Sistema Híbrido e On-line de Detecção e Classificação de Tráfego Malicioso

Diego Abreu[1,2], **Antônio Abelém**[1,2]

[1] Grupo de Estudos em Redes Comunicação Multimídia - GERCOM

[2]Universidade Federal do Pará - UFPA

Abstract. *Several Machine Learning methodologies have been proposed to improve security in computer networks and reduce the damage caused by the action of malicious agents. However, detecting and classifying attacks with high accuracy and precision is still a major challenge in today's networks. This paper proposes an online attack detection and network traffic classification system, which hybridly combines Stream Machine Learning, Deep Learning, and Ensemble technique. Using multiple stages of data analysis, the system can detect the presence of malicious traffic flows and classify them according to the type of attack they represent. The system was evaluated in three network security datasets, in which it obtained accuracy and precision above 90% with a reduced false alarm rate.*

Resumo. *Diversas metodologias de Aprendizado de Máquina têm sido propostas para melhorar a segurança em redes de computadores e reduzir os danos causados pela ação de agentes maliciosos. Entretanto, detectar e classificar ataques de forma acurada e precisa ainda é um grande desafio nas redes atuais. Este artigo propõe um sistema on-line de detecção de ataques e classificação de tráfego de rede, que combina de forma híbrida Aprendizado de Máquina por Stream, o Aprendizado Profundo e a técnica Ensemble. Utilizando múltiplos estágios de análise dos dados, o sistema detecta a presença de fluxos de tráfegos maliciosos e classifica-os de acordo com o tipo de ataque que eles representam. O sistema foi avaliado em três bases de dados de referência em segurança, nas quais obteve acurácia e precisão acima de 90%, com taxa de falso alarme reduzida.*

1. Introdução

No cenário atual, cresce cada vez mais a quantidade e diversidade de ameaças à segurança das redes de computadores [Liang and Kim 2021]. Assim, detectar ataques e proteger a rede tem se tornado uma tarefa muito desafiadora para os mecanismos de segurança, como Sistema de Detecção de Intrusão (*Intrusion Detection System - IDS*) e Sistema de Prevenção de Intrusão (*Intrusion Prevention System - IPS*) [Maidamwar et al. 2022]. Os principais desafios enfrentados no monitoramento e prevenção de ataques são a grande quantidade de dados e os custos e tempo da análise e do processamento envolvidos. Devido a heterogeneidade da rede, os ataques podem vir de diversas fontes e atacar diversos dispositivos, gerando um fluxo constante dos dados a serem analisados [Lobato et al. 2016]. Isso resulta na grande demora na detecção dos ataques e na quantidade de falsos alarmes gerados pelos sistemas atuais de monitoramento [Liang and Kim 2021].

Nesse contexto, diversas técnicas de Aprendizado de Máquina têm sido propostas para detectar a presença de atividade de agentes maliciosos na rede, o que pode significar a ocorrência de um ataque. Em destaque, técnicas de Aprendizado Profundo (*Deep Learning*) e técnicas baseadas em *Ensemble* têm obtido resultados significativos em termos de precisão e acurácia [Salih et al. 2021].

Entretanto, muitas dessas propostas tratam a detecção de ataques como uma tarefa de aprendizagem *off-line*, na qual os modelos de aprendizagem são treinados e, em seguida, aplicados ao sistema *on-line* de detecção. Desta forma, essas abordagens tendem a desconsiderar o caráter dinâmico e adversário dos ataques na rede, no qual a mudança de conceito (*concept drift*) se manifesta, tanto em alterações nas estatísticas do alvo de aprendizagem e de predição quanto nas características (*features)* de rede utilizadas. Como consequência disto, os modelos treinados, de forma *off-line* se tornam obsoletos rapidamente, o que compromete o funcionamento e a segurança da rede. Além disso, é importante não apenas detectar a ocorrência de uma ação maliciosa na rede, mas também conseguir caracterizar e classificar os ataques de forma correta. De acordo com o tipo específico de ataque identificado, pode-se tomar decisões e ações distintas nas redes buscando interromper o ataque em andamento e prevenir a ocorrência futura deste ataque [Maidamwar et al. 2022].

Nossa proposta para este trabalho busca realizar a detecção de ação maliciosa na rede e a classificação desse tráfego malicioso de acordo com classes de ataques. Nossa abordagem consiste em realizar, de forma *on-line*, múltiplos estágios de análise dos dados do fluxo da rede, de forma a aplicar técnicas específicas de AM de acordo com a tarefa mais apropriada para esses estágios. Nossa hipótese de pesquisa é que é possível desenvolver um sistema que seja capaz de detectar e classificar ataques dinâmicos e aliar as vantagens de técnicas AM por *Stream* (fluxo de dados), técnicas de Aprendizado Profundo e baseadas em *Ensemble*.

O restante do artigo está organizado da seguinte forma: a seção 2 apresenta a fundamentação teórica; a seção 3 e 4 apresentam os trabalhos relacionados e o sistema proposto; a seção 5 apresenta o estudo de caso, enquanto que a seção 6 apresenta os resultados dos experimentos e as discussões; por fim a seção 7 apresenta as conclusões os trabalhos futuros.

2. Fundamentação Teórica

Nesta seção é apresentada a fundamentação teórica para o desenvolvimento do sistema proposto neste trabalho. O sistema proposto combina técnicas de Aprendizado de Máquina por *Ensemble*, Profundo e por *Stream*, buscando obter as principais vantagens de cada metodologia.

2.1. Aprendizado de Máquina por Ensemble

O Aprendizado de Máquina por *Ensemble* é uma metodologia de aprendizado que utiliza diferentes algoritmos de classificação para criar modelos de predição e combinar os resultados individuais a fim de obter um resultado coletivo otimizado [Polikar 2012]. Entre as diversas abordagens de *Ensemble* podemos destacar a Votação Majoritária (*Majority Voting*) [Polikar 2012]. Nessa abordagem, cada modelo faz a sua predição acerca do pertencimento de cada instância à uma classe, o que representa um voto. A classe que

recebe o maior número de votos é então escolhida para a instância analisada. A Figura 1 apresenta o funcionamento da Votação Majoritária.

