UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA

Heitor Eugênio Gonçalves

Redes Neurais Artificiais combinadas a Algoritmos Genéticos para detecção de intrusões em rede de computadores

Uberlândia

2025

Heitor Eugênio Gonçalves

Redes Neurais Artificiais combinadas a Algoritmos Genéticos para detecção de intrusões em rede de computadores

Dissertação de mestrado apresentada ao Programa de Pós-graduação em Engenharia Elétrica da Universidade Federal de Uberlândia como requisito para a obtenção do título de Mestre em Ciências.

Área de concentração: Processamento da Informação.

Orientador: Prof. Dr. Éderson Rosa da Silva

Uberlândia-MG
2025

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da UFU, MG, Brasil.

G635h
2025

Gonçalves, Heitor Eugênio, 1999-
Redes Neurais Artificiais combinadas a Algoritmos Genéticos para detecção de intrusões em rede de computadores [recurso eletrônico] / Heitor Eugênio Gonçalves. - 2025.

Orientador: Éderson Rosa da Silva.
Dissertação (Mestrado) - Universidade Federal de Uberlândia, Programa de Pós-graduação em Engenharia Elétrica.
Modo de acesso: Internet.
Disponível em: http://doi.org/10.14393/ufu.di.2025.5236
Inclui bibliografia.
Inclui ilustrações.

1. Engenharia Elétrica. I. Silva, Éderson Rosa da, 1984-, (Orient.). II. Universidade Federal de Uberlândia. Programa de Pós-graduação em Engenharia Elétrica. III. Título.

CDU: 621.3

André Carlos Francisco
Bibliotecário-Documentalista - CRB-6/3408

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
Coordenação do Programa de Pós-Graduação em Engenharia Elétrica
Av. João Naves de Ávila, 2121, Bloco 3N - Bairro Santa Mônica, Uberlândia-MG, CEP 38400-902
Telefone: (34) 3239-4707 - www.posgrad.feelt.ufu.br - copel@ufu.br

ATA DE DEFESA - PÓS-GRADUAÇÃO

Programa de Pós-Graduação em:	Engenharia Elétrica				
Defesa de:	Dissertação de Mestrado nº 810 - PPGEELT				
Data:	Trinta de Setembro de Dois mil e vinte e cinco	Hora de início:	9h00	Hora de encerramento:	11h00
Matrícula do Discente:	12312EEL010				
Nome do Discente:	Heitor Eugênio Gonçalves				
Título do Trabalho:	Redes Neurais Artificiais combinadas a Algoritmos Genéticos para detecção de intrusões em rede de computadores				
Área de concentração:	Processamento da Informação				
Linha de pesquisa:	Processamento Digital de Sinais e Redes de Comunicação				
Projeto de Pesquisa de vinculação:	Coordenador do Projeto: Éderson Rosa da Silva. Título Do Projeto: Desenvolvimento e simulação de técnicas de alocação de recursos em redes de comunicação. Vigência Do Projeto: 2018 - atual.				

Reuniu-se através de videoconferência, a Banca Examinadora, designada pelo Colegiado do Programa de Pós-graduação em Engenharia Elétrica, assim composta:

Doutores: André Luiz Aguiar da Costa (UFU), Myke Douglas de Medeiros Valadão (SiDi) e Éderson Rosa da Silva, orientador do discente.

Iniciando os trabalhos, o presidente da mesa, Prof. Dr. Éderson Rosa da Silva apresentou a Comissão Examinadora e o candidato, agradeceu a presença do público, e concedeu ao discente a palavra para a exposição do seu trabalho. A duração da apresentação do discente e o tempo de arguição e resposta foram conforme as normas do Programa.

A seguir, o senhor presidente concedeu a palavra, pela ordem sucessivamente, aos examinadores, que passaram a arguir o candidato. Ultimada a arguição, que se desenvolveu dentro dos termos regimentais, a Banca, em sessão secreta, atribuiu o resultado final, considerando o candidato:

APROVADO.

Esta defesa faz parte dos requisitos necessários à obtenção do título de Mestre. O competente diploma será expedido após cumprimento dos demais requisitos, conforme as normas do Programa, a legislação pertinente e a regulamentação interna da UFU.

Nada mais havendo a tratar foram encerrados os trabalhos. Foi lavrada a presente ata que após lida e achada conforme, foi assinada pela Banca Examinadora.

Documento assinado eletronicamente por **Myke Douglas de Medeiros Valadão**, **Usuário Externo**, em 30/09/2025, às 10:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Ederson Rosa da Silva**, **Professor(a) do Magistério Superior**, em 30/09/2025, às 10:48, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

Documento assinado eletronicamente por **Andre Luiz Aguiar da Costa**, **Professor(a) do Magistério Superior**, em 30/09/2025, às 10:49, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site https://www.sei.ufu.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **6721160** e o código CRC **707045FB**.

Referência: Processo nº 23117.068078/2025-86 SEI nº 6721160

Dedicado à minha família, em especial à minha mãe, aos meus amigos e colegas de mestrado, ao meu orientador e professor Éderson e a todos os demais profissionais da Universidade Federal de Uberlândia.

Agradecimentos

Agradeço ao meu professor e orientador Éderson por todos os seus ensinamentos e pelo tempo dedicado apoiando-me nesta pesquisa.

Agradeço à minha mãe, que sempre me incentivou a estudar.

Sou grato ao corpo docente da Universidade Federal de Uberlândia, por todos os aprendizados que recebi durante os meus anos de mestrado, e aos demais funcionários da universidade por todos os serviços prestados. Também agradeço a Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) - Código de Financiamento 001 - pelo apoio prestado para a realização deste trabalho.

Resumo

Gonçalves, H. E. *Redes Neurais Artificiais combinadas a Algoritmos Genéticos para detecção de intrusões em rede de computadores*. Universidade Federal de Uberlândia, Uberlândia, Brasil, 2025.

Este trabalho apresenta uma metodologia para detecção de intrusões em redes de computadores fundamentada na integração de Algoritmos Genéticos (AGs) e Redes Neurais Convolucionais (CNNs). A proposta contempla, de forma simultânea, a seleção automática de atributos que melhor caracterizam o tráfego da rede e o ajuste de hiperparâmetros da CNN, com o objetivo de maximizar métricas de desempenho e minimizar a latência de inferência. Utilizando o conjunto de dados CIC-IDS2018, os experimentos evidenciaram que o AG de seleção de atributos foi capaz de identificar um subconjunto de 20 variáveis que preserva a capacidade discriminativa do modelo e, o AG de ajuste de hiperparâmetros, por sua vez, produziu uma configuração de CNN com desempenho ligeiramente superior, atingindo acurácia de 99,78%, *F1-score* de 99,66% e *Recall* de 99,82%, com latência média de apenas 0,0529 ms por amostra. A análise qualitativa demonstrou que os atributos selecionados apresentam relevância direta na caracterização de padrões de tráfego, enquanto os hiperparâmetros otimizados resultaram em melhor separabilidade entre fluxos benignos e maliciosos. As matrizes de confusão indicaram baixa incidência de falsos positivos e falsos negativos, reforçando a robustez da abordagem. Os resultados obtidos superaram os de trabalhos relacionados na literatura, validando a eficácia da proposta.

Palavras chaves: Algoritmos Genéticos, Detecção de Intrusões, Otimização de Hiperparâmetros, Redes Neurais Convolucionais, Seleção de Atributos.

Abstract

Gonçalves, H. E. *Redes Neurais Artificiais combinadas a Algoritmos Genéticos para detecção de intrusões em rede de computadores*. Universidade Federal de Uberlândia, Uberlândia, Brasil, 2025.

This work presents a methodology for intrusion detection in computer networks based on the integration of Genetic Algorithms (GAs) and Convolutional Neural Networks (CNNs). The proposal simultaneously contemplates the automatic selection of attributes that best characterize network traffic and the adjustment of CNN hyperparameters, with the aim of maximizing performance metrics and minimizing inference latency. Using the CIC-IDS2018 dataset, the experiments showed that the attribute selection GA was able to identify a subset of 20 variables that preserves the model's discriminative capacity, and that the hyperparameter adjustment GA, in turn, produced a CNN configuration with slightly superior performance, achieving an accuracy of 99.78%, an F1-score of 99.66%, and a recall of 99.82%, with an average latency of only 0.0529 ms per sample. Qualitative analysis demonstrated that the selected attributes are directly relevant in characterizing traffic patterns, while the optimized hyperparameters resulted in better separability between benign and malicious flows. Confusion matrices indicated a low incidence of false positives and false negatives, reinforcing the robustness of the approach. The results obtained surpassed those of related studies in the literature, validating the effectiveness of the proposal.

Keywords: Convolutional Neural Networks, Feature Selection, Genetic Algorithms, Hyperparameter Optimization, Intrusion Detection.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Siglas

Adam Adaptive Moment Estimation
Adamax Variant of Adam Optimizer
AG Algoritmo Genético
BI-LSTM Bidirectional Long Short-Term Memory
CART Classification and Regression Tree
CIC-IDS2018 Canadian Institute for Cybersecurity - Intrusion Detection System 2018
CNN Rede Neural Convolucional
CPU Central Processing Unit
DDoS Distributed Denial of Service
DNN Deep Neural Network
DoS Denial of Service
ELU Exponential Linear Unit
FN Falsos Negativos
FP Falsos Positivos
F1 F1-Score (média harmônica entre precisão e recall)
G-Mean Geometric Mean
GMM Gaussian Mixture Model
GPU Graphics Processing Unit
IDE Integrated Development Environment
IDS Intrusion Detection System
LR Logistic Regression
LSTM Long Short-Term Memory
MLP Multilayer Perceptron
ReLU Rectified Linear Unit
RNA Rede Neural Artificial
RNR Rede Neural Recorrente
SELU Scaled Exponential Linear Unit
SGD Stochastic Gradient Descent

Capítulo 1

Introdução

1.1 Contexto e Motivação

A segurança cibernética emergiu como uma das áreas mais críticas para o desenvolvimento de tecnologias modernas. A Internet, que há algumas décadas era uma rede simples de comunicação, transformou-se em uma rede global que conecta bilhões de dispositivos, sistemas e usuários ao redor do mundo. Esse fenômeno de globalização digital, impulsionado pela *Internet das Coisas* (IoT), pela computação em nuvem, e pelas novas redes móveis 5G, criou um ambiente vasto e complexo para ciberataques. Cada dispositivo, rede e sistema interconectado apresenta uma possível vulnerabilidade que pode ser explorada por agentes maliciosos.

A *Internet das Coisas* representa uma das mais significativas transformações tecnológicas do século XXI, com bilhões de dispositivos inteligentes conectados à Internet. Estes dispositivos, que vão desde termostatos e câmeras de segurança até equipamentos médicos e automóveis, aumentam exponencialmente a superfície de ataque das redes. De acordo com a *European Union Agency for Cybersecurity* (ENISA), o número de dispositivos IoT pode superar 25 bilhões até 2030, o que amplia consideravelmente os riscos de ataques cibernéticos [1].

Adicionalmente, a computação em nuvem, uma tecnologia que permite o armazenamento e processamento de grandes volumes de dados em servidores remotos, também aumentou a complexidade das redes. Embora traga benefícios como escalabilidade e flexibilidade, a computação em nuvem também apresenta desafios significativos em termos de proteção de dados e da infraestrutura de segurança. De acordo com o *Cisco Annual Internet Report*, em 2023, mais de 94% das cargas de trabalho estavam migrando para a nuvem, o que torna os provedores de serviços em nuvem alvos frequentes de ciberataques [2].

Além disso, o advento das redes móveis 5G aumentou a velocidade de comunicação e a conectividade, o que impulsionou ainda mais a digitalização de serviços e sistemas. No entanto, isso também ampliou o campo de atuação para os atacantes, criando novos vetores de ataque, como a manipulação de dispositivos de borda, que se conectam diretamente às redes móveis e à infraestrutura crítica [3].

Com a ampliação da superfície de ataque e o crescimento do tráfego global de dados, surge a necessidade urgente de sistemas de defesa cibernética mais sofisticados e rápidos. A detecção de intrusões, como parte essencial das arquiteturas de segurança, deve evoluir para enfrentar esses novos desafios. O aumento exponencial do tráfego de dados, por exemplo, torna a tarefa de monitorar e proteger redes complexas ainda mais difícil. Sistemas de Detecção de Intrusões (IDS) precisam ser capazes de identificar ameaças de forma rápida, precisa e, mais importante, em tempo real.

1.2 Histórico dos Ataques Cibernéticos

A evolução dos ataques cibernéticos reflete uma escalada de sofisticação e impacto. Nos anos 1980, os primeiros vírus e *worms* começaram a surgir, destacando-se o *Morris Worm* em 1988, que afetou cerca de 10% dos sistemas conectados à Internet na época [4]. Embora simples em comparação com os ataques atuais, este evento foi o precursor de uma série de incidentes de segurança que se seguiram.

Na década de 2000, a proliferação de *worms*, como o *Slammer Worm*, exemplificou o poder destrutivo de ataques que podiam se espalhar rapidamente, infectando milhares de sistemas em questão de minutos. O *Slammer*, que afetou bancos de dados do Microsoft SQL Server, causou paralisia em grandes áreas da internet e deixou claro que a segurança deveria ser repensada para lidar com ataques em larga escala [5].

A partir de 2010, a ameaça cibernética se diversificou com o uso de *botnets* massivas, como a *Mirai Botnet*. Essa *botnet* foi responsável por alguns dos maiores ataques *Distributed Denial of Service* (DDoS) da história, explorando dispositivos IoT mal protegidos, como câmeras de segurança e roteadores. Esses ataques atingiram alvos estratégicos, como provedores de Internet, afetando milhões de usuários em todo o mundo [1]. A sofisticação das técnicas de ataques de DDoS tem sido acompanhada pelo aumento da largura de banda dos ataques, ultrapassando os 1 Tbps, e causando interrupções que podem durar horas ou dias.

Além disso, ataques direcionados, como o *Stuxnet* em 2010, marcaram um ponto de inflexão ao demonstrar como *malwares* podem ser usados para atacar sistemas industriais críticos. O Stuxnet foi projetado para sabotar o programa nuclear iraniano, atingindo diretamente os sistemas SCADA, que controlam equipamentos industriais. Este ataque destacou a necessidade de proteção robusta em sistemas críticos e de uma abordagem mais holística para a segurança, que inclua a defesa cibernética em nível de infraestrutura física [6].

