João Pedro Pires Silva

Comparação entre o impacto de diferentes Datasets em modelos de Redes Neurais Artificiais na detecção de intrusões em redes de computadores

Uberlândia, MG

2026

João Pedro Pires Silva

Comparação entre o impacto de diferentes Datasets em modelos de Redes Neurais Artificiais na detecção de intrusões em redes de computadores

Trabalho de Conclusão de Curso da Engenharia Eletrônica e de Telecomunicações da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia Eletrônica e de Telecomunicações.

Universidade Federal de Uberlândia - UFU
Faculdade de Engenharia Elétrica - FEELT

Orientador: Prof. Dr. Éderson Rosa da Silva

Uberlândia, MG
2026

Agradecimentos

Agradeço a meus pais e meu irmão pelo apoio na realização deste trabalho e pelo apoio durante toda a graduação. Agradeço também meus amigos e colegas que deixaram toda essa jornada mais leve e única. Sou grato por todo o corpo docente da Universidade Federal de Uberlândia, em especial ao professor e orientador Éderson por todos os ensinamentos e tempo dedicado.

Resumo

Este estudo aborda a aplicação de Redes Neurais Artificiais (RNAs) voltadas à prevenção de intrusões em redes de computadores, fundamentando-se na premissa de que o sucesso de um modelo de inteligência artificial é pautado pelo aprendizado, estrutura de raciocínio, custo computacional e, primordialmente, pelo banco de dados utilizado. O objetivo central deste projeto é analisar como a estrutura dos valores e a organização de um conjunto de dados afetam os parâmetros e a efetividade de uma RNA. A metodologia consiste no treinamento de um modelo de rede neural do tipo *Transformer* utilizando dois bancos de dados distintos: o UNSW-NB15, gerado pela ferramenta *IXIA PerfectStorm* no *Cyber Range Lab* da *University of New South Wales* (UNSW) Canberra, e sua respectiva versão *Netflow*. Para garantir a comparabilidade, os parâmetros do modelo são mantidos constantes, permitindo a análise dos resultados de acurácia, *recall* e *F1-Score* em ambos os cenários, além da realização de uma validação cruzada entre os bancos de dados. A fundamentação teórica explora a estrutura inspirada no sistema neural biológico e a evolução tecnológica para mecanismos de auto-atenção (*self-attention*), que permitem identificar dependências de longo alcance de forma eficiente. O trabalho também discute técnicas de otimização e regularização, como o uso dos otimizadores Adam e AdamW, além da técnica de *Dropout* para mitigar o *overfitting* e garantir a capacidade de generalização do agente diante de informações inéditas. Assim, a partir dos testes e das análises realizadas, conclui-se que a organização das informações dos ataques de rede impacta diretamente no sucesso do processo de treinamento e na performance final do modelo de segurança.

Palavras-chaves: *Datasets*; Prevenção de Intrusões; Redes Neurais Artificiais; Segurança de Redes; *Transformer*.

Abstract

This study addresses the application of Artificial Neural Networks (ANNs) focused on intrusion prevention in computer networks, based on the premise that the success of an artificial intelligence model is guided by its learning process, reasoning structure, computational cost, and, primarily, the database utilized. The central objective of this project is to analyze how the structure of values and the organization of a dataset affect the parameters and effectiveness of an ANN. The methodology consists of training a Transformer-based neural network model using two distinct databases: the UNSW-NB15, generated by the IXIA PerfectStorm tool at the University of New South Wales (UNSW) Canberra Cyber Range Lab, and its respective Netflow version. To ensure comparability, the model parameters will be kept constant, allowing for the analysis of accuracy, recall, and F1-Score results in both scenarios, in addition to a cross-validation between the databases. The theoretical framework explores the structure inspired by the biological neural system and the technological evolution toward self-attention mechanisms, which allow for the efficient identification of long-range dependencies. The work also discusses optimization and regularization techniques, such as the use of Adam and AdamW optimizers, as well as the Dropout technique to mitigate overfitting and ensure the agent's generalization capability when facing unseen information. Therefore, with testing and evaluating the results, it is concluded that the organization of network attack information directly impacts the success of the training process and the final performance of the security model.

Key-words: Artificial Neural Network; Datasets; Intrusion Prevention; Network Security; Transformer.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BERT	Bidirectional Encoder Representations from Transformers
DNS	Domain Name System
DoS	Denial of Service
FFN	Feedforward Neural Network
FTP	File Transfer Protocol
GELU	Gaussian Error Linear Unit
GPT	Generative Pre-trained Transformer
HTTP	Hypertext Transfer Protocol
IDS	Intrusion Detection System
ML	Machine Learning
MLP	Multi-Layer Perceptron
NIDS	Network Intrusion Detection System
PLN	Processamento de Linguagem Natural
RNA	Rede Neural Artificial
RNN	Recurrent Neural Network
SMTP	Simple Mail Transfer Protocol
TCP	Transmission Control Protocol

Lista de símbolos

x_i	Sinal de entrada do neurônio
w_{ij}	Peso sináptico da ligação entre neurônios
b	Termo de viés (*bias*)
ϕ	Função de ativação não-linear
y_j	Sinal de saída do neurônio
Q	Vetor de Consulta (*Query*)
K	Vetor de Chave (*Key*)
V	Vetor de Valor (*Value*)
d_k	Dimensão das chaves no mecanismo de atenção
d_{model}	Dimensão do vetor de representação (*embedding*)
k	Número de partições na validação cruzada (*folds*)

Sumário

1 Introdução

O estudo de Redes Neurais Artificiais (RNAs) é pautado em algumas vertentes como: o aprendizado do modelo, a estrutura de raciocínio da rede, o custo computacional da RNA e o banco de dados utilizado para o treinamento do agente. Dessa forma, é notório que existem vários fatores que contribuem para o sucesso de um modelo, onde pequenas alterações corroboram para o aumento da acurácia, termo este que é amplamente utilizado no tema para indicar o sucesso de uma RNA.

Dado a popularização dessa tecnologia, surgiram diversas novas maneiras de se utilizar um modelo de rede neural, como o uso de RNAs para a prevenção de intrusões em redes de computadores. A partir de *datasets* que simulam variados cenários de tráfego virtual benignos e corrompidos por diversos tipos de ataques, torna-se possível treinar um modelo capaz de rotular esses ataques e preveni-los.

Por tratar-se de uma vasta opção de tipos de ataques, o tipo de RNA utilizado para esse serviço é imprescindível, onde, existem maneiras de reconhecer padrões capazes de evitar ataques mesmo que esses não sejam representados no *dataset* em questão. Isso torna-se possível a partir do tipo de RNA utilizado.

Para discorrer acerca do modelo específico de rede neural utilizado neste projeto é necessário um entendimento acerca das diferenças entre tipos de RNA. O ponto de partida para discorrer do assunto são os modelos de *Recurrent Neural Network* (RNN), onde o estudo de retroprogagação (*backpropagation*) foi introduzido por (1) e que permite os dados sequenciais e temporais serem analisados a partir dos pesos das conexões e do gradiente da função. De maneira paralela também existem modelos baseados em mecanismos de atenção, *Transformers*, modelos que não utilizam recorrência, ou seja, processam os valores de forma paralela como elucidado por (2).

Neste trabalho, o objetivo será explorar o tema de prevenção de ataques de rede através de RNAs e como a organização das informações desses ataques impactam no sucesso do agente. Portanto, para realizar os treinamentos serão utilizados dois bancos de dados que paralelamente integram o mesmo conjunto de dados porém a partir de abordagens diferentes, o *dataset* UNSW-NB15 e sua versão *NetFlow* o *dataset* NF-UNSW-NB15, descritos em (3) e (4), respectivamente.

O mesmo modelo de RNA será utilizado para ser treinado a partir dos dois diferentes bancos de dados, onde os parâmetros serão mantidos a fim de comparar os resultados de acurácia, *recall* e *F1-Score* dos dois *datasets* e pontuar as diferenças de prevenção a partir da estrutura dos dados capturados.

1.1 Motivação da Pesquisa

Com o advento do tema Inteligência Artificial e a popularização de recursos relacionados a essa tecnologia, tornou-se de maior uso comum as RNAs de forma geral. Com isso, em prol da tentativa de aumentar o nível de performance de um modelo o estudo a seguir foca em como a estrutura dos dados de treinamento de uma rede neural impacta no sucesso de sistemas de detecção de intrusão. Essa comparação aborda a análise de dois *datasets* e explora as diferenças dos mesmos na posição de *training set* para o mesmo modelo de rede neural. A escolha do campo de IDS é aprofundar o estudo na área de redes de computadores e assim analisar as problemáticas e as soluções de segurança de rede para os usuários.

1.2 Objetivos

O principal objetivo desse projeto é analisar como a estrutura dos valores e a organização de um conjunto de dados afeta os parâmetros de uma RNA. Busca-se ver como é o funcionamento de uma rede neural, principalmente sua efetividade, a partir de como é estruturado o *dataset* utilizado para o treinamento do sistema. Nota-se, ainda, que nesse trabalho uma validação cruzada entre os bancos de dados é explorada. Assim, busca-se uma maior efetividade na proteção dos usuários diante dos ataques estudados garantindo um maior sucesso para sistemas de detecção de intrusões perante redes de computadores.

1.3 Hipóteses

A partir da estruturação do modelo e do entendimento das particularidades de cada *dataset*, assim como os ataques de rede que estão inseridos nos mesmos, é possível abordar se um sistema é mais eficaz que o outro ou se existem fatores mais influentes a serem tratados. Com isso, surge a necessidade de uma maior especificação e uma padronização dos parâmetros utilizados para uma análise mais completa das abordagens testadas. Fica evidente, então, como é imprescindível que para comparar o mesmo modelo de RNA os parâmetros utilizados devem ser pensados em prol de atingir a imparcialidade, como é ilustrado no Capítulo 3.

1.4 Organização do Documento

A organização do documento é conforme descrito a seguir. No Capítulo 2, encontra-se a fundamentação teórica, assim como referências bibliográficas e principais trabalhos correlatos. Encontra-se também conceitos relacionados ao tema Redes Neurais Artificiais e ao tema Redes de Computadores. No Capítulo 3 exibe-se detalhes técnicos, metodologia do

projeto, arquitetura dos dados e organização do código utilizado. Em seguida, no Capítulo 4 os resultados são apresentados e discutidos, bem como os valores de acurácia para cada *dataset* e um comparativo entre o funcionamento do modelo dado cada *training set*. Por fim, no Capítulo 5 têm-se a conclusão obtida assim como possíveis pontos a serem discutidos e melhorados em futuros projetos.