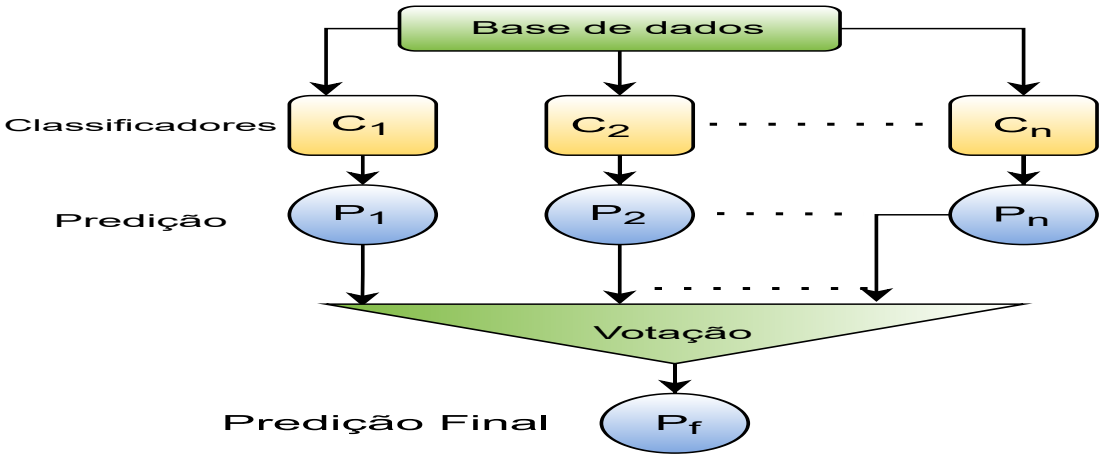

Figura 1. Votação Majoritária.

Como a Figura 1 mostra, para cada instância, cada classificador gera a sua predição de forma independente, sendo que vários classificadores diferentes podem ser utilizados nesse processo. A predição final é determinada de acordo com a maioria dos votos dados. Esse processo é feito até que todas as instâncias da base de dados tenham sido classificadas.

Neste trabalho, são utilizados como classificadores três métodos de AM supervisionados: *Random Forest*, K-ésimo Vizinho mais Próximo (*k-nearest neighbors* - kNN) [Altman 1992] e Máquina de Vetores de Suporte (*Support Vector Machine* - SVM) [Lorena and De Carvalho 2007]. Esses classificadores foram escolhidos pois são frequentemente utilizados no contexto de detecção de ataques e também por terem funcionamento diferentes entre si, o que melhora a sua utilização dentro da abordagem de Votação Majoritária [Salih et al. 2021]. Assim, a decisão final do estágio do sistema que utiliza o AM por *Ensemble* será com base na votação entre esses três classificadores.

2.2. Aprendizado de Máquina Profundo

O Aprendizado Profundo é um ramo do Aprendizado de Máquina baseado em Redes Neurais Artificiais (RNA). No contexto de detecção de ataques de rede uma das técnicas de Aprendizado Profundo mais utilizadas são as Redes Neurais Recorrentes (*Recurrent Neural Networks* - RNNs) [Rego and Nunes 2021]. As RNN são projetadas para reconhecer padrões em sequências de dados que estabeleçam um ordenamento lógico. As RNN salvam a saída de uma camada anterior, e as alimentam de volta à camada de entrada, para ajudar a prever o resultado da camada atual.

Uma das arquiteturas mais utilizadas em RNN são as *Long Short-Term Memory Networks* (LSTMs) [Malhotra et al. 2015]. As RNN podem trabalhar com sequência longas de dados, como séries temporais, porém existe uma limitação na recuperação de informações de sequências anteriores. As RNN do tipo LSTM buscam mitigar essa limitação das RNN utilizando células de memória de longo prazo, com portões específicos para o controle de informações. A Figura 2 apresenta a estrutura de uma célula LSTM.

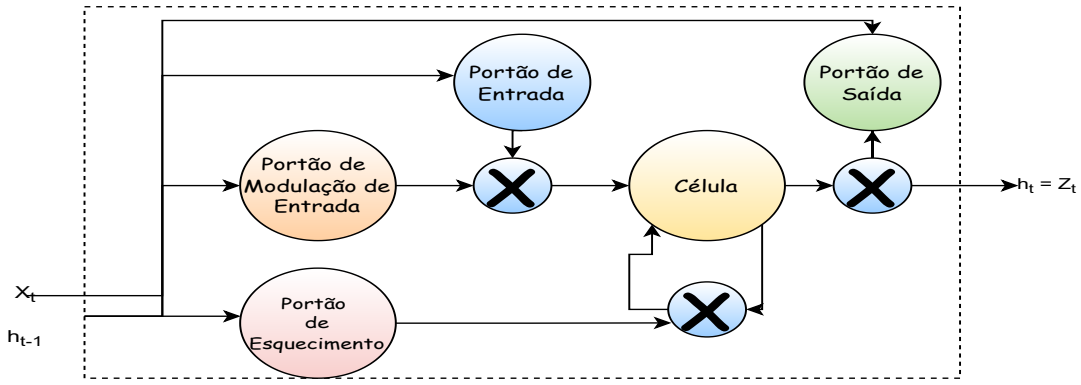

Figura 2. Estrutura de uma célula LSTM.

Como a Figura 2 apresenta, o método LSTM é composto por três portões, que são Redes Neurais, os quais controlam o fluxo de informações da rede e decidem o que é importante para a memória. O Portão do Esquecimento recebe a entrada X_t, que representa a entrada no momento presente, e h_{t-1}, que representa a saída da célula anterior, e decide qual informação deve ser descartada. O Portão de Entrada, adiciona informações consideradas úteis ao estado da célula. Já no Portão de Saída é realizado a extração de informações consideradas úteis ao estado da célula atual, as quais vão ser enviadas para a próxima célula, representada por Z_t. O LSTM também possui um Portão de Modulação de Entrada, o qual é responsável por determinar um valor baseado na entrada atual e na saída da célula anterior.