Mais recentemente, ataques como os de *ransomware*, que visam criptografar os dados e exigir resgates em criptomoedas, tornaram-se uma grande ameaça para empresas e governos. Exemplos incluem o ataque ao sistema de saúde do Reino Unido, que paralisou hospitais em 2017, e o ataque global do WannaCry, que afetou mais de 150 países. Tais incidentes aumentaram a pressão sobre as organizações para que implementem sistemas de defesa mais robustos, incluindo proteção contra *malware* avançado e ataques de engenharia social [7].

1.2.1 Impacto dos Ataques DDoS

Os ataques DDoS continuam sendo uma das formas mais prevalentes de ataque cibernético. Estes ataques têm o potencial de paralisar serviços essenciais, como sistemas bancários, governamentais e empresas de tecnologia, sobrecarregando os servidores com um tráfego massivo. Em 2020, a *NETSCOUT* reportou um aumento de mais de 30% no número de ataques DDoS em relação ao ano anterior, com um crescimento alarmante no número de ataques superiores a 1 Tbps [8].

Os ataques DDoS não se limitam apenas a bloquear serviços, mas também a minar a confiança dos consumidores em empresas e plataformas digitais. A perda de acessibilidade e a interrupção dos serviços têm consequências financeiras graves, além de danos à reputação. Empresas que não possuem um sistema de defesa robusto contra esses ataques enfrentam perdas financeiras diretas, além de prejuízos indiretos com a perda de clientes e a exposição a riscos de segurança. Estes ataques são frequentemente utilizados como cortinas de fumaça, criando distrações enquanto outras formas de intrusão ocorrem em paralelo, como a exfiltração de dados [9].

1.3 Evolução dos Sistemas de Detecção de Intrusão

Os Sistemas de Detecção de Intrusões (IDS) têm evoluído em resposta ao aumento da complexidade dos ataques. Inicialmente baseados em simples assinaturas, os primeiros IDS eram eficazes para identificar ataques conhecidos, mas falhavam em detectar novas ameaças. Com o tempo, novas abordagens foram desenvolvidas, como a detecção por anomalia, que visa identificar comportamentos incomuns dentro da rede. No entanto, a detecção de intrusões avançadas exigiu a adoção de técnicas mais sofisticadas, como aprendizado de máquina e redes neurais.

Os IDS podem ser classificados de acordo com a forma como identificam comportamentos maliciosos. A seguir, são descritos os principais tipos:

- **Baseados em Assinaturas:** funcionam de forma semelhante a antivírus tradicionais, procurando por padrões específicos e conhecidos de ataques. Esses sistemas são rápidos e precisos na identificação de ameaças previamente documentadas, mas apresentam limitações na detecção de ataques novos ou modificados. A constante evolução do *malware*, com o uso de técnicas de evasão, compromete a eficácia dos IDS baseados exclusivamente em assinaturas, tornando-os insuficientes em um cenário de segurança cibernética dinâmico.

- **Baseados em Anomalias:** surgiram como resposta à crescente complexidade dos ataques. Esses sistemas aprendem o comportamento "normal" de uma rede e identificam desvios desse padrão. Embora eficazes na detecção de ataques inéditos, enfrentam uma alta taxa de falsos positivos, pois o comportamento de rede pode variar naturalmente. Isso

dificulta a distinção entre variações legítimas e comportamentos maliciosos, exigindo sistemas mais sofisticados para mitigar esse problema.

- **Híbridos:** combinam os dois métodos anteriores, utilizando assinaturas para detectar ataques conhecidos e técnicas de detecção por anomalia para reconhecer novas ameaças. Além disso, esses sistemas podem incorporar algoritmos de aprendizado profundo, como redes neurais convolucionais (CNNs), capazes de identificar padrões mais complexos no tráfego da rede.

1.4 Tecnologias Emergentes

Nos últimos anos, o campo da detecção de intrusões em redes de computadores tem sido cada vez mais impulsionado por uma tecnologia central: a *inteligência artificial* (IA) aplicada à cibersegurança — em especial aos IDS. Estudos recentes mostram que, diante da complexidade crescente do tráfego de rede, da diversidade de vetores de ataque e da necessidade de adaptação a ameaças desconhecidas, a IA representa uma tendência tecnológica emergente para suportar a defesa automática e adaptativa de sistemas críticos [10–12].

Dentro desse panorama, diversas técnicas - como redes neurais profundas, aprendizado por reforço, algoritmos evolutivos, otimização de hiperparâmetros - têm sido exploradas. É importante diferenciar: a tecnologia refere-se ao uso amplo da IA para IDS; as técnicas são os métodos específicos empregados dentro dessa tecnologia, por exemplo, redes convolucionais, algoritmos genéticos, aprendizagem por reforço.

Em particular, a mais recente literatura aborda a aplicação de aprendizado profundo (*deep learning*) para IDS em ambientes emergentes como Internet das Coisas (IoT), *Edge Computing* e *Software-Defined Networking* (SDN) — o que reforça o caráter de "tecnologia emergente" desse tipo de proteção [10, 11].

Ao final desta subseção, vale ressaltar que a abordagem adotada neste trabalho, que combina redes neurais com AG para otimização de parâmetros e seleção de características, está alinhada com essa tendência de tecnologia emergente de IA para detecção de intrusões.

1.5 Solução Proposta

Como parte da solução proposta, constata-se que os AGs oferecem uma reconhecida capacidade de explorar espaços de busca de alta dimensionalidade de forma eficiente, evitando a dependência exclusiva de escolhas manuais ou heurísticas pouco generalizáveis [13, 14].

No contexto da detecção de intrusões, essa característica se torna relevante porque os fluxos de rede e os conjuntos de atributos utilizados para modelagem frequentemente apresentam elevada dimensionalidade, heterogeneidade e evolução contínua [13].

Por sua vez, as técnicas de aprendizado profundo, em especial CNNs, têm demonstrado uma forte aptidão para o reconhecimento de padrões complexos em dados de rede, inclusive padrões de tráfego anômalo ou desconhecido. Essa capacidade amplia o escopo da tecnologia emergente de IA, permitindo a detecção proativa de ataques novos ou não previamente assinados [10, 11].

Deste modo, a solução proposta — que integra AG para otimização e seleção de características, aliado a uma arquitetura de rede neural para classificação de tráfego — manifesta-se como uma instância alinhada à tendência tecnológica emergente em IDS: trata-se de um sistema IA-centrado, adaptativo e voltado para ambientes de rede complexos.

Na fase de implementação, os AGs atuam na busca automática das melhores combinações de parâmetros e subconjuntos de atributos, reduzindo a dimensionalidade e melhorando a eficiência do treinamento. Em seguida, a CNN realiza a classificação do tráfego de rede em benigno ou malicioso. A junção desta abordagem permite que a solução proponha avanço em termos de generalização, adaptabilidade e redução de intervenção manual.

1.6 Objetivos

O objetivo geral desta pesquisa consiste em desenvolver, implementar e validar uma abordagem híbrida baseada na integração de AGs e CNNs para a detecção de intrusões em redes de computadores. Pretende-se, com isso, alcançar simultaneamente dois propósitos fundamentais: maximizar a eficácia da detecção, por meio de métricas de desempenho consolidadas, e minimizar a latência de processamento, assegurando que o modelo seja compatível com cenários de operação em tempo quase real.

Para atingir esse objetivo amplo, foram definidos os seguintes objetivos específicos:

- Investigar a eficiência do uso de AGs para a seleção automática de atributos relevantes do conjunto CIC-IDS2018, de forma a identificar subconjuntos de variáveis que melhor representem os padrões de tráfego normal e malicioso;
- Aplicar AGs na otimização dos hiperparâmetros da CNN, buscando definir automaticamente funções de ativação, número de unidades em cada camada, otimizadores e funções de custo mais adequados ao problema;
- Projetar e treinar uma CNN enxuta, equilibrando simplicidade estrutural com alto desempenho preditivo, de forma a garantir tanto eficiência computacional quanto escalabilidade;
- Avaliar o modelo proposto utilizando métricas amplamente reconhecidas na literatura, como acurácia, *F1-score*, *G-Mean* e tempo médio de inferência, de modo a quantificar de forma abrangente sua eficácia e eficiência;
- Comparar os resultados obtidos com trabalhos relacionados, assegurando uma análise crítica que situe a contribuição deste trabalho no contexto do estado da arte em detecção de intrusões.

1.7 Contribuições

As contribuições deste trabalho podem ser analisadas sob diferentes perspectivas: metodológica, experimental e aplicada. No plano metodológico, destaca-se a proposição de uma arquitetura híbrida de AG e CNN que utiliza dois AGs distintos e complementares, um voltado à seleção de atributos e outro dedicado à definição de hiperparâmetros. Essa estratégia, embora ainda pouco explorada na literatura, demonstrou ser eficaz na otimização conjunta de insumos e parâmetros de modelos de aprendizado profundo.

No âmbito experimental, a pesquisa demonstrou que a seleção automática de atributos permitiu reduzir significativamente a dimensionalidade de entrada sem comprometer o desempenho, ao contrário, melhorando a acurácia da rede. Simultaneamente, a otimização conduzida pelo segundo AG identificou configurações de hiperparâmetros que potencializaram o poder discriminativo da CNN, resultando em métricas de desempenho superiores às reportadas por modelos de referência. Adicionalmente, os experimentos comprovaram que o modelo proposto mantém baixa latência de inferência, requisito essencial para aplicações em redes de alta velocidade.

Sob a perspectiva aplicada, este trabalho oferece uma solução viável e escalável para sistemas de detecção de intrusões, com potencial de utilização em ambientes reais. A redução do tempo de resposta frente a tráfego malicioso possibilita maior eficiência em ações mitigadoras, reduzindo riscos de indisponibilidade e danos em infraestruturas críticas. Dessa forma, a pesquisa contribui não apenas para o avanço científico na área de segurança em redes, mas também para a aplicação prática de modelos de aprendizado de máquina em cenários que exigem alto desempenho e confiabilidade.

1.8 Estrutura do Documento

A presente dissertação está organizada em capítulos, cada um com objetivos específicos, conforme descrito a seguir:

- **Capítulo 2 — Revisão Bibliográfica**: apresenta um panorama das principais técnicas de detecção de intrusões, destacando abordagens clássicas e tecnologias emergentes relevantes para o estado da arte.
- **Capítulo 3 — Fundamentação Teórica**: descreve os conceitos essenciais que sustentam o trabalho, com ênfase em redes neurais convolucionais e algoritmos genéticos, incluindo princípios de funcionamento e aplicações no contexto de segurança da informação.
- **Capítulo 4 — Metodologia**: detalha as etapas de desenvolvimento do modelo proposto, desde a preparação dos dados e configuração experimental até a integração entre seleção de atributos e ajuste de hiperparâmetros por meio de algoritmos genéticos.

- **Capítulo 5 — Resultados e Discussão**: apresenta os experimentos realizados, analisa o desempenho do modelo em múltiplas métricas e compara os resultados com trabalhos da literatura, destacando os avanços alcançados.
- **Capítulo 6 — Conclusões e Perspectivas Futuras**: sintetiza as principais contribuições da pesquisa, discute suas implicações práticas e aponta possíveis direções para trabalhos futuros.

Capítulo 2

Revisão Bibliográfica

A detecção de intrusões em redes de computadores é um campo que tem evoluído rapidamente desde os primeiros sistemas baseados em assinaturas até o emprego de técnicas modernas de Inteligência Artificial (IA) e aprendizado profundo. Esta seção apresenta uma revisão atualizada das principais abordagens reportadas na literatura, suas limitações e as lacunas que motivam a presente pesquisa.

Historicamente, Lunt [15] realizou uma das primeiras revisões abrangentes de técnicas de detecção de intrusões, categorizando os métodos então existentes e apontando limitações intrínsecas aos sistemas baseados exclusivamente em assinaturas. Axelsson [16] posteriormente formalizou uma taxonomia de IDS, distinguindo três categorias principais: baseados em assinaturas, em anomalias e híbridos. Esses trabalhos clássicos servem de base conceitual para o desenvolvimento de abordagens mais recentes.

Sharafaldin et al. [17] contribuíram significativamente para o avanço da pesquisa em detecção de intrusões ao desenvolverem a série de bases de dados do Canadian Institute for Cybersecurity (CIC), incluindo CICIDS2017, CSE-CIC-IDS2018 e, posteriormente, CIC-DDoS2019. A CICIDS2017 introduziu um conjunto abrangente de tráfego realista com múltiplos vetores de ataque distribuídos ao longo de vários dias, tornando-se referência na avaliação de IDS. Já a CSE-CIC-IDS2018 — base utilizada neste trabalho — ampliou substancialmente a complexidade ao incorporar uma topologia de rede maior, sete cenários distintos de ataque e um conjunto expandido de características extraídas via CICFlowMeter, conferindo maior diversidade e representatividade aos fluxos. Por fim, a CIC-DDoS2019 especializou-se em ataques de negação de serviço distribuída, reunindo ampla variedade de vetores DDoS modernos. Em conjunto, essas bases compartilham metodologia de captura, rotulação e extração de fluxos, mas diferem quanto ao foco, variedade de ataques e volume de tráfego, tornando a CICIDS2018 especialmente adequada para estudos que demandam generalização em cenários amplos e heterogêneos.

Drewek-Ossowicka et al. [18] revisaram mais de 150 estudos que empregaram técnicas de aprendizado profundo para detecção de intrusões, destacando o papel das Redes Neurais Convolucionais (CNN) e das Redes Neurais Recorrentes (RNN), especialmente as Long Short-Term Memory (LSTM), por sua capacidade de capturar dependências temporais.

Lopez-Martin et al. [19] propuseram um classificador de tráfego IoT com duas camadas convolucionais (32 e 64 filtros, ativação ReLU) seguidas por uma camada LSTM. Aplicado à base UNSW-NB15, o modelo alcançou 98,2 % de acurácia, superando *Support Vector Machine* (SVM) e *Random Forest*. O estudo, contudo, apresentou alta demanda computacional, dificultando sua aplicação em tempo real.

Shieh et al. [20] desenvolveram um modelo Bidirectional LSTM (BI-LSTM) combinado a um Gaussian Mixture Model (GMM) para detecção de ataques DDoS. O modelo atingiu 96,8 % de acurácia nas bases CIC-IDS2017 e CIC-DDoS2019, porém não explorou otimização automática de hiperparâmetros — lacuna explorada nesta dissertação.

Thirimanne et al. [21] apresentaram uma arquitetura de **Rede Neural Profunda (DNN)** de baixa latência, obtendo 99,1 % de acurácia no NSL-KDD com tempo de inferência inferior a 10 ms, mas sem explorar seleção de atributos ou técnicas evolutivas.

