UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

CONTRIBUIÇÕES AO ESTUDO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO EM HOSTS PARA ATAQUES DE DIA ZERO

MARCELL BRUNO SOUSA E SILVA

JOÃO PESSOA-PB
março-2026

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

CONTRIBUIÇÕES AO ESTUDO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO EM HOSTS PARA ATAQUES DE DIA ZERO

MARCELL BRUNO SOUSA E SILVA

JOÃO PESSOA-PB
março-2026

II

MARCELL BRUNO SOUSA E SILVA

CONTRIBUIÇÕES AO ESTUDO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO EM HOSTS PARA ATAQUES DE DIA ZERO

DISSERTAÇÃO DE MESTRADO

Orientador: Prof. Dr. IGUATEMI EDUARDO DA FONSECA
Coorientador: Prof. Dr. YURI DE ALMEIDA MALHEIROS BARBOSA

Catalogação na publicação
Seção de Catalogação e Classificação

S586c Silva, Marcell Bruno Sousa e.
Contribuições ao estudo de sistemas de detecção de intrusão em hosts para ataques de dia zero / Marcell Bruno Sousa e Silva. - João Pessoa, 2025.
74 f.

Orientação: Iguatemi Eduardo da Fonseca.
Coorientação: Yuri de Almeida Malheiros Barbosa.
Dissertação (Mestrado) - UFPB/CI.

1. HIDS (Sistema de Detecção de Intrusão em Hosts). 2. Ataque de Dia Zero. I. Fonseca, Iguatemi Eduardo da. II. Barbosa, Yuri de Almeida Malheiros. III. Título.

UFPB/BC CDU 004.056.53(043)

Elaborado por JONISMAR KENDYS DA SILVA LEAO - CRB-4/2332

UNIVERSIDADE FEDERAL DA PARAÍBA
CENTRO DE INFORMÁTICA
PROGRAMA DE PÓS-GRADUAÇÃO EM INFORMÁTICA

Ata da Sessão Pública de Defesa de Dissertação de Mestrado de Marcell Bruno Sousa e Silva, candidato ao título de Mestre em Informática na Área de Ciência da Computação, realizada em 15 de dezembro de 2025.

Aos quinze dias do mês de dezembro do ano de dois mil e vinte e cinco, às nove horas, reuniram-se os membros da Banca Examinadora constituída para julgar o trabalho do sr. Marcell Bruno Sousa e Silva, vinculado a esta Universidade sob a matrícula nº 20241006978, candidato ao grau de Mestre em Informática, na área de "Ciência da Computação", na linha de pesquisa "Sistemas de Computação", do Programa de Pós-Graduação em Informática, da Universidade Federal da Paraíba. A comissão examinadora foi composta pelos professores: Iguatemi Eduardo da Fonseca (PPGI), Orientador e Presidente da Banca; Yuri de Almeida Malheiros Barbosa, coorientador; Tiago Maritan Ugulino de Araújo (PPGI), Examinador Interno; e João José da Costa Gondim (UnB), Examinador Externo. Dando início aos trabalhos, o Presidente da Banca cumprimentou os presentes, comunicou a finalidade da reunião e passou a palavra ao candidato para que ele fizesse a exposição oral do trabalho de dissertação intitulado: "Contribuições ao Estudo de Sistemas de Detecção de Intrusão em Hosts para Ataques de Dia Zero". Concluída a exposição, o candidato foi arguido pela Banca Examinadora que emitiu o seguinte parecer: "***aprovado***". Do ocorrido, eu, Gilberto Farias de Sousa Filho, Coordenador do Programa de Pós-Graduação em Informática, lavrei a presente ata que vai assinada por mim e pelos membros da banca examinadora. João Pessoa, 15 de dezembro de 2025.

Prof. Dr. Gilberto Farias de Sousa Filho

Prof. Iguatemi Eduardo da Fonseca
Orientador (PPGI-UFPB)

gov.br Documento assinado digitalmente
IGUATEMI EDUARDO DA FONSECA
Data: 16/12/2025 09:44:02-0300
Verifique em https://validar.iti.gov.br

Prof. Yuri de Almeida Malheiros Barbosa
Coorientador (PPGI-UFPB)

gov.br Documento assinado digitalmente
YURI DE ALMEIDA MALHEIROS BARBOSA
Data: 16/12/2025 09:53:28-0300
Verifique em https://validar.iti.gov.br

Prof. Tiago Maritan Ugulino de Araújo
Examinador Interno (PPGI-UFPB)

gov.br Documento assinado digitalmente
TIAGO MARITAN UGULINO DE ARAUJO
Data: 18/12/2025 10:28:57-0300
Verifique em https://validar.iti.gov.br

Prof. João José da Costa Gondim
Examinador Externo (UnB)

gov.br Documento assinado digitalmente
JOAO JOSE COSTA GONDIM
Data: 18/12/2025 11:15:11-0300
Verifique em https://validar.iti.gov.br

CONTRIBUIÇÕES AO ESTUDO DE SISTEMAS DE DETECÇÃO DE INTRUSÃO EM HOSTS PARA ATAQUES DE DIA ZERO

Dissertação aprovada em ___/___/___ como requisito para a obtenção do título de Mestre em Informática do Centro de Informática da Universidade Federal da Paraíba.

BANCA EXAMINADORA:

_____

Prof. Dr. Iguatemi Eduardo da Fonseca – UFPB
(Orientador)

_____

Prof. Dr. Yuri de Almeida Malheiros Barbosa – UFPB
(Coorientador)

_____

Prof. Dr. Tiago Maritan Ugulino de Araujo – UFPB
(Examinador Interno)

_____

Prof. Dr. João José Costa Gondim – UnB
(Examinador Externo)

Dedico este trabalho a Deus e à minha esposa Ane Josana.

AGRADECIMENTOS

Eu gostaria de agradecer:

- ao meu orientador Iguatemi Eduardo da Fonseca pela orientação e suporte,
- ao meu co-orientador Yuri Malheiros pelo acompanhamento e suporte a dúvidas durante a implementação das técnicas LSTM e Transformers, bem como definição dos rumos do trabalho no contexto de inteligência artificial,
- ao professor Gerd do departamento de Química por ter disponibilizado GPUs mais robustas para a realização de testes.

RESUMO

No contexto de uso onipresente de mídias e dados digitais na sociedade contemporânea a cibersegurança emerge como uma das áreas mais críticas. Nesse contexto, sistemas de identificação de intrusões em computadores emerge como uma ferramenta essencial para garantia da segurança de dados e mídias digitais. Este trabalho teve como objetivo avaliar os desempenhos das técnicas LSTM, *Transformer* e Mapa de Kohonen (SOM) na tarefa de identificação de ataques de dia zero (*zero-day attacks*) em hosts. Os modelos avaliados foram empregados dentro do paradigma de classificação não supervisionada, a qual não utiliza rótulos para as classes no processo de treinamento. O paradigma não supervisionado é capaz de identificar ataques novos e de minimizar a menor disponibilidade de dados de ataque diante da imensa quantidade de dados oriundos de operações normais em *hosts*. As medidas de desempenho adotadas foram taxa de detecção de intrusões, taxa de falsos alarmes, número de parâmetros e número de operações de ponto flutuante dos modelos adotados, de forma a avaliar não só a efetividade final de cada modelo, mas também considerar as suas complexidades computacionais. Ficou demonstrado que a técnica que apresentou melhor desempenho foi o Transformer com AUC (*Area Under the Curve*) de 0,990, porém o SOM apresentou menor complexidade computacional com uma AUC intermediária.

PALAVRAS-CHAVE: Cibersegurança; Sistemas de detecção de intrusão; Ataques de dia zero; Aprendizado de máquina.

ABSTRACT

In the context of the ubiquitous use of digital media and data in contemporary society, cybersecurity emerges as one of the most critical areas. In this context, computer intrusion detection systems emerge as an essential tool for ensuring the security of digital data and media. This work aimed to evaluate the performance of LSTM, Transformer, and Kohonen Map (SOM) techniques in identifying zero-day attacks on hosts. The evaluated models were employed within the unsupervised classification paradigm, which does not use labels for classes in the training process. The unsupervised paradigm is capable of identifying new attacks and minimizing the reduced availability of attack data in the face of the immense amount of data originating from normal host operations. The performance measures adopted were intrusion detection rate, false alarm rate, number of parameters, and number of floating-point operations of the adopted models, in order to evaluate not only the final effectiveness of each model but also to consider its computational complexities. It was demonstrated that the technique that presented the best performance was the Transformer with an AUC (Area Under the Curve) of 0.990, however, the SOM presented lower computational complexity with an intermediate AUC.

KEYWORDS: Intrusion Detection System, IDS, LSTM, Transformer, Self Organizing Kohonen Map.

SUMÁRIO

Índice de figuras

Índice de tabelas

CAPÍTULO 1

INTRODUÇÃO

A sociedade contemporânea vive o que é frequentemente chamado de "Era da Informação", um período em que dados e informações se tornaram a base de praticamente todas as atividades econômicas, políticas, sociais e culturais. A Revolução Digital tem transformado todos os setores da vida humana, desde a indústria até a educação, a proliferação de dispositivos conectados à internet, a chamada Internet das Coisas (IoT – *Internet of Things*), e a digitalização de processos e serviços. Tal transformação gerou uma quantidade massiva de dados. Estes dados, por sua vez, são valiosos ativos que moldam o comportamento do mercado, as decisões de negócios e até as políticas governamentais. A análise de grandes volumes de dados, conhecida como *Big Data*, tornou-se uma ferramenta fundamental para a tomada de decisões mais assertivas, previsões mais precisas e soluções inovadoras para problemas complexos.

No entanto, à medida que a dependência de dados cresce, a segurança da informação se torna uma preocupação central. A proteção desses dados contra ameaças externas e internas não é apenas uma questão de privacidade, mas também de segurança nacional, confiabilidade econômica e estabilidade social. Em 2023, a cada segundo, eram gerados mais de 2,5 quintilhões de bytes de dados no mundo, tornando o campo da cibersegurança cada vez mais relevante e urgente [1].

Os dados, que antes eram considerados um mero subproduto das interações humanas com as máquinas, agora são o combustível das economias digitais, de forma que sua integridade, disponibilidade, autenticidade e privacidade, pilares da cibersegurança, são essenciais para o funcionamento da sociedade moderna.

1.1 Principais Desafios da Cibersegurança e HIDS

A segurança da informação consiste no esforço no sentido da proteção das informações em um contexto amplo, de forma que ações tradicionais de armazenamento e guarda de manuscritos também se enquadram no conceito de segurança da informação [2].

Dentro da área de segurança da informação há a área de cibersegurança ou *cybersecurity*, cujo campo de atuação se dá na proteção de dados armazenados ou em trânsito em contextos computacionais. Por conta da grande importância que os dados no contexto computacional desempenham na sociedade contemporânea, a cibersegurança emergiu como uma área de destaque. A cibersegurança tem por objetivo proteger a integridade, a disponibilidade e confidencialidade de dados computacionais, ou cibernéticos [2].

Alguns dos principais desafios da cibersegurança são: ataques de dia zero (*zero-day attacks*), *ransomware*, *phishing*, ataques de negação de serviço distribuída (DDoS – *Distributed Denial of Service*), malware e invasões por exploração de vulnerabilidades. Merecendo destaque estão os ataques de dia zero, por conta dos aspectos originais que os acompanham e a grande dificuldade de detecção e mitigação verificadas por parte de técnicas tradicionais.

De forma a tratar de forma estruturada os problemas enfrentados no contexto da cibersegurança, diversos frameworks foram propostos para guiar o estabelecimento de um sistema de cibersegurança, alguns deles são [3][4][5]:

- CIS *Critical Security Controls*;
- NIST *Cybersecurity Framework*;
- COBIT (*Control Objectives for Information and Related Technologies*);
- MITRE ATT&CK;
- FISMA (*Federal Information Security Management Act*);
- NIST SP 800-53;
- ITIL (*Information Technology Infrastructure Library*);
- NICE Framework (*National Initiative for Cybersecurity Education*).

Os diversos *frameworks* possuem diversas similaridades entre si, sendo as principais diferenças no tocante ao escopo de atuação, ou objetivos, e no tocante ao foco. Por exemplo, o CIS - *Critical Security Controls* propõe ações prescritivas para o estabelecimento de um sistema de defesa contra ataque cibernéticos [3], já o MITRE ATT&CK apresenta de forma estruturada as diversas fases de implementação de ataques cibernéticos.

Como o presente trabalho desenvolve comparações entre três diferentes técnicas de defesa cibernética, as técnicas de defesa apresentadas serão contextualizadas no framework CIS *Critical Security Controls*, porém é possível a devida contextualização nos demais frameworks citados. O framework CIS foi desenvolvido pela instituição *Center for Internet*

Security e na sua criação teve por objetivo disseminar melhores práticas de cibersegurança para pessoas e empresas [3]. É um framework bastante popular na comunidade de cibersegurança e em sua Versão 8 é estruturado em 18 controles, são eles [3]:

- Controle 1: Inventário e Controle de Ativos Institucionais;
- Controle 2: Inventário e Controle de Ativos de Software;
- Controle 3: Proteção de Dados;
- Controle 4: Configuração Segura de Ativos Institucionais e Software;
- Controle 5: Gestão de Contas;
- Controle 6: Gestão de Controle de Acesso;
- Controle 7: Gestão Contínua de Vulnerabilidades;
- Controle 8: Gestão de Registros de Auditoria;
- Controle 9: Proteções de E-mail e Navegador Web;
- Controle 10: Defesas Contra Malware;
- Controle 11: Recuperação de Dados;
- Controle 12: Gestão da Infraestrutura de Rede;
- Controle 13: Monitoramento e Defesa da Rede;
- Controle 14: Conscientização e Treinamento de Competências sobre Segurança;
- Controle 15: Gestão de Provedor de Serviços;
- Controle 16: Segurança de Aplicações;
- Controle 17: Gestão de Resposta a Incidentes;
- Controle 18: Testes de Invasão.

O Controle 13 tem por objetivo o monitoramento e a defesa da rede, e é implementado por meio de uma série de ações, dentre elas:

✔️ 13.1 Centralizar o alerta de eventos de segurança;

✔️ **13.2 Implantar solução de detecção de intrusão baseada em host (HIDS);**

✔️ 13.3 Implantar uma solução de detecção de intrusão de rede;

✔️ 13.6 Coletar logs de fluxo de tráfego da rede;

✔️ 13.11 Ajustar Limites de Alerta de Eventos de Segurança.

Algumas das principais técnicas contidas no Controle 13 são:

- sistemas de gerenciamento de informações e eventos de segurança (SIEM);
- sistemas de detecção de intrusão em redes (NIDS);
- **sistemas de detecção de intrusão em hosts (HIDS).**

Os sistemas do Controle 13 apresentam grande efetividade na detecção de ataques conhecidos, porém apresentam efetividade limitada na detecção de ataques de dia zero. Portanto, estudar e desenvolver novas técnicas para identificar vulnerabilidades e ataques de dia zero é um tema central em cibersegurança.

1.2 Ataques de Dia Zero e sua Relevância no Contexto de Segurança da Informação

O ataque de dia zero (zero-day attack) é um dos conceitos mais críticos no campo da segurança cibernética. Esse termo refere-se a uma exploração que ocorre antes que o fornecedor de um software ou sistema tenha conhecimento da vulnerabilidade associada. Em outras palavras, trata-se de uma falha de segurança previamente desconhecida, para a qual não existem correções, patches ou medidas de mitigação disponíveis no momento do ataque. O nome "dia zero" advém do fato de que os desenvolvedores possuem exatamente zero dias para corrigir a falha antes que ela seja explorada por agentes maliciosos [6].

A principal característica que diferencia os ataques de dia zero de outros tipos de ataques é justamente o elemento da novidade e imprevisibilidade. Enquanto ataques tradicionais, como *brute force*, *phishing*, DoS/DDoS ou exploração de vulnerabilidades já catalogadas (CVE — *Common Vulnerabilities and Exposures*), baseiam-se em técnicas amplamente documentadas e, em muitos casos, mitigáveis por boas práticas de segurança, os ataques de dia zero se destacam pela ausência de precedentes. Isso significa que, mesmo

sistemas devidamente atualizados e protegidos, ainda assim podem ser vítimas desse tipo de exploração.

Essa característica coloca os ataques de dia zero em um patamar de risco mais elevado quando comparados a outros vetores de ataque. Diferentemente de ameaças conhecidas, que podem ser reduzidas por meio de atualizações, antivírus ou monitoramento de assinaturas. As falhas exploradas em dia zero exigem abordagens proativas de defesa, como detecção baseada em comportamento, análise estatística de anomalias e sistemas de inteligência de ameaças capazes de identificar padrões atípicos de uso. Neste contexto, ataques de dia zero possuem relevância estratégica tanto para cibercriminosos quanto para atores estatais, uma vez que a posse de uma vulnerabilidade desconhecida confere vantagens significativas em espionagem, sabotagem ou obtenção de ganhos financeiros ilícitos. O mercado clandestino, inclusive, movimenta valores elevados na comercialização de *exploits* de dia zero, o que reforça sua importância no cenário global da segurança da informação [7].