2.3. Aprendizado de Máquina por *Stream*

Diferentemente do Aprendizado de Máquina *off-line*, no qual o conjunto de dados está disponível ao método de aprendizagem no momento do treino, no Aprendizado de Máquina por *Stream* os dados são classificados na medida em que chegam ao preditor. Os métodos de AM por *Stream* processam os dados uma instância, ou conjunto de instâncias, de cada vez. Desta forma, é possível usar essa metodologia para realizar a classificação de dados em tempo real, na medida em que os dados chegam ao sistema de aprendizado. A Figura 3 apresenta o processo de Aprendizado de Máquina por *Stream*.

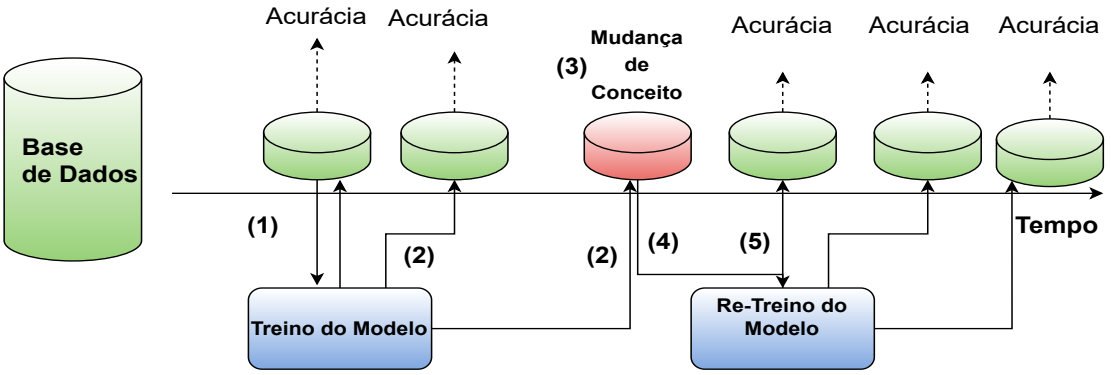

Figura 3. Processo de Aprendizado de Máquina por *Stream*.

Durante o processamento dos dados por *Stream* é comum ocorrer mudanças de conceito dos dados, alterações significativas dos valores dos atributos. No contexto de

detecção de ataques, essas mudanças de conceito podem ser causadas tanto por variações no comportamento normal da rede quanto por ação de agentes maliciosos em ataques à rede. Neste caso, o modelo de aprendizagem tende a perder parte de sua capacidade de predição, sendo necessário realizar o re-treino desse modelo. Os melhores algoritmos de AM buscam formas de ser mais robustos a essas mudanças de conceito de forma a evitar a diminuição da acurácia e precisão, e também evitar o re-treino do modelo.

Na Figura 3, vemos que os primeiros dados que chegam para serem analisados (1) são utilizados para o treino do modelo de aprendizagem. Esse modelo então é aplicado na próxima sequência de dados (2), realizando a predição e a classificação dos dados. Quando é detectada uma mudança de conceito (3), é necessário realizar um novo treinamento do modelo (4), para que este possa ser adaptado ao novo estado do fluxo de dados. Com o modelo adaptado é então feita novamente a predição dos dados para a próxima sequência de dados (5). Esse processo segue até que todos os dados sejam lidos.

O processo de seleção de característica é realizado a cada treino ou re-treino do modelo. A técnica utilizada além de trabalhar com *Stream* de dados, tem que ser rápida o suficiente para não atrasar a detecção e também deve ajudar a criar modelos mais robustos, capazes de se ajustar às mudanças de conceito, e assim evitar o re-treino. Em nossa proposta, a seleção de características é abordada de forma integrada tanto na detecção *on-line* dos modelos quanto na classificação por tipo de ataques, utilizando a seleção de características por clusterização [Abreu et al. 2020], uma técnica não supervisionada que evita o *overfitting* às classes majoritárias ao desconsiderar os rótulos dos dados na seleção do conjunto das características.

3. Trabalhos Relacionados

Em Lucas et al. (2021) [Lucas et al. 2021] é proposto um sistema de detecção de ataques com múltiplas camadas baseado em *Ensemble*. Os autores buscam combinar diversos métodos de classificação de forma exaustiva a fim de se propor a melhor configuração dos hiperparâmetros dos classificadores. O sistema é avaliado utilizando a base CICID17 [Sharafaldin et al. 2018], obtendo resultados de acurácia e precisão próximos do 100%. O trabalho, porém, não faz uma análise com relação ao tempo utilizado para gerar essas configurações otimizadas, e todo o processamento é feito de forma *off-line*. Em nossa proposta, também é utilizado um sistema de detecção e classificação de ataque com múltiplos estágios, porém, diferentemente do proposto em Lucas et al (2021), também será avaliado o tempo e o impacto que a geração e aplicação dos modelos causam no sistema. Além disso, também utilizaremos a abordagem *Ensemble* para compor o nosso sistema de detecção e utilizaremos a base CICIDS17 em nosso estudo de caso.