Nguyen et al. [22] compararam CNN, LSTM e arquiteturas híbridas em múltiplas bases, observando que CNNs apresentaram menor latência enquanto LSTMs alcançaram maior *recall* para ataques raros. O trabalho demonstrou que a escolha da arquitetura depende do cenário, mas não propôs mecanismos de ajuste dinâmico — o que motiva o uso de AGs neste estudo.

A Tabela 2.1 resume as principais características dos métodos clássicos e modernos, associando vantagens e desvantagens a partir de fontes consolidadas na literatura.

Tabela 2.1: Comparação entre métodos de detecção de intrusões.

Método	**Vantagens**	**Desvantagens**
Baseado em assinaturas	Alta precisão para ataques conhecidos; baixo custo computacional [15, 16].	Ineficaz contra ataques inéditos; exige atualização contínua.
Baseado em anomalias	Detecta ataques zero-day; adaptável a novos padrões [18].	Maior taxa de falsos positivos; requer treinamento complexo.
Híbrido	Combina pontos fortes das abordagens anteriores [20, 21].	Maior complexidade e consumo computacional.

A Tabela 2.2 apresenta um conjunto de trabalhos representativos da literatura recente, destacando a técnica principal empregada, o tipo de método e a base de dados utilizada, sem atribuição de qualquer avaliação comparativa.

Tabela 2.2: Trabalhos representativos: técnica, tipo de método e caracterização geral.

Referência	**Técnica principal**	**Tipo de método**	**Base de dados**	**Descrição**
Lopez-Martin et al. [19]	CNN + LSTM	Híbrido	UNSW-NB15	Análise baseada em arquiteturas profundas integradas
Shieh et al. [20]	BI-LSTM + GMM	Híbrido	CIC-IDS2017 / CIC-DDoS2019	Abordagem combinando redes recorrentes bidirecionais e modelos probabilísticos
Thirimanne et al. [21]	DNN	Supervisionado	NSL-KDD	Proposta centrada em rede neural profunda de múltiplas camadas
Nguyen et al. [22]	CNN, LSTM e híbrido	Comparativo	Diversas bases	Estudo comparativo entre diferentes arquiteturas de aprendizado profundo
Proposta (presente trabalho)	AG + CNN	Híbrido (IA evolutiva)	CIC-IDS2018	Integração de algoritmo genético com rede neural convolucional

A análise conjunta desses estudos evidencia a diversidade de abordagens aplicadas ao contexto de detecção de intrusões, variando desde modelos puramente supervisionados até arquiteturas híbridas e métodos evolutivos. Observa-se ainda que diferentes bases de dados têm sido utilizadas como referência, refletindo a ausência de um padrão único de avaliação no campo.

Além disso, nota-se que a literatura recente explora uma ampla gama de arquiteturas profundas, embora nem sempre incorpore procedimentos sistemáticos de seleção de atributos ou estratégias evolutivas de otimização, aspectos que motivam a presente investigação.

Capítulo 3

Fundamentação Teórica

3.1 Introdução às Redes Neurais Artificiais

As Redes Neurais Artificiais (RNAs) são modelos capazes de aprender padrões complexos a partir de dados. Neste trabalho, empregou-se uma variação de RNA denominada CNN, escolhida por sua capacidade de extrair representações discriminativas de dados estruturados. O processo de ajuste de pesos foi realizado via *backpropagation*, com função de custo e otimizador definidos automaticamente pelo Algoritmo Genético, permitindo configuração adaptada ao problema de detecção de intrusões.

3.2 Redes Neurais Convolucionais

As CNNs constituem uma classe de arquiteturas de aprendizado profundo inspiradas no funcionamento do córtex visual humano. Diferentemente das redes neurais totalmente conectadas (*Fully Connected Neural Networks*), que estabelecem conexões entre todos os neurônios de camadas adjacentes, as CNNs exploram o conceito de operações locais e de compartilhamento de pesos, o que resulta em modelos mais eficientes e com melhor capacidade de generalização em tarefas que envolvem dados estruturados em grades, como imagens, séries temporais e fluxos de tráfego de rede.

O componente central de uma CNN é a camada convolucional, na qual filtros (*kernels*) deslizam sobre a entrada aplicando operações de convolução. Cada filtro aprende a detectar padrões específicos, como bordas, variações de intensidade ou, no contexto de IDS, combinações de atributos de tráfego que caracterizam comportamentos normais ou maliciosos [23]. Matematicamente, a saída de uma convolução pode ser representada como

$$s(t) = (x * w)(t) = \sum_{\tau=-\infty}^{+\infty} x(\tau) \cdot w(t - \tau), \tag{3.1}$$

onde x representa o sinal de entrada e w o filtro convolucional. Esse processo é responsável

pela extração hierárquica de características, permitindo que camadas iniciais capturem padrões simples e camadas posteriores combinem tais padrões em representações mais complexas.

Além das camadas convolucionais, as CNNs geralmente incluem camadas de *pooling*, responsáveis por reduzir a dimensionalidade das representações intermediárias sem perder informações relevantes. A técnica mais comum é o *max pooling*, definida por [23]:

$$y_{i,j} = \max_{(m,n)\in\Omega_{i,j}} x_{m,n}, \tag{3.2}$$

no qual cada saída $y_{i,j}$ corresponde ao valor máximo em uma vizinhança local $\Omega_{i,j}$ da entrada x. Isso confere à rede invariância espacial parcial, além de diminuir o custo computacional.

Ao final do processo convolucional e de *pooling*, as representações extraídas são encaminhadas para camadas densas (*fully connected*), onde ocorre a combinação não-linear das características aprendidas para produzir a saída final, como a classificação entre tráfego normal e ataque.

A principal vantagem das CNNs está em sua capacidade de aprender automaticamente representações discriminativas a partir dos dados, dispensando a necessidade de extração manual de atributos. Essa característica torna as CNNs particularmente eficazes em sistemas de detecção de intrusões, uma vez que o tráfego de rede pode ser organizado em estruturas matriciais análogas a imagens, permitindo que a rede identifique padrões complexos de ataque com elevada acurácia e baixo tempo de inferência.

Em síntese, as CNNs oferecem um equilíbrio entre poder de representação e eficiência computacional, o que justifica sua adoção em diversos domínios da Inteligência Artificial, incluindo a cibersegurança.

O fluxograma da Figura 3.1 apresenta, de forma esquemática, o funcionamento de uma CNN aplicada à detecção de intrusões. Na etapa inicial, a entrada matricial $(H \times W \times C)$ corresponde ao tráfego de rede organizado em uma grade de atributos, a qual é processada pela camada convolucional, onde filtros especializados extraem padrões locais que revelam dependências entre atributos correlacionados. Em seguida, a ativação não-linear amplia a capacidade do modelo de representar relações complexas, enquanto a operação de *max pooling* reduz a dimensionalidade e preserva as informações mais relevantes. O vetor resultante é então achatado e encaminhado para camadas densas, responsáveis por integrar globalmente as características extraídas nas etapas anteriores, até que a camada de saída, com ativação sigmóide, retorne a probabilidade de um fluxo ser classificado como intrusão ou tráfego normal, evidenciando a hierarquia de abstração típica das CNNs na transformação de dados brutos em uma decisão final de classificação binária.

3.2.1 Representação de Tráfego de Rede para Entrada em CNN

Além das estratégias de regularização, a representação dos dados de entrada exerce influência determinante na qualidade do aprendizado. Neste trabalho, o vetor original de 20 atributos

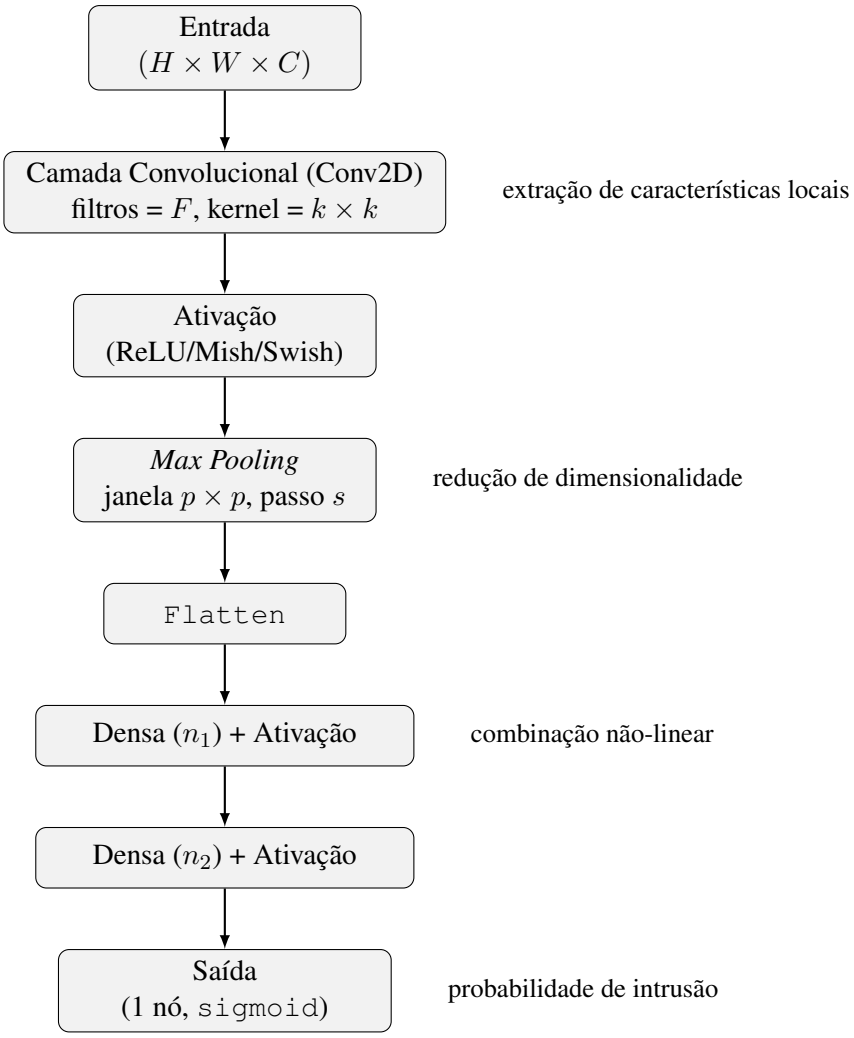

Figura 3.1: Fluxograma típico de uma CNN aplicada a IDS.

do tráfego de rede foi reorganizado na forma matricial 4×5, com canal único. Essa disposição não é arbitrária: ela busca preservar correlações semânticas entre atributos relacionados, como estatísticas referentes ao tráfego de envio e recebimento, posicionando-os em proximidade espacial na grade. Assim, os filtros convolucionais conseguem explorar tais relações locais de maneira mais eficaz, aproximando-se do princípio que fundamenta a detecção de padrões em imagens.

Para garantir equilíbrio entre os atributos, foi aplicada normalização *feature-wise*, baseada no *z-score* de cada coluna. Esse procedimento evita que atributos de maior magnitude numérica dominem a propagação direta da rede, assegurando que todas as variáveis contribuam de maneira equitativa para a aprendizagem. Dessa forma, o modelo é capaz de identificar padrões relevantes mesmo em atributos de menor escala, ampliando a robustez e a capacidade discriminativa da CNN.

3.3 Funções de Ativação

As funções de ativação são essenciais nas RNAs, pois introduzem não-linearidade no modelo, permitindo a aproximação de funções complexas. A escolha adequada da função de ativação afeta diretamente a capacidade de generalização, a estabilidade do gradiente e a velocidade de convergência. Além disso, para o algoritmo de retropropagação, é fundamental a derivada $\phi'(x)$ de cada função, uma vez que esta determina como o erro é propagado entre as camadas [23–27]. A seguir são apresentadas as funções de ativação utilizadas neste trabalho, juntamente com suas equações e respectivas análises.

Função Linear

A função linear é definida como

$$\phi(x) = x, \tag{3.3}$$

com derivada

$$\phi'(x) = 1. \tag{3.4}$$

Por ser estritamente linear, não introduz complexidade adicional ao modelo, mas é útil em camadas de saída em problemas de regressão.

Função ReLU

A *Rectified Linear Unit* (ReLU) é dada por

$$\phi(x) = \max(0, x), \tag{3.5}$$

com derivada

$$\phi'(x) = \begin{cases} 0, & x < 0, \\ 1, & x > 0. \end{cases} \tag{3.6}$$

A ReLU é amplamente utilizada devido à sua simplicidade computacional e à mitigação parcial do problema do desvanecimento do gradiente. Contudo, apresenta a limitação conhecida como *dying ReLU*, que ocorre quando neurônios ficam permanentemente inativos [23].

Função SELU

A *Scaled Exponential Linear Unit* (SELU) é definida por

$$\phi(x) = \begin{cases} k \cdot x, & x > 0, \\ k \cdot \alpha \cdot (e^x - 1), & x \leq 0, \end{cases} \tag{3.7}$$

onde $\alpha = 1,67326324$ e $k = 1,05070098$. Sua derivada é dada por

$$\phi'(x) = \begin{cases} k, & x > 0, \\ k \cdot \alpha \cdot e^x, & x \leq 0. \end{cases} \tag{3.8}$$

A SELU promove normalização automática da saída dos neurônios, favorecendo a estabilidade em arquiteturas profundas [25].

Função ELU

A *Exponential Linear Unit* (ELU) é expressa como

$$\phi(x) = \begin{cases} x, & x > 0, \\ \alpha \cdot (e^x - 1), & x \leq 0, \end{cases} \tag{3.9}$$

com $\alpha = 1.0$. Sua derivada é

$$\phi'(x) = \begin{cases} 1, & x > 0, \\ \phi(x) + \alpha, & x \leq 0. \end{cases} \tag{3.10}$$

A ELU suaviza a descontinuidade da ReLU em $x = 0$, permitindo gradientes não-nulos para entradas negativas [24].

Função Softmax

A função *Softmax* é definida para vetores de entrada $x \in \mathbb{R}^K$ como

$$\phi(x_j) = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \quad j = 1, \ldots, K. \tag{3.11}$$

A derivada envolve a matriz Jacobiana, dada por

$$\frac{\partial \phi(x_j)}{\partial x_i} = \begin{cases} \phi(x_j) \cdot (1 - \phi(x_j)), & i = j, \\ -\phi(x_i) \cdot \phi(x_j), & i \neq j. \end{cases} \tag{3.12}$$

O *Softmax* é utilizado principalmente na saída de classificadores multiclasse, garantindo distribuição de probabilidade [23].