2 Referenciais Teóricos

O desenvolvimento de sistemas de detecção de intrusão (IDS) baseados em modelos de aprendizado profundo (*Deep Learning*) fundamenta-se na convergência entre a arquitetura computacional, a representatividade dos dados e a natureza das ameaças cibernéticas. Conforme discorrido por Goodfellow et al. (5), as Redes Neurais Artificiais (RNAs) evoluíram de modelos lineares simples para estruturas multicamadas capazes de extrair características complexas de grandes volumes de informação, tornando-se ferramentas essenciais no reconhecimento de padrões de tráfego anômalo.

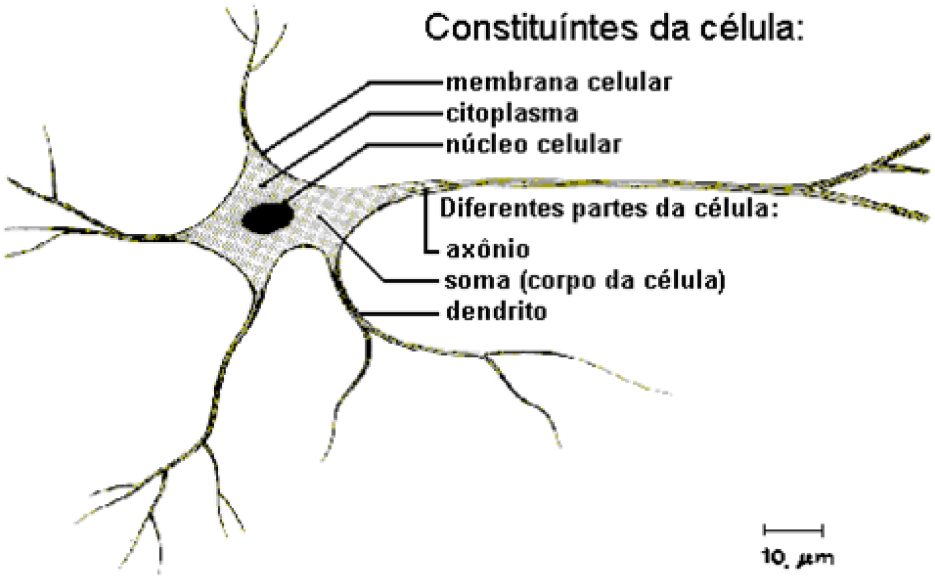
Nesse contexto, a eficácia de um modelo não reside apenas em sua arquitetura, mas na qualidade do *dataset* utilizado para o treinamento. Como apontado por Sarhan et al. (4), a estrutura e a taxonomia dos dados de rede sejam eles capturas de pacotes brutos ou fluxos agregados, definem o limite de desempenho que um classificador pode atingir. Além das abordagens convencionais, o advento dos mecanismos de *Self-Attention*, introduzidos por Vaswani et al. (2), permitiu que as redes neurais passassem a processar informações considerando correlações globais entre atributos, o que se mostra particularmente promissor para identificar a lógica sequencial de ataques de rede modernos.

A compreensão deste trabalho perpassa, portanto, pela análise integrada das RNAs, do papel estratégico dos conjuntos de dados, da inovação trazida pelos modelos de atenção e do panorama técnico das ameaças que buscam comprometer a integridade dos sistemas computacionais.

2.1 Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) são técnicas computacionais com uma estrutura inspirada no sistema neural de seres inteligentes que adquirem conhecimento a partir de experiência. Como apresentado na Figura 1 de uma célula neural, é possível discorrer acerca do funcionamento do seu semelhante digital.

Figura 1 – Exemplo de célula neural.



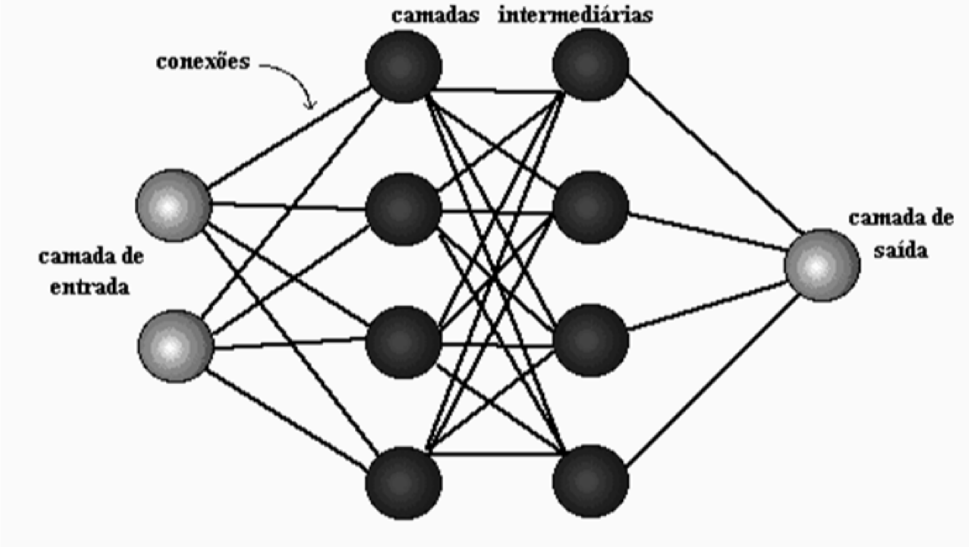
Fonte: (6)

O neurônio funciona como um cabo transmissor responsável pela ponte entre o processo de sinapse nervosa, onde há o recebimento, a condução e a transmissão de estímulos nervosos no interior dessas células. De forma resumida, esse processo ocorre através dos axônios (responsáveis pela transmissão), dendritos (responsáveis pelo recebimento dos sinais) e corpo da célula (responsável pela integração do pulso). A comunicação entre dois neurônios é o processo chamado de sinapse nervosa.

De forma parelha, uma Rede Neural Artificial é composta por camadas de unidades de processamento associadas através de pesos (sinapses). A entrada do modelo (dendritos) recebe os sinais, os nós (corpo celular) processam os valores e são capazes de avaliar cada "bloco de informação" indicando uma saída (axônios) da unidade baseando-se no grau de importância desse dado.

Grande parte dos modelos de RNA possuem regras de treinamento, onde os pesos são ajustados aos padrões apresentados. Usualmente as arquiteturas são divididas em camadas, assim sendo capazes de manter uma comunicação com os níveis posteriores de conexão, como visto na Figura 2.

Figura 2 – Exemplo de organização das camadas de uma RNA.



Fonte: (6)

A camada de entrada é onde os padrões são apresentados a rede, nas camadas intermediárias ocorre a maior parte do processamento, em que as conexões são ponderadas e as características da informação são extraídas. Por fim, na camada de saída o resultado é concluído.

Assim, para consolidar de forma matemática o funcionamento de uma rede neural artificial, conforme dito por (7), o funcionamento de um neurônio artificial baseia-se na soma ponderada de suas entradas x_i multiplicadas por pesos sinápticos w_{ij}, somadas a um termo de viés (*bias*) b. O resultado passa por uma função de ativação não-linear ϕ. A saída y_j é expressa pela Equação 2.1:

$$y_j = \phi\left(\sum_{i=1}^{n} w_{ij}x_i + b\right) \tag{2.1}$$

Uma característica primordial das RNAs é a habilidade de aprender a partir do ambiente e aumentar o rendimento do modelo. Esse processo de iteração pode ser dividido entre aprendizado supervisionado e aprendizado não supervisionado. Aprendizado supervisionado é quando um agente externo indica a saída esperada da rede, por exemplo, em uma situação com diversas imagens aleatórias e uma gama de imagens de automóveis, se a intenção é detectar imagens de veículos na saída do modelo torna-se possível verificar se a classificação da RNA está sendo feita de forma correta. No aprendizado não supervisionado não é possível validar a classificação na saída, ou seja, a categorização dos itens na saída do modelo não é realizada.

Uma apresentação dos N pares de entrada e saída no processo de aprendizagem de um modelo é chamada de ciclo. Nele, pode haver pequenas correções nos pesos: padrão

ou batch. O modo padrão apresenta correções à rede a cada apresentação do conjunto de treinamento, ou seja, em um ciclo N correções serão feitas. O modo *batch* apresenta correções por ciclo, ou seja, todo o conjunto é apresentado na saída, o erro médio é calculado e a partir dessa informação as correções são realizadas.

2.1.1 Redes Neurais de auto-atenção e seus parâmetros

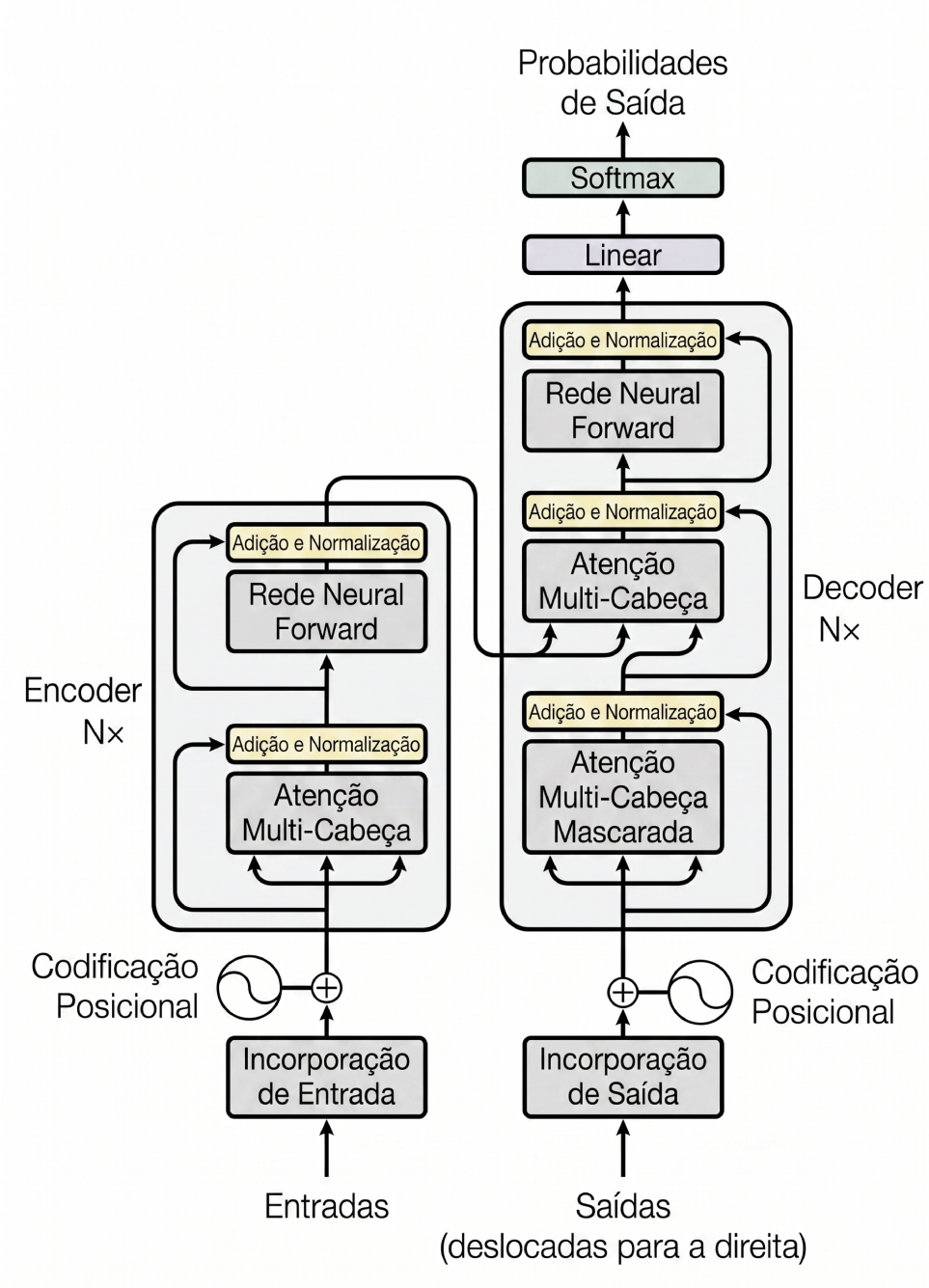
Alguns estudos datados de 2017, introduzidos por (2), trouxeram ideias de arquiteturas de RNAs que permitiram os modelos previamente conhecidos substituirem estruturas recorrentes e convolucionais pelo mecanismo de auto-atenção (*self-attention*), em que passou a ser possível identificar dependências de longo alcance de forma mais eficiente. Sua composição baseia-se em um sistema de codificador-decodificador com atenção de múltiplas cabeças, que hierarquiza a relevância dos *tokens* na sequência. Elementos como codificações posicionais, redes *feedforward*, normalização de camada e conexões residuais garantem a captura da ordem dos dados e a estabilidade do treinamento via *backpropagation*, algoritmo popularizado por (1) que visa minimizar erros de predição. Essa versatilidade fundamentou o sucesso de modelos como BERT e GPT, que redefiniram o estado da arte em diversos *benchmarks* da área. Um exemplo visual desta configuração é visto na Figura 3.