Em Tian et al. (2021) [Tian et al. 2021] é proposto uma abordagem de detecção e classificação de ataques em dois estágios para o contexto de Redes Definidas por Software (SDN - *Software Defined Networks*). Os autores utilizam um estágio para selecionar as características de rede usando um método bio-inspirado para se obter um conjunto otimizado das características mais relevantes. No segundo estágio, com o conjunto reduzido de características, é aplicado um classificador *Ensemble* que combina Árvore de Decisão (*Decision Tree* - DT), uma Rede Neural MLP (*Multilayer Perceptron*) e o kNN. O sistema proposto é avaliado nas bases NSL-KDD [Dhanabal and Shantharajah 2015] e UNSW-NB15 [Moustafa and Slay 2015], obtendo resultados próximos ao 100% para a

detecção binária, se o tráfego é malicioso ou benigno. Porém, na classificação por tipo de ataque, multiclasse, o sistema apenas tem bons resultados para os ataques com maior quantidade de dados disponíveis para o treino, as classes majoritárias. Isso se deve, além da otimização das características ter sido feita apenas de forma geral (para todos os ataques e não ataques específicos), ao fato do desbalanceamento das classes ter causado o sobreajuste (*overfitting*) dos modelos. Em nosso trabalho, a seleção de características é feita de forma contínua dentro do AM por *Stream*, selecionando de forma dinâmica as características de acordo com o fluxo de chegada dos dados, não sendo necessário um estágio prévio de análise das características, o qual, em um sistema *on-line* poderia causar um acréscimo significativo com relação ao tempo de detecção dos ataques. Além disso, as bases NSL-KDD e UNSW-NB15 serão utilizadas em nosso estudo de caso.

Khan et al. (2019) apresenta um sistema de dois estágios de detecção e classificação de ataques baseado em Aprendizado Profundo [Khan et al. 2019], utilizando em ambos estágios a Rede Neural do tipo *auto-encoder*. O primeiro estágio é responsável por detectar a presença de atividade maliciosa na rede, classificando os dados de tráfego de rede entre normal ou anormal. O segundo estágio é responsável por classificar os dados como pertencentes à classe normal ou entre uma das classes de ataque. O sistema proposto é avaliado nas bases de dados KDD99 e UNSW-NB15, obtendo acurácia de detecção da presença de ataques próxima de 100% para o KDD99 e próxima de 90% para o UNSW-NB15. Em termos de classificação dos ataques, para a base KDD99 os resultados em todos os ataques foram próximos de 100%, porém, assim como em Tian et al (2021), para o UNSW-NB15 o sistema proposto teve apenas bons resultados para os ataques de classes majoritárias. Os autores também apresentam os resultados do tempo necessário em cada estágio da sua abordagem, e também o tempo por instância analisada. Em nosso trabalho, a abordagem de Aprendizado Profundo irá compor o sistema proposto, com o objetivo de melhorar a precisão da classificação do comportamento normal da rede.

Em nossa pesquisa anterior [Abreu et al. 2021] é apresentado um sistema de Aprendizado por *Stream*, que integra a seleção de características com criação dos modelos e aplicação de janelas deslizantes para a detecção de ataques de rede. Técnicas de AM por *Stream* são utilizadas para detectar a ocorrência de ataques em fluxo de rede analisado de forma *on-line*, sendo que o HAT teve o melhor custo benefício entre acurácia e tempo. No sistema proposto neste trabalho, o Aprendizado por *Stream* irá compor a parte de detecção da ocorrência de ação maliciosa na rede. Desta forma, o sistema proposto neste trabalho explora as limitações presentes nos trabalhos relacionados apresentados nesta seção.

4. Sistema Proposto

De forma geral, o sistema proposto pode ser entendido como uma sequência de processos de Aprendizagem de Máquina. O sistema tem como entrada os dados do tráfego de rede que está sendo analisado e, como saída, o sistema apresenta os dados classificados de acordo com classes de tráfego e as métricas de avaliação do sistema. A Figura 4 apresenta o funcionamento do sistema. O sistema possui 3 estágios que realizam a classificação do tráfego de rede e a detecção de ataques.

Como a Figura 4 apresenta, o Estágio 1 é a primeira parte do sistema. O fluxo de dados da rede é analisado e classificado entre dados de tráfego considerados como

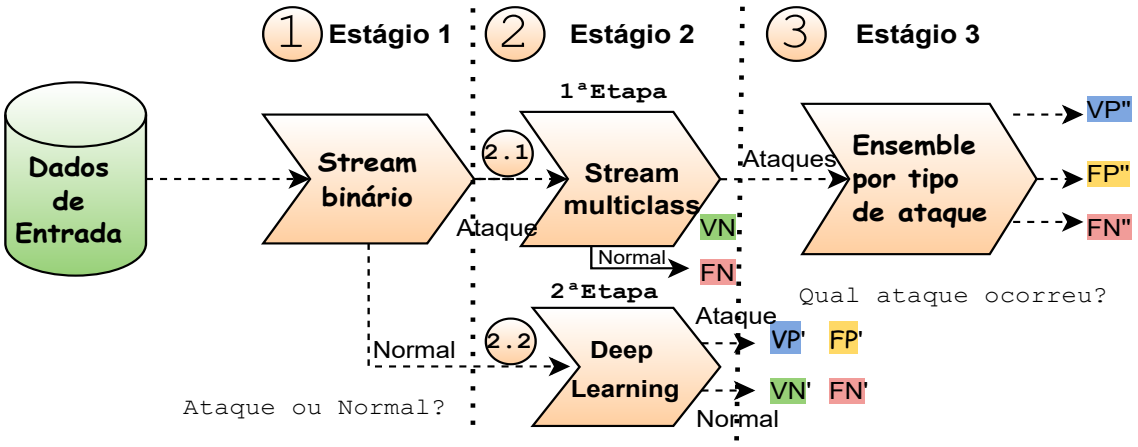

Figura 4. Sistema de Detecção e Classificação de tráfego de rede.

normais ou como de ataque. Assim, para esse estágio é implementado um sistema de detecção baseado em AM por *Stream* utilizando o método HAT. Nesse estágio o HAT tem um alvo binário e classifica os dados nas duas classes: tráfego normal e tráfego de ataque. A cada treino, ou re-treino do modelo, a técnica de seleção de características por clusterização [Abreu et al. 2020] é aplicada, melhorando o desempenho do HAT. As instâncias classificadas são enviadas em fluxo contínuo para o Estágio 2, na medida em que vão sendo rotulados pelo HAT. Os dados considerados como ataque são enviados para a Etapa 1 e os dados considerados como normal são enviados para a Etapa 2.