Portanto, compreender o conceito e a gravidade dos ataques de dia zero é fundamental para detecção e mitigação de intrusões. Ao contrário de ataques convencionais, que podem ser enfrentados com soluções já consolidadas, as ameaças de dia zero demandam estratégias adaptativas e de aprendizado contínuo, reforçando a necessidade de modelos de defesa baseados em inteligência artificial, aprendizado de máquina e análise preditiva, capazes de oferecer proteção mesmo diante do desconhecido.

1.3 Objetivos e Contribuições

O presente trabalho tem por objetivo geral propor HIDS baseado na arquitetira resultante da combinação de Transformer e sistema de detecção de mudanças, capaz de detectar ataques de dia zero. De forma secundária são desenvolvidas as seguintes arquiteturas como baselines de comparação: LSTM + sistema de detecção de mudanças, e SOM + sistema de detecção de mudanças. A Figura 1 apresenta um diagrama esquemático das arquiteturas descritas.

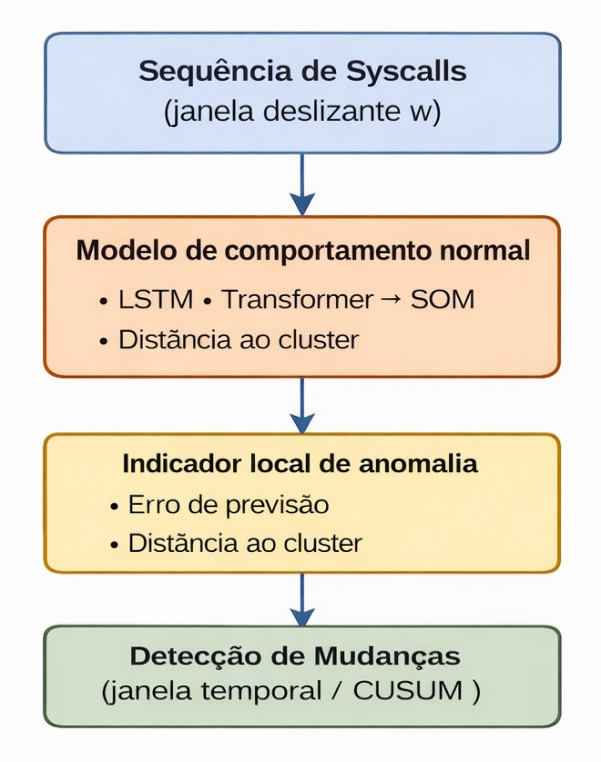

Figura 1: Diagrama de Blocos dos HIDSs Analisados

As avaliações dessas técnicas se deram quanto à capacidade de identificar com sucesso o máximo possível de intrusões em hosts, ao mesmo tempo em que apresenta uma quantidade mínima de falsas detecções de intrusão, bem como quanto às complexidades computacionais de cada modelo.

O sistema de detecção de mudanças utilizado teve por objetivo ajudar a reduzir a quantidade de falsos alarmes. Além disso, possibilitou conversão da natureza supervisionada do Transformer e do LSTM em natureza não supervisionada, e possibilitou comparações entre técnicas de natureza distinta.

Para alcançar o objetivo geral proposto, os seguintes objetivos específicos foram buscados:

- Investigar se o uso de técnicas de aprendizado de máquina do estado da arte é capaz de identificar ataques de dia zero em hosts;
- Averiguar a eficácia do uso de métricas de detecção para identificação de comportamentos anormais em hosts;
- Implementar e testar, em ambiente controlado, as técnicas descritas.

As contribuições do presente trabalho foram as seguintes:

- utilização de arquitetura *Transformer* para modelagem dos comportamentos normais dos *hosts*. Para o melhor do nosso conhecimento, esse é o primeiro trabalho da literatura que utiliza tal técnica combinada com técnica de detecção de mudanças no contexto de ataques de dia zero em HIDS;
- utilização de arquitetura Mapas Auto-organizáveis de Kohonen para modelagem dos comportamentos normais dos *hosts* no contexto da base de dados ADFA-LD;
- utilização de técnica de detecção de mudanças para detecção de comportamentos anormais em hosts;
- utilização no contexto de HIDS das métricas: número de parâmetros dos modelos e número de operações de ponto flutuante (FLOP - *Floating-Point Operations*) dos modelos.

1.4 Estrutura da Dissertação

O restante do texto está organizado da seguinte forma. No Capítulo 2 são apresentados os trabalhos relacionados às técnicas de HIDS. No Capítulo 3 é apresentado o referencial teórico, bem como as principais técnicas, métricas e bases de dados no contexto de HIDS. No Capítulo 4 são descritos os principais parâmetros e configurações utilizadas pelas metodologias descritas. No Capítulo 5 são apresentados os resultados obtidos. No Capítulo 6, as principais conclusões do trabalho são apresentadas, bem como ideias para desdobramentos futuros.

CAPÍTULO 2

TRABALHOS RELACIONADOS

Sistemas de detecção de intrusão em hosts (HIDS – *host intrusion detection system*) têm sido alvos de muitas pesquisas acadêmicas já que têm papel preponderante em identificar comprometimentos de hosts [8][9].

Atualmente o principal esforço nas pesquisas de HIDS se dão no sentido de identificar ataques de dia zero, uma vez que sistemas bem protegidos conseguem lidar bem com ataques conhecidos, e frequentemente falham em identificar ataques desconhecidos o que tem gerado enormes prejuízos a coorporações, governos e até mesmo a usuários comuns [8][9][6][7].

Comumente, HIDS são abordados como alternativa de identificação de intrusões quando NIDS falham no monitoramento do perímetro [9]. Porém é sabido que existem ataques que utilizam dispositivos de armazenamento portáteis como *pendrives*, cartões de memória e similares; nesses casos uma das principais alternativas para identificação dos ataques é o HIDS. Outro aspecto da utilização de HIDSs é a maior precisão e capacidade de detecção de intrusões do que NIDSs, visto que os dados dos *hosts* possuem maiores riquezas de detalhes do que os dados comumente utilizados em NIDSs [10].

De forma a estruturar melhor o entendimento científico na área de HIDSs, diversas revisões sistemáticas de literatura foram realizadas [8][9][11][12][13]. A revisão sistemática da literatura em [8] aborda de forma muito bem estruturada trabalhos acadêmicos de pesquisa na área de HIDS, abordando os principais conceitos e metodologias de detecção utilizadas nos projetos de HIDSs, aponta os principais trabalhos na área entre 2010 e 2022. Essa revisão sistemática é a principal fonte de informações do presente trabalho quanto ao estado da arte atual.

De forma a mitigar o lapso temporal da revisão acadêmica citada, bem como seu escopo que deixou de fora o universo dos dispositivos da internet das coisas (IoT) e a técnica SOM, outras revisões sistemáticas da literatura são abordadas [9][11][12][13] e é feito uma revisão sistemática da literatura entre 2021 e 2025 com os critérios descritos em [8] sem excluir dispositivos IoT e técnicas de clusterização como o SOM. As bases de dados

pesquisadas por [8] foram: Scopus Digital Library, IEEE Xplorer, ACM Digital Library. A string de busca nessas bases foi: "host intrusion detection system" or "host based intrusion detection" or "host anomaly detection" or "host based anomaly detection" or "host based anomaly intrusion detection" or "host based ids" or "host ids" or "host based misuse intrusion detection" or ("signature based intrusion detection" and "host").

Após o levantamento em [8], foram obtidos 606 artigos, após isso foram removidos: trabalhos duplicados; trabalhos não escritos em Inglês; trabalhos que abordavam IoT, cloud ou smart cities; trabalhos que não tinham sido publicados em revistas ou conferências. Após a remoção foi feito segunda seleção baseada na qualidade dos artigos, de forma que ao final restaram 65 artigos. Os artigos selecionados em [8] abordam o contexto de HIDS sob uma perspectiva mais alinhada à área de processamento de linguagem natural (PLN) sendo utilizados os *system calls* (SC), ou *syscalls*, dos sistemas operacionais como subsídio a tais processamentos. Acredita-se que a utilização de SCs em HIDSs seja mais seguro do que a utilização de logs de sistemas, uma vez que a produção de logs de sistemas é realizada por meio de regras de interpretações de dados brutos os quais frequentemente adicionam vieses e ruídos às informações, além de estar sujeito a alterações pelo atacante [10][14]. Outro problema resultante da utilização de dados de logs para HIDSs é o atraso decorrente das gerações dos logs, o que pode impactar negativamente em aplicações em tempo real [14].

Na revisão acadêmica em [9], de 2024, são considerados trabalhos que utilizaram técnicas de Aprendizado de Máquina (ML – *machine learning*) e técnicas de *deep learning* (DL), as quais foram subsidiadas por: SCs, arquivos do tipo *dynamic link library* (DLL), logs dos sistemas, chaves de registros de sistemas operacionais e análises de performance. Outro aspecto importante da revisão acadêmica em [9] é que o mesmo também apresenta soluções de detecção de intrusão em redes (NIDS – *network intrusion detection system*), bem como soluções de detecção de intrusão baseados em proveniência (PIDS). Acredita-se que tal esforço se dá por conta de similaridades contextuais entre NIDSs e HIDSs.

PIDS (*Provenance-based Intrusion Detection Systems*) tem por finalidade a representação dos fluxos de informações entre entidades de um sistema como Grafo Acíclico Direto (DAG) com o objetivo de reduzir a taxa de falsos alarmes [15]. Esse conceito não é explorado no presente trabalho.

Em 2021, redes de Mapas Auto-Organizáveis de Kohonen (SOMs) eram o padrão da indústria para IDSs não supervisionados dentro de contexto de bases de dados mais antigas [11]. Por conta disso foi feita uma revisão acadêmica focada nessa família de técnicas. Essa revisão abordou tanto HIDSs quanto NIDSs [11], e fez uma revisão conceitual dos principais

SOMs. Os principais trabalhos de aplicação dessas técnicas foram abordados ao contexto de IDS sendo destacados seus principais desafios e desempenhos [11]. Ficou claro em [11] que a utilização da distância Euclidiana gerava uma grande complexidade computacional e baixa capacidade de processamento inerente aos SOMs. Foram destacadas iniciativas para mitigar os problemas decorrentes da utilização da distância Euclidiana, e a técnica *Growing Hierarchical* SOM (GHSOM) como apresentando melhor desempenho entre as demais [11].

Dado à vasta e eficaz utilização de técnicas de processamento de linguagem natural no contexto de IDSs e a consolidação da técnica *Transformer* (TRF) como o estado da arte nessa área; em 2025 foi publicado uma revisão acadêmica com os principais trabalhos abordando IDS por meio de *Transformers* e LLMs [12]. Essa revisão abordou estudos com os seguintes objetivos quanto às intrusões: identificação supervisionada, identificação não supervisionada, identificação por meio de imagens, geração de novos dados de forma à balancear bases de dados, já que normalmente há mais dados normais de operação do que dados de intrusão [12]. Segundo essa revisão, os TRFs apresentaram desempenho superior a redes recorrentes, como o LSTM e GRU, além de apresentarem complexidade computacional mais baixa [12][16][17]. Já LLMs apesar de apresentarem desempenhos muito bons, apresentaram complexidades computacionais altas, dificuldades relacionadas à quantidade de dados necessária para realização de ajustes finos, e alta complexidade nos demais aspectos[16]. Apesar da tentativa em abordar NIDSs e HIDSs, fica clara a existência de mais trabalhos focados em NIDSs do que em HIDSs. É possível perceber também a inexistência de trabalho utilizando TRFs com SCs, já que os trabalhos citados em HIDS utilizam *logs* de sistemas.

Na revisão de [13] o foco é a aplicação de IDSs no contexto de IoT, em que os dispositivos apresentam severas restrições de processamento, de memória e de bateria. Nessa revisão foram abordados NIDSs, HIDSs e CIDSs (*Collaborative Intrusion Detection Systems*). No contexto de HIDS fica claro que o paradigma clássico de detecção por regras não pode ser aplicado em dispositivos IoT, uma vez que além de ter desempenho questionável por ser sensível a técnicas de ofuscação, requer capacidade computacional proibitiva para dispositivos IoT [13]. Uma vantagem de dispositivos IoT é seu padrão de comportamento mais simples, de forma a facilitar a identificação de padrões anormais [13]. Por exemplo, uma forma de ataque conhecida como *battery drain denial of service* pode ser identificada por meio do monitoramento da bateria do dispositivo [13].

2.1 Estudos Recentes Relacionados a HIDS

A string de busca foi a mesma utilizada em [8], a saber: "host intrusion detection system" or "host based intrusion detection" or "host anomaly detection" or "host based anomaly detection" or "host based anomaly intrusion detection" or "host based ids" or "host ids" or "host based misuse intrusion detection" or ("signature based intrusion detection" and "host"). O período de publicação dos artigos selecionados foi entre 2021 e 2026. Adicionalmente foram estabelecidos como critérios: artigos de língua inglesa, limitados à Ciência da Computação, artigos em revista ou em conferências, contendo a palavra chave "Host-based Intrusion Detection System".

O *Scopus Digital Library* apresentou 48 resultados, porém a maioria deles não tinha relação com o presente trabalho, dessa forma optou-se por adicionar a palavra chave "ADFA-LD". Com isso foram apresentados os principais 9 trabalhos, desses 9 trabalhos, um estava relacionado com o sistema operacional Windows, o que foge ao escopo do presente trabalho [18][19][20][21][22][23][24][25].

Como descritos em seções posteriores, as bases de dados utilizadas no presente trabalho são ADFA-LD, NGIDS e LID-DS 2021. Quando a palavra chave adicional foi substituída de "ADFA-LD" por "NGIDS", nenhum trabalho foi apresentado. Quando a palavra chave adicional foi substituída de "ADFA-LD" por "LID-DS", 5 trabalhos foram apresentados [20]. Desses 4 usaram o LID-DS 2019 [26][27][28][29], 1 usou o LID-DS 2021 [20]. O único trabalho que usou LID-DS 2021 usou também o ADFA-LD, e a abordagem foi de aprendizado por reforço, a qual não tinha por finalidade identificação de comportamentos anormais e assim detectar ataques de dia zero [20]. É importante deixar claro que o LID-DS 2019 e o LID-DS 2021 são bases de dados distintas e semelhança no nome remete ao mesmo grupo de pesquisa que desenvolveu ambas.

Utilizando os mesmos critérios para o *IEEE Xplorer* nenhum resultado foi obtido, já para o ACM Digital Library foram obtidos 16 artigos, desses, apenas 3 possuem relação com o presente trabalho [30][31][32].

Até 2025 não havia trabalho sobre HIDS utilizando a técnica Transformer e SCs, porém em 2025 foi publicada uma técnica com esse viés, no entanto, a abordagem apresentada se dá num contexto de arquitetura supervisionada [21]. Por conta disso, essa iniciativa utilizando Transformer não é capaz de identificar ataques de dia zero como é o caso em arquiteturas não supervisionadas que são as técnicas estudadas no presente trabalho.

Outro trabalho importante é um trabalho que criou vetores *embedding* a partir dos arquivos contidos no ADFA-LD usando técnicas clássicas de processamento de linguagem natural chegando a obter, por meio da distância de Levenshtein, uma *Area Under the Curve* (AUC) muito próxima de 1, que é o comportamento ideal. No entanto, a complexidade computacional dessa área é muito alta, visto que foi necessário 1,5 meses para o devido treinamento [22].

2.2 Classificações de HIDS

A revisão acadêmica de [8] apresenta classificação para HIDS baseada na forma de detecção (por regras ou assinaturas, e por detecção de anomalias). Já a revisão acadêmica de [9] apresenta classificação para HIDS baseada na forma de detecção bem como no tipo de resposta produzida (resposta ativa e resposta passiva).

O HIDS baseado em regra ou assinatura monitora o funcionamento do *host*, caso haja um comportamento que apresente a mesma assinatura de um ataque conhecido entende-se que houve um ataque, caso contrário, não houve ataque [8][9].

O HIDS baseado em detecção de anomalias monitora o funcionamento do *host* em busca de comportamentos que fujam da normalidade, caso um comportamento anormal seja identificado, é inferido que houve um ataque [8][9]. Esse tipo de técnica se baseia fortemente em técnicas de ML/DL.

Quanto à resposta, o HIDS pode produzir uma resposta passiva ou ativa. Quando a resposta consiste apenas em criar um alerta para o administrador, após identificação de um ataque, o HIDS é classificado como passivo [9]. Quando a resposta do HIDS é um alerta para o administrador e, além disso, o HIDS toma ações para mitigar a invasão o HIDS é classificado como ativo [9].

As revisões presentes em [11][12][13] não envidam esforços em categorizar as diferentes classificações de HIDS, se limitando, quando muito, em especificar técnicas supervisionadas e técnicas não supervisionadas.

2.3 Tipos de HIDS Baseados em Detecção de Anomalias

A revisão acadêmica de [8] apresenta de forma estruturada os diferentes tipos de abordagens de HIDS baseadas em detecção de anomalias, enquanto a revisão de [9] tem foco mais amplo. As seis famílias metodológicas principais de HIDS baseadas em detecção de anomalias são [8]:

- baseados nas frequências das palavras;

- baseados em N-gram [18][19][20][21][22][23][24];
- baseados em incorporação de palavras (*Word embedding-based*) [25];
- modelagem neural de linguagem;
- ontologia semântica;
- híbridos.

As abordagens de HIDS apresentadas em [11] utilizaram SCs agrupados como N-grams, enquanto as abordagens apresentadas em [12] utilizaram logs dos sistemas.