Esses mecanismos de atenção surgiram a partir de uma proposta de Vaswani (2) que impulsionou avanços significativos no campo, permitindo que os modelos foquem em partes específicas da informação disponível. O mecanismo opera através de três vetores principais: consulta (*query*), chave (*key*) e valor (*value*).

O processo, conhecido como *Scaled Dot-Product Attention*, calcula a compatibilidade entre a consulta e as chaves através do produto escalar. Para evitar que os gradientes se tornem demasiadamente pequenos em dimensões elevadas, o resultado é dividido pela raiz quadrada da dimensão das chaves. Em seguida, aplica-se uma função *softmax* para determinar os pesos que serão multiplicados pelos valores correspondentes.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.2}$$

Figura 3 – Arquitetura de um modelo *transformer*.

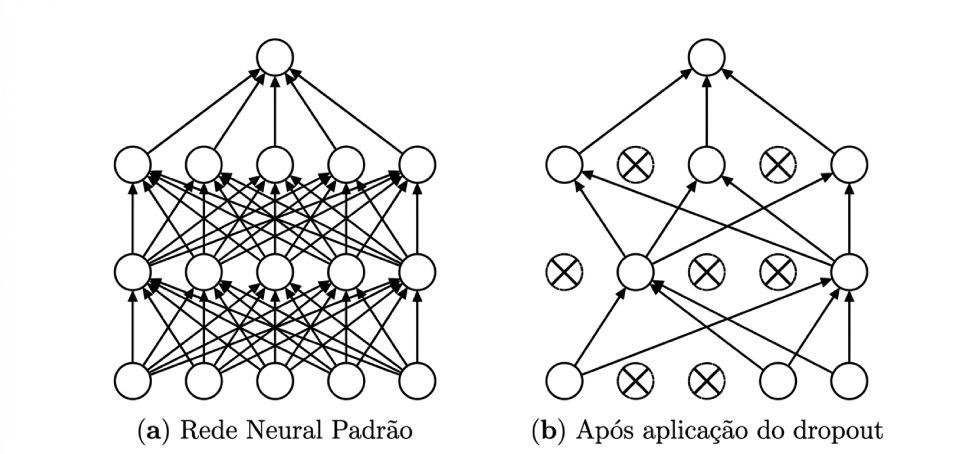


Fonte: Adaptado de (2)

A função *softmax* expressa na Equação 2.2 atua como um filtro seletor, priorizando os pares de chave-valor que possuem maior afinidade com a consulta. Esta capacidade de adaptação contextual é o que permite aos modelos de Processamento de Linguagem Natural (PLN) capturar relações complexas de longo alcance, tornando-os ferramentas essenciais para tarefas como tradução automática e sumarização. Para garantir o sucesso dessa abordagem, é importante também considerar as técnicas de otimização e regularização que são fundamentais para refinar os parâmetros do modelo e garantir que ele não apenas aprenda com os dados de treino, mas também apresente uma boa capacidade

de generalização diante de informações inéditas. Nesse contexto, destaca-se o uso do otimizador Adam e sua evolução, o AdamW, que se mostram eficazes ao adaptar as taxas de aprendizado de forma individual para cada parâmetro por meio de estimativas de momentos de primeira e segunda ordem. O diferencial do AdamW reside na incorporação do decaimento de peso (*weight decay*), que atua como uma penalidade para evitar o crescimento excessivo dos pesos e proporcionar maior estabilidade ao treinamento, mitigando problemas de convergência comuns em arquiteturas *Transformer*, como descreve (8). Somado a isso, o emprego do *dropout* reforça a robustez da rede ao excluir temporariamente neurônios e suas conexões durante o processo de aprendizado, o que impede a dependência mútua exagerada entre as unidades e previne o *overfitting*. O exemplo visual dessa técnica é visto na Figura 4.

Figura 4 – Arquitetura de um *dropout*.



(**a**) Rede Neural Padrão (**b**) Após aplicação do dropout

Fonte: Adaptado de (9)

2.1.2 FT-Transformer e Dados Tabulares

Embora a arquitetura *Transformer* tenha sido originalmente concebida para o Processamento de Linguagem Natural (PLN), avanços recentes permitiram a sua adaptação para o domínio de dados tabulares heterogêneos através do modelo FT-*Transformer* (*Feature Tokenizer + Transformer*) como elaborado por (10). Esta evolução é particularmente relevante no contexto de detecção de intrusões, onde a eficácia do modelo está intrinsecamente ligada à sua capacidade de extrair características complexas de fluxos de rede assim como em modelos que utilizam tecnologias exclusivas para dados tabulares, como o *Tabnet* (11).

O diferencial desta arquitetura reside no componente *Feature Tokenizer*, que transforma cada atributo individual, seja ele numérico ou categórico, em representações vetoriais chamadas de *embeddings*. Diferente das Redes Neurais Artificiais convencionais que processam entradas concatenadas, o FT-*Transformer* utiliza o mecanismo de auto-atenção

introduzido por (2) para analisar as interações globais entre esses *tokens* de características. Essa abordagem permite que a rede aprenda relações não-lineares sofisticadas entre campos distintos, como a correlação entre protocolos de transporte e indicadores temporais, fundamentais para a identificação de ataques contemporâneos de rastro reduzido (3, 4).

2.1.3 Métricas de avaliação de uma RNA

Para quantificar a eficácia de modelos de classificação, utiliza-se a Matriz de Confusão, que tabula as predições em relação aos rótulos reais. Conforme (12), os quatro resultados fundamentais são: Verdadeiros Positivos (VP), Verdadeiros Negativos (VN), Falsos Positivos (FP) e Falsos Negativos (FN).

A partir das categorias de resultados possíveis pode-se abordar a acurácia e o *F1-score*. A acurácia é a métrica mais intuitiva, representando a fração de predições corretas sobre o total de amostras. Sua formulação é dada pela Equação 2.3:

$$\text{Acurácia} = \frac{VP + VN}{VP + VN + FP + FN} \tag{2.3}$$

Apesar de sua popularidade, a acurácia pode ser enganosa em conjuntos de dados desbalanceados. Se 99% do tráfego for normal, um modelo que nunca detecta ataques ainda teria 99% de acurácia, falhando em seu propósito de segurança (13).

Para contornar as limitações da acurácia, utilizam-se a Precisão (*Precision*) e a Sensibilidade (*Recall*). A Precisão mede a proporção de ataques identificados que eram realmente ataques ($VP/(VP + FP)$), enquanto a Sensibilidade mede a capacidade do modelo de encontrar todos os ataques presentes no dado ($VP/(VP + FN)$).

O *F1-score* é a média harmônica entre precisão e sensibilidade, fornecendo uma métrica única que penaliza valores extremos em qualquer uma das duas. É definida pela Equação 2.4:

$$F1 = 2 \cdot \frac{\text{Precisão} \cdot \text{Sensibilidade}}{\text{Precisão} + \text{Sensibilidade}} \tag{2.4}$$

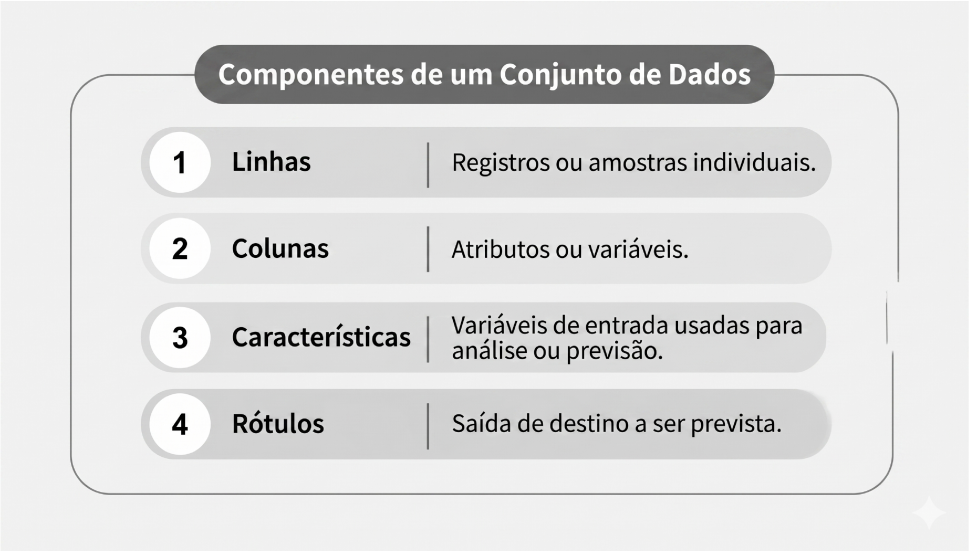
Segundo (13), o *F1-score* é superior para avaliar sistemas de detecção de intrusão, pois exige que o modelo seja preciso em seus alertas e abrangente em sua detecção simultaneamente.