De modo semelhante ao Estágio 1, na Etapa 1 do Estágio 2 os dados são classificados de acordo com o fluxo contínuo de chegada dos dados do tráfego da rede. Assim, o Aprendizado de Máquina por *Stream* é utilizado para classificar os dados que já foram sinalizados no Estágio 1 como ataque, agora entre tipos de ataques. Deste modo, a versão multiclasse, que considera como alvo múltiplas classes de dados, do HAT foi implementada. Desta forma, ao final desta etapa, os dados estarão classificados entre as diferentes classes dos dados, os tipos de ataque e a classe normal. Os dados que forem classificados como normal, terão sua análise finalizada, e receberão o rótulo final de classe normal. Deste modo, ao final da Etapa 1 do Estágio 2, temos Verdadeiros Negativos (VN), dados que realmente pertencem a classe normal, e Falsos Negativos (FN), dados que foram classificados incorretamente como normal, mas que pertencem a algum tipo de ataque. De outro modo, os dados que forem classificados entre os tipos de ataques seguem em fluxo contínuo para o Estágio 3, na medida em que são rotulados pelo Estágio 2.

A Etapa 2 do Estágio 2 consiste em realizar a verificação dos dados sinalizados no Estágio 1 como dados normais. Nessa etapa é aplicado um modelo de Aprendizagem Profunda para classificar os dados de forma multiclasse, entre as diferentes classes dos dados. Embora os dados de entrada dessa etapa tenham sido sinalizados como dados normais, é necessário uma segunda etapa de análise para reduzir a possibilidade de falsos positivos e de falsos negativos. Esta etapa, então, requer um modelo robusto, capaz de classificar os dados com mais precisão. Assim, a metodologia escolhida consiste em aplicar uma Rede Neural Recorrente do tipo LSTM, a qual será utilizada para classificar os dados de forma mais precisa. Ao final da Etapa 2, temos: Verdadeiros Negativos (VN'), dados normais

corretamente verificados como pertencentes ao fluxo normal da rede; Verdadeiros Positivos (VP'), dados de ataques que anteriormente haviam sido sinalizados como normal, porém que a Etapa 2 verificou que na verdade são pertencentes à ataques; Falso Positivos (FP'), dados de comportamento normal que foram classificados como ataque pela Etapa 2; Falso Negativos (FN'), ataques que não foram detectados nem pelo Estágio 1 e nem pela Etapa 2.

O Estágio 3 recebe os dados considerados como ataques no Estágio 2, separados por tipo de classe. Assim, esse estágio serve para verificar o real pertencimento dos dados a essa classe de modo a reduzir o número de falsos positivos. Nesse estágio, cada ataque tem um modelo específico, que irá verificar se esse dado pertence realmente a esse tipo de ataque, ou se trata de um falso positivo. Assim, nesse estágio, classificadores *Ensemble* são utilizados para gerar modelo específico e preciso para cada tipo de ataque. Os classificadores RF, kNN e SVM são aplicados aos dados do Estágio 3 e realizam a classificação baseada na votação majoritária de suas predições. Ao final do Estágio 3, os dados são classificados por tipo de ataque, e assim temos: Verdadeiros Positivos (VP''), ataques corretamente classificado em sua classe de ataque; Falso Positivos (FP''), ataques incorretamente classificados entre as classes de ataque, Falso Negativo (FN''), ataques incorretamente classificado como da classe normal.

Assim, para obter os resultados do desempenho total do sistema, os valores de Verdadeiro Positivo Total (VP_t), Falso Positivo Total (FP_t), Verdadeiro Negativo Total (VN_t), Falso Negativo Total (FN_t), são calculados seguindo as equações 1, 2, 3 e 4:

$$VP_t = VP' + VP'' \tag{1}$$

$$FP_t = FP' + VP'' \tag{2}$$

$$VN_t = VN + VN' \tag{3}$$

$$FN_t = FN + FN' + FN'' \tag{4}$$

O VP_t, Equação 1, e o FP_t, Equação 2, são relativos aos estágios onde ocorre a atribuição da classe de ataque, sendo que essa classificação pode ser correta (VP_t) ou incorreta (FP_t). Já o VN_t, Equação 3, e o FN_t, Equação 4, são relativos a atribuição de classe normal, tanto de instâncias verdadeiramente pertencentes a classe normal (VN_t), quando de dados de ataque que não foram corretamente identificados (FN_t). Com base nessas equações, são calculadas as métricas de Acurácia (ACC), Precisão (Prec), Taxa de Verdadeiro Positivo (*True Positive Rate* - TPR), Taxa de Falso Positivo (*False Alarm Rate* - FAR) e métrica F1 (*F1 score*), descritas em detalhes em Carvalho et al. (2019) [Carvalho et al. 2019], e o tempo, tanto com relação ao desempenho total do sistema quanto pelo desempenho parcial em cada estágio do sistema.

É importante ressaltar que no sistema proposto, os dados são enviados em fluxo contínuo entre os estágios. Na medida em que uma instância é rotulada, ela é enviada para a próxima parte do sistema, não sendo necessário terminar de processar todas as instâncias de um estágio para começar outro. Além disso, as Etapas 1 e 2 do Estágio 2 ocorrem de forma independente. Assim, o sistema é finalizado quando são rotuladas todas as instâncias da base de dados. Dependendo do tempo necessário em cada etapa, isso pode ocorrer tanto ao final do Estágio 3, quanto ao final da Etapa 2 do Estágio 2.