2.4 Bases de Dados Utilizadas nas Pesquisas de HIDS

A revisão de [8] apresentou de forma estruturada os diversos tipos de bases de dados utilizados nos trabalhos acadêmicos descritos, enquanto a revisão de [9] cita as correspondentes bases de dados associadas a cada trabalho acadêmico descrito de forma esparça. Algumas bases de dados utilizadas em pesquisas de HIDS são:

- DARPA/KDD
- PUS-DS
- UNM
- CANALI-WD
- Firefox DS
- ADFA-LD
- ADFA-WD
- NGIDS-DS
- AWSCTD
- LID-DS
- PLAID

Já as revisões em [13] e em [11] ratificam a base de dados DARPA/KDD como a base de dados mais utilizada entre os diversos trabalhos citados, mesmo sendo uma base de dados antiga e que não representa adequadamente sistemas computacionais modernos.

2.5 Métricas de Avaliação dos HIDS

Quanto às métricas de avaliação dos diversos modelos, na revisão em [8] são descritas 16 métricas de desempenho, na revisão em [9] são descritas 17 métricas, na revisão em [11] foram descritas 3 métricas de desempenho, na revisão em [12] foram descritas 6 métricas de desempenho e na revisão em [13] foram descritas 4 métricas de desempenho objetivas, além de um conjunto de métricas de desempenho subjetivas.

As métricas apresentadas por [11] são basicamente taxa de FPR (*false positive rate*), DR (taxa de detecção) e ACC (acurácia).

As métricas apresentadas por [8] são: DR, PRC (precisão), F1 (*F1-score*), FPR, FAR (*false alarme rate*), FNR (*false negative rate*), TNR (*true negative rate*), Matrix de Confusão, AUC (*area under the curve*), ACC, *Classification Error*, MER (*mean error rate*), MCC (*Matthew's correlation coefficient*), Tempo de Treinamento/Teste/Execução, BLEU *score*, TF-IDF e *Cossine Similarity*. [9].

As métricas apresentadas por [9] são: ACC, PRC, REC (*Recall*), F1, FPR, FAR, FNR, TNR, Misclassification rate, Curva ROC, AUC, Análise de Latência, Tempo de Treinamento/Teste/Execução, MAE (*Mean Absolute Error*), RRSE (*Root Mean Squared Error*), RAE (*Relative Absolute Error*), RRSE (*Root Relative Squared Error*).

Algumas das métricas apresentadas pela revisão de [12] são: ACC, REC, PRE, F1, MCC, FR (*fooling rate*), AS (*alert score*).

No contexto de IoT as principais métricas são: ACC, taxa de falso negativo, DR, FPR e técnicas diversas de medição de complexidade computacional para cada dispositivo em especial, já que a portabilidade das soluções em IoT é baixa [13].

Dentre os estudos mais recentes relacionados a HIDS, começam a surgir preocupações com as complexidades dos modelos adotados, de forma que começam a serem citados o tempos de treinamento de cada modelo e os tempos de teste [19][20]. No entanto, essas medidas variam de uma arquitetura computacional para outra.

Nos trabalhos acadêmicos citados no presente trabalho não foram citadas as métricas de desempenho: número de parâmetros de cada modelo e FLOP (*Floating Point Operations* – Operações em Ponto Flutuante). Essas métricas apresentam com maior exatidão as complexidades dos modelos e é uma das contribuições do presente trabalho. Na seção 4.8 estão descritas outras métricas que fundamentaram o presente trabalho.

2.6 Estado da Arte dos HIDSs

Para os contextos representados em cada base de dados há uma ou um conjunto de técnicas que representam o estado da arte, por exemplo:

- para a base de dados DARPA/KDD a técnica que é considerada como estado da arte e padrão da indústria, é a técnica SOM com (Taxa de Detecção de 99,99% para Taxa de Falso Alarme de 3,72%) [11];

- para a base de dados ADFA-LD as técnicas consideradas como estado da arte são *ensemble* (ou combinação) de LSTMs, ou mesmo técnicas de medição de distâncias utilizando *Embeddings* [18][22];
  - com a utilização de combinações de LSTM os melhores resultados constantes na literatura foram os seguintes valores de AUC: 0,928 [36], 0,997 [18],
  - com a utilização de LSTMs únicos, sem combinação, o melhor resultado constante na literatura foi o seguinte valor de AUC: 0,871 [18],
  - com a utilização de técnicas de medição de distâncias utilizando Embeddings o melhor resultado foi obtido pela distância de Levenshtein (Taxa de Detecção de 99% para Taxa de Falso Alarme de 2%).
- para a base de dados NGIDS-DS a principal técnica utilliza é a do pesquisador que a propôs. As técnicas utilizadas foram *Hierarchical Hidden Markov Models* (HHMMs) e *Gaussian Mixture Modeling* (GMM) combinada com Correntropy [10];
  - no contexto dessa base de dados o melhor resultado foi obtido pela técnica Corr-GMM com Taxa de Detecção de 96,57% para Taxa de Falso Alarme de 2,89%.
- em relação à base de dados LID-DS2021, é uma base de dados muito recente e por isso conta com poucos estudos associados. Um estudo importante de 2024 alega que o LID-DS2021 muito grande e complexo, por isso utiliza o LID-DS 2019 com a técnica Word2Vec para fazer embedding de SCs e LSTM como motor de decisão por essa técnica ser conhecida como capaz de lidar com obfuscação de ataques [33]. O único estudo encontrado que desenvolve solução através do LID-DS2021 atua a partir do paradigma de aprendizado por reforço, de forma que é incapaz de detectar ataques de dia zero [20].
  - no contexto do LID-DS2021 o único resultado disponível no momento não tem por finalidade a detecção de ataques de dia zero e são: Aurácia = 0,958, Precisão = 0,999, Recall = 0,939, F1-Score = 0,965.

Em relação ao paradigma de processamento de linguagem natural, paradigma esse que é o mais usado em HIDSs, a técnica que é o estado da arte no momento é a técnica Transformer [34].

Considerando apenas estudos utilizando Transformers no contexto de HIDSs as principais utilizações deste se dão utilizando logs de sistema em vez de syscalls [12]. Porém, há um estudo do ano corrente, de 2025, que utiliza Transformers dentro do contexto de

syscalls especificamente com a base de dados ADFA-LD, porém nesse estudo a abordagem de aprendizado de máquina utilizada foi a supervisionada, de forma que é incapaz de detectar ataques de dia zero [21].

2.7 Comentário Finais do Capítulo

Todos os estudos apresentados nesse capítulo tem em comum esforços no sentido de identificar **ataques de dia zero** que não são detectáveis por técnicas tradicionais.

As revisões de [8] e de [9] têm um foco mais geral e se complementam. A primeira aborda principalmente as técnicas de PLN aplicadas nos diversos trabalhos relacionados a HIDS e estrutura melhor as bases de dados bem como os diversos tipos de ataques cobertos por essas bases de dados.

Já a revisão de [9] por ser mais atual teve uma abordagem mais ampla, considerando soluções de NIDS, o paradigma do PIDS e diversos trabalhos abrangidos e/ou não devidamente detalhados na revisão de 2023 [8].

Já as revisões [11][12] tem foco específico na apresentação de desempenho das técnicas SOM e *Transformer*, sem abordar a inserção dessas técnicas no contexto mais amplo com as demais técnicas existentes.

A revisão em [13], por sua vez, tem foco em contexto de IoT, apresentando as potencialidades e desafios de IDSs nessa área.

Por conta do exposto, o presente trabalho se baseia fortemente nas revisões em [8] [11][12], considera algumas das abordagens apresentadas na revisão de [9], e cita os principais desafios apresentados por [13].

Por fim, foi feito levantamento dos principais trabalhos acadêmicos publicados entre 2021 até agosto de 2025, de forma que foi possível identificar que a base de dados ADFA-LD ainda é muito utilizada em pesquisas atuais, por exemplo na base de dados Scopus Digital Library foi possível constatar 48 trabalhos relacionados à HIDS, desses, 9 utilizavam ADFA-LD, dos quais 7 o utilizavam como única base de dados para análises.

Em relação às bases de dados mais recentes, NGIDS e LID-DS 2021, apesar de serem mais atuais e com maiores riquezas de informações, há poucos estudos relacionados, o que gera dificuldade em comparar as técnicas propostas com outras nesse contexto de bases de dados.

CAPÍTULO 3

REFERENCIAL TEÓRICO

No presente capítulo serão apresentados com maiores detalhes os principais conceitos descritos no capítulo anterior.

O principal conceito da área é que a quase totalidade das pesquisas em HIDS se dão no sentido de identificar ataques de dia zero, de forma que não é necessário dados atualizados de ataques para o treinamento dos modelos, apenas comportamentos normais dos *hosts* e os respectivos *syscalls*. Com isso as técnicas são capazes de identificar ataques que ainda não existem no tempo presente.

Ficou claro no capítulo anterior que os trabalhos científicos relacionados à área de HIDS estão associados a 6 famílias metodológicas principais [8], a saber:

- baseados nas frequências das palavras;
- baseados em N-gram;
- baseados em incorporação de palavras (*Word embedding-based*);
- modelagem neural de linguagem;
- ontologia semântica;
- híbridos.

3.1 HIDS baseado em frequência de palavras

HIDS baseado em frequência de palavras tem como objetivo inferir se houve ou não intrusão no sistema operacional utilizando apenas a frequência de ocorrência de cada *syscall* (SC) no contexto de um conjunto de *syscalls*. Essa metodologia é computacionalmente muito simples e não contém informação sobre sequências de SCs, o que a torna limitada quanto a capacidade de identificação de ataques. Adicionalmente, como o comportamento normal do sistema operacional varia com o tempo, a atualização dinâmica do comportamento normal do *host* é altamente desafiadora para a essa metodologia [8].

As principais variações dessa técnica são: *Term Frequency* (TF), *Inverse Document Frequency* (IDF) e *Bag-of-Words* (BoW).

3.2 HIDS baseado em N-gram

Uma sequência de SCs no tempo pode ser separada em sequências menores de tamanho finito, essas sequências menores são denominadas de N-grams. N-grams e SCs são similares a frases e palavras, respectivamente. Por exemplo, a sequência de SC “open, getrlimit, mmap, close”, pode originar dois N-grams de comprimento 3, a saber: “open, getrlimit, mmap” e “getrlimit, mmap, close”.

Um HIDS baseado em N-gram converte uma sequência de SC em uma sequência de N-grams e utiliza esses N-grams para inferir se houve ou não ataque ao *host*. Os HIDSs baseados em N-grams são muito utilizados, já que são capazes de armazenar as informações de contexto das sequências de syscalls, possuem boa capacidade de detecção e são capazes de identificar com sucesso ataques do tipo *mimicry* [8].

Ataques do tipo *mimicry s*ão ataques nos quais o atacante muda a ordem dos comandos, e por consequência, são capazes de superar técnicas baseadas em regras [35] e técnicas baseadas nas frequências das palavras.

As principais técnicas de HIDS baseado em N-gram são: N-gram (utiliza os N-grams diretamente no processamento), N-gram *frequency*, *Bag-of-n-grams*, *Lookahead Pairs* e Mapas Auto-Organizáveis de Kohonen (SOM) [8][11].

3.3 HIDS Baseado em Incorporação de Palavras (*Word Embedding-based*)

Nessa técnica os esforços se dão no sentido de realizar melhores representações numéricas vetoriais dos SCs. Frequentemente técnicas baseadas em incorporação de palavras apresenta melhores performances do que técnicas baseadas em N-gram [8].

Uma melhoria significativa obtida por meio dessa técnica se dá na representação das palavras para utilização em sistemas de aprendizado de máquina, já que comumente é utilizada a técnica *one hot encoding* na codificação das palavras para esse contexto.

As principais técnicas baseadas em incorporação de palavras são: Word2Vec, GloVe, FastText e *Graph Random State Embedding* (GRSE).

3.4 HIDS Baseado em Ontologia Semântica

Existem diversos fóruns, blogs e boletins de cibersegurança onde são divulgados dados e informações de ameaças cibernéticas. A técnica de ontologia semântica tem por objetivo inferir através desses dados os meios e as consequências associadas a tais ameaças cibernéticas [8].

A ontologia semântica é similar às técnicas tradicionais baseadas em regras, no entanto, é baseada em inteligência semântica (*semantic-reasoner*) e por isso, além de integrar as informações de ameaças existentes e identificá-las, também é capaz de identificar variações desses ataques conhecidos [8].

A ontologia semântica é baseada em processamento de linguagem natural (PLN), por exemplo, a técnica *Named-Entity Recognizer* (NER) foi treinada para extrair entidades de textos simples [8]. Uma das principais vantagens da ontologia semântica é a redução do tempo de varredura em um host para a busca de *malwares* [8].

3.5 Modelagem Neural de Linguagem

Nos últimos anos, modelos de linguagem natural tem aumentado de forma significativa suas performances na tarefa de capturar as relações entre palavras e na tarefa de prever a(s) próxima(s) palavra(s) de uma dada sequência de palavras. Essas técnicas têm sido adotadas no contexto de projeto de HIDS, uma vez que sequências de *syscalls* podem ser consideradas como sequências de palavras.

Como as técnicas recentes de PLN têm demonstrado uma excelente capacidade de identificar as relações entre as palavras em sequências de palavras, tais técnicas têm demonstrado uma drástica melhoria na capacidade de identificação de ataques cibernéticos [8][18][36].

Algumas das técnicas de PLN mais utilizadas no contexto de projeto de HIDS são: modelos de linguagem baseado em RNN-VED, modelo de linguagem baseado em LSTM, modelo de linguagem baseado em GRU [8][18][36].

3.6 Híbridos

As técnicas de HIDS híbridas têm como foco a combinação de mais de uma das técnicas descritas com o objetivo de melhorar os desempenhos dos HIDS.

Algumas combinações frequentes são: N-gram com TF-IDF (técnica baseada em frequência), N-gram com técnicas de incorporação de palavras, N-gram com técnicas de *data augmentation*, etc.

3.7 Bases de Dados Adotadas pela Comunidade Acadêmica para Pesquisas em HIDS

No capítulo anterior foram citadas algumas bases de dados utilizadas em pesquisas relacionadas a HIDS, no entanto, não foram apresentados maiores detalhes a respeito das mesmas, de forma que a presente seção trabalhará melhor esses detalhes.

As bases de dados utilizadas em questão são categorizadas como: públicas em contextos simulados, públicas em contextos reais, públicas em contextos híbridos, privadas em contextos simulados e privadas em contextos reais [8].

A seguir estão descritas algumas bases de dados para cada um dos tipos descritos no parágrafo anterior.

Bases de dados públicas em contextos simulados: DARPA, UNM. Firefox DS, NGIDS-DS, LID-DS, PLAID:

Bases de dados públicas em contextos reais: PUS, AWSCTD;

Bases de dados públicos em contextos híbridos: CANALI-WD, ADFA-LD, ADFA-WD;

Bases de dados privados em contextos simulados: bases de dados geradas pelas ferramentas Strace, LTTng, Ptrace, etc;

Base de dados privado em contexto real: base de dados da Symantec [8].

Na Figura 2 estão descritos a quantidade de trabalhos relacionados a cada um dos bancos de dados descrito apresentados em [8].

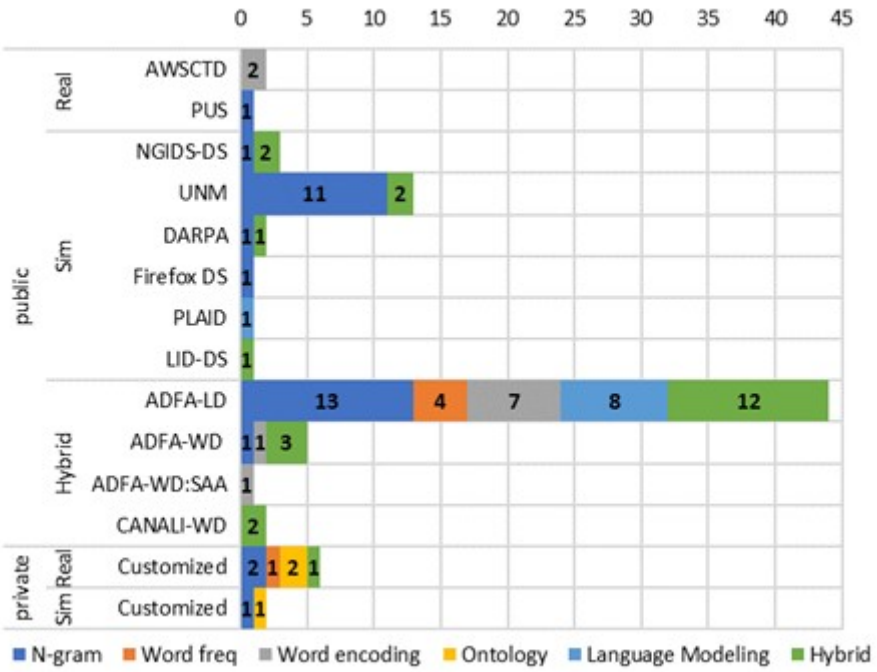

Figura 2 - Quantidade de trabalhos citados em [8] para cada base de dados descrita.

Como é possível observar na Figura 2, o ADFA-LD é uma base de dados consagrada e a base de dados mais utilizada em pesquisas sobre HIDS entre 2011 e 2022 [8].

