é usado como entrada por um módulo de *Classificação*, que aplica um modelo de aprendizagem de máquina para classificação do evento como por exemplo *normal* ou *ataque*. Por fim, é gerado um alerta caso seja identificado um ataque, através de um módulo de alerta *alerta*

Vários trabalhos propuseram técnicas baseadas em aprendizagem de máquina de alta precisão para detecção de intrusão, principalmente através de meios de reconhecimento de padrões [Vicentini et al. 2018]. No entanto, apesar dos resultados promissores, poucas ou nenhuma dessas propostas foram implantadas em ambientes de produção, portanto, permanecem principalmente como um tópico de pesquisa [Sommer and Paxson 2010]. Em geral, os autores assumem erroneamente que o comportamento dos ambientes de rede não muda com o tempo [Viegas et al. 2019]. Como resultado, pouco ou nenhum esforço é dado para facilitar o processo de atualização do modelo [Gates and Taylor 2006], que deve exigir o mínimo possível de eventos de rede rotulados e baixos custos computacionais. No entanto, as técnicas tradicionais de reconhecimento de padrões não são capazes de fornecer tais características, visto que quando ocorre uma atualização do modelo, o modelo desatualizado é descartado e um novo conjunto de dados de treinamento, com milhões de eventos rotulados deve ser construído [Molina-Coronado et al. 2020].

2.2. Deep Autoencoders

Nos últimos anos, esforços significativos de pesquisa foram feitos em técnicas de *deep neural network* [Mallmann et al. 2020], que atualmente fornece os resultados mais promissores em campos como classificação de imagens e detecção de objetos [Alam et al. 2020]. Uma dessas arquiteturas de *deep neural* são os *deep autoencoders*, que normalmente são usados para redução de dimensionalidade [Li et al. 2020], e eliminação de ruído de imagem [Gondara 2016], através de um processo de duas etapas, o *codificador* e o *decodificador*. O objetivo do *codificador* é realizar a compressão da entrada da *deep neural network*, por exemplo, mapeando um vetor de características de entrada de tamanho N para uma saída de tamanho $N/3$. Portanto, para atingir tal objetivo, o *codificador* deve aprender um conjunto de relações de características úteis de sua entrada. Enquanto, o objetivo do *decodificador* é realizar a descompressão da sua entrada, por exemplo, mapeando um vetor de características de entrada de tamanho $N/3$ de volta à sua saída de tamanho N. Assim, a saída do *deep autencoder* é uma representação aprendida das relações mais úteis entre suas características de entrada. A pesquisa sobre *deep autencoder* para detecção de intrusão, embora com várias propostas, ainda está em seu inicio [Yan et al. 2020]. Outras abordagens, buscam a redução da dimensionalidade dos dados [Li et al. 2020], ou mesmo a detecção de *outliers* [Cheng et al. 2021], visando como objetivo final melhorias na precisão da detecção de intrusão.

3. Trabalhos Relacionados

Nas últimas décadas, a detecção de intrusão por meio de técnicas baseadas em aprendizagem de máquina foi abordada por uma infinidade de trabalhos [Sommer and Paxson 2010, Gates and Taylor 2006]. Em geral, os autores tratam da detecção de intrusão como uma tarefa de classificação supervisionada por meio de abordagens de reconhecimento de padrões, dando pouca ou nenhuma atenção aos desafios de atualização do modelo [Viegas et al. 2019]. Por exemplo, [Ahmim et al. 2019] propõe um conjunto de classificadores baseados em árvore para detecção de intrusão. A abordagem proposta foi capaz de melhorar a precisão quando comparado a classificadores únicos, no entanto, o desafio de atualização do modelo é negligenciado. Outro conjunto de classificadores baseados em árvore é proposto por [Kevric et al. 2016], onde os autores avaliaram uma série de abordagens para combinação de classificadores. Da mesma forma, as atualizações do modelo não são abordadas.

Outras técnicas de *deep neural network* amplamente exploradas em tarefas de detecção de intrusão, são por exemplo, [Vinayakumar et al. 2019] que mostrou que as *deep neural network* produzem uma detecção mais precisa do que as técnicas de classificação mais superficiais. Da mesma forma, [Yang and Wang 2019] depende de *convolutional neural network* para extração de características de ataques em redes sem fio, o que também melhora a precisão da detecção quando comparada às *deep neural network* mais comuns. No entanto, apesar de técnicas baseadas em *deep neural network* para detecção de intrusão terem sido usadas por vários trabalhos, o desafio de atualização do modelo é geralmente esquecido. [Zhang et al. 2019] propõe uma abordagem de *transfer learning* com *neural networks* para detecção de intrusão. Sua técnica filtra dados de novos aplicativos e os usa com *transfer learning* para atualizações de modelo. No entanto, embora as atualizações do modelo sejam abordadas, os autores assumem que novos

serviços podem ser identificados e rotulados conforme necessário, uma suposição errada no NIDS devido o grande volume de dados gerado diariamente.