5. Estudo de Caso

Para avaliar o sistema proposto foi realizado um estudo de caso utilizando três bases de dados de referência em segurança de redes: NSL-KDD, UNSW-NB15 e CICIDS17. Essas bases contém dados de tráfego de rede, tanto de comportamento normal quanto de diversos tipos de ataques, e estão entre as mais utilizadas no contexto de detecção e classificação de ataques usando técnicas de AM [Khraisat et al. 2019]. A seguir é descrito brevemente as três bases usadas neste estudo de caso:

- **NSL-KDD**: Possui 41 características de rede e 148.517 instâncias relativas ao comportamento normal da rede e a quatro tipos de ataques: *Denial of Service Attack* (DoS), *User to Root* (U2R), *Remote to Local* (R2L) e ataques *Probe*.
- **UNSW-NB15**: Possui 48 características de rede e 257.673 instâncias relativas ao comportamento normal e 9 tipos diferentes de ataques: *Generic, Exploits, Fuzzers, DoS, Reconnaissance, Analysis, Backdoor, Shellcode, Worms*.
- **CICIDS17**: A base contém 2.830.743 instâncias e 84 características de rede, e agrega dados relativos ao comportamento normal da rede e de 7 classes de ataques: *Web Attack (SSH-Patator, FTP-Patator, Sql Injection, XSS), DoS (Hulk, GoldenEye, Slowloris, Slowhttptest, Heartbleed), Brute Force, Infiltration, Bot, Portscan, DDoS*.

As bases de dados são utilizadas como entrada de dados para o sistema proposto, lidas como um fluxo contínuo de dados. Os experimentos foram realizados utilizando uma máquina com processador Intel Core i5-5200U com 2.20 GHz e com 8 GB de memória RAM disponível, utilizando o sistema operacional Windows 10 x64.

6. Resultados e Discussões

Nesta seção são apresentados os resultados obtidos com o sistema proposto nas três bases de dados apresentadas no estudo de caso. A Tabela 1 apresenta o resultado obtido ao final do processo de classificação dos dados nas bases de dados NSL-KDD, UNSW-NB15 e CICIDS17, de acordo com as equações 1, 2, 3 e 4.

Na Tabela 1, podemos observar que o sistema proposto teve um desempenho acima de 90% nas métricas avaliadas, em todas as bases de dados utilizadas no estudo de caso. Além disso, o valor de falsos alarmes é inferior a 10% em todos os casos. O sistema teve ACC e TPR similares entre as bases NSL-KDD (98.30%) e CICIDS17 (98.63%), porém, a precisão e o F1 na base NSL-KDD (97.58% e 98.49%) é superior ao obtido no CICIDS17 (94.27% e 96.43%). Desta forma, temos que o sistema teve o seu melhor desempenho na base NSL-KDD, seguido de CICIDS17, e por último na UNSW-NB15, que teve os menores valores nas métricas avaliadas.

Tabela 1. Resultado Total do Sistema.

Base de Dados	ACC%	Prec%	TPR%	FAR%	F1%
NSL-KDD	98.30	97.58	99.42	2.42	98.49
UNSW-NB15	93.93	90.32	98.46	9.68	94.22
CICIDS17	98.63	94.27	98.70	5.73	96.43

Tabela 2. Desempenho por estágio - Base NSL.

Estágio	ACC%	Prec%	TPR%	FAR%	F1%
1	95.61	97.05	95.32	2.95	96.18
2. Etapa 1	99.81	98.60	96.44	1.40	97.51
2. Etapa 2	99.17	99.50	99.63	0.50	99.56
3	98.67	99.66	98.98	0.34	99.32

Tabela 3. Desempenho por estágio - Base UNSW-NB15.

Estágio	ACC%	Prec%	TPR%	FAR%	F1%
1	94.54	96.37	93.61	3.63	94.97
2. Etapa 1	98.02	98.32	99.63	1.68	98.97
2. Etapa 2	99.46	93.23	99.63	6.77	96.33
3	95.39	99.26	95.88	0.74	97.54

As tabelas 2, 3 e 4 apresentam os resultados para cada estágio do sistema, em cada uma das bases do estudo de caso. Nessas tabelas é possível observar o desempenho específico de cada estágio do sistema proposto, em termos das métricas avaliadas. De forma geral, podemos observar que os resultados de ACC, Prec, TPR e F1 obtidos em todos os estágios estão acima de 90%, e que os resultados de FAR também estão abaixo de 10%, assim como no resultado total do sistema, apresentado na Tabela 1. Uma exceção a isso é o Estágio 1 da base CICIDS17, Tabela 4, que teve Prec (88.12%) e TPR (71.3%) menores que 90%, e FAR (11.88%) superior aos 10%, sendo estes os piores resultados deste estudo de caso.

Além disso, podemos observar nas tabelas 2, 3 e 4, a tendência de as métricas ACC, Prec, TPR terem menor valor no Estágio 1, o qual aumenta no Estágio 2 e tem uma redução no Estágio 3. Isso ocorre devido ao comportamento da classificação utilizada em cada etapa do sistema. No Estágio 1, temos um sistema de classificação binário em *Stream*, o qual proporciona análise dos dados em tempo real, porém, pelo fato de que não existir um treino prévio dos modelos, o sistema tende a errar mais nesta classificação, ocasionando em menores valores nas métricas avaliadas. Já no Estágio 2, a Etapa 1 é uma classificação *Stream* multiclasse, a qual já não recebe grande parte dos dados considerados normais, assim tem um desempenho melhor que o sistema binário da Etapa 1. Já na Etapa 2, é implementado um método de Aprendizado Profundo, o qual também não recebe vários dados que foram considerados de ataque no Estágio 1, e assim, tende a ter um resultado melhor do que o Estágio 1. No caso do Estágio 3, é feita a decisão multiclasse de todos os dados que foram considerados como ataque, o que reduz a acurácia.