Função Softplus

A função *Softplus* é dada por

$$\phi(x) = \ln(1 + e^x), \tag{3.13}$$

com derivada

$$\phi'(x) = \frac{1}{1 + e^{-x}} = \sigma(x), \tag{3.14}$$

isto é, a própria função sigmóide. O *Softplus* pode ser entendido como uma versão suavizada da ReLU [23].

Função Mish

A função *Mish* é definida por

$$\phi(x) = x \cdot \tanh(\ln(1 + e^{x})), \tag{3.15}$$

cuja derivada é suave e não-nula para praticamente todo o domínio. Essa característica favorece um fluxo de gradiente mais estável em arquiteturas profundas [27].

Função Swish

A função *Swish* é expressa por

$$\phi(x) = x \cdot \sigma(x) = \frac{x}{1 + e^{-x}}, \tag{3.16}$$

com derivada

$$\phi'(x) = \phi(x) + \sigma(x)(1 - \phi(x)), \tag{3.17}$$

onde $\sigma(x)$ é a função sigmóide. A *Swish* mantém derivadas não-nulas em todo o domínio, sendo considerada uma alternativa moderna à ReLU [26].

Discussão Crítica

A análise das derivadas evidencia que funções como Linear e ReLU possuem simplicidade e baixo custo computacional, mas podem limitar a propagação de gradientes em cenários complexos. Por outro lado, SELU e ELU oferecem mecanismos de normalização e suavização, reduzindo a inatividade de neurônios. *Softmax* é indispensável para classificação multiclasse, enquanto *Softplus*, *Mish* e *Swish* são alternativas que preservam gradientes em regiões amplas, favorecendo arquiteturas profundas, como a CNN utilizada neste trabalho.

Resumo Comparativo

A Tabela 3.1 apresenta um conjunto abrangente das principais funções de ativação utilizadas em redes neurais modernas, incluindo suas equações, derivadas e vantagens práticas. Essa comparação evidencia como diferentes ativadores influenciam o fluxo de gradientes, a estabilidade numérica e a capacidade de generalização dos modelos. Funções clássicas, como Linear e ReLU, contrastam com variantes mais recentes, como *Swish* e *Mish*, que introduzem suavidade

e não linearidades mais ricas capazes de melhorar o desempenho em arquiteturas profundas. Além disso, funções como SELU e ELU destacam-se por mitigar problemas de desvanecimento do gradiente e favorecer processos de autonormalização, reforçando a importância da escolha adequada da ativação para cada tipo de tarefa e arquitetura.

Tabela 3.1: Funções de ativação utilizadas: equações, derivadas e principais vantagens.

Função	$\phi(x)$	$\phi'(x)$	**Principais vantagens**
Linear	x	1	Útil em saídas de regressão; estabilidade numérica.
ReLU	$\max(0, x)$	$\begin{cases} 0, & x < 0 \\ 1, & x > 0 \end{cases}$	Simples e barata; induz esparsidade; mitiga parcialmente o desvanecimento do gradiente [23].
SELU	$\begin{cases} kx, & x > 0 \\ k\alpha\,(e^x - 1), & x \leq 0 \end{cases}$	$\begin{cases} k, & x > 0 \\ k\alpha\, e^x, & x \leq 0 \end{cases}$	Autonormalização; treinamento estável em arquiteturas profundas [25].
ELU	$\begin{cases} x, & x > 0 \\ \alpha\,(e^x - 1), & x \leq 0 \end{cases}$	$\begin{cases} 1, & x > 0 \\ \phi(x) + \alpha, & x \leq 0 \end{cases}$	Permite ativações negativas; média próxima de zero; convergência mais rápida [24].
Softmax	$\phi(x_j) = \dfrac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}$	Jacobiana com termos cruzados $\partial\phi_j/\partial x_i$	Saída probabilística normalizada para multiclasse [23].
Softplus	$\ln(1 + e^x)$	$\sigma(x)$	ReLU suavizada; derivada não nula em $x < 0$; maior estabilidade [23].
Mish	$x\,\tanh(\ln(1 + e^x))$	Suave e não nula em quase todo domínio	Gradientes suaves; boa generalização reportada em tarefas profundas [27].
Swish	$x\,\sigma(x)$	$\phi(x) + \sigma(x)\,(1 - \phi(x))$	Suave e não monótona; desempenho superior à ReLU em redes profundas [26].

Notas: $\sigma(x) = \frac{1}{1+e^{-x}}$. Para SELU, usar $\alpha = 1{,}67326324$ e $k = 1{,}05070098$.

3.3.1 Derivadas e retropropagação

No processo de treinamento de redes neurais, a retropropagação (*backpropagation*) exige o cálculo das derivadas das funções de ativação, denotadas por $\phi'(x)$. Essas derivadas desempenham um papel fundamental, pois determinam a estabilidade do fluxo de gradientes ao longo das camadas. Em outras palavras, a escolha da função de ativação não impacta apenas a não-linearidade introduzida na propagação direta, mas também influencia se o gradiente será preservado, reduzido (*vanishing gradient*) ou amplificado (*exploding gradient*).

Por exemplo, a função ReLU, definida como

$$\phi(x) = \max(0, x), \tag{3.18}$$

apresenta derivada

$$\phi'(x) = \begin{cases} 0, & x < 0, \\ 1, & x > 0. \end{cases} \tag{3.19}$$

Isso significa que, para entradas negativas, a derivada é nula e o neurônio deixa de ser atualizado, fenômeno conhecido como *dead neuron*. Embora a ReLU tenha impulsionado o treinamento de redes profundas pela sua simplicidade e eficiência computacional, esse comportamento pode comprometer o aprendizado quando muitos neurônios mantém-se permanentemente inativos.

Em contraste, funções mais recentes como a *Swish* e a *Mish* apresentam derivadas suaves e não-nulas em uma faixa mais ampla. A função *Swish* é definida por

$$\phi(x) = x \cdot \sigma(x), \quad \text{onde } \sigma(x) = \frac{1}{1 + e^{-x}}, \tag{3.20}$$

cuja derivada pode ser expressa como

$$\phi'(x) = \sigma(x) + x \cdot \sigma(x) \cdot (1 - \sigma(x)). \tag{3.21}$$

Já a função *Mish* é dada por

$$\phi(x) = x \cdot \tanh(\ln(1 + e^{x})), \tag{3.22}$$

com uma derivada mais complexa, mas caracterizada pela suavidade contínua em todo o domínio real e ausência de regiões totalmente planas.

Essas propriedades tornam *Swish* [26] e *Mish* [27] menos suscetíveis ao problema do desvanecimento do gradiente, garantindo uma propagação de erro mais estável. Isso favorece tanto a convergência durante o treinamento quanto a capacidade de generalização da rede, especialmente em cenários de grande profundidade ou alta variabilidade dos dados.

Portanto, a análise de $\phi'(x)$ não deve ser vista como mera formalidade matemática, mas como uma ferramenta essencial para compreender os efeitos de cada função de ativação no processo de otimização [23, 28]. Em síntese, funções com derivadas suaves e não-nulas em faixas mais amplas do domínio contribuem para redes mais robustas e de treinamento mais eficiente.

3.4 Regularização

A regularização desempenha um papel crucial no treinamento de redes neurais, atuando como um conjunto de técnicas destinadas a mitigar o *overfitting*, ou seja, a tendência do modelo em memorizar excessivamente os dados de treinamento e perder capacidade de generalização. Para lidar com esse desafio, diversas estratégias foram aplicadas nesta pesquisa, combinando abordagens clássicas e avançadas.

3.4.1 *L2 Weight Decay*

O *weight decay* (ou regularização L2) consiste em adicionar um termo de penalização proporcional à norma dos pesos à função de perda. Formalmente, o termo $\frac{\lambda}{2}\|\theta\|^2$ é somado ao custo original, resultando em uma atualização dos parâmetros que favorece pesos de menor magnitude [23, 29]. Essa técnica contribui para reduzir a complexidade do modelo, evitando soluções instáveis e garantindo maior robustez frente a variações nos dados.

3.4.2 *Dropout* e *Alpha-Dropout*

O *Dropout* é uma técnica probabilística que desativa aleatoriamente neurônios durante o treinamento com probabilidade p, promovendo o aprendizado de representações redundantes e reduzindo a coadaptação entre unidades. Em termos práticos, equivale a treinar um *ensemble* de sub-redes menores, isto é, um conjunto de modelos parciais que compartilham pesos e cuja combinação efetiva resulta em maior capacidade de generalização [23].

No caso específico da função de ativação SELU, empregou-se o *Alpha-Dropout*, uma variante projetada para preservar as propriedades de auto-normalização da rede [25]. Diferentemente do *Dropout* tradicional, o *Alpha-Dropout* ajusta as ativações desligadas de forma a manter a média e variância inalteradas, assegurando que o fluxo de informação permaneça estável mesmo em arquiteturas profundas.

3.4.3 *Early Stopping*

Outra medida utilizada foi o *early stopping*, em que o processo de treinamento é interrompido quando o desempenho no conjunto de validação deixa de apresentar melhorias significativas. Essa técnica evita que o modelo continue ajustando-se a flutuações aleatórias do conjunto de treino, reduzindo a chance de sobreajuste e economizando recursos computacionais.

3.5 Otimizadores: teoria, prática e análise matemática

O processo de otimização em redes neurais visa minimizar a função de custo ajustando os parâmetros θ (pesos e vieses) da rede por meio de métodos iterativos baseados em gradientes.

Diferentes otimizadores variam na forma como calculam e aplicam essas atualizações, influenciando diretamente a convergência, estabilidade e velocidade do treinamento. Nesta seção são discutidos os principais otimizadores empregados neste trabalho, juntamente com suas formulações matemáticas e propriedades.

3.5.1 *Stochastic Gradient Descent* (SGD) e *Momentum*

O *Stochastic Gradient Descent* (SGD) é o método mais simples de otimização, onde os parâmetros são atualizados a partir do gradiente estimado em um mini-batch de dados:

$$\theta_t = \theta_{t-1} - \eta\, \hat{g}_t, \tag{3.23}$$

onde η é a taxa de aprendizado e $\hat{g}_t$ é o gradiente estocástico no passo t.

Para melhorar o comportamento do SGD, especialmente em superfícies irregulares, utiliza-se o conceito de *momentum*, que acumula gradientes passados e suaviza oscilações:

$$m_t = \mu m_{t-1} + \eta \nabla_\theta L(\theta_{t-1}), \qquad \theta_t = \theta_{t-1} - m_t, \tag{3.24}$$

onde $\mu \in [0, 1)$ é o coeficiente de *momentum*.

O *momentum* atua como um filtro exponencial que acelera a descida em direções persistentes e reduz o ruído estocástico [23, 30]. Isso é particularmente útil em regiões com vales estreitos ou platôs, comuns em funções de custo de alta dimensionalidade.

3.5.2 *Adaptive Moment Estimation* (Adam)

O Adam combina os benefícios do *momentum* com a adaptação da taxa de aprendizado para cada parâmetro. Ele calcula médias móveis de primeira e segunda ordem do gradiente:

$$g_t = \nabla_\theta L(\theta_{t-1}), \tag{3.25}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t, \tag{3.26}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2, \tag{3.27}$$

onde β_1 e β_2 controlam os decaimentos exponenciais.

Como essas estimativas são enviesadas no início, aplica-se correção:

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \qquad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}. \tag{3.28}$$

A atualização final é dada por:

$$\theta_t = \theta_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}, \tag{3.29}$$

onde ϵ é um termo de estabilidade numérica (tipicamente 10^{-8}).

O Adam é adaptativo por parâmetro, ou seja, cada peso possui sua própria taxa de aprendizado ajustada pela variância do gradiente [23, 31]. Essa característica torna o método robusto em problemas com gradientes esparsos ou heterogêneos. Os parâmetros recomendados na literatura são $\beta_1 = 0.9$, $\beta_2 = 0.999$ e $\epsilon = 10^{-8}$.

3.5.3 Adamax

O Adamax é uma variante do Adam baseada na norma infinito (ℓ_∞), substituindo a média quadrática dos gradientes pelo máximo móvel:

$$u_t = \max(\beta_2 u_{t-1}, |g_t|), \qquad \theta_t = \theta_{t-1} - \frac{\eta}{u_t} m_t. \tag{3.30}$$

Intuição: ao usar a norma infinita, o Adamax torna-se mais estável em situações com gradientes esparsos ou picos abruptos [31]. Embora menos popular que o Adam, pode oferecer robustez extra em cenários específicos.

3.5.4 Síntese Comparativa

Em resumo:

- O **SGD com *momentum*** é simples e eficiente, mas requer ajuste cuidadoso da taxa de aprendizado.
- O **Adam** oferece maior robustez e adaptação automática de taxas, sendo amplamente utilizado em CNNs modernas.
- O **Adamax** representa uma alternativa robusta em casos de gradientes extremos ou esparsidade.

A escolha do otimizador impacta diretamente a convergência do modelo. Neste trabalho, a seleção foi realizada de forma automática pelo Algoritmo Genético, permitindo explorar a adequação de cada método ao problema de detecção de intrusões.

3.6 Funções de Custo (*Loss*) e sua Interpretação

As funções de custo, também chamadas de funções de perda, são fundamentais no aprendizado supervisionado, pois quantificam o erro do modelo em relação aos dados observados. O processo de treinamento busca minimizá-las para ajustar os parâmetros da rede e melhorar a capacidade preditiva.

3.6.1 *Binary Crossentropy* (Entropia Cruzada Binária)

Indicada para problemas de classificação binária, em que o rótulo verdadeiro é $y \in \{0, 1\}$ e a predição do modelo é $\hat{y} \in (0, 1)$ (probabilidade estimada da classe positiva) [23, 32].

A função de perda por amostra é dada por:

$$\mathcal{L}(\hat{y}, y) = -\big[y \log(\hat{y}) + (1 - y) \log(1 - \hat{y})\big]. \tag{3.31}$$

- Equivale à *log-loss* para uma distribuição Bernoulli, medindo a discrepância entre a distribuição verdadeira (rótulo) e a predita.
- Minimizar essa função é equivalente a maximizar a verossimilhança do modelo sob a hipótese de dados Bernoulli independentes.
- Penaliza fortemente predições confidentes e erradas (por exemplo, $\hat{y} \to 1$ quando $y = 0$).
- É convexa em relação a $\hat{y}$, o que facilita otimização.

3.6.2 *Poisson Loss*

Indicada para problemas de regressão envolvendo contagens, onde o valor observado y é um número inteiro não negativo modelado como uma variável de Poisson com parâmetro λ (média e variância iguais) [32].