A aplicação dos *deep autoencoders* também está sendo explorada por vários trabalhos de detecção de intrusão nos últimos anos. [Li et al. 2020] lida com um *deep autoencoder* com uma abordagem de seleção de características do *random forest* para diminuir os custos computacionais de atualização do modelo. No entanto, os autores assumem que os eventos podem ser rotulados conforme a necessidade, e visa apenas diminuir os custos computacionais, independentemente da quantidade necessária de instâncias rotuladas nas atualizações do modelo. Por outro lado, [Yan et al. 2020] aplica uma série de codificadores automáticos para a tarefa de extração de características. A abordagem proposta é capaz de aumentar a precisão, enquanto a tarefa de atualização do modelo é negligenciada.

4. Definição do Problema

Em geral, os NIDS baseados em aprendizagem de máquina propostos na literatura não levam em consideração o comportamento adverso dos ambientes em rede [Sommer and Paxson 2010]. Nesta seção, investigamos como as mudanças no comportamento da rede ao longo do tempo impactam nas abordagens de classificação baseadas em aprendizagem de máquina.

4.1. Conjunto de dados MAWIFlow

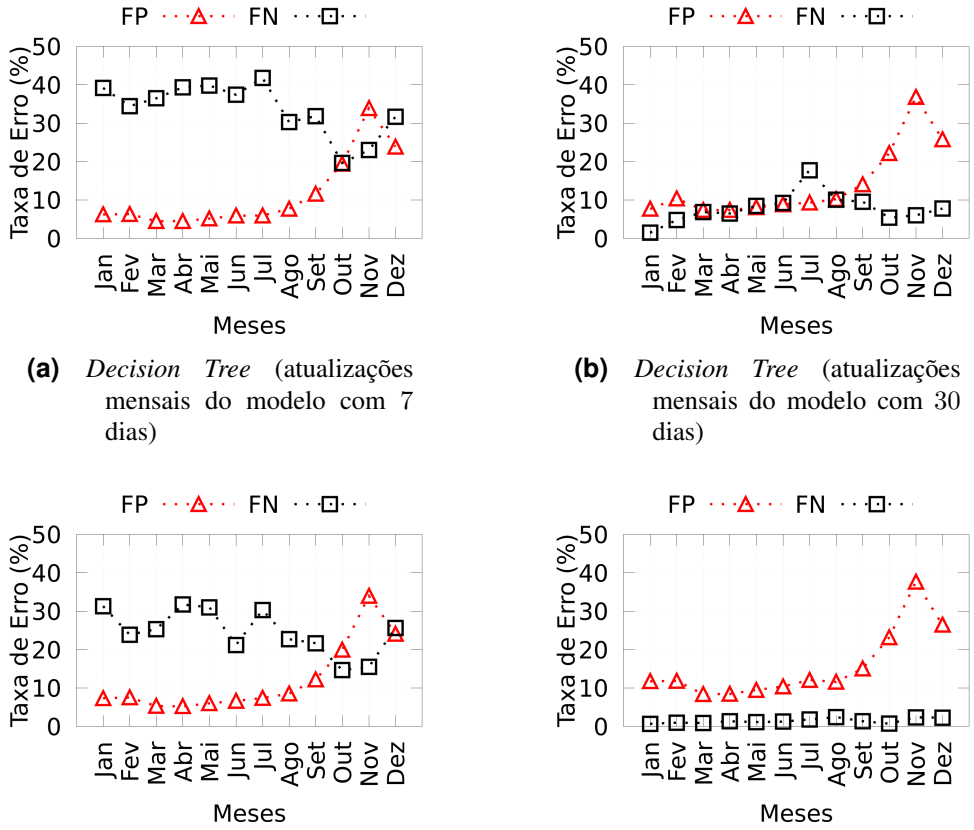
Os conjuntos de dados atuais disponíveis na literatura pressupõem que o comportamento do tráfego de rede não muda com o tempo. Como resultado, as abordagens propostas construídas por meio de tais dados são incapazes de avaliar o desempenho de sua abordagem proposta ao enfrentar as mudanças de comportamento do tráfego de rede ao longo do tempo.

Diante disso, nosso trabalho se baseia no conjunto de dados *MAWIFlow* [Viegas et al. 2019]. O conjunto de dados foi construído através do *Samplepoint-F* do arquivo MAWI [MAWI 2021], portanto, sendo feito do tráfego de rede real coletado diariamente por um intervalo de 15 minutos de um link de trânsito entre o Japão e os EUA. Todo o tráfego de rede coletado de 2016 foi usado para avaliar os NIDS baseados em aprendizagem de máquina amplamente usados. O conjunto de dados construído compreende mais de 8 TB de dados, compondo cerca de 300 bilhões de pacotes de rede. Para a rotulagem dos eventos, aplicamos uma técnica de aprendizagem de máquina não supervisionada do MAWILab [Fontugne et al. 2010], que rotula automaticamente os registros de entrada como *normal* ou *ataque*.