Bases de dados mais modernas, como NGIDS-DS e LID-DS, apesar de serem mais atuais, foram base de menos estudos que o ADFA-LD [10][8].

3.8 Tipos de Ataques Frequentemente Existentes em Bases de Dados Usadas para Estudos de HIDS

Os tipos de ataques comumente descritos nas bases de dados utilizados para estudos de HIDS são [8]:

- Remoto/local (R2L – *Remote to local*): ataque mais descrito nas bases de dados. Consiste em ataque no qual o atacante consegue executar remotamente atividades locais em um dado host;
- Usuário/administrador (U2R – *User to root*): por meio desse ataque um usuário normal consegue adquirir privilégios de usuário administrador. Algumas técnicas empregadas nesse tipo de ataque são utilização de dicionário, engenharia social e detecção de senhas (*password sniffing*);
- Execução de Código Arbitrário (ACE – *Arbitrary Code Execution*): esse ataque consiste no ganho de controle do atacante por meio da execução de um código malicioso no *host* explorando alguma vulnerabilidade;
- Força Bruta (*Brute Force*): por meio dessa técnica são feitas várias tentativas de acesso e validação com o intuito de coletar dados de forma a perpassar controles de confidencialidade de dados;
- Negação de Serviço (DoS – *Denial of Service*): esse ataque tem por objetivo ocupar todos os recursos de um *host* ou serviço de forma que requisições legítimas a esse *host* ou serviço não sejam atendidas;
- *Backdoor*: é um ataque que garante acesso remoto ao *host*, ou converte em texto simples informações criptografadas. O uso desse ataque permite ao atacante o acesso a senhas, permite a possibilidade de alterar o remover informações no sistema de armazenamento do *host*, bem como transferir dados por meio da rede. Esse ataque compromete a confidencialidade e a integridade de sistemas;
- *Worm*: é um tipo de malware capaz de se replicar para outros hosts não comprometidos de uma mesma rede sem a necessidade de intervenção humana. Esse tipo de ameaça é capaz de comprometer a disponibilidade de sistemas, já que utiliza

a banda de transmissão da rede e é capaz de ameaçar a integridade, já que tem o potencial de alterar arquivos nos hosts atacados;

- *Trojan*: é um tipo de *malware* que vem está disfarçado em softwares aparentemente legítimos, já que oferecem serviço(s) utilizado(s) pelo usuário. Frequentemente esse tipo de ameaça rouba dados dos usuários, mostra propagandas indesejadas, e nega alguns tipos de serviços aos usuários;
- Roubo de Dados: consiste em ameaça à confidencialidade de um *host* por meio do potencial roubo de informações presentes em bases de dados corporativos, dispositivos e servidores. Esse tipo de ameaça está presente em vulnerabilidades ou *bugs* dos sistemas dos *hosts;*
- *Probe*: esse tipo de ameaça está associado a varreduras nos sistemas internos dos *hosts*, como portas ativas, IPs, sistemas operacionais utilizados, etc. Essa ameaça é capaz de comprometer a confidencialidade;
- Vírus: é um tipo de *malware* capaz de colocar uma cópia de si mesmo em outros arquivos, como arquivos HTML, outros executáveis, etc. Uma vez que os arquivos infectados sejam executados, o vírus é executado também;
- Miscelâneos (*Miscellaneous*): nessa categoria estão todas as outras ameaças não citadas anteriormente, como o ataque com o *script lprcp* através do qual são alterados conteúdos de arquivos arbitrários [8].

Na Figura 3 estão representadas algumas bases de dados e os tipos de ataques contemplados nas mesmas.

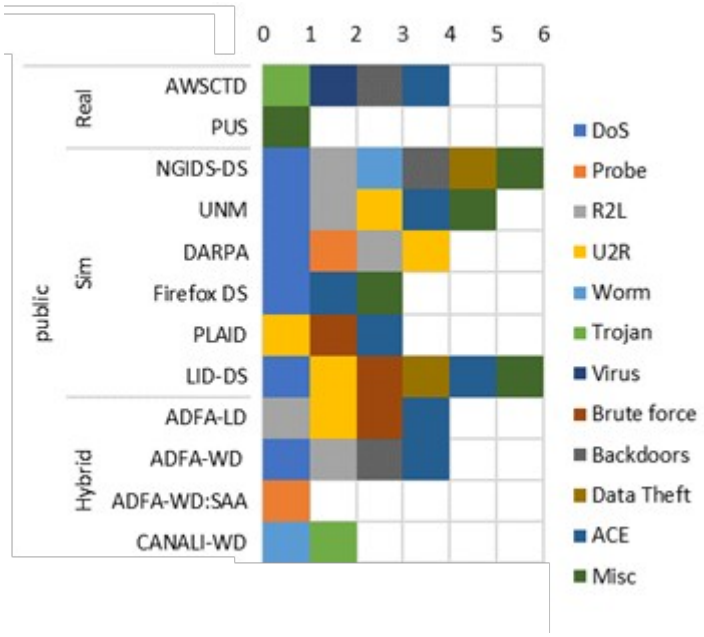

Figura 3 - Tipos de ataques presentes em algumas bases de dados em estudos sobre técnicas de HIDS, extraído de [8].

3.9 ADFA-LD

O Australian Defence Force Academy Linux Dataset (ADFA-LD) é um conjunto de dados amplamente utilizado para a avaliação de sistemas de detecção de intrusão em ambientes Linux.

Ele foi desenvolvido em 2013 pela University of New South Wales, em colaboração com a *Australian Defence Force Academy*, utilizando o sistema operacional Ubuntu 11.04, com o objetivo de suprir limitações presentes em datasets anteriores, como o DARPA 1998/1999, que já não refletiam com precisão as características de tráfego e ataques em sistemas modernos.

Os dados são apresentados como sequências de chamadas de sistema (*system calls*), codificadas como números em arquivos no formato .txt, e organizadas em três grupos:

- Training_Data_Master (normal): 833 arquivos de trace, totalizando cerca de 308 077 *system calls*.
- Validation_Data_Master (normal): 4.372 arquivos, com aproximadamente 2 122 085 *system calls*.
- Attack_Data_Master (malicioso): 746 arquivos, com cerca de 317 388 *system calls*.

Ao todo, são 5.951 arquivos .txt e aproximadamente 2.747.550 chamadas de sistema, sendo 746 traces maliciosos distribuídos entre seis tipos de ataques: Adduser, Hydra-FTP, Hydra-SSH, Java-Meterpreter, Meterpreter e Web-Shell.

Os traces foram coletados durante operações legítimas (como navegação web e uso de LaTeX) e durante ataques simulados (como brute force, exploits e rootkits), o que torna o ADFA-LD um dataset realista e valioso para pesquisas baseadas em aprendizado de máquina, detecção de anomalias e análise comportamental do sistema.

3.10 NGIDS-DS

O Next-Generation Intrusion Detection System Dataset (NGIDS-DS) é um conjunto de dados desenvolvido no cyber range de nova geração do Australian Centre for Cyber Security (ACCS), vinculado à University of New South Wales (UNSW) e à Australian Defence Force Academy (ADFA). Publicado em 2017, o dataset foi concebido no contexto da tese de doutorado de Waqas Haider, com o objetivo de suprir limitações de coleções anteriores e oferecer um cenário mais realista para a avaliação de sistemas de detecção de intrusão.

A coleta dos dados foi realizada em um ambiente de rede simulado, com Ubuntu 14.04 como sistema operacional principal e utilização de equipamentos de geração de tráfego (IXIA PerfectStorm). Foram reproduzidos padrões legítimos de acesso (como navegação, serviços corporativos e atividades de diferentes perfis de usuário) e diversos tipos de ataques, incluindo Exploit, DDoS, Backdoor, Shellcode, Reconhecimento, Worm e Generic. O experimento ocorreu ao longo de cinco dias completos, entre 11 e 16 de março de 2016, garantindo diversidade temporal e representatividade de eventos.

O NGIDS-DS é composto por múltiplos arquivos organizados em diferentes formatos:

- 99 arquivos de log de host (CSV), contendo registros detalhados de eventos do sistema;
- um arquivo de captura de pacotes (NGIDS.pcap), com aproximadamente 1 milhão de pacotes de rede;
- ground-truth.csv, fornecendo os rótulos de verdade (normal vs. ataque);
- feature-descr.csv, descrevendo os atributos extraídos;
- readme.txt, com documentação de suporte.

Do ponto de vista quantitativo, o dataset apresenta uma proporção aproximada de 90:1 entre registros normais e registros maliciosos nos logs de host, refletindo a predominância de tráfego legítimo em cenários reais. Essa característica reforça sua utilidade para pesquisas que exploram técnicas de aprendizado de máquina voltadas para a detecção de anomalias em ambientes desbalanceados.

Assim, o NGIDS-DS representa um avanço significativo em relação a datasets tradicionais, pois combina tráfego de rede e logs de host com rótulos precisos, permitindo análises híbridas e metodologias de detecção de intrusões mais robustas.

A principal desvantagem dessa base de dados é a baixa granularidade temporal dos eventos, o que torna muito difícil fazer os ordenamentos temporais das SCs e nas respectivas *threads* e entre elas [8].

3.11 LID-DS

De forma a mitigar os problemas das bases de dados anteriores foi desenvolvido o LID-DS. Contudo, por ser composto por duas bases de dados mais recentes (LID-DS 2019 e LID-DS 2021), carece de volume de trabalhos acadêmicos associados, deferentemente do ADFA-LD, por exemplo.

O *LID-DS (Leipzig Intrusion Detection Dataset)* é um conjunto de dados voltado para o estudo de sistemas de detecção de intrusão baseados em host (*Host-Based Intrusion Detection Systems*, HIDS), focado em chamadas de sistema (*system calls*) em sistemas Linux. Ele oferece uma riqueza de informação que permite avaliar métodos modernos de detecção de intrusos.

Foram lançadas pelo menos duas versões principais: **LID-DS 2019** e **LID-DS 2021**. A versão de 2019 concentra-se em logs de chamadas do sistema com parâmetros e metadados, enquanto a versão de 2021 amplia esse escopo, incluindo também tráfego de rede, entre outras melhorias [37].

A obtenção dos dados se deu em ambiente simulado, utilizando um host com sistema operacional Ubuntu versão 18.04. Os dados coletados foram: ID das *syscalls*, argumentos de chamada, valores de retorno, identificadores de processo/tarefa, *thread*, timestamps, nomes de processos e em alguns casos valores dos buffers de dados.

Os cenários de ataques descritos são mais atuais e complexos. Por exemplo, no LID-DS 2021 os cenários de ataque são: Bruteforce_CWE-307, CVE-2012-2122, CVE-2014-0160, CVE-2017-7529, CVE-2017-12635_6, CVE-2018-3760, CVE-2019-5418, CVE-2020-9484, CVE-2020-13942, CVE-2020-23839, CWE-89-SQL-injection, EPS_CWE-434, Juice-Shop, PHP_CWE-434, ZipSlip [37].

As principais críticas ao LID-DS 2021 são:

- foi desenvolvido em ambiente simulado com apenas um sistema operacional,
- exige hardware com capacidade de armazenamento e processamento considerável, uma vez que a quantidade de dados é muito grande.

CAPÍTULO 4

DESENVOLVIMENTO EXPERIMENTAL

4.1 Trabalho Proposto

O trabalho proposto nessa dissertação tem por objetivo comparar três técnicas de PLN quanto à acurácia, taxa de detecção, taxa de falsos alarmes, complexidade computacional, e interpretabilidade.

A base de dados utilizada é o ADFA-LD por ser uma base de dados bem consolidada na comunidade acadêmica, desenvolvida em ambiente híbrido (boa quantidade de dados reais), **atual**, e se baseia em *syscalls*. É uma base de dados bem explorada em diversos trabalhos acadêmicos, apresenta as vantagens associadas aos *syscalls*, visto que syscalls apresentam, em comparação a *logs* de sistemas, menos ruído advindo de interpretações e maior aplicabilidade a problemas de tempo real. Outra vantagem dessa base de dados é a não necessidade de conversão dos *syscalls* em valores numéricos, já que os SCs já estão representados em valores numéricos, sendo essas numerações correspondentes aos IDs de cada syscall no sistema operacional Ubuntu 11.04, os quais continuam válidos em versões mais recentes do Ubuntu [10].

A quantidade de ataques cobertos pela base ADFA-LD é quatro, esse fato se traduz em uma quantidade razoável de ataques cobertos, a saber: *Brute Force*, U2R (*User to Root*), R2L (*Remote to Local*), ACE (*Arbritary Code Execution*).

É importante destacar, no entanto, que os dados de ataque do ADFA-LD não foram utilizados nos treinamentos dos modelos, de forma que para os modelos esses ataques são ataques de dia zero.

As técnicas analisadas pelo presente trabalho são: LSTM [8], Transformer [38][34] [17] e uma versão modificada de um Mapa Auto organizável de Kohonen (SOM) [39].

A técnica LSTM foi utilizada com sucesso em diversos estudos [8][18][36][40] apresentando uma alta acurácia na identificação de ameaças, e baixa taxa de falso alarme, no entanto, apresenta interpretabilidade limitada, e é computacionalmente complexo e intensivo.

A técnica *Transformer* é o estado da arte na área de processamento de linguagem natural (PLN) desde 2019, pelo menos. Dessa forma apresenta melhor efetividade na identificação das relações contextuais entre as palavras de uma frase, de forma a prever a próxima palavra de uma sequência de palavras forma mais efetiva do que o LSTM [17][34] [41]. Adicionalmente, a técnica Transformer possibilita paralelismo na fase de treinamento, o que não é possível com o LSTM; apresenta complexidade computacional baixa no treinamento para sequências de palavras com tamanho baixo e apresenta complexidade computacional constante na fase de inferência [17][41].

A terceira técnica analisada no presente trabalho é uma versão modificada do SOM. Essa técnica foi utilizada com sucesso na identificação das aeronaves brasileiras com maiores riscos associados [39]. O SOM tem baixa capacidade computacional, e é altamente interpretável. Porém, no contexto do presente trabalho atua na análise direta de n-grams e isso pode ser uma limitação. Análise diretamente em n-grams comumente não apresenta desempenho tão bom quanto técnicas de modelagem neural da linguagem como LSTM e GRU [8]. Por conta disso, no presente trabalho foi feito um processo de otimização dos parâmetros do SOM utilizado.

Uma vez definido o tamanho das sequências, foram implementados SOM, LSTM e *Transformer* para tratar sequências de syscalls com o mesmo tamanho. Para a inferência da situação do *host* quanto a invasões, o LSTM e o Transformer estimam a próxima palavra de uma dada sequência, caso a palavra se concretize, é assumido que o sistema está em situação normal de funcionamento, caso a estimação da próxima palavra seja diferente da palavra observada, é assumido que houve intrusão no *host*.

Em relação ao SOM, a inferência de anormalidade se deu de forma diferente. Durante o seu treinamento foram identificados diversos *clusters*, cada *cluster* com uma distância máxima normal entre uma sequência normal e o centro desse *cluster*. Caso uma nova sequência tivesse uma distância menor ou igual a essa distância para um *cluster* mais próximo, considerou-se que a sequência representava comportamento normal do sistema, caso contrário, entendeu-se que houve intrusão.

A comparação entre essas diversas técnicas se deu por meio de técnica de detecção de mudanças. Essa técnica analisa a quantidade de eventos em uma janela de tempo conhecida, caso poucos eventos de anormalidade aconteçam essa técnica considera que estes são *outliers*, caso os eventos de anormalidade sejam frequentes dentro dessa janela de tempo, a técnica de detecção de mudanças infere que uma anormalidade, ou ataque, ocorreu [42][43] [44].

A Figura 4 apresenta um diagrama esquemático da combinação das técnicas descritas para a criação dos respectivos HIDSs.