A função de perda por amostra é dada por:

$$\mathcal{L}(\hat{\lambda}, y) = \hat{\lambda} - y \log(\hat{\lambda}) + \log(y!). \tag{3.32}$$

Na prática, o termo $\log(y!)$ não depende de $\hat{\lambda}$ e pode ser ignorado durante a otimização.

Interpretação:

- Representa a *negativa da log-verossimilhança* sob o modelo Poisson.
- Penaliza erros na predição da taxa $\hat{\lambda}$, incentivando que ela se aproxime do valor real y.
- Adequada para dados de contagem com variância proporcional à média.

Dado um vetor de probabilidades alvo $P = (P(1), \ldots, P(k))$ e uma predição $Q = (Q(1), \ldots, Q(k))$, ambas distribuições discretas sobre k classes, a Divergência de Kullback–Leibler (KL) é definida como [33]:

$$D_{\mathrm{KL}}(P \| Q) = \sum_{i=1}^{k} P(i) \log \frac{P(i)}{Q(i)}. \tag{3.33}$$

- Mede a dissimilaridade ou "distância" (não simétrica) entre as distribuições P (verdadeira) e Q (predita).

- É sempre não negativa e igual a zero somente se $P = Q$ ponto a ponto.
- É amplamente utilizada em aprendizado probabilístico para alinhar a distribuição predita do modelo com a distribuição alvo.
- Em classificadores com múltiplas classes, pode ser vista como uma generalização da entropia cruzada.

Em síntese, a escolha da função de custo deve considerar o tipo de problema e a natureza dos dados, de modo que a otimização corresponda a maximizar a verossimilhança (ou minimizar a discrepância) entre o modelo e a realidade.

3.6.3 Matriz de Confusão

A matriz de confusão sintetiza o desempenho do classificador, conforme Tabela 3.2:

Tabela 3.2: Matriz de confusão para classificação binária

	Previsto Positivo	**Previsto Negativo**
Verdadeiro Positivo	TP	FN
Verdadeiro Negativo	FP	TN

Exemplo: em IDS, um FP pode ser um falso alarme (tráfego legítimo classificado como ataque), enquanto um FN é um ataque não detectado, geralmente mais crítico.

3.7 Pré-processamento e Representação dos Atributos

3.7.1 Normalização

A normalização é uma etapa essencial no pré-processamento de dados para redes neurais, pois garante que os atributos de entrada estejam em escalas compatíveis. Sem esse ajuste, variáveis com magnitudes muito distintas podem dominar o processo de otimização, resultando em convergência lenta ou em soluções subótimas [23].

Duas técnicas usuais são amplamente empregadas. A primeira é a padronização (*z-score*), que transforma cada atributo para ter média zero e desvio padrão unitário. Essa abordagem é indicada quando os dados apresentam distribuição aproximadamente gaussiana. A segunda é a normalização min–max, que reescala os atributos para um intervalo definido, tipicamente $[0, 1]$ ou $[-1, 1]$. Essa técnica é recomendada quando se deseja preservar as relações proporcionais entre valores e quando os atributos possuem limites conhecidos.

A escolha entre padronização e min–max depende da natureza do conjunto de dados e do comportamento esperado das funções de ativação utilizadas na rede. Em ambos os casos, o

objetivo central é melhorar a estabilidade numérica, acelerar a convergência e favorecer um treinamento mais robusto.

3.7.2 Mapeamento para entrada 2D (4 × 5) da CNN

Neste trabalho, vetores com 20 atributos foram reestruturados para a matriz $4 \times 5 \times 1$. Observações importantes:

- **Ordem dos atributos:** proximidade semântica entre features adjacentes na matriz pode melhorar detecção por convoluções (preservar correlações locais).
- **Normalização por atributo:** evita dominância de features com escala maior.
- **Justificativa teórica:** operações locais de convolução exploram correlações entre atributos próximos na grade — uma representação similar a imagens tem sido adotada em trabalhos recentes de IDS via CNN [34, 35].

3.8 Integração com Algoritmos Genéticos

A integração de AGs com CNNs tem sido explorada na literatura como uma estratégia de busca e otimização de arquiteturas. Em geral, os AGs são aplicados tanto na seleção de atributos mais relevantes para classificação quanto no ajuste automático de hiperparâmetros das redes neurais. Trabalhos recentes apontam que essa combinação é promissora, pois alia a capacidade exploratória dos AGs à expressividade das CNNs, resultando em modelos mais eficientes e robustos. Essa abordagem visa reduzir a dimensionalidade, evitar sobreajuste e, simultaneamente, encontrar combinações de configuração que maximizem a acurácia e minimizem a latência.

3.8.1 Representação cromossômica

Dois tipos de cromossomos foram definidos de acordo com o objetivo da otimização:

1. **Seleção de atributos:** cada cromossomo possui 20 genes, cada um representando o índice de um atributo do conjunto original de 78 variáveis. O espaço de busca associado a essa tarefa é da ordem de $\binom{78}{20}$, o que torna inviável a exploração exaustiva e justifica o uso de heurísticas evolutivas.
2. **Hiperparâmetros da CNN:** o cromossomo codifica múltiplos aspectos da arquitetura e do treinamento, incluindo: quatro genes correspondentes às funções de ativação, três genes para o número de nós em camadas densas, um gene para o otimizador e um gene para a função de perda.

3.8.2 Operadores genéticos

O processo evolutivo emprega operadores clássicos adaptados ao problema:

- **Seleção:** método da roleta proporcional ao *fitness*, favorecendo indivíduos de maior desempenho sem eliminar a diversidade.
- **Cruzamento:** operador de Radcliff (semelhante a *blend crossover*), aplicado a genes contínuos. Formalmente:

$$\text{filho}_1 = \beta p_1 + (1-\beta)p_2, \quad \text{filho}_2 = (1-\beta)p_1 + \beta p_2, \tag{3.34}$$

  com $\beta \in [0, 1]$, onde p_1 e p_2 são os pais.
- **Mutação:** perturbação aleatória de genes com probabilidade p_m, permitindo explorar regiões não visitadas do espaço de busca.
- **Elitismo:** preservação do melhor indivíduo a cada geração, garantindo que a qualidade da população não decresça.

3.8.3 Função de aptidão

A aptidão de cada indivíduo é avaliada por meio das métricas de desempenho (acurácia ou F1 ponderado) obtidas após o treinamento da CNN correspondente. Além disso, o tempo de treinamento pode ser incorporado como critério adicional, permitindo balancear a busca entre precisão e custo computacional.

3.9 Integração de AGs com CNNs na Detecção de Intrusões

Diversos trabalhos recentes têm investigado a utilização conjunta de AGs e CNNs no contexto da segurança em redes. Os AGs atuam como mecanismos de busca e otimização, enquanto as CNNs oferecem elevado poder de representação para a análise de tráfego. Essa integração pode ocorrer em dois níveis principais: (i) seleção de atributos mais relevantes para reduzir a dimensionalidade do problema; e (ii) ajuste automático de hiperparâmetros da CNN, evitando a dependência de configuração manual.

Shieh et al. [36], por exemplo, aplicaram técnicas de otimização evolutiva em conjunto com redes recorrentes e CNNs para melhorar a detecção de ataques DDoS. Thirimanne et al. [37] utilizaram um esquema híbrido de busca de hiperparâmetros aliado a arquiteturas profundas. Outros autores, como Nguyen et al. [38], compararam diferentes estratégias de integração entre AGs e redes neurais, evidenciando ganhos de desempenho ao incorporar mecanismos evolutivos no processo de modelagem.

A análise desses trabalhos evidencia que a combinação AG–CNN tem se mostrado promissora, mas ainda carece de maior exploração no contexto de IDS com bases modernas e de grande escala, como o CIC-IDS2018, justificando a proposta desta dissertação.

3.10 Complexidade computacional

O custo computacional da abordagem é diretamente proporcional ao número de indivíduos, épocas de treinamento e custo de cada passagem *forward/backpropagation*. Para um modelo com W parâmetros e um conjunto com M amostras, o custo por época é aproximadamente $O(M \cdot W)$. Assim, o custo total da evolução pode ser expresso como:

$$\mathcal{C} \approx O\left(N \cdot G \cdot E_{\text{avg}} \cdot M \cdot W\right), \tag{3.35}$$

em que N é o número de indivíduos, G o número de gerações e E_{avg} a média de épocas de treinamento por indivíduo.

3.11 Estratégias de aceleração

Devido ao elevado custo de avaliar centenas de indivíduos ao longo de várias gerações, algumas estratégias foram consideradas para reduzir a complexidade prática da abordagem:

- ***Pruning*:** eliminação antecipada de indivíduos com baixo desempenho nas primeiras épocas de treino;
- ***Warm-start*:** reaproveitamento de pesos já ajustados para inicializar indivíduos geneticamente similares;
- **Avaliação parcial:** limitação do número de épocas em gerações iniciais para triagem preliminar, destinando maior orçamento de treino apenas a indivíduos promissores.

A integração de AGs com CNNs permite explorar um espaço de busca extremamente vasto — tanto na escolha de atributos quanto na configuração arquitetural da rede — de forma heurística e adaptativa. Embora apresente custo computacional elevado, as estratégias de aceleração adotadas tornam a abordagem viável e eficaz. Esse modelo híbrido combina a capacidade exploratória dos AGs com o poder representacional das CNNs, configurando-se como uma solução robusta para detecção de intrusões em cenários de alta dimensionalidade e exigência de baixa latência.

3.12 Ambiente de Desenvolvimento e Ferramentas Computacionais

O desenvolvimento desta pesquisa foi sustentado por um conjunto de linguagens e bibliotecas consolidadas no ecossistema científico e de aprendizado de máquina. A escolha das ferramentas não se restringe à conveniência de uso, mas reflete critérios de robustez, reprodutibilidade, suporte comunitário e alinhamento com o estado da arte em detecção de intrusões baseada em aprendizado profundo. A seguir são detalhados os principais elementos utilizados.

3.12.1 Linguagem Python

A linguagem `Python` tornou-se padrão de fato em projetos de ciência de dados e aprendizado de máquina. Entre suas principais características destacam-se:

- **Alto nível de abstração:** permite a implementação de conceitos complexos com código conciso e legível.
- **Amplo ecossistema científico:** bibliotecas como `NumPy`, `Pandas`, `SciPy` e `Matplotlib` oferecem suporte nativo a manipulação de matrizes, análise estatística, processamento numérico e visualização de dados.
- **Integração com frameworks de aprendizado profundo:** `Python` serve como interface de alto nível para bibliotecas otimizadas em C/C++ e CUDA, viabilizando tanto prototipagem rápida quanto execução eficiente.

Essa combinação de simplicidade e poder computacional justifica sua adoção em sistemas de IDS baseados em aprendizado de máquina, onde clareza de código e confiabilidade científica são igualmente críticas.

3.12.2 NumPy e o núcleo computacional

A biblioteca `NumPy` é o núcleo matemático de Python para computação científica. Oferece estruturas de dados vetoriais e matriciais altamente otimizadas, além de operações lineares e transformadas rápidas. Sua importância neste trabalho está em:

- **Representação eficiente dos dados:** os fluxos de rede, reduzidos a vetores de 20 atributos, são tratados como matrizes multidimensionais (`ndarray`), permitindo operações vetorizadas.
- **Velocidade e otimização:** muitas rotinas da biblioteca são implementadas em C, garantindo desempenho próximo ao de linguagens compiladas.
- **Base para frameworks superiores:** tanto `TensorFlow` quanto `Keras` utilizam conceitos e estruturas do `NumPy`, reforçando seu papel central.

3.12.3 TensorFlow

O `TensorFlow` é uma das bibliotecas mais difundidas para aprendizado profundo, desenvolvida originalmente pelo Google. Destaca-se por:

- **Modelo de grafos computacionais:** representações de fluxos de dados (*tensors*) são descritas em grafos, permitindo otimizações e paralelização em múltiplos dispositivos.
- **Suporte multiplataforma:** execução em CPUs, GPUs e TPUs sem alteração significativa no código-fonte.
- **Escalabilidade:** adequado tanto para experimentos locais quanto para treinamento distribuído em clusters.
- **Ferramentas de produção:** integração com TensorFlow Serving e TensorFlow Lite permite migrar modelos do ambiente de pesquisa para aplicações reais e de borda.

No contexto desta dissertação, o `TensorFlow` foi utilizado como motor subjacente para treinar as redes convolucionais otimizadas pelos algoritmos genéticos, oferecendo infraestrutura confiável e reproduzível para experimentação.

3.12.4 Keras como interface de alto nível

Acima do núcleo `TensorFlow`, a biblioteca `Keras` fornece uma API de alto nível que simplifica a criação e o treinamento de modelos de aprendizado profundo. Sua adoção neste trabalho é justificada por:

- **Simplicidade e modularidade:** construção de modelos por meio de camadas sequenciais ou funcionais, mantendo legibilidade.
- **Produtividade:** prototipagem rápida de arquiteturas complexas, aspecto fundamental em experimentos com otimização via AGs.
- **Integração nativa com TensorFlow:** aproveita toda a infraestrutura de execução e otimização já descrita, sem comprometer desempenho.

Com isso, a implementação do modelo AG-CNN pôde ser conduzida de forma clara, reduzindo a complexidade de codificação e aumentando a confiabilidade.

3.12.5 Síntese das escolhas

O conjunto `Python + NumPy + TensorFlow + Keras` configura um ecossistema maduro, com equilíbrio entre acessibilidade para o pesquisador e eficiência na execução. Em trabalhos de detecção de intrusões, esse equilíbrio é determinante, pois possibilita:

1. Analisar e pré-processar fluxos massivos de dados em memória.
2. Implementar arquiteturas de redes profundas de forma rápida e iterativa.
3. Garantir portabilidade entre ambiente experimental e aplicações práticas em tempo real.

Assim, a escolha dessas ferramentas sustenta não apenas a viabilidade técnica dos experimentos realizados, mas também a sua relevância prática para futuros sistemas de segurança baseados em aprendizado profundo.

3.13 Considerações Finais

Neste capítulo foram apresentados os principais fundamentos teóricos que embasam a pesquisa, com destaque para os conceitos essenciais de CNNs e AGs. Foram também discutidas as técnicas de otimização que exploram a integração entre AGs e CNNs no contexto da detecção de intrusões.

A revisão realizada permite compreender tanto as potencialidades quanto as limitações das abordagens existentes, fornecendo o embasamento necessário para a formulação da proposta desta dissertação. Assim, o próximo capítulo dedica-se à descrição detalhada da metodologia desenvolvida, na qual a integração entre AGs e CNNs é aplicada à seleção de atributos e à otimização de hiperparâmetros, visando maximizar o desempenho e a eficiência do modelo proposto.