O MAWILab emprega vários algoritmos de aprendizagem de máquina não supervisionados para encontrar anomalias nos dados do MAWI sem assistência especializada para a tarefa de rotulagem de eventos. As anomalias encontradas são rotuladas como *ataque*, enquanto os dados restantes são considerados eventos *normal*. Para a tarefa de extração de características, o BigFlow [Viegas et al. 2019] foi usado para agrupar os eventos em intervalos de 15 segundos enquanto extraía 21 características baseadas em fluxo de rede do trabalho do Nigel [MAWI 2021].

4.2. Comportamento diverso do tráfego de rede

A avaliação visa responder a duas questões principais de pesquisa: (***RQ1***) *Quantos dados de treinamento rotulados devem ser fornecidos para um treinamento adequado de NIDS*



(a) *Decision Tree* (atualizações mensais do modelo com 7 dias)

(b) *Decision Tree* (atualizações mensais do modelo com 30 dias)

(c) *Gradient Boosting* (atualizações mensais do modelo com 7 dias)

(d) *Gradient Boosting* (atualizações mensais do modelo com 30 dias)

Figura 1. Comportamento da precisão da detecção de intrusão longo do tempo no conjunto de dados mensal *MAWIFlow*. Os classificadores são atualizados em uma periodicidade mensal usando os últimos 7 ou 30 dias como dados de treinamento rotulados.

baseados em aprendizagem de máquina? (***RQ2***) *Qual é o comportamento da precisão dos algoritmos NIDS baseados em aprendizagem de máquina com atualizações periódicas do modelo e dados de treinamento rotulados?*

Dois classificadores amplamente usados na literatura foram avaliados, o *Decision Tree* (DT) e o *Gradient Boosting* (GBT). O classificador DT foi implementado por meio de um algoritmo C4.5, com um fator de confiança de $0,25$ e $gini$ como a métrica de qualidade de divisão do nó. Enquanto o classificador GBT depende de um valor de taxa de aprendizagem de $0,1$, com $desvio$ como a função de perda, usando 100 árvores de decisão como sua base de aprendizagem. Uma subamostragem aleatória sem reposição é usada no procedimento de treinamento para equilibrar a ocorrência entre as classes. Os classificadores foram implementados por meio da API $scikit-learn$ v0.24. Os classificadores foram avaliados de acordo com suas taxas de Falso-Positivo (FP) e Falso-Negativo (FN). O FP denota a proporção de instâncias *ataque* classificadas incorretamente como *ataque*, enquanto o FN denota a proporção das instâncias *normal* classificadas incorretamente como *normal*.

O primeiro experimento visa responder o $RQ1$ e avaliar o desempenho da precisão dos classificadores selecionados de acordo com o tamanho do conjunto de dados de treinamento durante a realização das atualizações mensais do modelo. Para atingir tal objetivo, cada classificador avaliado foi atualizado periodicamente, mensalmente, com os dados ocorridos nos últimos 7 ou 30 dias. O objetivo é avaliar se o número de instâncias rotuladas fornecidas pode ser diminuído sem impacto significativo na precisão do NIDS, devido ao alto custo envolvido na rotulagem dos eventos em ambientes de produção. A figura 1 mostra o desempenho da precisão dos classificadores avaliados com atualizações mensais usando 7 dias (Figs. 1a e 1c) ou 30 dias (Figs. 1b e 1d) dos dados rotulados durante a atualização do modelo. É possível observar que o DT e o GBT não foram capazes de fornecer alta taxa de detecção com apenas 7 dias de dados de treinamento de atualização do modelo. Entretanto, quando 30 dias de dados são usados nas atualizações do modelo, conforme mostrado nas Figuras 1b e 1d, as abordagens avaliadas foram capazes de fornecer maior precisão de classificação em todo conjunto de dados *MAWIFlow* até outubro de 2016. Nesse caso, em $Out.$, $Nov.$ e $Dez.$, ainda mais dados de treinamento rotulados devem ser fornecidos nas atualizações do modelo para aumentar suas precisões, tornando tais técnicas inviáveis para NIDS baseados em aprendizagem de máquina.