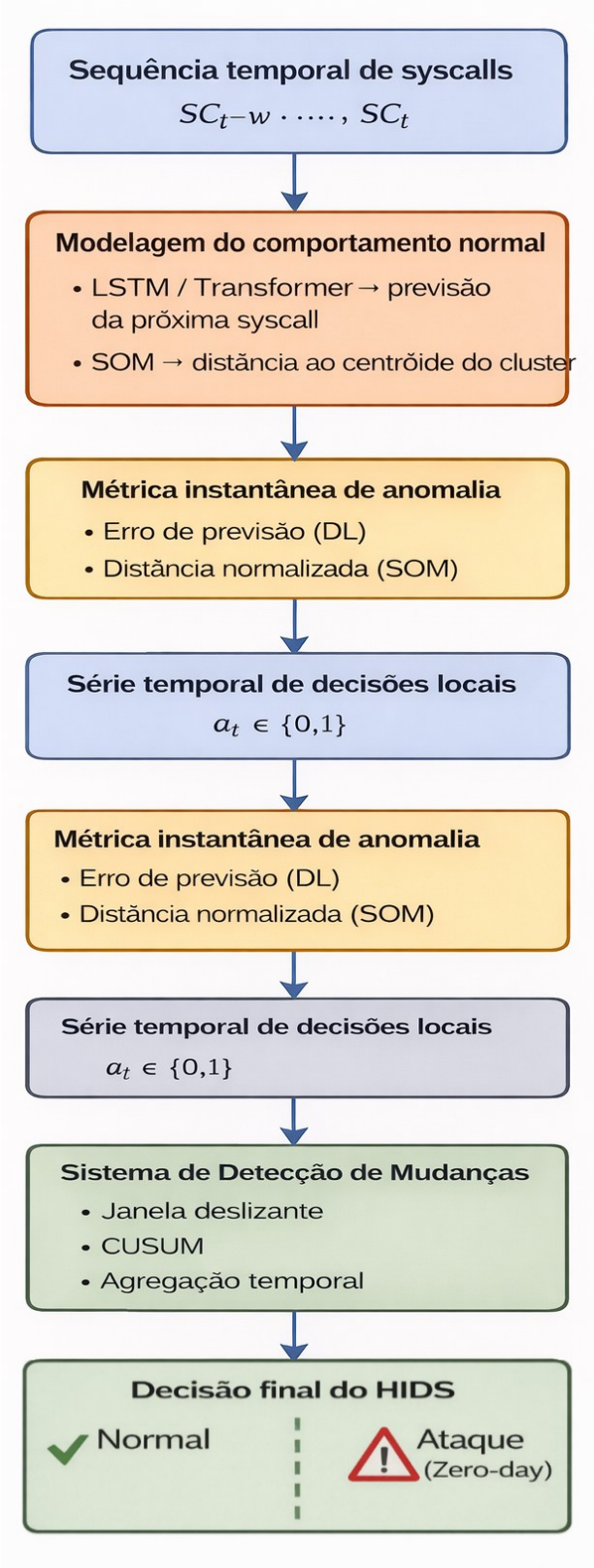

Figura 4: Diagrama Esquemático dos HIDSs
Contendo Sistema de Detecção de Mudanças

4.2 Máquina Utilizada

Os experimentos foram conduzidos em uma máquina com sistema operacional Ubuntu 22.04.5, com processador Core i5 9ª geração, GPU GTX 1650, memória RAM de 8 GB.

4.3 Bases de Dados Utilizadas

As bases de dados utilizadas foram ADFA-LD e LID-DS 2021. O NGIDS não foi utilizado por conta da dificuldade na determinação da ordem temporal dos *syscalls*, dificuldade essa também apontada em outros estudos [8].

A base de dados ADFA-LD foi escolhida como *benchmark* principal por ser uma base dados consolidada e muito estudada muitos estudos acadêmicos, com vários deles bem recentes, no corrente ano de 2025, o que possibilita uma melhor comparação dos desempenhos das técnicas propostas [8][21][22].

O LID-DS 2021 foi considerado nas análises por ser a base de dados mais recente entre as principais bases de dados para HIDS. Como existem poucos estudos relacionados a essa base de dados, uma análise de desempenho mais completa das técnicas propostas se tornaria inviável feita apenas nessa base [33].

O treinamento dos modelos foi feito na base de dados ADFA-LD uma vez que o tamanho e complexidade do LID-DS inviabilizam o treinamento dos modelos na mesma no tempo e escopo do presente trabalho, considerando também os recursos computacionais existentes. Essas limitações também foram apontadas por [33] que preferiu lidar a com a base de dados LID-DS 2019 [37].

As avaliações dos modelos foram realizados tanto na base ADFA-LD quanto na base LID-DS 2021.

No contexto do LID-DS 2021 foi analisado o cenário CVE-2017-12635_6. Nesse cenário ocorrem a CVE-2017-12635 e a CVE-2017-12636 concomitantemente, o que simula uma ataque real, consistindo de escaneamento de portas por meio do NMAP, um ataque de força bruta usando a combinação Hydra, login na base de dados, aumento dos privilégios dos atacantes, e finalmente a execução de código remoto [37].

4.4 LSTM

A arquitetura do LSTM tinha uma camada de embedding com 400 neurônios para otimizar as representações das sequências de SC, duas camadas com 400 células cada, e uma camada densa com 3 camadas, com respectivamente 17.600, 4.400 e 341 neurônios. Foi utilizado uma taxa de aprendizado de 0,0001, um *batch size* de 32, 305 épocas, regularização L1 e L2.

Essa arquitetura de LSTM foi escolhida pois foi uma das arquiteturas utilizadas no estudo que apresenta o estado da arte de HIDSs em contexto *single thread* com AUC igual a 0,928 e 0,984 em contexto *multi-thread* [18][36]. Essa arquitetura apresentou resultados similares às demais em termos de AUC [36].

Maiores detalhes sobre a técnica LSTM estão descritos no Apêndice A.

4.5 *Transformer*

A arquitetura do *Transformer* utilizado consistia apenas do codificador e tinha uma camada de embedding de tamanho 64, uma camada de auto atenção com 4 *heads*, uma camada de *fowarding* com 128 neurônios e uma rede neural densa ao final com 3 camadas, cada uma com 341 neurônios. No treinamento foram utilizados: taxa de aprendizado de 0,0001, um *batch size* de 32, e 305 épocas.

Essa arquitetura foi escolhida pois a quantidade de dados disponíveis para treinamento da base de dados ADFA-LD é baixa, de forma que não propicia condições para a devida convergência de treinamento de arquiteturas *Transformers* mais robustas.

A técnica *Transformer* está bem difundida, de forma que maiores detalhes estão descritos no Apêndice B.

4.6 Mapa Auto-Organizável de Kohonen (SOM)

O SOM em questão se caracteriza pela determinação do neurônio mais próximo ao dado de entrada, bem como os demais neurônios próximos por meio da distância de Manhattan, ou distância L1, conforme Equação 1.

$$d_i = ||X(t) - NT_i(t)||_0 \quad (1)$$

No SOM clássico, os neurônios considerados próximos ao neurônio mais apto seguem uma função exponencial quadrática, já no SOM proposto eles seguem uma função exponencial inversa, conforme Equação 2.

$$RT = (NN - 1)\, e^{\frac{-t}{NMA.T}} + 1 \qquad (2)$$

Já o ajuste de cada neurônio obedece à Equação 4.

$$NT_i(t+1) = NT_i(t) + \frac{\alpha}{i^{TD}}\left[X(t) - NT_i(t)\right] \qquad (3)$$

Em que:

- $X(t)$ ⇔ dado de entrada no tempo *t*;
- $NT_i(t)$ ⇔ neurônio *i* no tempo de treinamento *t*, também representa o centróide do *cluster I*;
- d_i⇔ distância do dado de entrada e o neurônio *i;*
- NN ⇔ número de neurônios totais;
- NMA ⇔ número total de dados de entrada;
- T ⇔ taxa de decaimento de vizinhança topológica;
- RT ⇔ número de neurônios treinados por $X(t)$;
- α ⇔ taxa de aprendizado;
- *TD* ⇔ taxa de decaimento de treinamento inter neurônios.

O último aspecto dessa versão do SOM é a determinação da distância máxima do centróide de cada *cluster* e o limiar de decisão, de forma que dados localizados entre o centróide e o limiar de decisão são considerados normais, enquanto dados além desse limiar são considerados maliciosos.

Para determinação desses raios foram consideradas duas abordagens:

- identificar a distância máxima do dado normal ao seu respectivo neurônio e considerar essa distância como limiar de normalidade;
- identificar um percentual dos dados normais com maior homogeneidade e estabelecer como limiar de normalidade a superfície que envolvesse tal percentual de dados.

Na Figura 5 é possível observar o *cluster* da Abordagem 1 em azul e o *cluster* da Abordagem 2 em verde. Os dados normais são representados por círculos e os dados de ataque são representados por estrelas.

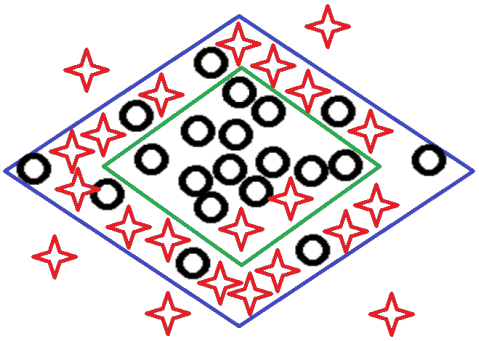

Figura 5: Dados normais em círculos, dados de ataque em formato de estrela, cluster verde delimitando percentual de dados principais e cluster azul delimitando todos os dados normais.

Na Abordagem 1 a distância é muito grande de forma que os *clusters* também classificam como normais muitos dados de ataque. Na Abordagem 2 os limiares de decisão levam em consideração a quantidade de dados mais próximos de cada neurônio, de forma a representar melhor dados mais homogêneos e desconsiderar um percentual dos dados normais que são mais similares aos dados de ataque.

A Abordagem 2 foi adotada no presente trabalho por conseguir classificar melhor os dados normais em detrimento dos dados de ataque, conferindo melhor ajuste aos tamanhos de cada *cluster* conforme suas realidades distintas. Após ajustes, chegou-se a conclusão que definir o raio de normalidade abrangendo 68,50% dos dados normais de treinamento em cada *cluster* fornecia a melhor solução.

As principais vantagens dessa versão modificada é a baixa complexidade computacional advinda da utilização da distância L1 e da boa interpretabilidade das fórmulas em relação às etapas originais do algoritmo.

O SOM em questão teve seus parâmetros ajustados por um algoritmo genético (AG) com população de 10 indivíduos, com torneios de dois indivíduos e 10% de mutação dos genes. A otimização levou em consideração um treinamento com 80% dos dados disponíveis

para treinamento e um conjunto de 20% de dados para validação. Ao final, os parâmetros otimizados estão representados na Tabela 1.

Tabela 1: Parâmetros Otimizados do SOM proposto.

	NN	α	TD	T	Raio
Parâmetros	182	0,04879	15,82	35,39	68,50%

A função de aptidão utilizada está descrita na Equação 4. É importante frisar que uma das principais dificuldades e até mesmo impossibilidade da utilização de um AG é a determinação de uma função custo adequada. Dessa forma, a função de aptidão utilizada foi obtida empiricamente a partir de diversos experimentos, conforme Equação 4.

$$FA = 9\,ACC^3 + 1{,}44\,DM^2 + 0{,}7\log\left(\frac{1}{DC + 0{,}01}\right) \quad (4)$$

Em que:

- ACC ⇔ acurácia no conjunto de validação (20% dos dados de treinamento não vistos no treinamento);
- DM ⇔ distância média dos neurônios ao centróide entre eles. Visa estimar a dispersão dos *clusters*;
- DC ⇔ distância média entre o centróide de cada *cluster* e sua superfície limite.

A função de aptidão utilizada se mostrou adequada na maioria das vezes, porém falhou diversas vezes. Com o intuito de evitar que o indivíduo mais apto tivesse desempenho inferior após cada geração, sua atualização era feita desde que o desempenho do novo indivíduo mais apto não fosse inferior nos dados de teste.

Para uma visualização do funcionamento do SOM proposto, é apresentado na Figura 6 um esboço gráfico de *clusters* tentando delimitar o espaço dos dados normais em detrimento dos dados de ataque.

É possível perceber que, quanto maior a dispersão dos *clusters* e quanto melhor a acurácia, melhor é o desempenho. Frequentemente, *clusters* menores conseguem representar melhor dados. A equação 4 procura expressar essa situação.

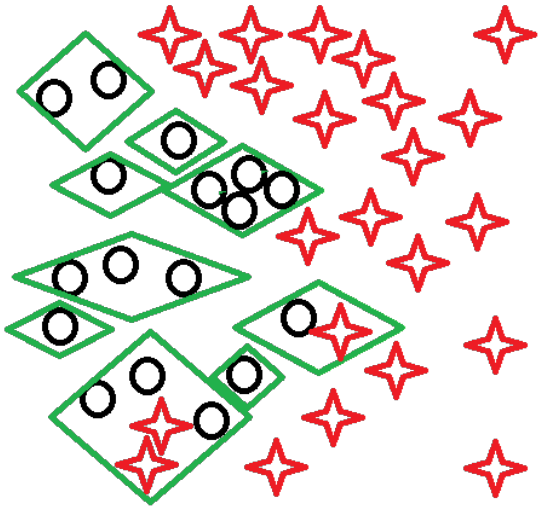

Figura 6: Esboço espacial do SOM proposto, no qual os círculos representam dados normais, as estrelas representam os dados de ataque e os losangos representam os clusters os quais agrupam os dados normais.

4.7 Técnica de Detecção de Mudanças

A detecção de mudanças (*change detection*) consiste em identificar momentos em que ocorrem uma alteração estatisticamente significativa no comportamento observado de uma série temporal ou fluxo de dados [44]. A técnica de detecção de mudanças inicialmente foi utilizada em sistemas de controle de qualidade e atualmente é amplamente utilizado na área de processamento digital de sinais [44]. É uma técnica que pode ser particularmente relevante em sistemas de detecção de intrusão, nos quais a distinção entre padrões normais e anômalos depende da capacidade de reconhecer transições sutis ou abruptas em processos computacionais ou de rede. Porém não é de nosso conhecimento que essa técnica tenha sido utilizada no contexto de HIDSs [44]. A Figura 7 ilustra um exemplo de uso da técnica.

Em termos gerais, a detecção de mudanças pode ser dividida em duas categorias:

- mudanças abruptas (*abrupt changes*), caracterizadas por transições repentinas, como o início de um ataque de força bruta ou a ativação de um malware;
- mudanças graduais (*gradual changes*), nas quais o comportamento se altera lentamente ao longo do tempo, tornando mais difícil a diferenciação entre atividade legítima e maliciosa.

Diversas abordagens têm sido empregadas para enfrentar essas duas categorias. Técnicas estatísticas clássicas utilizam testes de hipóteses, dentre elas se destaca a família de técnicas CUSUM (*Cumulative Sum*). Uma grande vantagem da técnica CUSUM é que é uma técnica não paramétrica, de forma que não necessita de conjecturas a respeito das distribuições de probabilidade das séries temporais monitoradas e em algumas situações apresenta resultado ótimo [42]. Alguns algoritmos do espectro CUSUM são: GLR, MLR, Brandt's GLR [44] e o *Page-Hinkley Test* [45].

No âmbito da segurança cibernética, a detecção de mudanças pode desempenhar papel fundamental ao permitir a identificação precoce de ataques que não seguem padrões previamente conhecidos [45]. Por exemplo, em ambientes de alta criticidade, a detecção de alterações na frequência de chamadas de sistema ou no volume de conexões de rede pode indicar a ocorrência de um comportamento anômalo antes mesmo que a assinatura de um ataque esteja disponível. Assim, a detecção de mudanças se consolida como um mecanismo essencial para complementar tanto abordagens de detecção baseadas em assinaturas quanto métodos baseados em detecção de comportamentos anormais.

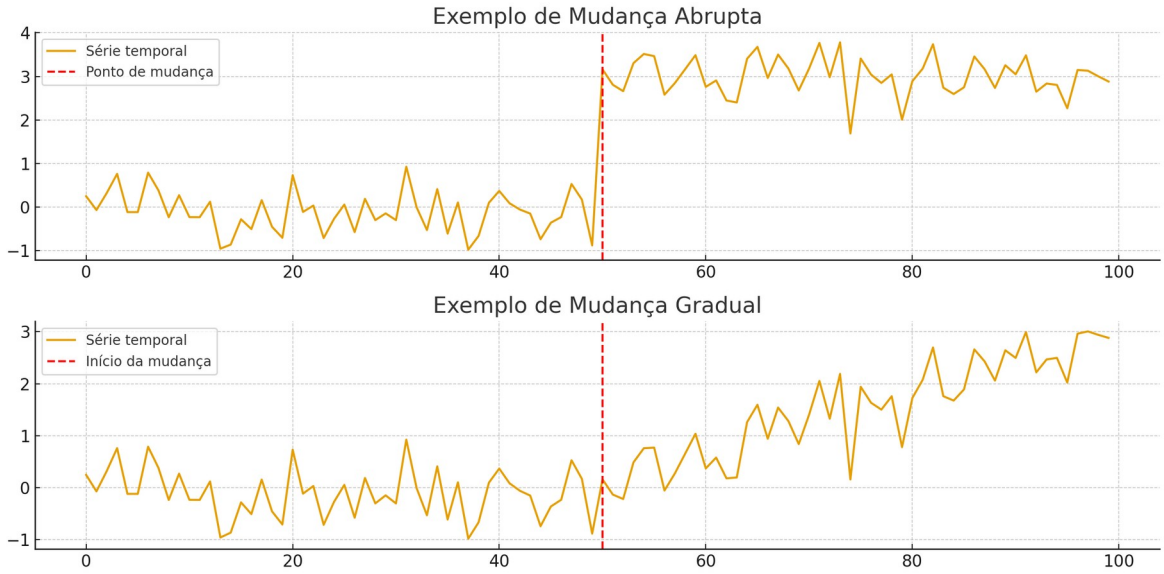

Figura 7: Exemplos de detecção de mudanças.

O presente trabalho utiliza um algoritmo da família CUSUM que consiste em uma janela deslizante w_n com 32 posições [46]. Cada uma dessas posições é preenchida com 32 duas classificações das técnicas Transformer, LSTM e SOM.

A Figura 8 exemplifica a janela w_n no instante de tempo 32 e no instante n, onde ela contém 32 classificações de um dado modelo.