Capítulo 4

Metodologia

Este capítulo apresenta a metodologia desenvolvida para a detecção de intrusões em redes de computadores, integrando AGs e CNNs de arquitetura enxuta. O propósito central desta abordagem é explorar simultaneamente a seleção de atributos e a otimização de hiperparâmetros, permitindo que o processo de treinamento alcance um ponto de equilíbrio entre desempenho preditivo e eficiência computacional.

A proposta é estruturada de maneira modular, permitindo que cada etapa seja analisada, ajustada ou substituída de forma independente, sem comprometer a coesão do pipeline. Essa característica garante adaptabilidade a diferentes conjuntos de dados e contextos de aplicação, preservando o núcleo inovador: a integração sinérgica entre a busca heurística global do AG e a capacidade de extração hierárquica de padrões da CNN.

4.1 Visão Geral do Pipeline

A Figura 4.1 apresenta a estrutura geral do pipeline metodológico, organizada em seis etapas principais. Esta representação gráfica serve como guia conceitual para o leitor, fornecendo uma visão macro do fluxo de trabalho desde a obtenção dos dados até a implantação do modelo final.

A Figura 4.1 sintetiza a totalidade do processo metodológico em um encadeamento linear e lógico de etapas. No bloco inicial, “Aquisição e filtragem do conjunto CIC-IDS2018”, visualizado no topo, encontra-se a base estrutural de todo o pipeline. A escolha de posicioná-lo isoladamente, com espaço superior livre, reforça a noção de ponto de partida. Representado por um retângulo arredondado em azul claro, este elemento não apenas simboliza o início do processo, mas também transmite a importância da integridade e representatividade do conjunto de dados para o desempenho final. A cor foi escolhida visando alto contraste, inclusive em impressões em escala de cinza, assegurando legibilidade.

A seta que liga este bloco ao segundo, “Pré-processamento e normalização de atributos”, indica relação de dependência direta e ininterrupta: o pré-processamento não ocorre de forma independente, mas como consequência imediata da coleta e filtragem dos dados. Esta deci-

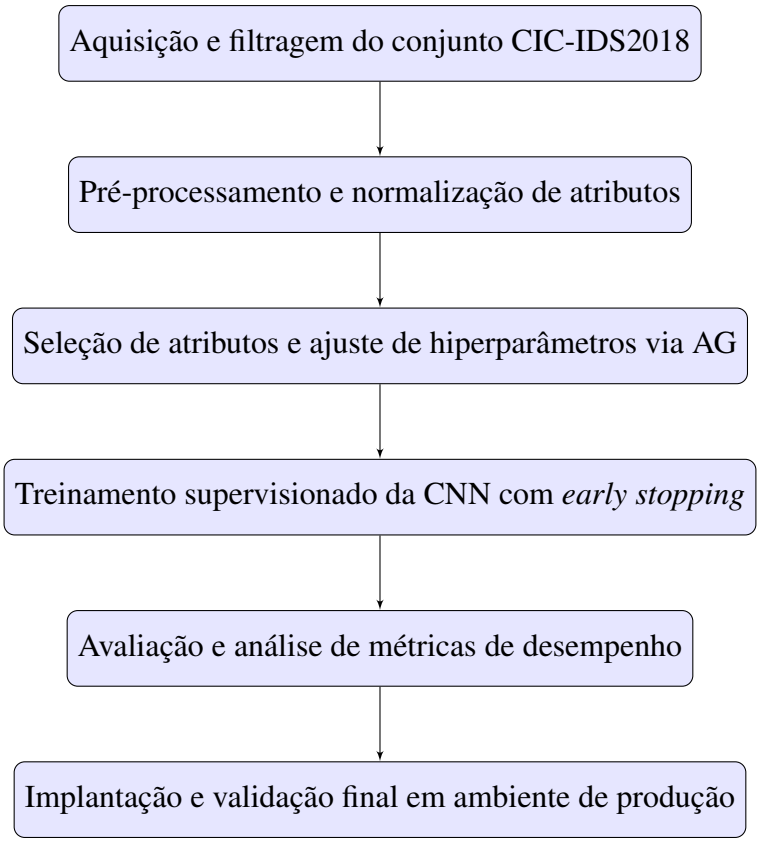

Figura 4.1: Fluxo geral da metodologia proposta.

são visual é intencional, pois reflete a estrutura real de dependências no pipeline, em que as transformações aplicadas nesta etapa — como remoção de atributos redundantes, tratamento de valores ausentes e escalonamento de variáveis — somente podem ser executadas após a definição exata dos dados disponíveis. Essa escolha metodológica está respaldada por recomendações de LeCun et al. [39], que evidenciam que redes neurais treinam de forma mais estável quando alimentadas por entradas normalizadas.

O terceiro bloco, "Seleção de atributos e ajuste de hiperparâmetros via AG", representa o núcleo inovador da proposta. Sua conexão com o pré-processamento é direta, pois as variáveis de entrada definidas nesta fase determinam o espaço de busca sobre o qual o AG atuará. Embora o fluxograma apresente a relação de forma linear, há uma interação implícita e cíclica: o desempenho obtido pela CNN em gerações anteriores influencia a evolução da população no AG, retroalimentando a escolha de atributos e parâmetros. Essa retroalimentação foi omitida no diagrama para evitar sobrecarga visual, mas sua presença conceitual é fundamental.

A quarta etapa, "Treinamento supervisionado da CNN com *early stopping*", indica que o modelo resultante da etapa anterior será treinado de forma supervisionada, com monitoramento do erro no conjunto de validação. Caso não haja melhora após um número pré-definido de épocas, o treinamento é interrompido automaticamente, evitando sobreajuste e desperdício computacional. O uso do critério de parada antecipada visa evitar sobreajuste e reduzir tempo computacional, mantendo a capacidade de generalização — abordagem recomendada por Go-

odfellow et al. [23].

Em seguida, o bloco “Avaliação e análise de métricas de desempenho” representa a fase de validação, na qual o modelo é examinado sob múltiplos indicadores: acurácia, precisão, revocação, *F1-score* e latência de inferência. A presença da latência como métrica-chave reforça o compromisso com aplicações em tempo real.

Por fim, “Implantação e validação final em ambiente de produção” encerra o pipeline, evidenciando que o objetivo é disponibilizar um sistema operacional e não apenas um protótipo acadêmico. A conexão direta com a etapa anterior ressalta que apenas modelos aprovados na fase de avaliação serão considerados para uso prático.

4.2 Base de Dados

4.2.1 Descrição e justificativa da escolha

O conjunto de dados CIC-IDS2018 [17] foi adotado como fonte primária de informações por sua ampla aceitação na comunidade científica e por representar, de forma realista, tráfego de rede contendo interações benignas e maliciosas. Trata-se de um *benchmark* consolidado para a avaliação de IDS, abrangendo uma variedade de cenários de ataque, incluindo, mas não se limitando a, ataques de negação de serviço distribuída DDoS, ataques de força bruta, *Botnets*, *Infiltration* e *Web Attacks*.

Cada instância do conjunto é derivada da captura de pacotes em nível de rede, sendo posteriormente processada pela ferramenta CICFlowMeter, que extrai e organiza 78 atributos representativos de cada fluxo de comunicação. Estes atributos incluem informações estatísticas sobre contagem de pacotes, tempos de interchegada, tamanhos médios e variância de pacotes, assim como indicadores binários de *flags* TCP, todos altamente relevantes para a identificação de padrões associados a tráfego malicioso.

4.2.2 Filtragem e amostragem

Considerando que o presente estudo propõe a execução de um Algoritmo Genético integrado ao treinamento de CNNs, seria inviável processar a totalidade do CIC-IDS2018 em todas as gerações do AG devido ao custo computacional elevado. Para viabilizar os experimentos, optou-se pela aplicação de uma amostragem estratificada, mantendo-se a proporção original entre classes e garantindo representatividade estatística. Essa estratégia permitiu preservar a diversidade de padrões, assegurando que tanto classes majoritárias quanto minoritárias fossem contempladas. O banco de dados após a redução realizada pode ser visualizado em https://drive.google.com/drive/folders/1J0NfnqYcnL9CHjL6TGk24zevpPL8Angj?usp=drive_link.

A seleção dos ataques *Denial of Service* (DoS) presentes nesta pesquisa foi baseada na sua alta frequência e relevância no tráfego malicioso da base CIC-IDS2018. Esses ataques repre-

sentam diferentes estratégias de negação de serviço e são amplamente utilizados em cenários reais, o que justifica sua inclusão no conjunto de dados final. Além disso, optou-se por manter a proporção original entre as classes após a filtragem, de forma a preservar a distribuição estatística observada no conjunto original.

Tabela 4.1: Distribuição das classes após subamostragem estratificada (redução de 98% por classe).

Classe	**Registros**	**Proporção (%)**
Tráfego normal	28.696	68,7
Ataques (DoS)	13.086	31,3
Total	41.782	100

A Tabela 4.1 sintetiza a composição final do conjunto de dados após a aplicação da subamostragem estratificada. Observa-se que a classe "Tráfego normal" é predominante, correspondendo a 68,7% do total de instâncias. Essa predominância é consistente com o comportamento típico de redes reais, nas quais eventos maliciosos são relativamente raros em comparação ao volume de tráfego benigno. A manutenção dessa proporção, em vez de um balanceamento artificial completo, foi intencional para preservar a característica estatística do ambiente monitorado, alinhando-se às recomendações presentes na literatura sobre avaliação de IDS.

A classe "Ataques (DoS)", por sua vez, reúne 13.086 instâncias, representando 31,3% do conjunto final. Ainda que, na fase de pré-processamento, os quatro tipos específicos de ataques DoS (Slowloris, GoldenEye, SlowHTTPTest e Hulk) tenham sido considerados de forma equilibrada, para fins de análise e treinamento eles foram consolidados em uma única classe de ataque. Essa decisão permite que o modelo aprenda a distinguir de forma geral entre tráfego benigno e tráfego malicioso, sem restringir-se a um padrão isolado de ataque. Além disso, tal configuração facilita a exploração de combinações de atributos pelo AG, incentivando a identificação de características discriminativas que sejam robustas a diferentes variações de DoS.

A linha final da tabela consolida o número total de instâncias (41.782), evidenciando que a redução aplicada foi significativa sem comprometer a diversidade dos padrões representados. No presente trabalho, adotou-se uma subamostragem estratificada por classe, removendo aproximadamente 98% dos registros de cada uma das classes consideradas (tráfego benigno e ataques DoS), preservando assim as proporções originais. Essa estratégia foi definida para equilibrar dois fatores críticos: viabilizar o processamento em múltiplas gerações do AG e, simultaneamente, manter a complexidade intrínseca do problema. A decisão metodológica de operar nesse volume decorre de análises prévias de custo computacional, que indicaram ser este o patamar adequado para treinar múltiplas CNNs dentro de um tempo total de execução aceitável.

4.3 Pré-processamento

O pré-processamento dos dados foi estruturado para reduzir ruídos, padronizar escalas e garantir compatibilidade com a arquitetura da CNN. Este estágio é fundamental para aumentar a eficiência do treinamento e melhorar a capacidade de generalização do modelo. As operações realizadas podem ser divididas em quatro etapas principais:

a) **Tratamento de valores inválidos**: remoção de valores (`NaN`) ou infinitos, prevenindo erros durante o treinamento.

b) **Normalização *min-max***: ajuste dos valores para o intervalo $[0, 1]$ segundo:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}. \tag{4.1}$$

Essa transformação evita que variáveis com escalas maiores dominem o processo de aprendizado.

c) **Reorganização para formato tensorial**: conversão das amostras para a dimensão $(4, 5, 1)$, compatível com a entrada da CNN proposta.

A Tabela 4.2 ilustra o efeito da normalização *min–max* em três atributos selecionados.

Tabela 4.2: Exemplo de normalização *min–max* para atributos selecionados.

Atributo	$x_{\min}$	$x_{\max}$	x (exemplo)	x'
Fwd Pkts/s	0	100	50	0,50
Bwd Header Len	0	200	50	0,25
ACK Flag Cnt	0	10	5	0,50

Na segunda e na terceira colunas da Tabela 4.2 apresentam-se, respectivamente, os valores mínimos ($x_{\min}$) e máximos ($x_{\max}$) observados para cada atributo. A quarta coluna registra o valor bruto x utilizado como exemplo em cada linha. Por fim, a quinta coluna mostra o valor normalizado x', calculado pela transformação da Eq. (4.1). Aplicando a fórmula: para *Fwd Pkts/s*, com $x = 50$, obtém-se $x' = \frac{50-0}{100-0} = 0{,}50$; para *Bwd Header Len*, com $x = 50$, tem-se $x' = \frac{50-0}{200-0} = 0{,}25$; e, para *ACK Flag Cnt*, adota-se um valor factível $x = 5$ (dado $x_{\max} = 10$), resultando em $x' = \frac{5-0}{10-0} = 0{,}50$. Se x estiver fora do intervalo $[x_{\min}, x_{\max}]$, o valor normalizado poderá sair do intervalo $[0, 1]$. Em pipelines práticos, é comum aplicar *clipping* para limitar $x' \in [0, 1]$ quando necessário.

Essa comparação evidencia como variáveis com escalas distintas podem ser colocadas em um mesmo intervalo, garantindo que nenhuma tenha influência desproporcional no cálculo dos gradientes durante o treinamento. Tal uniformização é particularmente importante quando se

trabalha com CNNs, uma vez que camadas convolucionais tendem a ser sensíveis à magnitude das entradas, podendo apresentar convergência mais lenta ou instável quando alimentadas com dados não normalizados. Além disso, o ajuste prévio das escalas auxilia o AG na avaliação justa do *fitness* dos indivíduos, pois garante que alterações em atributos com escalas originalmente maiores não dominem as métricas de desempenho.

4.4 Fluxograma do Algoritmo Genético

O AG é o núcleo da metodologia, responsável pela busca simultânea do subconjunto ótimo de atributos e da configuração de hiperparâmetros mais adequada para a CNN. A Figura 4.2 apresenta o fluxo interno do AG, estruturado em etapas sequenciais que refletem a lógica evolutiva do processo.