O segundo experimento visa responder à pergunta $RQ2$ e investigar mais a fundo como as abordagens selecionadas funcionam quando um mês de dados de treinamento rotulados é fornecido durante as atualizações mensais do modelo. O intervalo selecionado de um 1 mês de vida útil do modelo $vs.$ 1 mês de tamanho do conjunto de dados de treinamento rotulado, foi escolhido assumindo que o operador de rede teria acesso ao rótulo de todos os eventos *MAWIFlow*, uma suposição irreal nos ambientes de produção. No entanto, tal avaliação serve como uma linha de base de quão preciso um NIDS baseado em aprendizagem devidamente atualizado é capaz de alcançar, apesar de ser inviável na produção. As figuras 1b e 1d mostram o desempenho da precisão ao longo do tempo dos classificadores avaliados com atualizações mensais do modelo. É possível notar que quando uma quantidade adequada de dados de treinamento é fornecida, os classificadores são capazes de fornecer taxas de precisão razoáveis em toda a maioria do conjunto de dados *MAWIFlow*. No entanto, em alguns meses, como $Out.$, $Nov.$ e $Dez.$, a taxa de precisão diminui drasticamente. A tarefa de atualização do modelo não é facilmente realizada, levando-se em consideração que mais de 30 dias de dados rotulados devem ser fornecidos para construção adequada do modelo. Portanto, para lidar com as mudanças de comportamento do tráfego de rede ao longo do tempo, as abordagens baseadas em aprendizagem de máquina devem ser capazes de fornecer facilidade nas atualizações do modelo, levando em consideração tanto o número de dados de treinamento necessários quanto os custos computacionais do retreinamento do modelo.

5. Um modelo de detecção de intrusão baseado em *autoencoders* desatualizado

Para enfrentar o desafio acima mencionado do comportamento variado de ambientes de rede, propomos um novo modelo de detecção de intrusão baseado em *transfer learning* com *deep autoencoder*. A abordagem proposta visa facilitar a tarefa de atualização do modelo considerando os custos computacionais e a quantidade de dados de treinamento rotulados necessários. Para atingir tal objetivo, o modelo proposto na (figura 2) é implementado em dois módulos: *Classificação* e *Facilidade na Atualização do Modelo*.

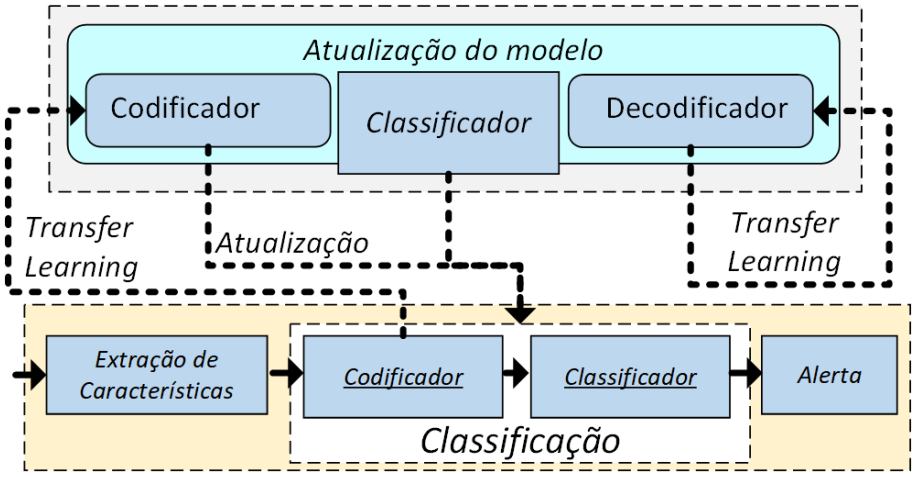


Figura 2. Proposta do modelo de detecção de intrusão por meio do uso de *deep autoencoder* com *transfer learning*.

O procedimento de classificação começa com um evento de rede a ser classificado, coletado do ambiente monitorado. O comportamento dos dados coletados é então extraído por um módulo de extração de características, que produz um conjunto de características comportamentais que compõem um vetor de características. O vetor de características compreende um conjunto de valores que representam o comportamento do evento analisado, que é usado como entrada para uma camada do *codificador* (Figura 2, *codificador*). O *codificador* produz um conjunto adicional de características comportamentais, que é usado como entrada por um classificador (por exemplo, GBT), que o classifica como *normal* ou *ataque*. Com isso, nossa proposta faz com que as camadas do *codificador* de um *deep autoencoder* sejam usadas como um módulo de extração de características adicional, que leva em consideração o comportamento dos eventos da rede. A lógica de nossa proposta é que o *codificador*, construído por meio de *transfer learning*, seja capaz de ampliar conhecimento prévio sobre dados de rede. Nas atualizações do modelo, a proposta reaproveita os pesos das camadas *codificadoras* e do *decodificadoras* do *deep autoencoder* desatualizado. Como consequência, as atualizações do modelo são executadas por meio da abordagem de *transfer learning*, diminuindo os custos computacionais e a quantidade de dados de treinamento rotulados necessários.

As próximas subseções descrevem com mais detalhes nosso modelo proposto, incluindo os componentes utilizados para a implementação.

5.1. Classificação

O NIDS tradicional baseado em aprendizagem de máquina realiza a classificação dos eventos de rede por meio da avaliação de características extraídas do fluxo de rede. Como consequência, o procedimento de treinamento de tais propostas exige o fornecimento de quantidades inviáveis de eventos rotulados. Isso porque o conjunto de funcionalidades utilizado não leva em consideração suas variações ao longo do tempo, de acordo com as mudanças de comportamento do tráfego da rede.