2.2 Datasets

O *Dataset* para um modelo de rede neural é o conjunto de dados responsável por treinar, testar e validar um modelo. Considerado o grande combustível para alimentar

algoritmos, geralmente possuem formato tabular com linhas e colunas bem definidas, como pode ser visto na Figura 5, fornecendo assim informações acessíveis.

Figura 5 – Componentes de um *dataset*.



Fonte: Adaptado de (14)

Responsável por armazenar e categorizar uma coleção de dados acerca de certo assunto específico, um conjunto de *Datasets* pode ser chamado de *Database*, como previsto nas informações da Figura 6. Dessa forma, torna-se importante categorizar as camadas de informação para o treinamento de modelos de redes neurais onde esses valores implicam diretamente na eficiência do algoritmo que está sendo desenvolvido.

Figura 6 – Diferenças conceituais e práticas entre tipos de dados.

Aspecto	Dados	Dataset	Banco de Dados
Definição	Fatos brutos ou informações básicas sem contexto.	Uma coleção estruturada de entradas de dados relacionadas.	Uma coleção organizada de datasets armazenados sistematicamente.
Estrutura	Desorganizados, carecem de estrutura inerente.	Organizado em linhas e colunas.	Organizado em tabelas, muitas vezes em múltiplas dimensões.
Papel	Espinha dorsal para datasets e bancos de dados.	Estrutura dados e fornece insights significativos.	Define relacionamentos entre recursos de forma abrangente.
Manipulação	Não podem ser manipulados diretamente devido à falta de estrutura.	Podem ser analisados ou visualizados usando ferramentas como Tableau, Power BI, Python.	Podem ser manipulados com consultas, transações e scripts.
Uso	Precisa de pré-processamento e transformação antes do uso.	Usado para análise de dados, modelagem e visualização.	Usado para consultas, transações e gerenciamento de aplicações.

Fonte: Adaptado de (14)

Para o tema de detecção de intrusão, o revisionismo histórico da área revela uma transição de modelos estáticos e redundantes para bases de dados que priorizam a fidelidade

do tráfego e a complexidade das interações humanas. De forma breve, é interessante elucidar previamente características inatas entre os *datasets* que tratam o tema de formas diferentes: KDD99, um dos pioneiros na área de IDS e UNSW-NB15 e NF-UNSW-NB15 que são os objetos de estudo deste trabalho e representam diferentes abordagens para um mesmo conjunto de informação em diferentes pontos de vista.

O KDD *Cup* 1999 é derivado do experimento DARPA 1998 realizado pelo *MIT Lincoln Lab*. Por quase duas décadas, ele foi o padrão absoluto para testes de algoritmos de detecção de intrusão. Contudo, (15) conduziu uma revisão crítica fundamental, expondo que o *dataset* carecia de variabilidade estatística real. O tráfego de fundo era gerado sinteticamente com distribuições matemáticas simples que não refletiam o ruído de uma rede real. Além disso, a presença maciça de registros duplicados levava os modelos de *Machine Learning* a um (*overfitting*) em classes majoritárias, ignorando ataques raros mas críticos. Apesar de obsoleto para sistemas modernos, o KDD99 ainda é citado como a base conceitual que permitiu o nascimento da área.

Em resposta às falhas dos predecessores, (3) desenvolveram o UNSW-NB15 no *Cyber Range Lab* da Universidade de New South Wales. Este *dataset* representou um avanço significativo ao utilizar a ferramenta IXIA *PerfectStorm* para emular tráfego (HTTP, FTP, SMTP, DNS) misturado a atividades de ataque contemporâneas. Diferente do KDD99, o UNSW-NB15 foca em ataques de baixa volumetria e alta complexidade, como o *Analysis* e os *Fuzzers*. Ele introduziu 49 atributos de fluxo que permitem ao modelo analisar não apenas o conteúdo do pacote, mas o comportamento temporal da conexão, sendo ideal para validar arquiteturas baseadas em atenção que buscam padrões em sequências de dados. Sua versão *NetFlow* desenvolvida por (4) compactua com uma modernização dos padrões extraídos possibilitando a garantia de uma melhor aplicação de modelos de ML para a detecção de intrusões.

Assim, entende-se que a eficácia de uma Rede Neural Artificial (RNA) na detecção de intrusões depende diretamente da representatividade e da variedade das amostras contidas nos *datasets* de treinamento. A sistematização dos ataques apresentada posteriormente serve como base para avaliar se os dados de entrada refletem adequadamente a realidade das ameaças modernas, garantindo que o modelo aprenda a distinguir padrões sutis entre o tráfego legítimo e as diferentes modalidades de ataque malicioso.

A evolução histórica dos Sistemas de Detecção de Intrusão (IDS) baseados em aprendizagem profunda revela que a qualidade das predições está intrinsecamente ligada à taxonomia e ao balanceamento dos dados brutos. *Datasets* tradicionais enfrentam o desafio de capturar o tráfego de rede moderno, muitas vezes apresentando redundâncias ou falta de exemplos de ataques emergentes. A transformação de fluxos de pacotes em características estatísticas como duração de conexão, contagem de *bytes* e estados de *flags* TCP é o procedimento técnico que permite que a RNA identifique as assinaturas comportamentais

dos ataques.

Portanto, esta revisão acerca da trajetória dos modelos é uma etapa essencial para a análise comparativa entre as bases de dados, objetivo central deste estudo. Logo, busca-se identificar como a origem e a estruturação dos dados impactam a precisão e a capacidade de generalização das arquiteturas neurais propostas, especialmente em cenários onde classes de ataque são minoritárias em relação ao tráfego normal (problema de desbalanceamento de classes).

2.3 Ataques de Rede

Ataques de Rede são de forma categórica a ação criada que visa computadores ou elementos eletrônicos de um sistema de informação afim de acessar, destruir ou controlar dados ou assumir o controle de uma rede de comunicação. Por existir diversos tipos de ataques de rede, dado a fundamentação do trabalho, os tipos de ataques explorados são: *Fuzzers, Analysis, Backdoors, DoS, Exploits, Generic, Reconnaissance, Shellcode* e *Worms*. Os ataques são categorizados em diferentes famílias e a descrição de funcionamento de cada um é disponibilizada em (3). Para ataques que se destacam em reconhecimento e identificação de vulnerabilidades tem-se o reconhecimento (*Reconnaissance*), que consiste na coleta de informações sobre a infraestrutura alvo; a análise (*Analysis*), que envolve o estudo de tráfego, códigos e protocolos para identificar fragilidades; e o *Fuzzing*, técnica automatizada que submete sistemas a entradas aleatórias com o objetivo de revelar falhas. Ainda nessa perspectiva investigativa, incluem-se os ataques genéricos à criptografia (*Generic*), que exploram propriedades comuns de algoritmos de bloco sem depender de sua estrutura específica. Esses métodos têm como principal objetivo mapear vulnerabilidades e preparar etapas posteriores do ataque. Por outro lado, há os ataques voltados à exploração efetiva e ao comprometimento do sistema alvo, caracterizados pela ação direta sobre suas vulnerabilidades. A exploração (*Exploits*) consiste no uso prático de falhas para obtenção de acesso não autorizado ou execução de comandos indevidos, podendo envolver a injeção de *Shellcode* para controle do ambiente comprometido ou a instalação de *Backdoors* que garantam acesso persistente. Nesse contexto, destacam-se também os *Worms*, que possuem capacidade de autorreplicação e propagação automática em redes, ampliando o alcance da infecção. Inclui-se ainda o ataque de negação de serviço (DoS), cujo objetivo é tornar um serviço indisponível por meio do envio massivo de requisições ou sobrecarga de recursos computacionais, comprometendo a disponibilidade do sistema. Em conjunto, tais ataques visam afetar a confidencialidade, a integridade e, sobretudo, a disponibilidade das informações.

A sistematização apresentada na Tabela 1 evidencia que a segurança de rede não é um estado estático, mas um processo contínuo de antecipação e resposta. Ao agrupar os

Tabela 1 – Classificação sistemática dos ataques de rede explorados

Tipo de Ataque	**Classificação**	**Impacto Principal**
Fuzzers	Varredura	Instabilidade de serviço
Analysis	Penetração	Vazamento de informação
Backdoors	Penetração	Controle remoto persistente
DoS	Apreensão	Indisponibilidade
Exploits	Penetração	Execução de código arbitrário
Worms	Varredura	Propagação lateral automática

Fonte: Adaptado de (3).

ataques em categorias assim como descreve (3) torna-se possível compreender o ciclo de vida de uma invasão, desde a sondagem inicial até o comprometimento total dos ativos. No contexto deste trabalho, essa categorização é fundamental, pois a eficácia de uma Rede Neural Artificial na detecção de intrusões depende diretamente da representatividade e da variedade das amostras contidas nos *datasets* de treinamento.

2.4 Trabalhos Relacionados

A principal inspiração do trabalho são as pesquisas de (3) e (4) e as metodologias de ML idealizadas por (2). Todavia, com a evolução de *datasets* que captam tráfegos de rede e das redes neurais artificiais, surge a possibilidade do uso de novas tecnologias de IDS de forma variada, como visto em (16) em que diferentes tipos de RNAs são utilizadas para comparar métricas de detecção a partir de um mesmo conjunto de informação. O tema também é idealizado pelo próprio autor do *dataset* UNSW-NB15, previsto em sua publicação (17).

2.5 Considerações Finais

É a partir de toda a fundamentação teórica que torna-se possível prosseguir com a parte de estrutura e teste do modelo em questão. Através da informação dividida entre três blocos majoritários: Redes Neurais Artificiais; *Datasets* e Ataques de Rede a metodologia do projeto é definida no Capítulo 3 com todas as particularidades do modelo. As definições abordadas são novamente pontuadas e referenciadas a partir do uso de cada uma, assim, é possível abordar de forma mais criteriosa o algoritmo de teste do projeto.

3 Metodologia

Visando atingir os objetivos do trabalho, a arquitetura do sistema foi dividida em quatro módulos: (i) *Datasets*, (ii) Modelo de RNA *Transformer*, (iii) Treinamento do modelo e (iv) Análise dos valores obtidos.