As tabelas 5, 6, 7 apresentam os resultados obtidos por tipo de classe de tráfego, para cada base de dados. Nessas tabelas são destacados os resultados de detecção de cada classe e os falsos alarmes gerados pelo sistema para essa classe específica. Na Tabela 5, temos o desempenho por tipo de classe para a base NSL-KDD. Podemos observar que a classe normal tem o maior TPR (99.31%) e o menor FAR (0.13%), seguido do *DoS*, *U2R*, *Probe* e *R2L*. Isto segue a tendência de as classes com maior quantidade de dados

Tabela 4. Desempenho por estágio - Base CICIDS17.

Estágio	ACC%	Prec%	TPR%	FAR%	F1%
1	92.47	**88.12**	**71.3**	11.88	78.87
2. Etapa 1	94.95	95.40	98.82	4.60	97.08
2. Etapa 2	99.74	97.65	98.43	2.35	98.04
3	94.33	99.70	94.50	0.30	97.03

disponíveis terem melhores resultados. Por outro lado o U2R teve um FAR significativamente superior aos demais, com 6.75%, isso pode ser resultado da pouca quantidade de dados disponível para treino para essa classe.

Tabela 5. Desempenho por Tipo de Classe de Tráfego - Base NSL-KDD.

Classe	TPR%	FAR%	Instâncias
Normal	99.31	0.13	77.054
DoS	99.26	0.23	53.385
U2R	96.50	**6.76**	252
R2L	94.70	0.69	3.749
Probe	95.79	0.47	14.077

Tabela 6. Desempenho por Tipo de Classe de Tráfego - Base UNSW-NB15.

Classe	TPR%	FAR%	Instâncias
Analysis	91.29	4.63	2.677
Backdoor	83.36	5.63	2.329
DoS	**77.30**	0.97	16.353
Exploits	91.73	0.40	44.525
Fuzzers	82.98	3.03	24.246
Generic	95.44	0.50	58.871
Reconnaissance	89.24	2.35	13.987
Shellcode	**71.43**	**32.84**	1.511
Worms	90.91	**52.38**	174
Normal	99.59	0.03	93.000

Na Tabela 6 é apresentado o resultado por tipo de ataque da base UNSW-NB15. Podemos observar, novamente, que a classe de tráfego Normal teve o maior TPR (99.59%) e menor FAR (0.03%), seguindo também das classes de tráfego com maior quantidade de dados como *Generic* e *Exploits*. Entretanto, alguns dos ataques mais frequentes tiveram TPR significativamente menor de 90%, com o *DoS* (77.30%) e *Fuzzers* (82.98%). Além disso, é possível observar que os ataques *Shellcode* e *Worms* tiveram FAR expressivos, muito superiores aos demais, respectivamente 32.84% e 52.38%. Isso ajuda a explicar o fato de que o sistema teve o seu pior desempenho geral, observado na Tabela 1, na base UNSW-NB15.

Na Tabela 7 é apresentado o desempenho por classe de tráfego na base CICIDS17. Nessa tabela temos que, assim como nas outras bases, a classe normal tem o melhor valor

Tabela 7. Desempenho por Tipo de Classe de Tráfego - Base CICIDS17.

Classe	TPR%	FAR%	Instâncias
DoS	94.50	1.14	252.672
PortScan	93.87	1.37	158.930
DDoS	94.36	0.81	128.027
Brute Force	93.46	1.49	13.835
Web Attack	96.79	7.05	2.180
BoT	94.40	**10.36**	1.966
Infiltration	90.63	**59.72**	36
Normal	99.70	0.16	2.273.097

de TPR (99.70%) e de FAR (0.16%). Todos os ataques têm TPR acima de 90% e a maioria tem FAR menor que 10%. Os ataques *BoT* e *Infiltration* tem FAR acima de 10%, respectivamente 10.36% e 59.72%. Assim com o *U2R* da base NSL-KDD e o *Worms* da base UNSW-NB15, o ataque *Infiltration* possui a menor quantidade de dados disponíveis, o que impacta diretamente, não só no TPR mas também no FAR, já que os modelos gerados para esses ataques não são precisos o suficientes para classificar corretamente esses ataques. Assim, de forma geral, o tráfego normal teve a melhor detecção em todas as bases de dados. Isso se deve, principalmente, à Etapa 2 do Estágio 2, onde o LSTM é utilizado para realizar a verificação dos dados considerados como normal no Estágio 1. Esse fator contribui para a redução de falso positivo no sistema, já que a maioria do tráfego normal é corretamente classificado.

A Tabela 8 apresenta o tempo, medido em segundos, relativo aos experimentos do estudo de caso. São apresentados, para cada base de dados, o tempo total necessário para detectar e classificar os dados de tráfego de rede, a quantidade de instâncias identificadas por segundo (inst./s), e o tempo gasto em cada estágio do sistema proposto. Podemos ver que o tempo necessário para a base CICIDS17 é muito superior ao das outras bases de dados. Isso ocorre devido ao seu tamanho ser muito superior às demais bases analisadas. Por outro lado, a velocidade com que os dados são processados nesta base é muito maior.

Tabela 8. Tempo de detecção por estágio e por base de dados.

Base de Dados	Tempo Total(s)	inst./s	1 (s)	2.1 (s)	2.2 (s)	3 (s)
NSL-KDD	114.88	196.22	43.35	43.88	86.17	74.25
UNSW-NB15	272.88	301.70	158.11	194.26	270.88	255.85
CICIDS17	2254.84	1129.86	1959.74	867.75	2253.84	936.57

Além disso, podemos observar que a Etapa 2 do Estágio 2 apresenta o tempo determinante para o tempo total do sistema, sendo o estágio mais demorado do sistema. Isso se deve ao fato de que nessa etapa é realizado o processo de classificação dos dados utilizando o LSTM, o qual tende a ser um método mais demorado que o utilizado nos demais estágios. Porém, como os resultados das tabelas 2,3 e 4, demonstram, essa etapa possui um alto desempenho, com ACC e Prec próximos de 100%, e também, como as tabelas 5,6 e 7, expressam, isso é refletido, principalmente, no alto valor de TPR e baixo FAR da classe Normal. Assim, embora seja o estágio mais demorado, essa etapa mostra-se fundamental ao sistema proposto.