Diante disso, nossa proposta aproveita as camadas do *codificador* de um *deep autoencoder*, como um estágio adicional de extração de características. A justificativa de tal abordagem é que as camadas *codificadoras*, que são continuamente atualizadas por meio

da técnica de *transfer learning*, serão capazes de retratar as relações significativas nos valores de características ao longo do tempo, portanto, extraindo características significativas que serão capazes de resistir a longos períodos de tempo, apesar das mudanças no tráfego de rede diário. Para fazer uso de tal abordagem, nosso procedimento de classificação (Figura 2), extrai as características tradicionais baseadas no fluxo de rede coletadas e os usa como entrada para as camadas *codificadoras*, que produzem as características significativas estabelecidas por meio de um processo de treinamento contínuo utilizando *transfer learning*. A saída do *codificador* é usada como entrada para um classificador (por exemplo, GBT), que atribui um rótulo de evento relacionado, por exemplo, *normal* ou *ataque*.

5.2. Facilidade nas Atualizações do Modelo

Abordagens com atualização do modelo tradicionais em NIDS baseados em aprendizagem de máquina descartam o modelo desatualizado e constroem um novo no conjunto de dados de treinamento rotulado atualizado. Como consequência, tais propostas exigem quantidades significativas de dados rotulados durante as atualizações do modelo (e.g., Figs. 1c vs 1d), tornando-os inviáveis para serem usados adequadamente na produção.

Para enfrentar esse desafio, nossa proposta se baseia em *deep autoencoders* e *transfer learning* nas atualizações do modelo (Figura 2, *Facilidade na Atualização do Modelo*), como uma abordagem para diminuir os custos computacionais e a quantidade de dados rotulados necessários para o treinamento. Por um lado, os *deep autoencoders* são usados como um estágio adicional de extração de características com o objetivo de representar as mudanças de comportamento do tráfego de rede ao longo do tempo. Isso porque, como é continuamente atualizado por meio do *transfer learning*, é capaz de extrair relações significativas das características de entrada ocorridas com o passar do tempo, de acordo com as mudanças no comportamento do tráfego da rede ao longo do tempo. Por outro lado, o *transfer learning* é usado para diminuir os custos computacionais durante a atualização do modelo, bem como manter o comportamento conhecido já presente nas camadas do *deep autoencoder*. Como consequência, nosso modelo proposto é capaz de diminuir a quantidade de dados de treinamento rotulados necessários e os custos computacionais para cada atualização do modelo. Em resumo, a tarefa de atualização do modelo é executada da seguinte maneira. As camadas *codificadoras* desatualizadas são usadas pelo módulo de classificação e enviadas para os*decodificadores* com o conhecimento armazenado. O *deep autoencoder* é atualizado por meio da técnica de *transfer learning*, usando uma pequena janela de tempo de eventos, por exemplo, uma semana de dados, levando em consideração que o comportamento conhecido da rede já está disponível no modelo desatualizado do *deep autoencoder*. Depois que a tarefa de atualização do *deep autoencoder* é realizada, as camadas *codificadoras* são usadas como entrada por um classificador (por exemplo, GBT), que também é atualizado, levando em consideração o novo *deep autoencoder*.

6. Avaliação

Esta seção avalia ainda mais o modelo de detecção de intrusão proposto, respondendo a três questões principais de pesquisa: (***RQ3***) *Como a abordagem proposta funciona sem atualizações de modelo e utilizando poucos dados de treinamento rotulados?* (***RQ4***) *A abordagem proposta é capaz de realizar atualizações do modelo com menos instâncias*

de treinamento? (***RQ5***) *A abordagem de transfer learning proposta diminui os custos computacionais?*

O modelo proposto (Figura 2) foi implementado e avaliado com o conjunto de dados *MAWIFlow* (consulte a seção 4.1). O *deep autoencoder* foi implementado com a seguinte arquitetura:

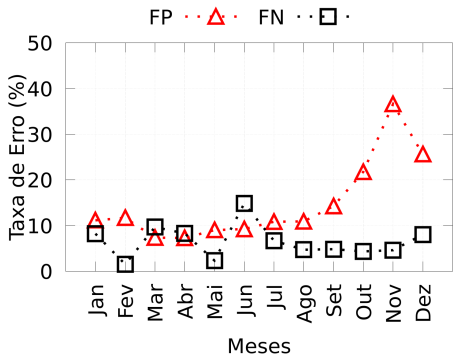
- *Entrada*. As 21 características extraídas são alimentadas como entrada para o *deep autoencoder*.
- *Codificador*. Três camadas $densas$ foram implementadas com uma função de ativação $relu$, com 512, 256 e 128 características respectivamente.
- *Codificador de Saída*. Uma camada de saída *codificadora* foi implementada com uma função de ativação $linear$ e 10 características.
- *Decodificador*. Três camadas *decodificadoras* foram implementadas com a função de ativação $relu$, com 128, 256 e 512 características respectivamente.
- *Decodificador de Saída*. Uma camada de saída do *decodificador* com 21 características e uma função de ativação $sigmoid$.