3.1 Datasets

Este módulo enuncia as características utilizadas no projeto para cada *Dataset*, onde descrições inatas de cada um dos conjuntos de informação a serem testados serão elucidados e diferenciados

3.1.1 Dataset UNSW-NB15

Como foi previamente informado, o *Dataset* UNSW-NB15 foi desenvolvido no *Cyber Range Lab* do *Australian Centre for Cyber Security* (ACCS), de (3). Este *dataset* foi selecionado por superar limitações de bases de dados obsoletas, como o KDD99, ao refletir com maior precisão o tráfego de rede contemporâneo e ataques de rastro reduzido (*low footprint*). A infraestrutura de geração dos dados utilizou a ferramenta IXIA *PerfectStorm* para simular um ambiente de rede real com três servidores virtuais, roteadores e *firewalls*, garantindo uma composição híbrida entre tráfego normal e atividades maliciosas.

Tabela 2 – Categorias de atributos do conjunto UNSW-NB15.

Categoria	**Qtd.**	**Exemplos de Atributos**
Fluxo (*Flow*)	5	*srcip, sport, dstip, dsport, proto*
Básicos (*Basic*)	13	*state, dur, sbytes, dbytes, sttl, dttl*
Conteúdo (*Content*)	8	*swft, dmeansz, trans_depth, res_bdy_len*
Tempo (*Time*)	9	*sjit, djit, stime, ltime, tcprtt, synack*
Adicionais	12	*ct_srv_src, ct_dst_ltm, ct_src_dport_ltm*
Total	**49**	-

Fonte: Adaptado de (3).

O fluxo metodológico para a extração das características segue a arquitetura descrita nos seguintes processos: captura de arquivos brutos (*pcap*) via *Tcpdump*, processamento pelas ferramentas *Argus* e *Bro-IDS*, até a aplicação de doze algoritmos customizados para a criação de atributos complexos. O resultado é uma base consolidada com 49 características, detalhadas nas Tabela 2 adaptada de (3), que abrangem desde informações básicas de fluxo até indicadores temporais e de conteúdo. No total, o *dataset* compreende nove famílias de ataques (conforme a distribuição descrita pela Tabela 3), fornecendo uma base abrangente para o treinamento e validação de sistemas de detecção de intrusão (NIDS).

Tabela 3 – Distribuição de registros no *dataset* UNSW-NB15.

Categoria	**Nº de Registros**	**Descrição Curta**
Normal	2.218.761	Transações naturais.
Generic	215.481	Técnicas contra cifras de bloco.
Exploits	44.525	Exploração de vulnerabilidades conhecidas.
Fuzzers	24.246	Dados aleatórios para suspender programas.
DoS	16.353	Interrupção de serviços de rede.
Reconnaissance	13.987	Ataques de coleta de informações.
Analysis	2.677	Port scan, spam e penetrações html.
Backdoors	2.329	Acesso furtivo ao sistema.
Shellcode	1.511	Payload para exploração de vulnerabilidades.
Worms	174	Replicação para outros computadores.
Total	**2.540.044**	-

Fonte: Adaptado de (3).

3.1.2 Dataset NF-UNSW-NB15

A variante NF-UNSW-NB15, documentada por (4), representa uma evolução metodológica ao converter as capturas *pcap* originais para o formato *NetFlow* versão 9 através da ferramenta *nProbe*. Esta abordagem foca-se na sumarização do tráfego em fluxos, o que, segundo os autores, oferece maior escalabilidade e menor custo de armazenamento em comparação com a captura total de pacotes. Conforme descrito na metodologia de extração de (4), foram selecionadas 12 características fundamentais baseadas no cabeçalho dos pacotes, conforme a Tabela 4. Esta versão consolidada do NF-UNSW-NB15 totaliza 1.623.118 fluxos, com uma distribuição de 4,46 % de amostras de ataque e 95,54 % de tráfego benigno como descrito na Tabela 5, permitindo uma avaliação de modelos de *Machine Learning* em cenários de rede de alta velocidade onde a extração de atributos complexos seria inviável.

Tabela 4 – Principais atributos *NetFlow* extraídos para o NF-UNSW-NB15.

Atributo	**Categoria**	**Descrição**
IPV4_SRC_ADDR	Fluxo (*Flow*)	Endereço IPv4 de origem.
IPV4_DST_ADDR	Fluxo (*Flow*)	Endereço IPv4 de destino.
L4_SRC_PORT	Fluxo (*Flow*)	Porta de origem da camada de transporte.
L4_DST_PORT	Fluxo (*Flow*)	Porta de destino da camada de transporte.
PROTOCOL	Fluxo (*Flow*)	Identificador do protocolo IP.
L7_PROTO	Conteúdo (*Content*)	Identificador do protocolo de aplicação.
TCP_FLAGS	Básicos (*Basic*)	Acumulado de todas as *flags* TCP.
IN_BYTES	Básicos (*Basic*)	Número de bytes recebidos.
OUT_BYTES	Básicos (*Basic*)	Número de bytes enviados.
IN_PKTS	Básicos (*Basic*)	Número de pacotes recebidos.
OUT_PKTS	Básicos (*Basic*)	Número de pacotes enviados.
FLOW_DURATION_MS	Tempo (*Time*)	Duração do fluxo em milissegundos.

Fonte: Adaptado de (4).

Tabela 5 – Distribuição de fluxos no *dataset* NF-UNSW-NB15.

Classe	**Contagem**	**Descrição**
Benign	1.550.712	Fluxos normais não maliciosos.
Exploits	24.736	Sequências de comandos via vulnerabilidades.
Fuzzers	19.463	Dados aleatórios para causar falhas.
Reconnaissance	12.291	Coleta de informações (probes).
Generic	5570	Ataques contra criptografia.
DoS	5051	Tentativa de sobrecarga de recursos.
Outros	5295	Analysis, Backdoor, Shellcode e Worms.
Total	1.623.118	-

Fonte: Adaptado de (4).

3.2 Modelo de RNA Transformer

Este módulo é responsável pelo desenvolvimento de toda a infraestrutura do modelo de RNA utilizado no projeto. O mesmo é dividido em três etapas: (i) Parâmetros de entrada, (ii) Estrutura geral do modelo e (iii) Refinamento. O hardware utilizado foi o mesmo para todos os conjuntos de dados, como detalhado na Tabela 6 a seguir.

Tabela 6 – Especificações do *hardware* utilizado nos experimentos.

Componente	**Especificação Técnica**
Processador	Intel(R) Core(TM) i5-13420H (13ª Gen, 2,10 GHz)
Memória RAM	16,0 GB (3200 MT/s)
Placa de Vídeo	Intel(R) UHD Graphics (128 MB)
Armazenamento	SSD (477 GB)

Fonte: Autoria própria

3.2.1 Parâmetros de entrada

Para ambos os *datasets*, o objetivo foi preservar a integridade dos parâmetros para assegurar que os resultados refletissem o desempenho do modelo em condições operacionais padronizadas. A definição da arquitetura da rede neural FT-*Transformer* priorizou o equilíbrio entre a eficiência computacional e a capacidade de generalização, evitando o *overfitting* através de uma estrutura otimizada.

Conforme detalhado na Tabela 7, a configuração do modelo baseia-se em uma única camada de *encoder* com dimensão de *embedding* (d_{model}) de 32. Essa escolha estabelece uma infraestrutura enxuta, adequada para a complexidade dos dados de tráfego de rede sem incorrer em uma profundidade excessiva que poderia levar à dispersão de gradientes. Para capturar interdependências e relações não-lineares latentes, foram empregadas duas cabeças de atenção, permitindo que o modelo processe diferentes perspectivas do espaço de características simultaneamente. A dimensão da camada *feedforward* foi fixada em 64, respeitando a proporção de $2 \times d_{model}$. Esta configuração permite a projeção dos dados em

um espaço de maior dimensionalidade para a extração de padrões complexos, mantendo o controle sobre o número de parâmetros para evitar a memorização pura. Para a ativação das camadas internas, selecionou-se a função *Gaussian Error Linear Unit* (GELU). Diferente da ReLU, a GELU introduz uma suavidade não-linear ponderada pela distribuição normal, o que otimiza o fluxo de informações em arquiteturas baseadas em atenção. Por fim, aplicou-se uma taxa de *dropout* de 0,2 como mecanismo de regularização, forçando a rede a aprender representações mais resilientes ao ignorar aleatoriamente uma fração dos neurônios durante o treinamento, assegurando que o foco permaneça na estrutura intrínseca dos dados.

Tabela 7 – Parâmetros estruturais da arquitetura FT-*Transformer*.

Componente	**Valor**	**Descrição**
Embeddings (d_{model})	32	Dimensão do vetor de representação (EMB_DIM).
Cabeças de Atenção	2	Número de cabeças no multi-head attention.
Camadas Encoder	1	Número de blocos Transformer empilhados.
Feedforward Dim	64	Dimensão da camada densa interna ($2 \times$ EMB_DIM).
Função de Ativação	GELU	Ativação utilizada no encoder do Transformer.
Dropout	0,2	Taxa de descarte para regularização.

Fonte: Autoria própria.

As estratégias de otimização e treinamento, apresentadas na Tabela 8, foram delineadas para enfrentar o desequilíbrio severo de classes, um desafio recorrente em cenários de segurança cibernética. A implementação do otimizador AdamW com uma taxa de aprendizado de 0,0005 e um coeficiente de *weight decay* de 0,00001 visa uma convergência estável, enquanto o agendador de taxa de aprendizado ajusta o passo de descida conforme a evolução da perda de validação. Crucialmente, o mecanismo de interrupção prematura (*Early Stopping*) é guiado pelo Macro *F1-score* com uma paciência de seis épocas. O tamanho de *batch* foi definido em 512, visando otimizar a eficiência do processamento paralelo e a estabilidade da estimativa do gradiente durante as 50 épocas máximas de treinamento e para o ajuste dinâmico do passo de descida, implementou-se um *scheduler* do tipo Plateau, que reduz a taxa de aprendizado em 50 % após quatro épocas consecutivas sem melhora na perda de validação. Esta escolha métrica é fundamental, pois garante que o modelo seja otimizado para identificar eficazmente tanto o tráfego benigno quanto os ataques minoritários, impedindo que a alta acurácia global muitas vezes inflada pela classe majoritária mascare um desempenho insuficiente na detecção de ameaças raras.

Tabela 8 – Configurações e hiperparâmetros de treinamento.

Configuração	**Valor**	**Descrição**
Otimizador	AdamW	Algoritmo de otimização com decaimento de peso.
Taxa de Aprendizado	5e-4	Valor inicial (Learning Rate).
Weight Decay	1e-5	Penalidade L2 para os pesos da rede.
Tamanho do Lote	512	Quantidade de amostras por iteração (Batch Size).
Épocas Máximas	50	Limite de passagens pelo conjunto de dados.
Paciência	6	Limite de épocas sem melhora no F1-score.
Scheduler	Plateau	Redução da LR em 50% após 4 épocas de estagnação.

Fonte: Autoria própria.