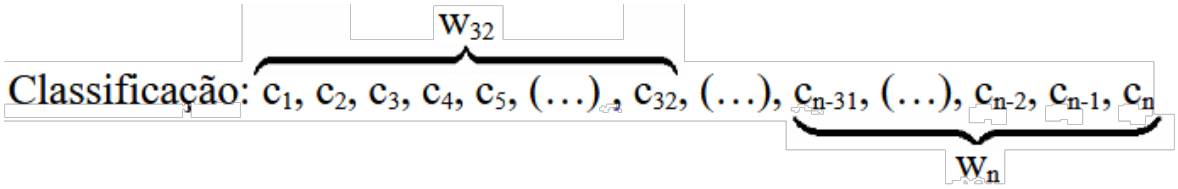

Figura 8: Diagrama representando a janela deslizante do CUSUM ao longo de uma sequência de classificações de um modelo.

No caso do Transformer e do LSTM, cada classificação c_n tem o valor igual a zero caso tenha acertado o próximo syscall da sequência, o que representa um estado não malicioso do host, e igual a 1 caso tenha errado, o que é considerado como sendo um *outlier* ou indício de invasão.

No caso do SOM, cada classificação c_n tem o valor igual a zero tenha caso o syscall da sequência seja classificado como normal, ou não malicioso, e igual a 1 caso sido classificado como anormal, porém uma anormalidade apenas é considerada como *outlier* ou indício de invasão.

A Tabela 2 apresenta um exemplo da janela w_{32}.

Tabela 2: Exemplo da janela w32.

	c1	c2	c3	c4	...	c29	c30	c31	c32
Classificação	1	0	0	1	-	0	1	1	1

Caso todas as posições de w_n sejam zero não ocorreu intrusão, caso todas as posições sejam iguais a 1 ocorreu invasão, caso haja um meio termo entre essas duas situações, a decisão se houve ou não intrusão depende de um limiar de decisão.

Para a utilização de um limiar de decisão é necessário o somatório de todas as posições de w_n, caso a soma seja igual a zero, não houve intrusão, caso seja igual a 32, houve intrusão, caso seja um valor maior do que zero ou menor do que 32 é necessário que tal valor seja comparado a um limar de decisão. Caso seja maior ou igual a esse limar de decisão, infere-se que houve intrusão, caso contrário infere-se de que não houve intrusão.

4.8 Métricas Utilizadas

Os objetivos das métricas eram dois, o primeiro era averiguar os desempenhos de cada modelo individualmente e o segundo era averiguar os desempenhos de cada modelo acoplado com o sistema de detecção de mudanças.

Para avaliação de cada modelo individualmente buscou-se identificar o percentual de classificações corretas de cada modelo dentro de contexto de dados de operações normais e dentro do contexto de dados de ataque. Já para avaliação dos modelos após acoplamento com sistema de detecção de mudanças a métrica adotada foi a curva ROC com a respectiva métrica AUC - *Area Under the Curve*.

Com relação às medidas de complexidade computacional foram adotadas as medidas de desempenho: número de parâmetros de cada modelo e FLOP (número de operações em ponto flutuante de cada modelo).

4.9 Métricas de Avaliação do LSTM e do Transformer

Para avaliar os modelos LSTM e Transformer foi utilizado o percentual de acerto das estimativas das próximas palavras das sequências de palavras, tanto dos dados normais quanto dos dados de ataque.

Esse percentual também é conhecido como acurácia (ACC). No caso em questão foram avaliadas as acurácias dos modelos para dados unicamente normais e para dados unicamente de ataque, caso os valores dessas duas acurácias fossem significativamente diferentes entendia-se que esse comportanmento destoante dos modelos para dados normais e para dados de ataque facilitariam a identificação de comportamentos anormais, ou ataques, por parte de outra técnica, no contexto do presente trabalho essa técnica é a técnica de detecção de mudanças.

A previsão de uma próxima palavra a partir de uma sequência de palavras pode ser vista como um processo de classificação de múltiplas classes, com cada classe correspondendo a uma *syscall* diferente.

Dessa forma ACC pode ser definida matematicamente como:

$$\text{ACC} = \text{número de classificações corretas} / \text{número total de classificações}$$

4.10 Métricas de Avaliação do SOM

Diferentemente do LSTM e do Transformer, o SOM tem por finalidade a realização de classificação binária do tipo 0 ou 1, ou do tipo intrusão ou não intrusão. Nesse caso uma matriz de confusão clássica pode desempenhar papel preponderante.

A ideia do SOM era identificar o número de classificações de situações de normalidade tanto submetido a dados somente de normalidade quanto a dados somente de ataque e com isso identificar as diferenças de desempenho.

Para criar a matriz de confusão é necessário a definição das seguintes métricas:

- TP_{SOM} (*true positive*) – parâmetro que mede, dado um n-gram, a quantidade de vezes que o SOM classifica **corretamente** o n-gram como normal;
- TN_{SOM} (true negative) – parâmetro que mede, dado um n-gram, a quantidade de vezes que o SOM classifica **corretamente** o n-gram como **anormal (ou malicioso)**;
- FP_{SOM} (false positive) – parâmetro que mede, dado um n-gram, a quantidade de vezes que o SOM classifica **erroneamente** o n-gram como normal;
- FN_{SOM} (false negative) – parâmetro que mede, dado um n-gram, a quantidade de vezes que o SOM classifica **erroneamente** o n-gram como **anormal (ou malicioso)**.

	Dados Classificados		
Dados Reais		normal	anormal
	normal	TP_{SOM}	FN_{SOM}
	anormal	FP_{SOM}	TN_{SOM}

Uma vez submetidos a dados puramente normais, a quantidade de dados classificados de fato normais é TP_{SOM}, enquanto a totalidade dos dados normais é (TP_{SOM} + FN_{SOM}). Dessa forma, o Percentual de Desempenho do modelo é:

Percentual de Desempenho = TP_{SOM} / (TP_{SOM} + FN_{SOM})

Sendo assim, é possível inferir que esse percentual de desempenho corresponde à medida de desempenho recall (REC).

Para o caso de dados puramento anormais a quantidade de dados classificados como normais é FP_{SOM}, já a quantidade de dados anormais totais é dado por ($FP_{SOM} + TN_{SOM}$). Logo, nesse caso o Percentual de Desempenho é definido por:

Percentual de Desempenho = $FP_{SOM} / (FP_{SOM} + TN_{SOM})$

Não é do conhecimento do autor medida de desempenho equivalente à fórmula apresentada. No presente trabalho esse medida será chamada de sensibilidade complementar (SENC).

4.11 Métricas de Avaliação dos Modelos Acoplados com Sistema de Detecção de Mudanças

Para os modelos já acoplados com sistema de detecção de mudanças, as métricas de desempenho adotadas foram curva ROC e AUC. Para melhor explicitação dessas métricas é necessário primeiro definir os seguintes parâmetros:

- TP (*true positive*) – parâmetro que mede a quantidade de vezes que o modelo identifica **corretamente** que houve um ataque;
- TN (*true negative*) – parâmetro que mede a quantidade de vezes que o modelo identifica **corretamente** que **não** houve um ataque;
- FP (*false positive*) – parâmetro que mede a quantidade de vezes que o modelo identifica **erroneamente** que houve um ataque;
- FN (*false negative*) – parâmetro que mede a quantidade de vezes que o modelo identifica **erroneamente** que **não** houve um ataque;
- DR (*detection rate*) – parâmetro que mede a taxa de detecção, ou seja, dado o universo total de ataque, identifica o percentual que o modelo foi capaz de **identificar corretamente** como ataque;

- FAR (*false alarme rate*) – parâmetro que mede a taxa de falso alarme de detecção, ou seja, dado o universo total de dados normais, identifica o percentual de vezes nos quais o modelo **identificou erroneamente** que havia um ataque.

Fazendo a matriz de confusão:

	Dados Classificados		
Dados Reais		anormal	normal
	anormal	TP	FN
	normal	FP	TN

DR pode ser definida pela seguinte equação:

DR = TP / (TP + FN).

FAR pode ser definido como:

FAR = FP / (FP + TN)

Os valores de FAR e DR variam conforme um limiar de decisão ajustável, por isso é possível a criação de um gráfico no qual no eixo da abcissas está o FAR e no eixo das ordenadas se encontra o DR recebe o nome de curva ROC. A área abaixo da curva ROC recebe o nome de AUC, como na Figura 9.

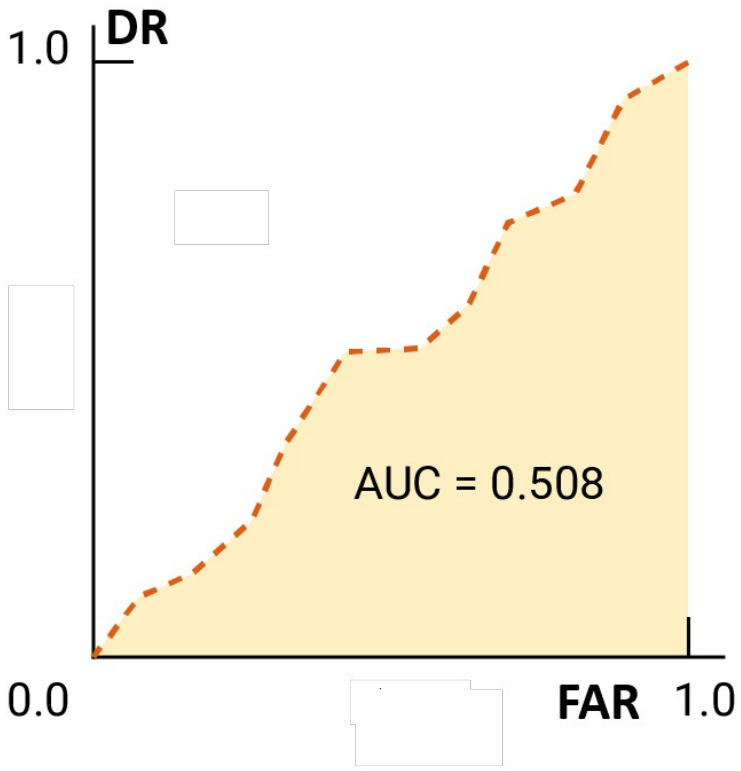

Figura 9: Curva ROC, adaptado de https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=pt-br.

Os valores de AUC variam de 0 a 1. Quando um sistema de detecção binária é puramente aleatório a AUC é 0,5, similar a jogar uma moeda pra cima e prever a face que ficará pra cima. Quando o sistema é capaz de acertar 100% das vezes com uma taxa de falsos alarmes de 0%, tem-se o comportamento ideal com AUC igual a 1.

CAPÍTULO 5

RESULTADOS E DISCUSSÕES

5.1 N-gram

Os N-grams utilizados consistiam em sequências de *system calls*. Estudos prévios afirmavam que eram necessárias sequências longas para o funcionamento satisfatório de um HIDS utilizando o paradigma de NLP, porém não especificaram os tamanhos dos N-grams [18][36]. Por conta disso, foram testados diversos valores na arquitetura LSTM e chegou-se a conclusão que a diferença de acurácia de treinamento era alta entre os tamanhos 5 e 11, porém baixa entre os tamanhos de n-gram 11 e 12. Dessa forma, optou-se por n-grams com tamanho 11, já que tornaria os modelos menos complexos.

É importante destacar que nessa estapa de definição dos tamanhos dos n-grams não foi utilizada a técnica de detecção de mudanças.

5.2 Acurácia

A primeira métrica de desempenho observada foi o número de acertos da próxima palavra do LSTM e do *Transformer* em relação ao número total de N-grams. Essa métrica também é conhecida como acurácia.

No contexto de dados normais da base de teste do ADFA-LD, os números de acertos do LSTM e do *Transformer* foram, respectivamente, 71,77% e 70,11%. Já no contexto de dados de ataque os números de acertos foram 35,56% e 30,03% respectivamente. Portanto, a diferença entre a acurácia com dados normais e a acurácia com dados maliciosos foi de 36,21% para o LSTM e 40,07% para o *Transformer*.

Nos dados de teste, o SOM classificou corretamente como normais 58,65% das sequências de palavras normais, já nas sequencias de palavras de ataque classificou corretamente como normais 23,32%. A distância entre esses desempenhos foi de 35,33%, ou seja, próximo ao resultado obtido no LSTM, porém, consideravelmente inferior ao desempenho do Transformer.

Quanto maior é a diferença de percentual de acertos, ou acurácia, entre dados normais e dados de ataque apresentado pela técnica, melhor é sua capacidade de identificar padrões anormais [14].

5.3 *Area Under the Curve* (AUC)

Para obtenção do parâmetro AUC é necessário obter a curva ROC com os desempenhos das técnicas analisadas quanto à taxa de detecção bem como quanto à taxa de falsos alarmes. Para obtenção da curva ROC, decidiu-se utilizar técnica de detecção de mudanças CUSUM - *Cumulative Sum* com janela de 32 amostras [42]. Essa técnica é amplamente utilizada na área de detecção de mudanças, e em diversos cenários apresenta comportamento ótimo [42]. Um dos desdobramentos do CUSUM é a menor sensibilidade a *outliers*, porém, caso *outliers* ocorram de forma frequente em uma janela de tempo conhecida, é possível inferir por meio do CUSUM que ao menos uma mudança ocorreu [43] [44].

Cada classificação errada de cada uma das técnicas alterava para o valor 1 o último elemento da janela deslizante do CUSUM de tamanho 32. A soma dos 32 elementos era comparado a um limiar no instante de tempo de cada palavra. Com a variação do limiar de zero a 33 foi possível obter a curva ROC para a base de dados ADFA-LD constante na Figura 8.

A partir da Figura 8, é possível afirmar que o desempenho do *Transformer* foi consideravelmente superior ao desempenho do LSTM e do SOM. O LSTM foi superior ao SOM, porém numa faixa considerável de valores de Taxa de Falsos Alarmes (FAR – *False Alarm Rate*), o SOM teve desempenho superior.

Quanto à métrica *Area Under the Curve* (AUC), a arquitetura baseada no *Transformer* apresentou AUC de 0,990, a baseada no LSTM apresentou um AUC de 0,841, enquanto que a arquitetura baseada no SOM apresentou AUC de 0,804.

Segundo [8], a técnica de estado da arte na área de HIDS em 2023 era a combinação de diversos classificadores LSTM ligeiramente diferentes e aplicados em um contexto de multi processamento [18]. Nessa situação, no contexto da base de dados ADFA-LD, a técnica que apresentou melhor resultado foi a WaveNet Relu (AUC: 0,997), seguida pela WaveNet_1 (AUC: 0,993) e LSTM_2 (AUC: 0,984) [18]. Esse artigo também abordou a base de dados PLAID, onde conseguiu AUC de 0,996 [18].

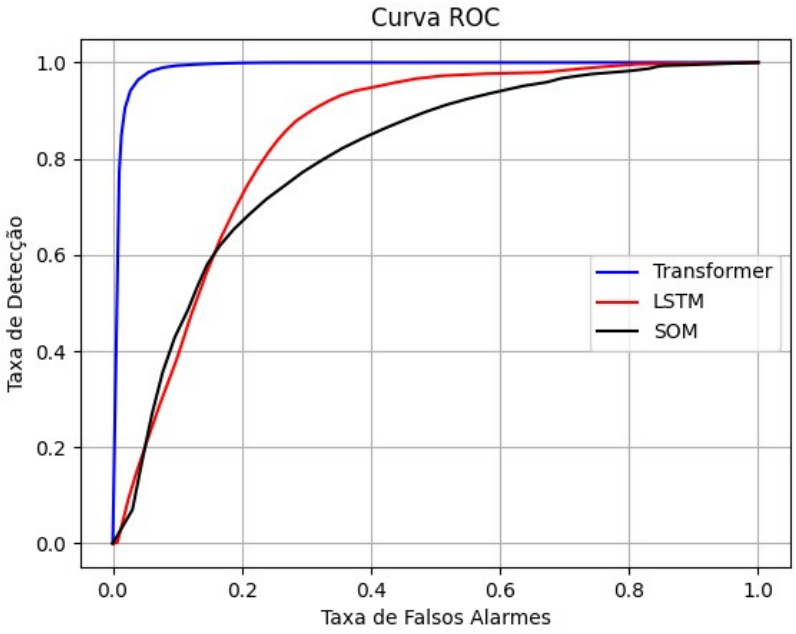

Figura 10: Curva ROC para comparação entre Transformer, SOM e LSTM, contexto do ADFA-LD

Dentre as técnicas avaliadas no presente estudo a técnica Transformer apresentou AUC 0,990 em contexto com um processador, o que se mostrou superior aos desempenhos das redes recorrentes, mesmo no contexto de multi processamento e se aproximou do desempenho das redes WaveNet [18]. Por isso, é possível considerar a rede Transformer proposta como fazendo parte do estado da arte em termos de desempenho.

Outro aspecto é a quantidade de dados utilizados no treinamento, uma vez que a arquitetura *Transformer* utilizada no presente estudo utilizou uma quantidade de dados de treinamento consideravelmente menor para alcançar o mesmo desempenho; visto que em [18] também foram utilizados no treinamento parte dos dados de validação da base de dados ADFA-LD.

É importante também fazer um recorte dos desempenhos das diversas técnicas em relação à taxa de detecção e a taxa de falsos alarmes para um dado limiar de decisão, conforme feito em diversos estudos, um vez que em muitos deles é o única métrica de desempenho produzida [10][11][22]. Os desempenhos das três arquiteturas em termos de DR e FAR para um limiar de decisão igual a 29 estão dispostos na Tabela 3 .