7. Conclusão e Trabalhos Futuros

A detecção de ataques de rede de forma acurada e precisa é um grande desafio nas redes atuais. Neste trabalho foi proposto um sistema *on-line* que utiliza de forma híbrida técnicas de AM em três estágios de detecção e classificação de ataques na rede. A avaliação feita utilizando três bases de dados de referência mostrou que o sistema proposto é tem acurácia e precisão acima de 90%, com taxas de falso alarmes reduzidas.

Foram apresentados os resultados por tipo de ataque, nos quais o sistema obteve bons resultados para a maioria das classes, mesmo para os ataques que tem poucos dados disponíveis, com valores de TPR acima de 90% em vários casos. Isso é reflexo da adequação dos métodos de AM a cada estágio do sistema, que, de forma híbrida, combinou o AM por *Stream* para dar maior velocidade à detecção, e o AM por Aprendizado Profundo e *Ensemble* para melhorar a precisão e acurácia do sistema.

Como trabalhos futuros, podem ser exploradas a aplicação de outras técnicas de AM e outras bases de dados utilizando o sistema proposto. Além disso, a implementação distribuída do sistema pode ser abordada utilizando SDN, balanceando e alocando em dispositivos diferentes cada estágio do sistema. O sistema proposto pode ser implementado em um IDS ou IPS que seja capaz de processar os dados de entrada e aplicar os métodos de AM. Por exemplo, o Estágio 1 e a Etapa 1 do Estágio 2, aplicam métodos de AM que não precisam de muita memória ou tempo, e podem ser implementadas em *switches* SDN. Já, a Etapa 2 do Estágio 2 e o Estágio 3, podem ser implementados em dispositivos dedicados, que tenham maior poder de processamento.

Agradecimentos

Agradecemos ao apoio da Fundação de Amparo a Pesquisa do Estado de São Paulo (FAPESP), por meio do processo no 2020/04031-1.

Referências

Abreu, D., Carvalho, I., and Abelém, A. (2021). Seleção de Características em Stream para a Detecção de Ataques de Rede. In *Anais do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 197–210, Porto Alegre, RS, Brasil. SBC.

Abreu, D., Carvalho, I., Abelém, A. J., Menasché, D., Leão, R. M., and Silva, E. (2020). Seleção de Características por Clusterização para Melhorar a Detecção de Ataques de Rede. In *Proceedings of the 38th Brazilian Symposium on Computer Networks and Distributed Systems*, pages 295–308, Porto Alegre, RS, Brasil. SBC.

Altman, N. S. (1992). An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *The American Statistician*, volume 46, páginas 175-185(3).

Carvalho, D. V., Pereira, E. M., and Cardoso, J. S. (2019). Machine learning interpretability: A survey on methods and metrics. *Electronics*, 8(8):832.

Dhanabal, L. and Shantharajah, S. (2015). A study on nsl-kdd dataset for intrusion detection system based on classification algorithms. *International Journal of Advanced Research in Computer and Communication Engineering*, 4(6):446–452.

Khan, F. A., Gumaei, A., Derhab, A., and Hussain, A. (2019). A novel two-stage deep learning model for efficient network intrusion detection. *IEEE Access*, 7:30373–30385.

Khraisat, A., Gondal, I., Vamplew, P., and Kamruzzaman, J. (2019). Survey of intrusion detection systems: techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22.

Liang, X. and Kim, Y. (2021). A survey on security attacks and solutions in the iot network. In *2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0853–0859. IEEE.

Lobato, A. G. P., Andreoni Lopez, M., and Duarte, O. (2016). Um sistema acurado de detecção de ameaças em tempo real por processamento de fluxos. *XXXIV Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos-SBRC*.

Lorena, A. C. and De Carvalho, A. C. (2007). Uma introdução às support vector machines. *Revista de Informática Teórica e Aplicada*, 14(2):43–67.

Lucas, T. J., da Costa, K. A., Moraes, E. A., Júnior, P. R. H., and das Neves, M. J. (2021). Stacking-based committees para detecção de ataques em redes de computadores-uma abordagem por exaustão. In *Anais do XXXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, pages 644–657. SBC.

Maidamwar, P. R., Bartere, M. M., and Lokulwar, P. P. (2022). Implementation of network intrusion detection system using artificial intelligence: Survey. In *Proceedings of the 2nd International Conference on Recent Trends in Machine Learning, IoT, Smart Cities and Applications*, pages 185–198. Springer.

Malhotra, P., Vig, L., Shroff, G., Agarwal, P., et al. (2015). Long short term memory networks for anomaly detection in time series. In *Proceedings*, volume 89, pages 89–94.

Moustafa, N. and Slay, J. (2015). Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set). In *2015 Military Communications and Information Systems Conference (MilCIS)*, pages 1–6.

Polikar, R. (2012). Ensemble learning. In *Ensemble machine learning*, pages 1–34. Springer.

Rego, R. C. S. and Nunes, R. C. (2021). Detecção de ataques web: Explorando redes neurais recorrentes com redutor de dimensionalidade. In *Anais do XXI Simpósio Brasileiro em Segurança da Informação e de Sistemas Computacionais*, pages 183–196. SBC.

Salih, A., Zeebaree, S. T., Ameen, S., Alkhyyat, A., and Shukur, H. M. (2021). A survey on the role of artificial intelligence, machine learning and deep learning for cybersecurity attack detection. In *2021 7th International Engineering Conference "Research & Innovation amid Global Pandemic"(IEC)*, pages 61–66. IEEE.

Sharafaldin, I., Lashkari, A. H., and Ghorbani, A. A. (2018). Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *ICISSp*, pages 108–116.

Tian, Q., Han, D., Hsieh, M.-Y., Li, K.-C., and Castiglione, A. (2021). A two-stage intrusion detection approach for software-defined iot networks. *Soft Computing*, 25(16):10935–10951.