O argumento central para a manutenção destes parâmetros invariáveis em todos os experimentos reside no rigor metodológico do isolamento de variáveis. Ao fixar a arquitetura da rede e as configurações de execução para ambos os conjuntos de dados, qualquer variação observada no desempenho (como precisão e *F1-score*) pode ser atribuída exclusivamente à forma como o tráfego foi coletado e estruturado. Esta abordagem permite comparar a eficácia dos dados brutos do *dataset* UNSW-NB15 com a representação baseada em fluxos do padrão *NetFlow*, permitindo uma conclusão científica fundamentada sobre a relevância da engenharia de características e da proveniência dos dados na eficácia de sistemas de detecção de intrusão baseados em modelos de atenção contemporâneos.

3.2.2 Estrutura geral do modelo

O modelo implementado neste estudo baseia-se na arquitetura FT-*Transformer* (*Feature Tokenizer + Transformer*), que representa uma adaptação contemporânea dos mecanismos de atenção para o domínio de dados tabulares heterogêneos. Conforme discorrido por Goodfellow et al. (5), a evolução das RNAs permitiu a extração de características complexas em grandes volumes de dados, e a estrutura geral deste modelo operacionaliza esse conceito ao tratar cada atributo do tráfego de rede como um *token* individual. O componente inicial, denominado *Feature Tokenizer*, projeta cada característica numérica ou categórica para um espaço de *embedding* de dimensão $d_{model} = 32$, permitindo que a rede identifique a relevância latente de cada campo do protocolo de forma isolada.

A sequência de *tokens* gerada é processada por um bloco *Encoder* do Transformer, baseado na proposta original de (2). Configurado com duas cabeças de atenção (*Multi-Head Self-Attention*), este mecanismo permite que o modelo capture correlações globais e dependências de longo alcance entre atributos, como a relação entre o volume de pacotes e a duração do fluxo. A saída desta etapa atravessa uma camada de *Feedforward Neural Network* (FFN) com dimensão de expansão igual a 64 e ativação GELU, garantindo a estabilidade do treinamento. Por fim, a representação agregada é enviada a uma camada linear que realiza a classificação entre tráfego benigno ou as categorias de ataques detalhadas

por (3).

3.2.3 Refinamento

O processo de refinamento do modelo foi delineado para enfrentar o desafio do desbalanceamento de classes, fator que, como apontado por (4), define os limites de desempenho de um classificador de rede. Para mitigar a tendência de favorecimento da classe majoritária (normal), integrou-se o *WeightedRandomSampler* no *pipeline* de treinamento. Este refinamento ajusta as probabilidades de amostragem durante o aprendizado, garantindo que ataques com rastro reduzido (*low footprint*) tenham representatividade estatística em cada ciclo, forçando o modelo a distinguir assinaturas comportamentais raras.

Complementarmente, o refinamento incluiu técnicas de otimização e regularização fundamentais para a generalização do modelo. Utilizou-se o otimizador AdamW com uma taxa de aprendizado de 0,0005 e decaimento de peso (*weight decay*) 0,00001, que proporciona maior estabilidade ao treinamento em arquiteturas Transformer (2). O ajuste fino é reforçado pelo agendador *ReduceLROnPlateau*, que reduz a taxa de aprendizado em um fator de 0,5 após quatro épocas de estagnação na perda de validação. Somado ao emprego de *Dropout* de 0,2, estas estratégias impedem a dependência mútua exagerada entre as unidades, assegurando que o refinamento resulte em uma detecção eficaz de ameaças sem comprometer a integridade do sistema por *overfitting*.

3.3 Treinamento do modelo

Para a realização do treinamento do modelo o mesmo fundamento foi utilizado para ambos os *datasets*. Uma metodologia uniforme foi utilizada em prol de garantir a mesma padronização para cada um dos conjuntos de informação. Assim, o fluxo pode ser definido em duas partes: validação cruzada estratificada (*Stratified K-fold*) e um teste independente para cada conjunto.

3.3.1 Validação cruzada estratificada (*Stratified K-fold*)

Dada a natureza desbalanceada dos dados de intrusão de rede, onde o tráfego normal prevalece sobre as classes maliciosas, utilizou-se a técnica de *Stratified K-Fold* com $k = 3$ dobras (*folds*). A escolha pela estratificação é crucial, pois garante que cada dobra de validação mantenha a proporção original das classes de ataque e tráfego benigno, mitigando vieses de amostragem e permitindo que o Macro *F1-score* obtido reflita a capacidade real de detecção para todas as categorias (4).

Durante o ciclo de treinamento de cada dobra, integrou-se o *WeightedRandomSampler* no carregamento das *batches*. Este componente ajusta a probabilidade de seleção das amostras, atribuindo pesos maiores às instâncias das classes minoritárias. Este refinamento metodológico força o FT-*Transformer* a dar a mesma importância estatística a ataques raros, como *Worms* ou *Analysis*, que possuem representatividade mínima no *dataset* original (3).

3.3.2 Critérios de parada

Antes da realização de testes independentes e síntese das informações para prosseguir com a análise dos dados é imprescindível abordar os critérios de parada do modelo, fatores esses que corroboram diretamente com os resultados a serem obtidos e analisados.

Embora o limite superior tenha sido estabelecido em 50 épocas, o controle efetivo do treinamento foi gerido pelo mecanismo de *Early Stopping*. Este monitora continuamente o Macro *F1-score* no conjunto de validação e interrompe a execução caso não ocorra evolução significativa após seis épocas. Esta abordagem, aliada ao uso de *Dropout* e normalização de camada, assegura que o estado final dos pesos represente o ponto ótimo de generalização, prevenindo a memorização excessiva (*overfitting*) das assinaturas específicas de tráfego presentes no conjunto de treinamento.

3.3.3 Teste independente

Após a validação cruzada, o passo seguinte foi o treinamento final utilizando a totalidade dos dados de treino. Nesta fase, reservou-se uma fração de 10 % do conjunto para o monitoramento interno do *Early Stopping*. A performance final do sistema foi então validada através da inferência sobre o conjunto de teste (*test hold-out*), composto por arquivos independentes fornecidos pelos autores dos *datasets*. As métricas extraídas desta etapa, consolidadas através de matrizes de confusão e relatórios de classificação, permitem a comparação direta entre a eficácia dos dados brutos (UNSW-NB15) e a representação baseada em fluxos do padrão *NetFlow* (NF-UNSW-NB15).

3.4 Análise dos valores obtidos

A análise de desempenho do FT-*Transformer* é fundamentada no conjunto de métricas geradas para cada um dos *datasets* em teste. Para assegurar a robustez estatística das conclusões, cada experimento proveu um conjunto individual de resultados para cada um dos três *folds* da validação cruzada, permitindo observar não apenas a eficácia média, mas também a variância do modelo diante de diferentes partições de dados.

Um ponto crítico nesta análise reside na distinção entre a aprendizagem de padrões reais e o fenômeno de *overfitting*. Conforme definido por (5), o *overfitting* ocorre quando o modelo atinge um erro extremamente baixo no conjunto de treino, mas falha em generalizar para dados inéditos, capturando ruídos e flutuações aleatórias como se fossem características intrínsecas da classe. Em sistemas de detecção de intrusão, este problema é agravado pela presença de classes minoritárias e tráfego altamente repetitivo, o que pode levar a rede a memorizar assinaturas específicas em vez de aprender o comportamento anômalo global.

Para mitigar este risco, a arquitetura implementada utiliza a técnica de *dropout*, configurada com uma taxa de 0,2. De acordo com (18), o *dropout* funciona através da desativação aleatória e temporária de neurônios (e das suas conexões) durante cada iteração do treino. Esta técnica impede que as unidades da rede desenvolvam uma co-dependência excessiva, forçando cada neurônio a aprender características mais robustas e úteis em diferentes contextos.

Enquanto o *overfitting* representa uma falha na capacidade de abstração do modelo, o *dropout* atua como uma forma de regularização que simula o treino de uma combinação de múltiplas redes neurais menores, resultando numa arquitetura final com maior poder de generalização. Assim, no momento em questão, a análise da curva de perda de validação face à perda de treino servirá para confirmar se a taxa de *dropout* escolhida foi eficaz em manter a convergência estável sem comprometer a integridade da detecção.

Além das métricas agregadas, os relatórios de classificação (*Classification Reports*) de cada dobra permitiram uma inspeção detalhada sobre o desempenho por classe de ataque. Tal abordagem evidencia se o refinamento via *WeightedRandomSampler* obteve sucesso constante em elevar a taxa de detecção de classes minoritárias em ambos os formatos de representação. Dessa forma, os valores obtidos fornecem o suporte empírico necessário para a discussão subsequente sobre a relevância da engenharia de características na segurança cibernética moderna.

3.5 Considerações Finais

O algoritmo utilizado para os testes foi feito na linguagem de programação *Python* com o auxílio principal das bibliotecas *Pytorch*, *Scikit-learn* e *Pandas*. O sistema operacional da máquina que executou o programa é o *Windows 11 Pro* 64 *bits*. No capítulo 4 os valores obtidos para cada um dos *datasets* são discutidos e comparados com gráficos e tabelas e no Apêndice A um *link* de um repositório é disponibilizado com todos os arquivos referentes ao projeto, desde o código fonte até planilhas utilizadas para analisar os valores obtidos.

4 Resultados e Discussões

A análise de desempenho do modelo FT-*Transformer* foi conduzida através da avaliação comparativa entre o *dataset* UNSW-NB15 original e sua versão sumarizada em fluxos, o NF-UNSW-NB15. Para garantir o rigor estatístico, os resultados foram coletados individualmente para cada uma das três dobras (*folds*) da validação cruzada, permitindo observar a estabilidade e a variância da arquitetura *Transformer* diante das diferentes representações de tráfego.

Os valores obtidos são demonstrados em tabelas e gráficos. Vale ressaltar que a comparação entre cada *dataset* é fundamentada não só a partir dos valores individuais de cada *fold*, mas também pelas médias gerais obtidas.

Inicialmente, para apontar o desempenho generalista de cada *dataset* denota-se a acurácia de cada *fold* e a média entre os valores obtidos.

Tabela 9 – Comparativo de acurácia global por *folds* entre os *datasets*.

Dataset	**Fold 1**	**Fold 2**	**Fold 3**	**Média**
UNSW-NB15 (Bruto)	62 %	65 %	65 %	64 %
NF-UNSW-NB15 (NetFlow)	90 %	94 %	78 %	87 %

Fonte: Autoria própria.

Os valores consolidados de acurácia, apresentados na Tabela 9, revelam uma discrepância notável entre as representações. O formato *NetFlow* proporcionou uma acurácia média de 87 %, superando significativamente os 64 % obtidos com os dados brutos. Este resultado sugere que a agregação de pacotes em fluxos estatísticos remove ruídos de baixa relevância para a classificação binária de tráfego (benigno vs. malicioso).

Tabela 10 – Comparativo de Macro *F1-score* por *folds* entre os *datasets*.