Tabela 3: Avaliação de desempenho em termos de *Detection Rate* (DR) e de *False Alarm Rate* (FAR)

Arquiteturas	Detection Rate (DR)	False Alarm Rate (FAR)
LSTM	9,2%	2,4%
Transformer	**94,1%**	**2,7%**
SOM	7%	3%

O SOM é mais sensível a mudanças no limiar de decisão, de forma que seu FAR subiu consideravelmente com o limiar de decisão 29, para o limiar de decisão 32 seu **DR 7%** e seu **FAR 3%**. Nessa situação onde o limiar de decisão é 32, o HIDS considera como ataque tão somente se todos os elementos da janela de 32 posições considerarem que de fato ocorreu ataque, caso contrário o HIDS considera que não houve ataque.

5.4 Complexidade Computacional

Frequentemente pesquisas em HIDS consideram apenas o desempenho das técnicas, sequer considerando ou avaliando as complexidades computacionais dos modelos. Além de desconsiderar as complexidades dos modelos, ainda costumam adotar estratégias ensemble o que aumenta consideravelmente o problema já que nessa abordagem vários modelos rodam em paralelo [18][36].

De forma a mitigar esse problema, diversos estudos têm adotado a estratégia de avaliar os tempos de treinamento e tempos de teste [22][47]. No entanto, esses tempos dependem diretamente das máquinas envolvidas, variando entre máquinas até mesmo com a presença ou não de GPUs.

De forma a medir de forma mais objetiva as complexidades dos modelos, o presente trabalho propõe a medição de complexidade computacional no contexto de HIDS as métricas: números de parâmetros dos modelos e FLOP (*Floating Point Operations* – Número de Operações em Ponto Flutuante*)*. Números de parâmetros dos modelos e FLOP são métricas amplamente difundidas no contexto de *deep learning*, porém, em nenhum dos artigos citados nessa dissertação essas métricas foram utilizadas. De forma que é possível afirmar que é a primeira vez que essas métricas são utilizadas no contexto de HIDS.

Avaliando as arquiteturas propostas quanto ao número de parâmetros e FLOP obtem-se a Tabela 4.

Tabela 4: Quantidade de parâmetros e de operações de ponto flutuante realizadas no LSTM, Transformer e no SOM.

	FLOP (*Floating Point Operations*)	Parâmetros
LSTM	738,7 M	159,0 M
Transformer	1,08 M	255,4 k
SOM	3,8 k	2 k

Apesar de AUC mais baixa em relação ao LSTM e ao *Transformer*, a técnica SOM se apresenta promissora por conta da baixa complexidade computacional, tanto do ponto de vista de memória quanto do ponto de vista de processamento, como demonstrado na Tabela 4.

A despeito dos excelentes desempenhos citados das diversas técnicas consideradas estado da arte, há estudos que apontam que uma AUC igual a 0,810 é um bom valor [21]. Em relação ao SOM proposto, seu desempenho foi bom (AUC: 0,804) já que seu desempenho é próximo à uma AUC de 0,810, tornando possível a utilização do SOM em aplicações não críticas com limitações de hardware e disponibilidade de apenas um processador.

Além do disposto anteriormente, o SOM apresentou desempenho ligeiramente superior a duas arquiteturas relevantes na comunidade científica e que foram analisadas no contexto de um processador em [18], a saber: CNN_RNN_1 (AUC = 0,802) e LSTM_2 (AUC = 0,793).

Tabela 5: Desempenho do SOM proposto em relação ao CNN_RNN_1 e LSTM_2 [18].

	AUC
SOM	0,804
CNN_RNN_1 [18]	0,802
LSTM_2 [18]	0,793

É importante destacar que a arquitetura LSTM_2 (AUC 0,793) que teve desempenho inferior ao SOM, no contexto de um processador, teve um desempenho excelente no contexto de multi processamento, chegando a uma AUC de 0,984.

Dessa forma é possível afirmar que um ensemble de SOM é plenamente viável dada sua baixa complexidade computacional, e que, a exemplo do LSTM_2, pode fornecer desempenhos excepcionais em contexto *multi-thread*.

5.5 Desempenho no LID-DS 2021

No contexto dos dados do LID-DS 2021 foi analisado o modelo Transformer no cenário descrito pelo CVE-2017-12635_6.

No contexto de dados normais, o Transformer acertou 23,51% das próximas palavras, enquanto no contexto de dados de ataque acertou 89.15% das próximas palavras. A amplitude nesse caso é -65,64%.

Esperava-se que o nível de acertos das próximas palavras do Transformer fosse maior no contexto de dados normais, uma vez que foi treinado com esse tipo de dado no ADFA-LD, porém é possível perceber que o padrão dos *syscalls* com ID até 341 do LID-DS 2021 é diferente do padrão equivalente no ADFA-LD. Um fator que contribuiu para essa diferença de padrões de *syscalls* foi o fato de o LID-DS 2021 ter mais syscalls do que os 340 *syscalls* existentes no ADFA-LD, e esses *syscalls* a mais foram desconsiderados da análise, pois o modelo os desconhecia.

Um aspecto surpreendente foi o nível de acerto de 89,15% nas sequências de ataque o que destaca o fato desses dados de ataque se assemelharem com os dados normais contidos no ADFA-LD.

A grande diferença de comportamento da arquitetura baseada no *Transformer* nos dados normais e nos dados de ataque destaca que *essa* arquitetura foi capaz de identificar claramente diferenças entre sequências normais e sequências de ataque, o que torna essa técnica promissora quando treinada nos dados do LID-DS 2021.

Caso o *Transformer* continue apresentando essa capacidade identificar diferenças claras entre sequências de ataque e sequências normais quando for treinado com dados atualizados do próprio LID-DS 2021, ele será capaz de identificar intrusões sem a necessidade de outras informações contidas no LID-DS 2021, como os argumentos dos *syscalls*.

Os criadores do LID-DS 2021 acharam por bem coletar outras informações além dos *syscalls*, como os seus argumentos, porque entenderam que para identificação de alguns

ataques apenas *syscalls* eram insuficientes. A arquitetura baseada em *Transformer*, no entanto, pode por em cheque essa percepção.

CAPÍTULO 6

CONCLUSÕES E RECOMENDAÇÕES

No presente trabalho foi realizada ampla pesquisa bibliográfica de trabalhos relacionados com HIDS seguindo a abordagem de utilização de técnicas de aprendizado de máquina (ML) e chamadas de sistema operacional (*system calls* - SCs) agrupados em N-grams.

As principais técnicas de ML utilizadas no contexto de HIDS eram LSTM e *Self-Organizing Maps* (SOM).

A técnica LSTM foi implementada no contexto do ADFA-LD utilizando uma arquitetura consolidada como estado da arte [36] e robusta, com alta complexidade computacional de forma a ser obtido o melhor resultado possível.

Já a técnica SOM aplicada foi uma técnica proposta em [39], no qual apresentou um incrível desempenho no tocante a identificação de aeronaves risco, bem como a constatação de acidentes e incidentes com as aeronaves identificadas após a identificação.

Por fim, foi implementada uma arquitetura *Transformer* simples, com poucos parâmetros para avaliar seu desempenho em relação às demais técnicas.

De forma a possibilitar a comparação entre técnicas de natureza distinta, foi utilizada a técnica de detecção de mudanças CUSUM. As métricas de desempenho adotadas foram percentual de acertos de cada modelo, AUC, FLOP, Detection Rate (DR), False Alarm Rate (FAR) e quantidade de parâmetros dos modelos de cada técnica.

Em relação à diferença de percentual de acertos e AUC, a arquitetura Transformer utilizada apresentou desempenho consideravelmente superior às demais técnicas no conexto do ADFA-LD, apresentando também uma baixa complexidade computacional evidenciada por meio de baixos valores de FLOP e de quantidade de parâmetros do modelo.

A melhor arquitetura no quesito complexidade computacional foi a arquitetura baseada em SOM com os menores valores de FLOP e de quantidade de parâmetros de modelo.

Para FARs em torno de 2% a arquitetura Transformer proposta também apresentou desempenho supreendente bom na casa dos 90% (DR 94,1% para FAR 2,7%), enquanto que as arquiteturas baseadas em LSTM e em SOM apresentavam DR menores que 10% (LSTM: DR 9,2% para FAR 2,4%, e SOM: DR 7% para FAR 3%).

De forma geral o LSTM apresentou desempenho de detecção superior ao SOM, porém, em uma faixa de valores de FAR, o SOM apresentou DRs superiores ao LSTM. Por exemplo para o FAR 9,6% o SOM apresentou DR 42,8%, enquanto que o LSTM apresentou DR 38,2% (desempenho inferior) para um FAR 9,9% (quantidade maior de falsos alarmes).

Para aplicações com maiores capacidades de processamento que dispositivos IoT, a arquitetura *Transformer* proposta se destaca, uma vez que apresenta desempenho superior à combinação de diversos classificadores LSTM, e uma complexidade computacional várias ordens de grandeza inferior.

Outra vantagem da arquitetura *Transformer* proposta é que necessita apenas um processador para apresentar desempenho superior ao conjunto de LSTM em ambiente de multi processamento constante da literatura vigente [18].

Em relação ao SOM proposto, este pode ser utilizado em aplicações com grande limitação de recursos computacionais como processamento, memória e bateria; e que apresentem uma maior tolerância a riscos cibernéticos [13].

Apesar do desempenho do SOM em termos de percentual de acertos e de AUC seja inferior ao LSTM, é possível obter um aumento de desempenho caso seja utilizada a estratégia de combinação de vários classificadores SOM, como foi feito com sucesso pelo LSTM.

No contexto do LID-DS 2021, a arquitetura *Transformer* treinada no ADFA-LD não funcionou da forma como foi projetada, no entanto, foi capaz de fornecer um comportamento capaz de identificar com clareza diferenças entre sequências de *syscalls* normais e sequências de *syscalls* de ataque.

A arquitetura Transformer proposta ensejou o registro de software constante no Processo No: BR512025004978-1.

Para trabalhaos futuros sugere-se a exploração do aspecto da explicabilidade da arquitetura Transformer bem como a combinação esemble de arquiteturas baseadas no SOM.

REFERÊCIAS BIBLIOGRÁFICAS

[1] MARR, Bernard. The impact of big data on the world economy. *Forbes,* 2023. Disponível em: https://www.forbes.com. Acesso em: 17 nov. 2025.

[2] BLAKLEY, Bob; MCDERMOTT, Ellen; GEER, Dan. Information security is information risk management. In: Proceedings of the 2001 workshop on New security paradigms. p. 97-104. 2001.

[3] CENTER FOR INTERNET SECURITY. *CIS Controls: Version 8*. East Greenbush: CIS, 2021. Disponível em: https://www.cisecurity.org. Acesso em: 17 nov. 2025.

[4] ALMUHAMMADI, Sultan; ALSALEH, Majeed. Information security maturity model for NIST cyber security framework. *Computer Science & Information Technology*, v. 7, n. 3, p. 51–62, 2017.

[5] TAHERDOOST, Hamed. Understanding cybersecurity frameworks and information security standards—a review and comprehensive overview. *Electronics*, v. 11, n. 14, p. 2181, 2022.

[6] GUO, Yang. A review of machine learning-based zero-day attack detection: Challenges and future directions. *Computer Communications*, v. 198, p. 175–185, 2023.

[7] MEAKINS, Joss. A zero-sum game: the zero-day market in 2018. *Journal of Cyber Policy*, v. 4, n. 1, p. 60–71, 2019.

[8] SWORNA, Zarrin Tasnim; MOUSAVI, Zahra; BABAR, Muhammad Ali. NLP methods in host-based intrusion detection systems: A systematic review and future directions. *Journal of Network and Computer Applications*, p. 103761, 2023.

[9] SATILMIŞ, Hami; AKLEYLEK, Sedat; TOK, Zaliha Yüce. A systematic literature review on host-based intrusion detection systems. *IEEE Access*, v. 12, p. 27237–27266, 2024.

[10] HAIDER, Waqas. *Developing reliable anomaly detection system for critical hosts: a proactive defense paradigm*. 2018. Tese (Doutorado) — UNSW Sydney, Sydney, 2018.

[11] QU, Xiaofei et al. A survey on the development of self-organizing maps for unsupervised intrusion detection. *Mobile Networks and Applications*, v. 26, p. 808–829, 2021.

[12] KHEDDAR, Hamza. Transformers and large language models for efficient intrusion detection systems: A comprehensive survey. *Information Fusion*, p. 103347, 2025.

[13] ARNABOLDI, Luca; MORISSET, Charles. A review of intrusion detection systems and their evaluation in the IoT. *arXiv*, 2021. Disponível em: https://arxiv.org/abs/2105.08096. Acesso em: 17 nov. 2025.

[14] CREECH, Gideon; HU, Jiankun. A semantic approach to host-based intrusion detection systems using contiguous and discontiguous system call patterns. *IEEE Transactions on Computers*, v. 63, n. 4, p. 807–819, 2013.

[15] ZIPPERLE, Michael et al. Provenance-based intrusion detection systems: A survey. *ACM Computing Surveys*, v. 55, n. 7, p. 1–36, 2022.

[16] AHMED, Syed Wahaj; KIENTZ, Fabio; KASHEF, Rasha. A modified transformer neural network (MTNN) for robust intrusion detection in IoT networks. *In: 2023 international telecommunications conference* (ITC-Egypt). IEEE. p. 663-668, 2023.

[17] VASWANI, Ashish et al. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

[18] RING IV, John H. et al. Methods for host-based intrusion detection with deep learning. *Digital Threats: Research and Practice*, v. 2, n. 4, p. 1–29, 2021.

[19] SATILMIŞ, Hami; AKLEYLEK, Sedat; TOK, Zaliha Yüce. Development of various stacking ensemble-based HIDS using ADFA datasets. *IEEE Open Journal of the Communications Society*, 2025.

[20] KIM, Yongsik et al. Reinforcement learning-based generative security framework for host intrusion detection. *IEEE Access*, 2025.

[21] SENOUSSI, Nour El Houda et al. Transformer-Based Approach for Intrusion Detection System. In: 2025 *International Symposium on iNnovative Informatics of Biskra* (ISNIB). IEEE p. 1-6, 2025.

[22] OUARDA, Lounis; MALIKA, Bourenane; BRAHIM, Bouderah. Towards a better similarity algorithm for host-based intrusion detection system. *Journal of Intelligent Systems*, v. 32, n. 1, p. 20220259, 2023.

[23] DWARKANATH, Samleti Sandeep; ARUNA, R. Multiclass cyber-attack classification approach based on the Krill Herd Optimized Deep Neural Network (KH-DNN) model for WSN. *International Journal of Modeling, Simulation, and Scientific Computing*, v. 13, n. 05, p. 2250056, 2022.

[24] LU, Yijun; TENG, Shaohua. Application of sequence embedding in host-based intrusion detection system. In: 2021 IEEE 24th international conference on computer supported cooperative work in design (CSCWD). IEEE p. 434-439, 2021.

[25] SUBBA, Basant; GUPTA, Prakriti. A tfidf-vectorizer and singular value decomposition based host intrusion detection system framework for detecting anomalous system processes. *Computers & Security*, v. 100, p. 102084, 2021.

[26] JORAVIYA, Nidhi; GOHIL, Bhavesh N.; RAO, Udai Pratap. Ab-HIDS: An anomaly-based host intrusion detection system using frequency of N-gram system call features and ensemble learning for containerized environment. *Concurrency and Computation: Practice and Experience*, v. 36, n. 23, p. e8249, 2024.

[27] AJAYI, Oluwagbemiga et al. Developing cross-domain host-based intrusion detection. *Electronics*, v. 11, n. 21, p. 3631, 2022.

[28] GRIMMER, Martin; KAELBLE, Tim; RAHM, Erhard. Improving host-based intrusion detection using thread information. In: *International Symposium on Emerging Information Security and Applications*. Cham: Springer International Publishing, p. 159-177, 2021.

[29] PARK, Daekyeong et al. Host-based intrusion detection model using siamese network. *IEEE Access*, v. 9, p. 76614–76623, 2021.

[30] TANWAR, Akshat et al. Intrusion detection system based ameliorated technique of pattern matching. In: Proceedings of the 4th International Conference on Information Management & Machine Intelligence, p. 1-4, 2022

[31] JI, Jun et al. Design of Intrusion Detection System for Layout Problem Based on Cloud Platform. In: Proceedings of the 2023 5th International Conference on Internet of Things, Automation and Artificial Intelligence, p. 728-732, 2023.

[32] ARQANE, Aouatif et al. A review of intrusion detection systems: Datasets and machine learning methods. In: Proceedings of the 4th International Conference on Networking, Information Systems & Security, p. 1-6, 2021.

[33] LÉO, Henrique Pessôa. *Syscall Security Components*. 2024. Dissertação de Mestrado. Universidade do Porto (Portugal).

[34] VERÍSSIMO, Vinícius et al. A study on the use of sequence-to-sequence neural networks for automatic translation of brazilian portuguese to libras. In: Proceedings of the 25th Brazillian Symposium on Multimedia and the Web, p. 101-108, 2019

[35] WAGNER, David; SOTO, Paolo. Mimicry attacks on host-based intrusion detection systems. In: Proceedings of the 9th ACM Conference on Computer and Communications Security, p. 255-264, 2002.

[36] KIM, Gyuwan et al. LSTM-based system-call language modeling and robust ensemble method for designing host-based intrusion detection systems. *arXiv*, 2016. Disponível em: https://arxiv.org/abs/1611.01726. Acesso em: 17 nov. 2025.