A arquitetura do *deep autoencoder* implementada usou a fórmula $rmse$ como função de $loss$ e o otimizador $adam$. Para cada procedimento de construção de modelo, do zero ou nas atualizações do modelo. 1000 épocas são executadas com um tamanho de lote de 512. A arquitetura foi implementada por meio da API $keras\ v.2.4.0$ e a API $tensorflow\ v.2.4.1$. É importante notar que o conjunto de parâmetros usado foi definido de forma semelhante aos trabalhos relacionados, e não foram encontradas diferenças significativas ao variá-los. O classificador (Figura 2) foi implementado através do algoritmo *Gradient Boosting* (GBT). Da mesma forma como feito na Seção 4, o algoritmo usa o valor da taxa de aprendizagem $0,1$, com $desvio$ como a função de perda, usando 100 árvores de decisão como seu algoritmo base. Uma subamostragem aleatória sem reposição é usada no procedimento de treinamento para equilibrar a ocorrência entre as classes.

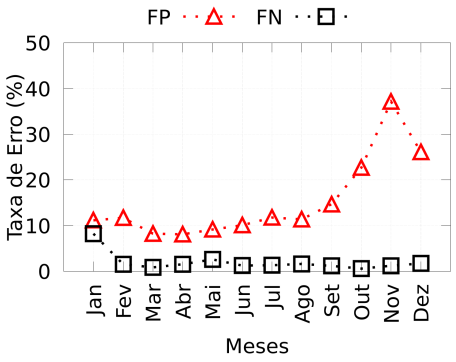
6.1. Atualização com menos dados

Para responder à pergunta $RQ3$, investigamos ainda se nosso modelo proposto é capaz de fornecer alta precisão de classificação com menos dados de treinamento rotulados, mesmo quando nenhuma atualização do modelo é realizada. Para atingir tal objetivo, diferente da Seção 4 que usa 30 dias de dados de treinamento rotulados, realizamos o procedimento de treinamento de nossa proposta com apenas 7 dias de eventos rotulados. Portanto, nosso modelo proposto é treinado com os dados que ocorreram no conjunto de dados *MAWIFlow* de 1 a 7 de janeiro do ano de 2016, embora não seja atualizado ao longo do tempo, avaliando a mais extrema situação que nossa proposta poderia enfrentar em ambientes de produção.

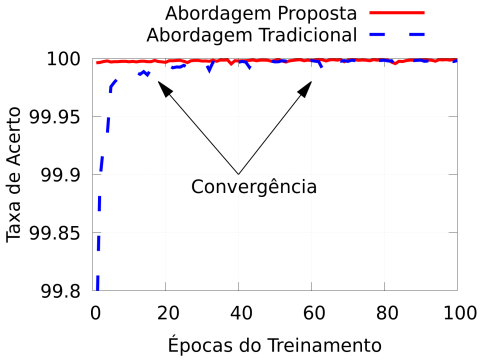
A Figura 3a mostra a precisão da classificação do nosso modelo proposto ao longo do tempo com apenas 7 dias de dados de treinamento rotulados e sem nenhuma atualização do modelo realizada. É possível notar que nosso modelo proposto fornece precisão da classificação semelhante, mesmo quando comparado a abordagens tradicionais que dependem de um conjunto de 30 dias de dados rotulados para o treinamento e realização das atualizações periódicas do modelo (Figura 1). Por exemplo, se comparar a técnica sem o *deep autoencoder* (Figura 1c), melhorou a taxa de FN em 3,8% e a taxa de



(a) Treinamento de modelo em janeiro com apenas 7 dias, e nenhuma atualização ao longo do tempo



(b) Atualizações do modelo em uma periodicidade mensal, usando apenas 7 dias de dados



(c) Comparação da convergência de treinamento no mês de fevereiro

Figura 3. Desempenho de abordagem proposta com o conjunto de dados *MAWI-Flow* (Seção 4.1).

FP em 0,3%, com apenas $\approx$ 22% de dados de treinamento rotulados, embora não sejam atualizados ao longo do tempo. Portanto, a abordagem proposta, com *deep autoencoder*, é capaz de fornecer detecção de intrusão de alta precisão, exigindo menos dados de treinamento.

Para responder à pergunta $RQ4$, avaliamos como as atualizações periódicas do modelo afetam nosso modelo proposto. Para atingir essa meta, realizamos atualizações mensais do modelo também com apenas 7 dias de dados. Por exemplo, em primeiro de fevereiro, nossa abordagem proposta é atualizada com os dados que ocorreram entre 25 a 31 de janeiro. Lembrando que nosso procedimento de atualização do modelo amplia o modelo desatualizado do *deep autoencoder* por meio de uma abordagem de *transfer learning* (Seção 5.2).