Dataset	**Fold 1**	**Fold 2**	**Fold 3**	**Média**
UNSW-NB15 (Bruto)	37 %	42 %	42 %	40 %
NF-UNSW-NB15 (NetFlow)	42 %	42 %	38 %	41 %

Fonte: Autoria própria.

Embora a acurácia global do *NetFlow* seja superior, a análise do Macro *F1-score* na Tabela 10 equaliza as conclusões. Esta métrica, que penaliza o modelo por falhas em classes minoritárias, manteve-se em patamares próximos (40 % vs 41 %). Este fato comprova que o desbalanceamento intrínseco das nove famílias de ataques permanece como o principal desafio para o sistema de detecção, independentemente da engenharia de características aplicada.

Utilizando os valores obtidos e analisando-os a partir de uma matriz de confusão descritas nas Figuras 7 e 8 é possível visualizar de maneira mais clara o desbalanceamento sugerido. Para o entendimento das representações vale ressaltar que os Verdadeiros Positivos indicam os acertos do modelo, os Falsos Negativos indicam as falhas de detecção e os Falsos Positivos indicam reconhecimentos errôneos, alarmes falsos. Com isso, tem-se o *Support* Médio que é a média de quantas amostras daquela classe específica estavam presentes, em média, por rodada de teste. Para a representação em específico, a soma de amostras de Verdadeiros Positivos e Falsos Negativos.

Figura 7 – *Heatmap* da matriz de confusão do *dataset* UNSW-NB15

UNSW-NB15				
Classe	***Support* Médio**	**Verdadeiros Positivos (TP)**	**Falsos Negativos (FN)**	**Falsos Positivos (FP)**
0	18667	14435	4232	145
1	582	549	33	10431
2	667	191	476	1052
3	6061	3536	2525	2828
4	378	365	13	3511
5	3497	1888	1609	647
6	11131	3377	7754	479
7	4088	68	402	332
8	43	39	4	1026
9	13333	13066	267	40

Fonte: Autoria própria

Figura 8 – *Heatmap* da matriz de confusão do *dataset* NF-UNSW-NB15

NF-UNSW-NB15				
Classe	***Support* Médio**	**Verdadeiros Positivos (TP)**	**Falsos Negativos (FN)**	**Falsos Positivos (FP)**
0	74591	668831	77079	0
1	11272	5260	6012	1889
2	1425	1377	12873	21056
3	1553	1134	419	17584
4	5691	3889	1802	3103
5	655	4631	1919	6555
6	1994	1449	545	22186
7	794	749	45	2219
8	409	403	6	896
9	53	44	9	186

Fonte: Autoria própria

Nota-se que as classes nas Figuras 7 e 8 indicam os 10 tipos de tráfego encontrados nos *datasets*, sendo eles divididos entre tráfego normal e cada ataque propriamente dito. As cores apresentadas nas representações foram definidas de forma quantitativa a partir de uma formatação condicional onde os tons de verde indicam os maiores valores e os tons de vermelho indicam os menores valores. O critério de endereçamento entre classe e tipo de tráfego foi feito a partir da função *.unique().tolist()* da biblioteca *Pandas* que extrai diferentes valores de uma estrutura de informação a partir da ordem de aparição de cada uma. Dessa forma, na Tabela 11 é possível relacionar as informações dos mapas de confusão com as particularidades de UNSW-NB15 e NF-UNSW-NB15.

Tabela 11 – Relação entre ID da Classe e Categoria para UNSW-NB15 e NF-UNSW-NB15.

ID da Classe	**Categoria UNSW-NB15**	**Categoria NF-UNSW-NB15**
0	Normal	Normal
1	*Backdoor*	*Fuzzers*
2	*Analysis*	*Exploits*
3	*Fuzzers*	*Backdoor*
4	*Shellcode*	*Reconnaissance*
5	*Reconnaissance*	*Generic*
6	*Exploits*	*DoS*
7	*DoS*	*Shellcode*
8	*Worms*	*Analysis*
9	*Generic*	*Worms*

Fonte: Autoria própria

Embora o volume nominal absoluto de registros do *dataset* UNSW-NB15 seja superior ao da versão *Netflow* NF-UNSW-NB15, a magnitude dos valores observados nas matrizes de confusão (Figuras 7 e 8) apresenta uma inversão proporcional. Esse fenômeno justifica-se por dois fatores, a distribuição percentual interna do NF-UNSW-NB15 é severamente concentrada na classe tráfego benigno (Normal), que detém a quase totalidade das amostras e também, ao comportamento adaptativo do classificador frente a esse cenário desbalanceado que resultou em um efeito cumulativo de falsas predições (Falsos Positivos e Falsos Negativos como na Classe 2 da Figura 8). Portanto, a concentração massiva de acertos na classe majoritária somada à dispersão de erros nas classes minoritárias eleva a contagem volumétrica visual da matriz na versão *NetFlow*, mesmo este partindo de um espaço amostral global inferior.

Diante dessas representações e análises, é notório que o desbalanceamento severo de classes, acentuado na versão *NetFlow* do *dataset* NF-UNSW-NB15, exerce uma força desproporcional na acurácia global do modelo. Como as classes majoritárias (Classe 0) detêm a maior parte do suporte, os acertos concentrados nelas inflam a acurácia geral. No entanto, a análise individual via matriz de confusão revela que o modelo falha em generalizar para classes minoritárias, apresentando altas taxas de Falsos Negativos e Falsos Positivos nelas. Assim, dado que os algoritmos de aprendizado tendem a otimizar a função de perda global minimizando o erro médio por amostra, o modelo prioriza o aprendizado das características das classes majoritárias. Como resultado, o acerto massivo na Classe 0 infla artificialmente a acurácia geral do sistema, mascarando uma degradação severa no desempenho de detecção das classes minoritárias.

4.1 Média de acurácia e impacto do desbalanceamento no Macro F1-Score

A discrepância acentuada entre as acurácias médias (64 % para dados brutos vs. 87 % para *NetFlow*) sugere que a sumarização estatística dos fluxos simplifica significativamente a fronteira de decisão primária da rede. No formato *NetFlow*, o modelo consegue atingir uma precisão de quase 100 % na classe benigna (Classe 0), indicando que as 12 características selecionadas por (4) são preditores extremamente fortes de tráfego normal.

Em contraste, o modelo treinado com os dados brutos do UNSW-NB15, apesar de possuir 49 características, apresenta *recall* inferior para o tráfego normal. Isso demonstra que o excesso de atributos brutos pode introduzir ruído e redundâncias que dificultam a convergência do mecanismo de auto-atenção, validando a tese de que um sistema de características orientada a fluxos é superior para a classificação volumétrica global.

A análise torna-se mais complexa ao observar o Macro *F1-score*. O fato de ambos os *datasets* apresentarem médias quase idênticas (aproximadamente 0,40) revela o impacto severo do desbalanceamento de classes. Conforme os relatórios detalhados nas dobras, a alta acurácia do *NetFlow* é impulsionada quase inteiramente pelo acerto na classe majoritária. Quando o modelo é desafiado a identificar classes minoritárias como *Worms* ou *Analysis*, a performance degrada significativamente em ambas as representações.

4.2 Estabilidade e fenômenos de convergência

As curvas de aprendizagem extraídas dos experimentos fornecem evidências visuais sobre a estabilidade do treinamento. Os gráficos expressam as diferenças observadas a cada dobra, onde é possível comparar a resposta de cada teste a partir das épocas (eixo horizontal dos gráficos) e da magnitude de cada métrica (eixo vertical dos gráficos). Mantendo um padrão de visualização, nos itens a), c) e e) das Figuras 9 e 10 é possível encontrar os termos perda de validação (*Val Loss*) e perda de treinamento (*Train Loss*) que relacionados indicam como o modelo é capaz de aprender a partir dos dados fornecidos. Nesse sentido, quanto menor e estável o *Train Loss* maior essa capacidade do modelo e quanto maior o *Val Loss* maior a dificuldade do modelo de aprender além do que foi fornecido, ou seja, surge a dificuldade de decidir com precisão a partir da informação não proveniente do conteúdo de teste. Vale ressaltar também que se há um aumento da curva de perda de validação enquanto há uma descida da curva de perda de treinamento é possível afirmar que existe *overffiting*.

Para os itens b), d) e f) das Figuras 9 e 10 encontra-se a validação de Macro *F1-Score*, que indica a média entre precisão e *recall* do modelo. No *dataset* bruto, a convergência é mais lenta e linear, refletindo uma aprendizagem estável porém limitada pela complexidade

do espaço de características. No NF-UNSW-NB15, a ascensão do Macro F1 nas primeiras 10 épocas é muito mais agressiva, porém acompanhada de uma volatilidade maior nas dobras finais. É a partir dessa métrica que pode-se extrair a capacidade do modelo de classificar o tipo de tráfego processado.

Figura 9 – Curvas de performance por *fold* UNSW-NB15

Fonte: Autoria própria

Figura 10 – Curvas de performance por *fold* NF-UNSW-NB15

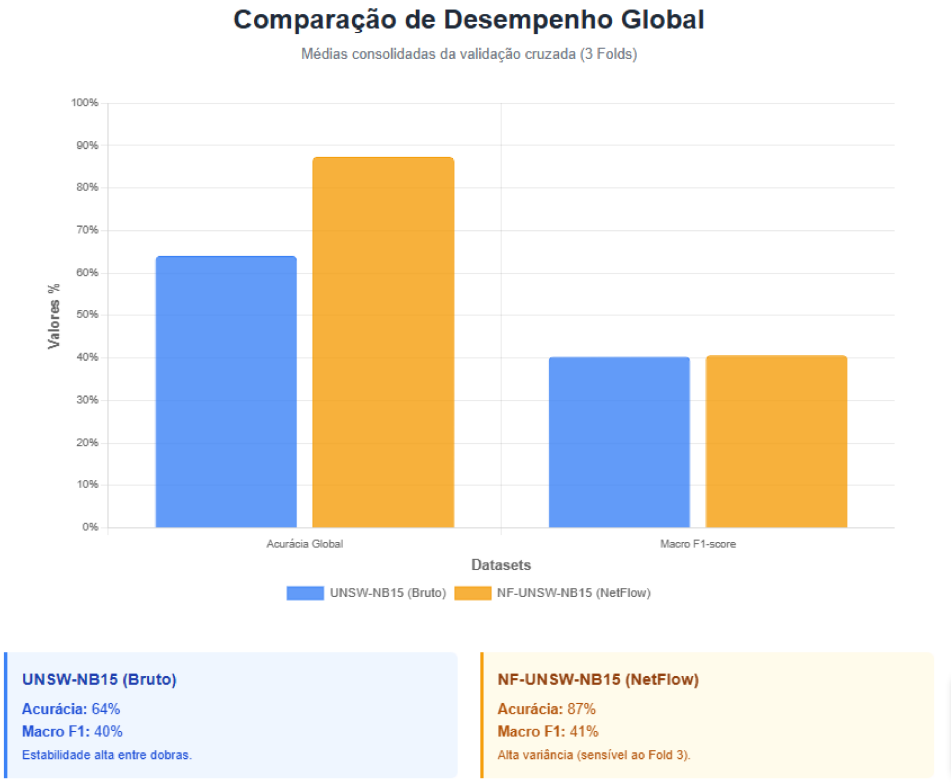
Fonte: Autoria própria

A análise visual dos valores revela uma volatilidade latente na representação *NetFlow* que não é imediatamente perceptível na tendência das curvas, mesmo que ao passar das dobras seja possível observar o aparecimento de dentes acentuados (*jitter*), vista no item f) da Figura 10. Ao retomar os valores de Macro *F1-Score* torna-se possível apontar a diferença de 16 % entre as dobras de validação que surgiram a partir da dificuldade do modelo de potencializar o acerto na classe majoritária, criando um isolamento quase perfeito da normalidade, porém mantendo a mesma barreira de detecção para intrusões de baixo rastro observada no tráfego bruto.