[37] GRIMMER, Martin et al. Dataset Report: LID-DS 2021. *In: International Conference on Critical Information Infrastructures Security*. Cham: Springer Nature Switzerland, p. 63-73, 2022.

[38] WU, Felix et al. Pay less attention with lightweight and dynamic convolutions. *ArXiv*, 2019. Disponível em: https://arxiv.org/abs/1901.10430. Acesso em: 17 nov. 2025.

[39] E SILVA, Marcell Bruno Sousa. New type of aeronautical risk assessment: performance of Kohonen Self-Organizing Maps in identifying Brazilian aircraft with greater associated risks. *Journal of Genetic Engineering and Bio Research*, v. 4, n. 3, p. 340–350, 2022.

[40] SUTSKEVER, Ilya et al. Sequence to sequence learning with neural networks. *ArXiv*, 2014. Disponível em: https://arxiv.org/abs/1409.3215. Acesso em: 17 nov. 2025.

[41] LANKFORD, Séamus; AFLI, Haithem; WAY, Andy. Transformers for low-resource languages: Is Féidir Linn!. *ArXiv*, 2024. Disponível em: https://arxiv.org/abs/2403.01985. Acesso em: 17 nov. 2025.

[42] FLYNN, Thomas; YOO, Shinjae. Change detection with the kernel cumulative sum algorithm. In: 2019 IEEE 58th Conference on Decision and Control (CDC). IEEE, p. 6092-6099, 2019.

[43] TAKEUCHI, Jun-ichi; YAMANISHI, Kenji. A unifying framework for detecting outliers and change points from time series. *IEEE Transactions on Knowledge and Data Engineering*, v. 18, n. 4, p. 482–492, 2006.

[44] GUSTAFSSON, Fredrik. *Adaptive filtering and change detection*. New York: Wiley, 2000.

[45] FRANCIS, Navya; ALI, Anooja. Identifying and Managing Concept Drift in Machine Learning Through Page-Hinkley Test: Approaches, Obstacles, and Resolutions. In: International Conference on Recent Trends in Computing. Singapore: Springer Nature Singapore, p. 33-46, 2024.

[46] E SILVA, Marcell Bruno Sousa; BARRETO, André Noll. Spectrum sensing in cognitive radio networks through change detection technique. In: 2014 International Telecommunications Symposium (ITS). IEEE, p. 1-5, 2014.

[47] ZHANG, Yifei et al. Syscall-BSEM: Behavioral semantics enhancement method of system call sequence for high accurate and robust host intrusion detection. *Future Generation Computer Systems*, v. 125, p. 112–126, 2021.

[48] AL-SELWI, Safwan Mahmood et al. RNN-LSTM: From applications to modeling techniques and beyond—systematic review. *Journal of King Saud University – Computer and Information Sciences*, p. 102068, 2024.

[49] ABIBULLAEV, Berdakh; KEUTAYEVA, Aigerim; ZOLLANVARI, Amin. Deep learning in EEG-based BCIs: A comprehensive review of transformer models, advantages, challenges, and applications. *IEEE Access*, v. 11, p. 127271–127301, 2023.

APÊNDICE A - LSTM

Long Short-Term Memory (LSTM) é um tipo de Rede Neural Recorrente (RNN) projetada especificamente para lidar com o problema do desaparecimento do gradiente, que torna as RNNs tradicionais inadequadas para aprender dependências de longo prazo em dados sequenciais. As LSTMs são capazes de processar dados sequenciais e reter informações de etapas anteriores na sequência, permitindo-lhes prever efetivamente as etapas futuras. Essa característica as torna altamente adequadas para tarefas que envolvem dependências de longo prazo.

As LSTMs utilizam um conjunto de 'portas' para regular o fluxo de informações para dentro e para fora da célula de memória, o que lhes permite mitigar o problema do desaparecimento do gradiente. Esses portões são:

- porta de Esquecimento (*Forget Gate*): Decide quais informações devem ser descartadas da célula de memória. Ele recebe a entrada atual e o estado oculto anterior e produz um número entre e para cada número no estado da célula anterior. Um significa 'manter completamente' e um significa 'descartar completamente'.
- porta de Entrada (*Input Gate*): Decide quais novas informações serão armazenadas na célula de memória. Ele tem duas partes: uma camada sigmoide que decide quais valores serão atualizados e uma camada tanh que cria um vetor de novos valores candidatos para adicionar ao estado.
- porta de Saída (*Output Gate*): Decide qual será o valor de saída. Ele usa uma camada sigmoide para decidir quais partes do estado da célula serão a saída e, em seguida, passa o estado da célula através de tanh e o multiplica pela saída da camada sigmoide.

A arquitetura de uma célula LSTM é ilustrada na Figura 4, mostrando como esses portões interagem para controlar o fluxo de informações e manter a memória de longo prazo.

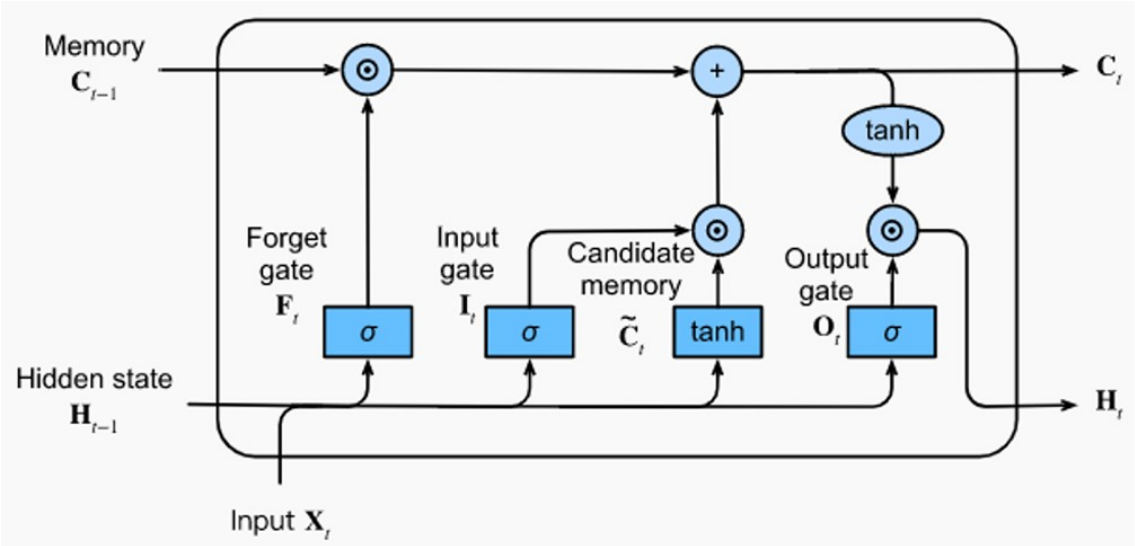

Figura 11: Arquitetura de uma célula Long Short-Term Memory (LSTM), adaptado de [48].

APÊNDICE B - TRANSFORMERS

Este anexo visa fornecer uma compreensão didática dos princípios fundamentais dos Transformers [49].

A arquitetura Transformer, introduzida no artigo seminal "Attention Is All You Need" por Vaswani et al. em [17], revolucionou o campo do Processamento de Linguagem Natural (PLN) e, posteriormente, diversas outras áreas, incluindo visão computacional e BCIs. Diferentemente de modelos anteriores como RNNs e LSTMs, os Transformers abandonam a abordagem sequencial e se baseiam inteiramente em mecanismos de atenção, permitindo o processamento paralelo de dados e a captura de dependências de longo alcance de forma mais eficiente.

Os componentes-chave da arquitetura Transformer incluem:

- Mecanismo de Autoatenção (*Self-Attention Mechanism*): Permite que o modelo pondere a importância de diferentes partes dos dados de entrada em relação umas às outras. Isso é crucial para entender o contexto e as relações entre os elementos de uma sequência.
- Codificação Posicional (*Positional Encoding*): Como os Transformers processam as sequências em paralelo, eles não possuem uma noção inerente da ordem dos elementos. A codificação posicional adiciona informações sobre a posição relativa de cada elemento na sequência, garantindo que o modelo possa considerar a ordem dos dados.
- *Multi-Head Attention*: Permite que o modelo foque em diferentes partes da entrada simultaneamente, capturando diversas relações e aspectos dos dados. Isso é alcançado através da execução de múltiplos mecanismos de atenção em paralelo e da concatenação de seus resultados.

As vantagens da arquitetura Transformer incluem a capacidade de paralelização, escalabilidade e flexibilidade. O processamento paralelo de todos os pontos de dados resulta em tempos de treinamento mais rápidos. Além disso, são altamente escaláveis, com modelos maiores capazes de capturar padrões intrincados em grandes conjuntos de dados, levando a um desempenho de ponta em várias tarefas. Embora inicialmente projetados para tarefas de

PLN, os Transformers demonstraram grande potencial em outros domínios, como visão e BCIs [49].

O sucesso do modelo Transformer inicial levou ao desenvolvimento de inúmeras variantes adaptadas para diferentes tarefas, como BERT, GPT e Vision Transformers [49]. A Figura 1 apresenta uma visão geral da arquitetura Transformer.

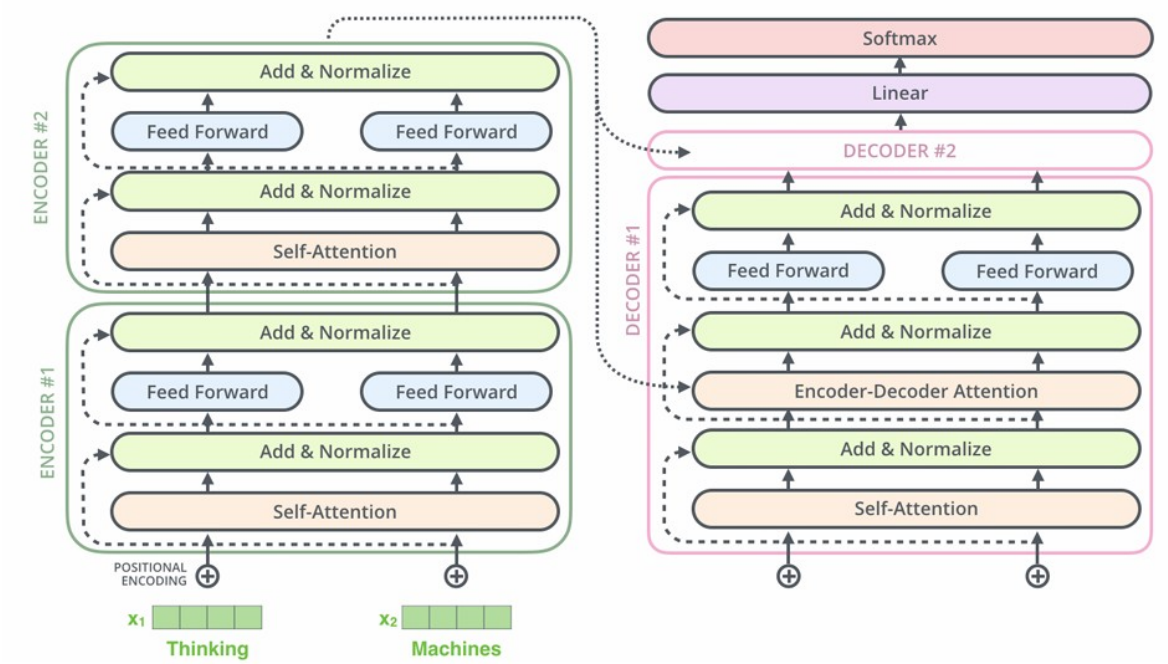

Figura 12: Visão geral da arquitetura Transformer [49].

B.1 Mecanismo de Autoatenção (*Self-Attention*)

O mecanismo de autoatenção é o coração da arquitetura Transformer, permitindo que o modelo avalie a importância de diferentes palavras (ou tokens) em uma sequência em relação a outras palavras na mesma sequência. Isso é fundamental para capturar dependências de longo alcance e entender o contexto de cada elemento. Em vez de processar cada palavra sequencialmente, como em RNNs, a autoatenção permite que o modelo considere todas as palavras de uma vez, atribuindo pesos de atenção que indicam a relevância de cada palavra para a representação da palavra atual.

Matematicamente, a autoatenção é calculada a partir de três vetores principais para cada palavra de entrada: Query (Q), Key (K) e Value (V). Esses vetores são obtidos a partir

da incorporação (embedding) da palavra de entrada, multiplicada por três matrizes de peso distintas que são aprendidas durante o treinamento. O processo envolve os seguintes passos:

- Cálculo das Pontuações de Atenção: Para cada Query, calcula-se uma pontuação de atenção com todas as Keys. Isso é feito geralmente através do produto escalar entre Q e K. Um produto escalar alto indica que a Query e a Key são semanticamente semelhantes, sugerindo uma forte relação entre as palavras correspondentes,
- Escala: As pontuações de atenção são divididas pela raiz quadrada da dimensão dos vetores Key (d_k). Isso ajuda a estabilizar o gradiente durante o treinamento, evitando que os produtos escalares se tornem muito grandes,
- Função Softmax: A função softmax é aplicada às pontuações escaladas para normalizá-las, transformando-as em probabilidades. Essas probabilidades indicam o quanto cada palavra na sequência deve ser "atendida" ao processar a palavra atual,
- Ponderação dos Valores: As probabilidades de atenção são multiplicadas pelos vetores Value (V). Isso resulta em uma soma ponderada dos Values, onde as palavras mais relevantes recebem maior peso. O resultado é o vetor de saída para a palavra atual, que encapsula o contexto da palavra em relação a toda a sequência.

Este mecanismo permite que o Transformer capture relações complexas e dependências contextuais que seriam difíceis de modelar com arquiteturas sequenciais tradicionais. A capacidade de "olhar" para toda a sequência simultaneamente é o que confere aos Transformers sua notável eficiência e poder de representação.

B2 ***Codificação Posicional (Positional Encoding)***

Ao contrário das RNNs que processam sequências token por token, os Transformers processam todos os tokens de entrada em paralelo. Embora isso confira uma grande vantagem em termos de velocidade de treinamento, o modelo perde a informação sobre a ordem posicional dos tokens na sequência. Para compensar essa perda, os Transformers incorporam a Codificação Posicional.

A codificação posicional é um vetor adicionado ao embedding de cada token de entrada. Esses vetores contêm informações sobre a posição absoluta ou relativa de cada token na sequência. Dessa forma, o modelo pode distinguir entre tokens que são semanticamente idênticos, mas que aparecem em diferentes posições na frase, o que é crucial

para a compreensão do significado. Por exemplo, em frases como "o gato persegue o rato" e "o rato persegue o gato", as palavras são as mesmas, mas a ordem muda completamente o significado. A codificação posicional permite que o Transformer capture essa diferença.

Existem diferentes abordagens para a codificação posicional, sendo a mais comum a utilização de funções seno e cosseno de diferentes frequências. Essa escolha permite que o modelo aprenda a codificar posições relativas, o que é benéfico para lidar com sequências de comprimentos variados. A fórmula para a codificação posicional é a seguinte:

```
 PE(pos, 2i) = sin(pos / 10000^(2i/d_model))
PE(pos, 2i+1) = cos(pos / 10000^(2i/d_model))
```

Onde:

- pos <=> posição do token na sequência,
- i <=> dimensão dentro do vetor de codificação posicional,
- d_model <=> dimensão do modelo (a dimensão dos embeddings e dos vetores de saída do Transformer).

Essa abordagem garante que cada posição tenha uma codificação única e que as relações de distância entre as posições sejam preservadas. A codificação posicional é um elemento vital que permite aos Transformers processar sequências de forma não sequencial, mantendo a capacidade de entender a ordem e a estrutura dos dados.

B3 Multi-Head Attention

O mecanismo de autoatenção, por si só, é poderoso, mas o Transformer o aprimora ainda mais com a introdução do Multi-Head Attention (Atenção Multi-Cabeça). Em vez de realizar uma única função de atenção, o Multi-Head Attention executa o mecanismo de atenção em paralelo várias vezes, com diferentes conjuntos de matrizes de projeção (Q, K, V). Isso permite que o modelo aprenda diferentes representações de subespaços para as informações de Query, Key e Value.

Cada "cabeça" de atenção é independente e aprende a focar em diferentes aspectos da sequência de entrada. Por exemplo, uma cabeça pode aprender a focar em relações sintáticas, enquanto outra pode se concentrar em relações semânticas. Após a execução paralela de todas as cabeças, os resultados de cada cabeça são concatenados e, em seguida, projetados

linearmente para a dimensão de saída desejada. Isso permite que o modelo capture uma gama mais rica e diversificada de dependências e informações contextuais.

Formalmente, o Multi-Head Attention é definido como:

```
MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O
  onde head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)
```

Em que:

- h <=> número de cabeças de atenção;
- W_i^Q , W_i^K , W_i^V <=> matrizes de peso para a i-ésima cabeça de atenção, que são aprendidas durante o treinamento;
- W^O <=> matriz de peso para a projeção linear final;
- Attention <=> função de atenção que calcula a autoatenção por meio de operações entre matrizes.

O uso de múltiplas cabeças de atenção confere ao Transformer uma capacidade aprimorada de modelar relações complexas e de capturar informações de diferentes perspectivas, tornando-o mais robusto e eficaz em uma variedade de tarefas.