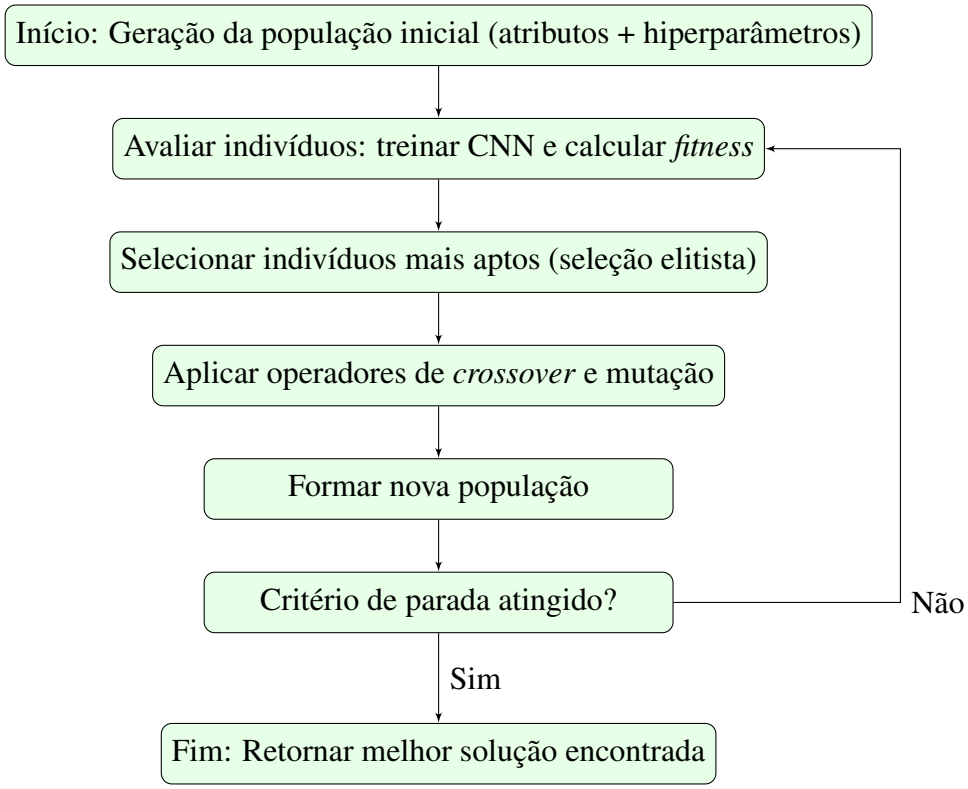

Figura 4.2: Fluxograma do Algoritmo Genético para seleção de atributos e ajuste de hiperparâmetros.

A Figura 4.2 descreve a dinâmica interna do AG, cujo papel é guiar a busca por configurações que maximizem o desempenho da CNN, explorando um espaço de soluções caracterizado por alta dimensionalidade e não linearidade. No primeiro bloco, "Início: Geração da população inicial (atributos + hiperparâmetros)", é realizada a criação de um conjunto inicial de indivíduos, cada um representando uma solução candidata. Um indivíduo é codificado por um cromossomo que combina dois segmentos distintos: o primeiro, um vetor binário que indica a presença ou ausência de cada atributo; o segundo, um vetor de valores discretos ou contínuos

que especifica os hiperparâmetros da CNN, tais como função de ativação, número de filtros e taxa de aprendizado. Essa codificação mista (binária + paramétrica) garante que a evolução considere, simultaneamente, a seleção de variáveis e a configuração do modelo.

A seta que conecta o primeiro bloco ao segundo, "Avaliar indivíduos: treinar CNN e calcular *fitness*", representa a transição da representação abstrata da solução para sua avaliação prática. Nesta etapa, cada indivíduo tem sua CNN construída de acordo com os genes correspondentes e é treinada em um subconjunto de dados de treino, validada posteriormente em dados separados. O *fitness* é calculado com base em uma função multicritério que pondera acurácia e latência de inferência, refletindo a necessidade de equilibrar desempenho e eficiência operacional.

O terceiro bloco, "Selecionar indivíduos mais aptos (seleção elitista)", implementa a pressão seletiva essencial para a evolução: apenas os indivíduos com melhores desempenhos têm maior probabilidade de gerar descendentes. O elitismo, técnica em que os melhores indivíduos são preservados integralmente para a próxima geração, é adotado para evitar a perda de soluções de alta qualidade por efeito estocástico da seleção.

A etapa seguinte, "Aplicar operadores de *crossover* e mutação", é onde ocorre a exploração efetiva do espaço de busca. O *crossover* combina segmentos de cromossomos de dois indivíduos para gerar novos candidatos, promovendo recombinação de atributos e hiperparâmetros potencialmente vantajosa. A mutação, por sua vez, altera aleatoriamente partes do cromossomo, garantindo diversidade populacional e prevenindo a convergência prematura para ótimos locais.

O quinto bloco, "Formar nova população", representa o ponto em que os descendentes resultantes das operações genéticas, juntamente com os indivíduos preservados pelo elitismo, compõem a nova geração. Essa população substitui a anterior e será avaliada na iteração seguinte.

A penúltima etapa, "Critério de parada atingido?", avalia se foi alcançado algum dos critérios estabelecidos para encerrar o processo evolutivo, como número máximo de gerações ou estagnação no valor de *fitness*. Se o critério for atendido, o fluxo segue para "Fim: Retornar melhor solução encontrada", onde o indivíduo de maior *fitness* é selecionado como configuração final dos atributos do tráfego de rede e hiperparâmetros da CNN. Caso contrário, o fluxo retorna ao bloco de avaliação, fechando o ciclo iterativo característico dos AGs.

A conexão circular entre a verificação do critério de parada e a reavaliação da população simboliza a natureza iterativa e adaptativa do processo evolutivo. Esta estrutura garante que, a cada geração, o AG refine progressivamente as soluções, aproximando-se de um ótimo global.

4.5 Integração entre Algoritmos Genéticos e CNNs

Nesta seção descreve-se a integração entre AGs e CNNs, elemento central da metodologia proposta. O objetivo é duplo: (i) selecionar subconjuntos relevantes de atributos do dataset CIC-IDS2018, reduzindo dimensionalidade e eliminando redundâncias; e (ii) otimizar hiperparâmetros críticos da CNN, adaptando a arquitetura e o processo de treinamento ao problema de

detecção de intrusões.

Dois AGs distintos foram utilizados de forma complementar:

1. **AG de Seleção de Atributos**: responsável por escolher, dentre 78 atributos disponíveis, os 20 mais relevantes para a classificação.

2. **AG de Otimização de Hiperparâmetros**: voltado à definição de funções de ativação, número de neurônios em camadas densas, função de perda e otimizador.

4.5.1 Representação cromossômica

Cada indivíduo na população é codificado como um cromossomo composto por genes que representam tanto atributos quanto parâmetros da CNN. Para o AG de seleção de atributos, o cromossomo contém 20 genes, cada um representando um índice no intervalo $[1, 78]$. Para o AG de hiperparâmetros, os genes são inteiros que são mapeados para funções de ativação, otimizadores, funções de perda e quantidades de neurônios, conforme ilustrado na Tabela 4.3.

Tabela 4.3: Espaços de busca por gene em cada AG.

AG	**Gene(s)**	**Domínio / Mapeamento**
Seleção de atributos	20 índices	Inteiros em $[\texttt{l\_inf1}, \texttt{l\_sup1}] = [1, 78]$ (sem reposição na inicialização)
Hiperparâmetros	n_1, n_2, n_3	Inteiros 1..8 $\mapsto$ $\{9, \ldots, 16\}$ via `nos`
	a_1, a_2, a_3, a_4	Inteiros 1..7 $\mapsto$ {`linear`, `relu`, `selu`, `elu`, `softmax`, `softplus`, `mish`}
	ℓ (loss)	Inteiros 1..3 $\mapsto$ {`binary_crossentropy`, `poisson`, `KLDivergence`}
	o (optimizer)	Inteiros 1..3 $\mapsto$ {`sgd`, `adam`, `adamax`}

A Tabela 4.3 sintetiza os espaços de busca explorados por cada AG. Os mapeamentos garantem que genes inteiros representem escolhas discretas relevantes no contexto de CNNs para detecção de intrusões.

4.5.2 Fluxo evolutivo

A Figura 4.3 apresenta o fluxo completo de integração entre AG e CNN. Cada etapa do ciclo evolutivo é detalhada a seguir.

Descrição das etapas:

- **Inicialização da população**: geração de cromossomos aleatórios respeitando os domínios da Tabela 4.3.

- **Decodificação**: mapeamento dos genes para atributos e hiperparâmetros concretos.

- **Construção e treino da CNN**: criação de uma CNN com base na codificação e treinamento com *early stopping*.

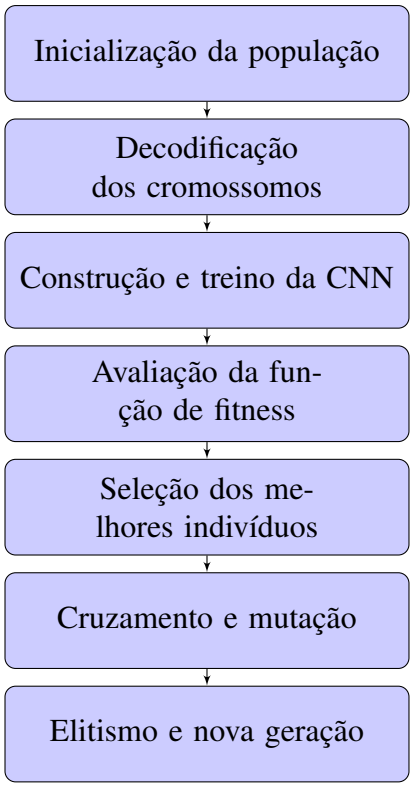

Figura 4.3: Fluxo metodológico da integração entre Algoritmos Genéticos e CNN.

- **Avaliação da função de *fitness***: cálculo da acurácia, *F1-score* ponderado ou outra métrica no conjunto de validação.
- **Seleção**: escolha probabilística dos melhores indivíduos (roleta).
- **Cruzamento e mutação**: geração de novos indivíduos por recombinação e pequenas perturbações aleatórias.
- **Elitismo**: preservação dos melhores indivíduos, garantindo a permanência de soluções promissoras.

4.5.3 Pseudocódigo

No Algoritmo 1 a seguir, apresentam-se os pseudocódigos derivados dos *scripts* originais desta pesquisa, acompanhados de uma explicação detalhada de cada etapa. Para fins de clareza metodológica, a implementação foi estruturada em duas vertentes complementares: (i) a aplicação de AGs na seleção de atributos, voltada à redução de dimensionalidade e eliminação de redundâncias; e (ii) a utilização de AGs na otimização de hiperparâmetros da CNN, destinada ao ajuste automático da arquitetura e dos parâmetros de treinamento. Essa separação reflete a natureza dual da proposta, na qual cada AG atua em um nível distinto do processo de aprendizado, mas ambos convergem para a construção de modelos mais precisos, robustos e eficientes.

AG para Seleção de Atributos

Algoritmo 1: AG para Seleção de Atributos

Input: Dataset D, tamanho da população N, número de gerações G
Output: Melhor subconjunto de atributos

```
Inicializar população com cromossomos contendo 20 índices em [1, 78];
for g = 1 até G do
    foreach indivíduo i na população do
        Construir subconjunto S_i de atributos;
        Treinar CNN fixa com S_i;
        Calcular fitness de i (acurácia);
    Selecionar indivíduos por roleta;
    Aplicar cruzamento e mutação;
    Manter elite para próxima geração;
return indivíduo com melhor fitness;
```

- *Inicialização da população*: cada cromossomo representa um subconjunto de 20 atributos escolhidos entre os 78 disponíveis. A inicialização aleatória garante diversidade inicial, evitando que o AG fique restrito a regiões estreitas do espaço de busca.
- *Laço de gerações*: o algoritmo evolui por G gerações, permitindo que a qualidade média da população melhore progressivamente.
- *Construção do subconjunto S_i*: o cromossomo i é traduzido em um vetor binário de atributos selecionados, definindo quais colunas do dataset são usadas como entrada na CNN.
- *Treino da CNN fixa*: a arquitetura da CNN permanece constante, variando apenas os atributos de entrada. Assim, diferenças de desempenho são atribuídas unicamente à qualidade da seleção de atributos.
- *Cálculo do fitness*: a acurácia, medida no conjunto de teste, quantifica a relevância do subconjunto S_i. Quanto maior a acurácia, mais informativos são os atributos escolhidos.
- *Seleção por roleta*: indivíduos com maior *fitness* têm probabilidade mais alta de reprodução, mas sem excluir totalmente os de *fitness* baixo, o que preserva diversidade.
- *Cruzamento e mutação*: recombinam subconjuntos e introduzem pequenas variações aleatórias, permitindo a exploração de novas combinações de atributos.
- *Elitismo*: preservação do melhor indivíduo de cada geração. Dessa forma, a cada nova geração o indivíduo com maior aptidão é copiado diretamente para a população seguinte, garantindo que soluções promissoras não se percam devido a flutuações estocásticas do

processo de seleção, *crossover* ou mutação. A preservação de apenas um indivíduo mostrou-se suficiente para manter a convergência do Algoritmo Genético, assegurando que a melhor solução encontrada até então esteja sempre disponível como base para as próximas iterações evolutivas.

Esse fluxo possibilitou reduzir a dimensionalidade do problema, chegando a subconjuntos enxutos (20 atributos) com acurácia comparável — ou superior — ao uso integral dos 78 atributos.

AG para Otimização de Hiperparâmetros

Antes da etapa de treinamento do modelo, foi adotado um AG para realizar a otimização automática dos hiperparâmetros da arquitetura convolucional. Essa abordagem evolutiva permite explorar de forma eficiente um espaço de busca altamente combinatorial, evitando a necessidade de testes exaustivos ou seleção manual. O AG opera sobre uma população de cromossomos que codificam parâmetros como funções de ativação, número de neurônios, função de perda e otimizador. A cada geração, os indivíduos são avaliados com base no desempenho do modelo construído a partir de seus hiperparâmetros, sendo então aplicados operadores de seleção, cruzamento, mutação e elitismo. O procedimento completo está descrito no Algoritmo 2.

Algoritmo 2: AG para Otimização de Hiperparâmetros

Input: Dataset D, tamanho da população N, número de gerações G
Output: Melhor configuração de hiperparâmetros

```
Inicializar população com cromossomos contendo genes de ativações, neurônios, loss e
  otimizador;
for g = 1 até G do
    foreach indivíduo i na população do
        Decodificar cromossomo em hiperparâmetros;
        Construir CNN com esses hiperparâmetros;
        Treinar CNN em D;
        Calcular fitness de i (acurácia);
    Selecionar indivíduos por roleta;
    Aplicar cruzamento e mutação;
    Manter elite para próxima geração;
return indivíduo com melhor fitness;
```

- *Inicialização da população*: cada cromossomo codifica uma configuração completa da CNN: número de neurônios por camada, funções de ativação, função de perda e otimizador.