Dessa forma, um ponto crítico identificado foi a queda de desempenho no *Fold 3* do *NetFlow*, onde a acurácia recuou para 78 %. A inspeção dos *logs* revela que neste *fold* específico houve uma confusão maior entre fluxos de *Fuzzers* e tráfego normal. Isso sugere

que a representação compacta do *NetFlow* pode omitir detalhes contextuais importantes que, em certas partições de dados, tornam ataques volumétricos indistinguíveis de fluxos benignos para a rede neural.

Figura 11 – Comparação de acurácia e Macro *F1-score* entre *datasets*.



Fonte: Autoria própria

Em suma, como visto na Figura 11, o confronto visual entre a ascensão da acurácia e a estagnação do *F1-score* revela que a superioridade do formato *NetFlow* (0,87) em relação ao bruto (0,64) é impulsionada predominantemente pela facilidade de isolamento do tráfego benigno. No entanto, a paridade observada no Macro *F1-score* (aproximadamente 0,41) confirma que a sumarização estatística não é suficiente para mitigar o desbalanceamento severo de ataques raros. A discussão aponta que, embora o modelo *Transformer* seja altamente eficaz na extração de padrões globais, a detecção de intrusões específicas permanece dependente da densidade das classes, sugerindo que a eficácia real do IDS reside na combinação entre a representação de fluxo e técnicas avançadas de reamostragem.

Tabela 12 – Resultados de Macro e *Weighted F1-Score* para o conjunto UNSW-NB15 (Bruto).

Métrica	**Fold 1**	**Fold 2**	**Fold 3**	**Média**
Macro *F1-score*	37 %	42 %	42 %	40,3 %
Weighted F1-score	68 %	70 %	69 %	69 %

Fonte: Autoria própria.

Tabela 13 – Resultados de Macro e *Weighted F1-Score* para o conjunto NF-UNSW-NB15 (*NetFlow*).

Métrica	**Fold 1**	**Fold 2**	**Fold 3**	**Média**
Macro *F1-score*	42 %	42 %	38 %	40,6 %
Weighted F1-score	93 %	96 %	86 %	91,6 %

Fonte: Autoria própria.

O fator de generalização do modelo a partir das limitações impostas sejam de *hardware* ou de configuração do sistema podem ser mais visíveis a partir de uma breve comparação entre Macro *F1-Score* e *Weighted F1-Score* (média ponderada), métrica essa que analisa valores a partir do número de amostras por divisão de classe, ou seja, para *datasets* em que o número de tráfego benigno é significativamente maior os valores serão ponderados para um limiar ideal (100 %) que destoa do objetivo de um sistema IDS, como descrito nas Tabelas 12 e 13.

5 Conclusão

Neste trabalho um modelo de rede neural não convolucional foi desenvolvido em prol de comparar dois *datasets* que registram fluxos de rede, diferenciados entre fluxos que sofreram ataques de intrusão e que não sofreram ataques de intrusão. A mesma RNA foi treinada com as duas bases de dados mantendo os mesmos parâmetros de configuração a fim de elucidar as diferenças encontradas a partir das particularidades de construção e organização de cada *dataset*.

A partir das simulações computacionais, as métricas de acurácia, taxa de falsos positivos e capacidade preditiva evidenciaram o impacto crítico do desbalanceamento severo de classes na estabilização dos limites de decisão do modelo. Diante desse cenário de assimetria de dados, a adoção da versão *NetFlow* demonstrou ser um fator determinante: a padronização e a agregação temporal do tráfego em fluxos comportamentais reduzem significativamente a redundância estatística e o ruído de pacotes brutos. Como consequência direta, o modelo de RNA obteve melhores margens de separabilidade linear e não linear, blindando o classificador contra o *overfitting* na classe majoritária (tráfego benigno). Assim, este trabalho constata de maneira incisiva que o *design* de um IDS contemporâneo depende tanto da sensibilidade do algoritmo às oscilações volumétricas quanto da qualidade da representação baseada em fluxos para mitigar os vieses de distribuição de dados.

Para trabalhos futuros é possível refinar os parâmetros iniciais do modelo a fim de alcançar um melhor resultado perante à falsos positivos e falsos negativos assim como comparar um número maior de *datasets* que tenham configurações mais destoantes entre si. Além disso, nota-se também que existem lacunas no *software* onde os recursos de *deep learning* foram limitados a partir das restrições de configuração do computador utilizado para os testes. Por fim, também seria interessante o incremento de recursos de captação de ataques novos para que a rede seja capaz de interagir com fluxos corrompidos mesmo que o tipo de ataque não esteja previamente nos bancos de dados.

Referências Bibliográficas

1 RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning representations by back-propagating errors. *Nature*, Nature Publishing Group, v. 323, n. 6088, p. 533–536, 1986. DOI: https://doi.org/10.1038/323533a0.

2 VASWANI, A. et al. Attention is all you need. In: *Advances in Neural Information Processing Systems*. [s.n.], 2017. p. 5998–6008. DOI: https://doi.org/10.48550/arXiv.1706.03762.

3 MOUSTAFA, N.; SLAY, J. Unsw-nb15: A comprehensive data set for network intrusion detection systems. In: *2015 Military Communications and Information Systems Conference (MilCIS)*. [S.l.: s.n.], 2015. p. 1–6. DOI: https://doi.org/10.1109/MilCIS.2015.7348942.

4 SARHAN, M. et al. Netflow datasets for machine learning-based network intrusion detection systems. *IEEE Access*, v. 9, p. 11791–11804, 2021. DOI: https://doi.org/10.1109/ACCESS.2021.3050914.

5 GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. MIT Press, 2016. DOI: https://doi.org/10.7551/mitpress/10206.001.0001.

6 CARVALHO, A. C. P. L. F. de. Redes neurais artificiais. *ICMC-USP*, 2017. Disponível em: <https://sites.icmc.usp.br/andre/research/neural/>.

7 HAYKIN, S. S. *Neural Networks and Learning Machines*. Third. Upper Saddle River, NJ: Pearson Education, Inc., 2009. ISBN 978-0131471399.

8 LOSHCHILOV, I.; HUTTER, F. Decoupled weight decay regularization. In: *International Conference on Learning Representations (ICLR)*. [s.n.], 2019. DOI: https://doi.org/10.48550/arXiv.1711.05101.

9 CAVALCANTE, R. G. S. A transformer-based architecture neural network approach to email message autocomplete. *Repositório Institucional da UFAL*, 2024. Disponível em: <https://www.repositorio.ufal.br/bitstream/123456789/13566/1/A%20transformer-based%20architecture%20neural%20network%20approach%20to%20email%20message%20autocomplete.pdf>.

10 GORISHNIY, Y. et al. Revisiting deep learning models for tabular data. In: RANZATO, M. et al. (Ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2021. v. 34, p. 18932–18943. DOI: https://doi.org/10.48550/arXiv.2106.11959.

11 ARIK, S. Ö.; PFISTER, T. Tabnet: Attentive interpretable tabular learning. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. [S.l.: s.n.], 2021. v. 35, n. 8, p. 6679–6687. DOI: https://doi.org/10.1609/aaai.v35i8.16826.

12 FAWCETT, T. An introduction to roc analysis. *Pattern Recognition Letters*, Elsevier, v. 27, n. 8, p. 861–874, 2006. DOI: https://doi.org/10.1016/j.patrec.2005.10.010.

13 POWERS, D. M. W. Evaluation: From precision, recall and f-measure to roc, informedness, markedness correlation. *Journal of Machine Learning Technologies*, BioInfo Publications, v. 2, n. 1, p. 37–63, 2011. ISSN 2229-3981. DOI: https://doi.org/10.48550/arXiv.2010.16061.

14 GEEKSFORGEEKS. What is a dataset? *GeeksforGeeks*, 2023. Disponível em: <https://www.geeksforgeeks.org/data-science/what-is-dataset/>.

15 MCHUGH, J. Testing intrusion detection systems: A critique of the 1998 darpa/lincoln laboratory old intrusion detection system evaluation as performed by lincoln laboratory. *ACM Transactions on Information and System Security (TISSEC)*, ACM New York, NY, USA, v. 3, n. 4, p. 262–294, 2000. DOI: https://doi.org/10.1145/382912.382923.

16 GONçALVES, H. E. *Comparação da eficiência de modelos de Redes Neurais Artificiais na detecção de intrusões em redes de computadores*. Dissertação (Trabalho de Conclusão de Curso (Graduação em Engenharia Eletrônica e de Telecomunicações)) — Universidade Federal de Uberlândia, Uberlândia, MG, 2023. Orientador: Prof. Dr. Éderson Rosa da Silva. Disponível em: <https://repositorio.ufu.br/handle/123456789/37246>.

17 MOUSTAFA, N.; TURNBULL, B.; CHOO, K.-K. R. An ensemble intrusion detection technique based on proposed statistical flow features for protecting network traffic of internet of things. *IEEE Internet of Things Journal*, PP, p. 1–1, 09 2018. DOI: https://doi.org/10.1109/JIOT.2018.2869578.

18 SRIVASTAVA, N. et al. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, v. 15, n. 1, p. 1929–1958, 2014. Disponível em: <http://jmlr.org>. DOI: https://doi.org/10.48550/arXiv.1207.0580.

APÊNDICE A – Arquivos do projeto

Aqui encontra-se uma pasta com o arquivo Python utilizado para o desenvolvimento da RNA utilizada, assim como os bancos de dados utilizados e os valores obtidos nos testes: <https://drive.google.com/drive/folders/1nSA1bfJzCOr7gKzaCoWUGTGvqFsfibqF?usp=drive_link>