A Figura 3b mostra a precisão da classificação da nossa proposta com atualizações mensais do modelo. É possível notar uma melhoria na precisão da classificação quando as atualizações do modelo são realizadas, mesmo com apenas uma semana de dados de treinamento rotulados. As atualizações mensais do modelo em nossa abordagem proposta foram capazes de melhorar a taxa de FN em 5,4% quando comparada à abordagem

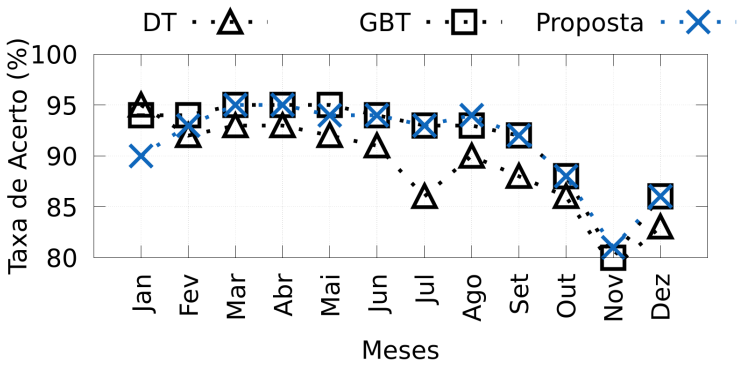


Figura 4. Taxa média de acerto das abordagens avaliadas no conjunto de dados *MAWIFlow* com atualizações mensais do modelo. As técnicas tradicionais são atualizadas com 30 dias, enquanto nossa proposta é atualizada com apenas 7 dias de dados rotulados, podemos fornecer resultados semelhantes enquanto exigimos apenas ≈ 22% de dados de treinamento rotulados.

sem atualização (Figura 3a). Também investigamos o desempenho da nossa abordagem proposta quando comparado às técnicas tradicionais com atualizações periódicas do modelo. A Figura 4 mostra a taxa de acerto média (média de $1 - FP$ e $1 - FN$) quando atualizações mensais do modelo são realizadas. Lembrando que nossa abordagem proposta usa apenas 7 dias de dados de treinamento, enquanto GBT e DT dependem de 30 dias de dados de treinamento. Nossa abordagem proposta também é capaz de melhorar a taxa média de acerto em $3,4\%$ quando comparado ao DT tradicional. Além disso, se comparado ao GBT, mas sem nossa abordagem proposta de *deep autoencoder* (Fig. 1d) diminui a taxa de acerto em apenas $0,2\%$, enquanto exige apenas $\approx 22\%$ de instâncias de treinamento.

Finalmente, para responder à questão $RQ5$, investigamos como nossa abordagem proposta pode diminuir os custos computacionais durante as atualizações do modelo. Para atingir tal objetivo, avaliamos o número de épocas necessárias para nosso *deep autoencoder* durante a tarefa de treinamento do modelo, ou seja, a convergência do treinamento. A Figura 3c mostra a convergência do modelo na atualização do modelo em fevereiro de 2016 no conjunto de dados *MAWIFlow*. É possível notar que nossa abordagem proposta se baseia no *deep autoencoder* com *transfer learning* para convergir o modelo nas atualizações com períodos de treinamento significativamente menores. Em resumo, nossa abordagem proposta foi capaz de diminuir o tempo computacional nas atualizações do modelo em uma média de 28%, reduzir os dados rotulados necessários para apenas $\approx 22\%$ de dados, enquanto melhorava a taxa de FN em até $23,9\%$ (Fig. 3b $vs.$ 1a).

7. Conclusão

Este trabalho desafiou a suposição da literatura sobre como as mudanças no comportamento do tráfego de rede afetam o NIDS baseado em aprendizagem de máquina, mostrando que isso pode afetar significativamente a precisão da classificação, e ao mesmo tempo exige o fornecimento de quantidades inviáveis de dados de treinamento rotulados durante as atualizações do modelo. Além disso, propusemos um novo modelo de detecção de intrusão baseado em *deep autoencoder* com *transfer learning* para ser usado como um

estágio de extração de conhecimento do ambiente desatualizado. Nossa abordagem proposta foi capaz de fornecer taxas de precisão significativamente altas, enquanto exigia custos computacionais mais baixos nas atualizações do modelo e menos instâncias de treinamento a serem fornecidas durante as atualizações do modelo. Como trabalhos futuros, planejamos estender nossa abordagem proposta para abordar *multi-view* e *ensemble de classificadores* visando maior expectativa de vida do modelo.

Agradecimentos

Os autores agradecem ao CNPq pelo apoio financeiro parcial ao projeto, processo: 430972/2018-0.

Referências

Ahmim, A., Maglaras, L., Ferrag, M. A., Derdour, M., and Janicke, H. (2019). A novel hierarchical intrusion detection system based on decision tree and rules-based models. In *2019 15th International Conference on Distributed Computing in Sensor Systems (DCOSS)*. IEEE.

Alam, M., Samad, M., Vidyaratne, L., Glandon, A., and Iftekharuddin, K. (2020). Survey on deep neural networks in speech and vision systems. *Neurocomputing*, 417:302–321.

Bulle, B. B., Santin, A. O., Viegas, E. K., and dos Santos, R. R. (2020). A host-based intrusion detection model based on OS diversity for SCADA. In *IECON 2020 The 46th Annual Conference of the IEEE Industrial Electronics Society*. IEEE.

Cheng, Z., Zhu, E., Wang, S., Zhang, P., and Li, W. (2021). Unsupervised outlier detection via transformation invariant autoencoder. *IEEE Access*, 9:43991–44002.

Fontugne, R., Borgnat, P., Abry, P., and Fukuda, K. (2010). MAWILab: Combining diverse anomaly detectors for automated anomaly labeling and performance benchmarking. In *Proc. of the 6th Int. Conf. on emerging Networking EXperiments and Technologies (CoNEXT)*.

Gates, C. and Taylor, C. (2006). Challenging the anomaly detection paradigm: A provocative discussion. In *Proc. of the Workshop on New Security Paradigms (NSPW)*, pages 21–29.

Gondara, L. (2016). Medical image denoising using convolutional denoising autoencoders. In *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE.

Kevric, J., Jukic, S., and Subasi, A. (2016). An effective combining classifier approach using tree algorithms for network intrusion detection. *Neural Computing and Applications*, 28(S1):1051–1058.

Li, X., Chen, W., Zhang, Q., and Wu, L. (2020). Building auto-encoder intrusion detection system based on random forest feature selection. *Computers & Security*, 95:101851.

Mallmann, J., Santin, A. O., Viegas, E. K., dos Santos, R. R., and Geremias, J. (2020). PPCensor: Architecture for real-time pornography detection in video streaming. *Future Generation Computer Systems*, 112:945–955.

MAWI (2021). MAWI Working Group Traffic Archive - Samplepoint F.

Molina-Coronado, B., Mori, U., Mendiburu, A., and Miguel-Alonso, J. (2020). Survey of network intrusion detection methods from the perspective of the knowledge discovery in databases process. *IEEE Trans. on Network and Service Management*, 17(4):2451–2479.

Ramos, F., Viegas, E., Santin, A., Horchulhack, P., dos Santos, R. R., and Espindola, A. (2021). A machine learning model for detection of docker-based APP overbooking on kubernetes. In *ICC 2021 - IEEE International Conference on Communications*. IEEE.

Sommer, R. and Paxson, V. (2010). Outside the closed world: On using machine learning for network intrusion detection. In *2010 IEEE Symposium on Security and Privacy*. IEEE.

Tomio, R. L., Viegas, E. K., Santin, A. O., and dos Santos, R. R. (2021). A multi-view intrusion detection model for reliable and autonomous model updates. In *ICC 2021 - IEEE International Conference on Communications*. IEEE.

Vicentini, C., Santin, A., Viegas, E., and Abreu, V. (2018). A machine learning auditing model for detection of multi-tenancy issues within tenant domain. In *2018 18th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID)*. IEEE.

Vicentini, C., Santin, A., Viegas, E., and Abreu, V. (2019). SDN-based and multitenant-aware resource provisioning mechanism for cloud-based big data streaming. *Journal of Network and Computer Applications*, 126:133–149.

Viegas, E., Santin, A., Bessani, A., and Neves, N. (2019). BigFlow: Real-time and reliable anomaly-based intrusion detection for high-speed networks. *Future Generation Computer Systems*, 93:473–485.

Viegas, E., Santin, A. O., and Jr, V. A. (2021). Machine learning intrusion detection in big data era: A multi-objective approach for longer model lifespans. *IEEE Transactions on Network Science and Engineering*, 8(1):366–376.

Vinayakumar, R., Alazab, M., Soman, K. P., Poornachandran, P., Al-Nemrat, A., and Venkatraman, S. (2019). Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7:525–550.

Yan, Y., Qi, L., Wang, J., Lin, Y., and Chen, L. (2020). A network intrusion detection method based on stacked autoencoder and LSTM. In *ICC 2020 IEEE Int. Conf. on Communications (ICC)*. IEEE.

Yang, H. and Wang, F. (2019). Wireless network intrusion detection based on improved convolutional neural network. *IEEE Access*, 7:64366–64374.

Zhang, J., Li, F., Wu, H., and Ye, F. (2019). Autonomous model update scheme for deep learning based network traffic classifiers. In *2019 IEEE Global Communications Conference (GLOBECOM)*. IEEE.