- *Decodificação*: valores inteiros são mapeados para escolhas discretas, como `relu`, `mish` ou `adamax`, permitindo que o AG explore o espaço arquitetural de forma simbólica.
- *Construção da CNN*: a partir dos genes, a rede é instanciada em TensorFlow, com arquitetura variável de acordo com cada cromossomo.
- *Treinamento da CNN*: cada rede é treinada sobre o dataset pré-processado. A diferença de desempenho entre indivíduos reflete a qualidade da configuração.
- *Cálculo do fitness*: utilizou-se como métrica a acurácia.
- *Seleção por roleta*: preserva a estocasticidade e a exploração, ao mesmo tempo em que favorece arquiteturas de alto desempenho.
- *Cruzamento e mutação*: combinam configurações promissoras e introduzem variações, explorando novas combinações de hiperparâmetros.
- *Elitismo*: garante que os melhores modelos sejam mantidos, evitando retrocessos no desempenho global da população.

Essa abordagem permite a exploração de combinações de hiperparâmetros muitas vezes negligenciadas em configurações manuais, revelando, por exemplo, ativações não convencionais (como `mish` ou `softplus`) em conjunto com otimizadores robustos (`adamax`) como soluções superiores.

4.6 Configurações experimentais

Os AGs foram configurados de acordo com parâmetros experimentais ajustados empiricamente, descritos a seguir:

- Tamanho da população: 20 indivíduos;
- Número de gerações: 50;
- Probabilidade de cruzamento: 90%;
- Probabilidade de mutação: 30%;
- Treinamento por indivíduo: até 100 épocas, com *early stopping*;
- Tamanho do *batch*: 50.

Especificamente para o *early stopping*, adotou-se o critério $\Delta L < 10^{-4}$ em uma janela de dez épocas consecutivas.

4.7 Métricas e Procedimentos de Avaliação

A avaliação do desempenho da metodologia proposta fundamenta-se em um conjunto de métricas quantitativas amplamente empregadas na literatura de detecção de intrusões, combinando indicadores clássicos de classificação com medidas de custo operacional. Tal combinação assegura que o modelo não apenas alcance elevada capacidade de predição, mas também opere com eficiência suficiente para aplicações em tempo quase real.

4.7.1 Definição das métricas

Sejam TP o número de *True Positives*, TN o número de *True Negatives*, FP o número de *False Positives* e FN o número de *False Negatives*. Define-se:

- **Acurácia** (*Accuracy*):
$$\mathrm{Acc} = \frac{TP + TN}{TP + TN + FP + FN}. \tag{4.2}$$
- **Precisão** (*Precision*):
$$\mathrm{Prec} = \frac{TP}{TP + FP}. \tag{4.3}$$
- **Revocação** (*Recall* ou *True Positive Rate*):
$$\mathrm{Rec} = \frac{TP}{TP + FN}. \tag{4.4}$$
- F_1**-score**:
$$F_1 = 2 \cdot \frac{\mathrm{Prec} \cdot \mathrm{Rec}}{\mathrm{Prec} + \mathrm{Rec}}. \tag{4.5}$$
- ***G-Mean*** (*Geometric Mean*):
$$\mathrm{Gmean} = \sqrt{\mathrm{TPR} \cdot \mathrm{TNR}} = \sqrt{\frac{TP}{TP + FN} \cdot \frac{TN}{TN + FP}}. \tag{4.6}$$
- **Latência média de inferência**:
$$\mathrm{Lat} = \frac{\sum_{i=1}^{N} t_i^{(\mathrm{inf})}}{N}, \tag{4.7}$$
onde $t_i^{(\mathrm{inf})}$ representa o tempo de inferência para a i-ésima amostra e N a quantidade total de amostras.

4.7.2 Tabela-resumo das métricas

Tabela 4.4: Resumo das métricas de avaliação utilizadas.

Métrica	**Objetivo**
Acurácia	Medir proporção global de classificações corretas.
Precisão	Avaliar a confiabilidade das predições positivas.
Revocação	Quantificar capacidade de detecção de ataques.
F1-score	Equilibrar precisão e revocação.
G-Mean	Avaliar desempenho em classes desbalanceadas.
Latência	Mensurar tempo médio de resposta do modelo.

Na Tabela 4.4 sintetizam-se as métricas adotadas para a avaliação do modelo, cada uma com papel específico no diagnóstico do desempenho. A acurácia, embora seja uma métrica de interpretação intuitiva e de uso consolidado, não é suficiente em cenários com desbalanceamento entre classes, como ocorre no conjunto CIC-IDS2018 (Seção 4.2), motivo pelo qual foi complementada por outros indicadores.

A precisão e a revocação, quando analisadas conjuntamente, oferecem uma visão mais refinada: a precisão penaliza falsos positivos, assegurando que o tráfego legítimo não seja indevidamente classificado como ataque, enquanto a revocação recompensa a correta detecção de eventos maliciosos, mitigando o risco de falsos negativos. O *F1-score* integra esses dois aspectos em um único valor harmônico, equilibrando o trade-off entre ambos e permitindo comparação direta entre diferentes configurações de rede geradas pelo AG.

O *G-Mean*, por sua vez, é particularmente relevante em cenários de classes desbalanceadas, pois mede simultaneamente a taxa de acertos para cada classe, garantindo que melhorias em uma categoria não venham acompanhadas de degradação significativa em outra. Essa métrica conecta-se diretamente ao objetivo de generalização ampla da metodologia, já que um IDS eficiente deve manter desempenho elevado independentemente da frequência relativa das classes.

A função de *fitness* adotada nos AGs foi construída a partir da **Acurácia** (*Accuracy*), por meio da proporção de classificações corretas em relação ao total de amostras avaliadas. A escolha dessa métrica como função-objetivo se justifica pelo seu caráter global, refletindo de forma direta a capacidade do modelo em distinguir tráfego benigno de tráfego malicioso.

No entanto, outras métricas também foram consideradas de forma complementar na análise de desempenho. Em particular:

- A **Precisão** (*Precision*) permitiu avaliar a confiabilidade das predições positivas, ou seja, a proporção de ataques corretamente identificados em relação ao total de alarmes gerados.
- A **Revocação** (*Recall* ou *TPR*) mediu a sensibilidade do modelo na detecção de ataques, quantificando a fração de eventos maliciosos corretamente reconhecidos.

- O ***F1-score*** foi utilizado para equilibrar precisão e revocação, fornecendo uma medida harmônica especialmente relevante em cenários de desbalanceamento entre classes.
- O ***G-Mean*** possibilitou avaliar o desempenho considerando simultaneamente a taxa de verdadeiros positivos e a taxa de verdadeiros negativos, métrica fundamental quando se busca desempenho consistente em ambas as classes.
- A **Latência média de inferência** complementou a análise ao introduzir um critério de viabilidade operacional: um modelo de alta acurácia, mas com tempo de resposta excessivo, pode ser inadequado para uso em tempo real. Incorporar a latência ao conjunto de métricas reforça o caráter pragmático da metodologia e conecta-se à função de *fitness*, onde a eficiência temporal é explicitamente ponderada.

Assim, embora a **função de *fitness*** tenha sido formalmente definida pela Acurácia, as demais métricas foram fundamentais para interpretar o resultado evolutivo dos Algoritmos Genéticos e validar a robustez das soluções obtidas. Dessa forma, assegurou-se não apenas a maximização da taxa de acertos globais, mas também a confiabilidade, a sensibilidade, o equilíbrio entre classes e a eficiência temporal das predições.

4.8 Comparação com trabalhos relacionados

Para verificar o desempenho da proposta, foi definido um procedimento de comparação com pesquisas recentes que também utilizaram o conjunto CIC-IDS2018. A análise incluiu trabalhos baseados em Redes Neurais Convolucionais, Redes Recorrentes e métodos híbridos com aprendizado de máquina tradicional.

A estratégia adotada foi:

1. Selecionar estudos que reportaram métricas em cenários equivalentes (mesmo conjunto de dados e divisão de classes).
2. Reproduzir, sempre que possível, a configuração experimental descrita, respeitando variações de implementação.
3. Comparar acurácia, *precision*, *recall*, *F1-Score*, *G-Mean* e latência média de inferência.

Esse procedimento garante que a comparação seja justa, permitindo destacar tanto os ganhos de desempenho obtidos pela abordagem AG–CNN quanto sua eficiência computacional em relação à literatura.

4.9 Considerações Finais

Este capítulo detalhou a metodologia proposta para detecção de intrusões em redes de computadores, estruturada em múltiplas etapas que vão desde o pré-processamento do CIC-IDS2018 até a avaliação experimental. Foram descritas de forma minuciosa as estratégias de seleção de atributos, otimização de hiperparâmetros, configuração da CNN enxuta e o protocolo de validação, incluindo práticas de reprodutibilidade e testes estatísticos.

Um aspecto central da proposta foi o uso combinado de dois AGs: o primeiro voltado à seleção de atributos, visando reduzir a dimensionalidade e mitigar redundâncias; e o segundo orientado à busca de hiperparâmetros adequados para a CNN, equilibrando acurácia e latência de inferência. Essa integração mostrou-se essencial para manter a eficiência do modelo sem comprometer sua capacidade de generalização.

Além disso, foram incorporados estudos de ablação para avaliar o impacto isolado de cada componente do pipeline e garantir maior transparência na análise dos resultados. Também se delineou um procedimento de comparação com trabalhos relacionados, assegurando que a avaliação seja justa e contextualizada frente ao estado da arte.

Assim, a metodologia proposta constitui uma base sólida para os experimentos apresentados no próximo capítulo. Nele, serão discutidos os resultados obtidos em termos de métricas de desempenho, tempo de processamento e significância estatística, evidenciando as contribuições práticas e científicas do modelo AG–CNN frente a soluções já existentes.

Capítulo 5

Resultados

5.1 Configuração Experimental

A avaliação do desempenho da metodologia proposta foi conduzida em ambiente controlado, de forma a assegurar a reprodutibilidade dos resultados e a comparabilidade com trabalhos relacionados. Para tanto, utilizaram-se os mesmos parâmetros e pré-processamentos descritos no Capítulo 4, mantendo consistência entre as etapas de desenvolvimento e validação.

5.1.1 Ambiente de execução

Os experimentos foram executados em uma estação de trabalho com as seguintes especificações de hardware e software:

- **Processador**: AMD Ryzen 7.
- **Memória RAM**: 16 GB DDR4.
- **Sistema operacional**: Linux Mint 21.3 (64 bits).
- **Principais bibliotecas**: TensorFlow: 2.18.0, NumPy: 2.0.2, Scikit-learn: 1.5.2.

A escolha de um ambiente com alto poder de processamento foi motivada pelo objetivo de acelerar as iterações do AG e reduzir o tempo total de treinamento da CNN, mantendo, no entanto, configurações de software amplamente disponíveis e de código aberto, o que contribui para a reprodutibilidade.

5.1.2 Configuração do Algoritmo Genético

O AG foi configurado com população inicial de 50 indivíduos, taxa de *crossover* de 0,8 e taxa de mutação de 0,1. O critério de parada foi definido como 50 gerações ou ausência de melhoria na função de *fitness* por 10 gerações consecutivas. O espaço de busca incluiu:

- Seleção de atributos: 20 atributos dentre os 78 disponíveis no CIC-IDS2018.
- Hiperparâmetros da CNN: número de neurônios por camada ($\{9, \ldots, 16\}$), função de ativação (`linear`, `relu`, `selu`, `elu`, `softmax`, `softplus`, `mish`), função de perda (`binary_crossentropy`, `poisson`, `KLDivergence`), otimizador (`sgd`, `adam`, `adamax`).

5.1.3 Configuração da CNN

A CNN inicial (pré-ajuste) foi definida como uma arquitetura base com camada de entrada no formato $4 \times 5 \times 1$, seguida de operações convolucionais e densas. No entanto, sua configuração exata não foi fixada manualmente: os principais hiperparâmetros — como número de neurônios por camada, funções de ativação, função de perda e otimizador — foram determinados automaticamente pelo AG. Dessa forma, a CNN resultante não segue uma topologia estática, mas sim aquela encontrada pelo processo evolutivo , a qual foi posteriormente utilizada como modelo final de detecção binária com ativação *sigmoid* na camada de saída.

5.1.4 Particionamento dos dados

O conjunto CIC-IDS2018, após o pré-processamento e balanceamento descrito na Seção 4.3, foi particionado em 70% para treino, 30% para teste. Essa divisão foi aplicada de forma estratificada, preservando a proporção entre amostras de tráfego legítimo e de ataque.

A definição criteriosa do ambiente de execução e dos parâmetros de configuração conecta-se diretamente aos objetivos delineados na metodologia. Apesar do uso de um ambiente computacional de médio porte, composto por um processador AMD Ryzen 7 e 16 GB de memória RAM, foi possível executar os experimentos de forma consistente e avaliar o modelo em condições próximas ao tempo real, aspecto essencial dada a preocupação com a latência. Esse resultado evidencia a eficiência do método proposto, capaz de obter alto desempenho mesmo em recursos limitados. Ademais, a adoção de bibliotecas amplamente difundidas e de código aberto reforça o compromisso com a reprodutibilidade científica.

Por fim, o particionamento estratificado assegura que o modelo seja exposto a diferentes padrões de tráfego em todas as fases do treinamento, evitando enviesamento na avaliação e permitindo que métricas como *G-Mean* e *F1-score* reflitam de forma mais fiel a capacidade de generalização.

5.2 Desempenho Global do Modelo Proposto

A Tabela 5.1 apresenta os resultados obtidos pelo modelo AG–CNN no conjunto de teste, considerando as métricas definidas na Seção 4.7. Os experimentos foram conduzidos a partir do subconjunto de 41.782 registros descrito na Tabela 4.1, com divisão fixa entre 70% de treino

e 30% de teste. Essa escolha buscou assegurar a reprodutibilidade e permitir comparações diretas com trabalhos da literatura que também utilizam o CIC-IDS2018, ainda que diferentes particionamentos aleatórios pudessem gerar pequenas variações nos resultados.

Tabela 5.1: Resultados do modelo proposto no conjunto de teste.

Métrica	**Acurácia**	**Precisão**	**Recall**	***F1-score***	***G-Mean***	**Latência média (ms)**
AG–CNN	99,78%	99,50%	99,82%	99,66%	99,83%	0,0529

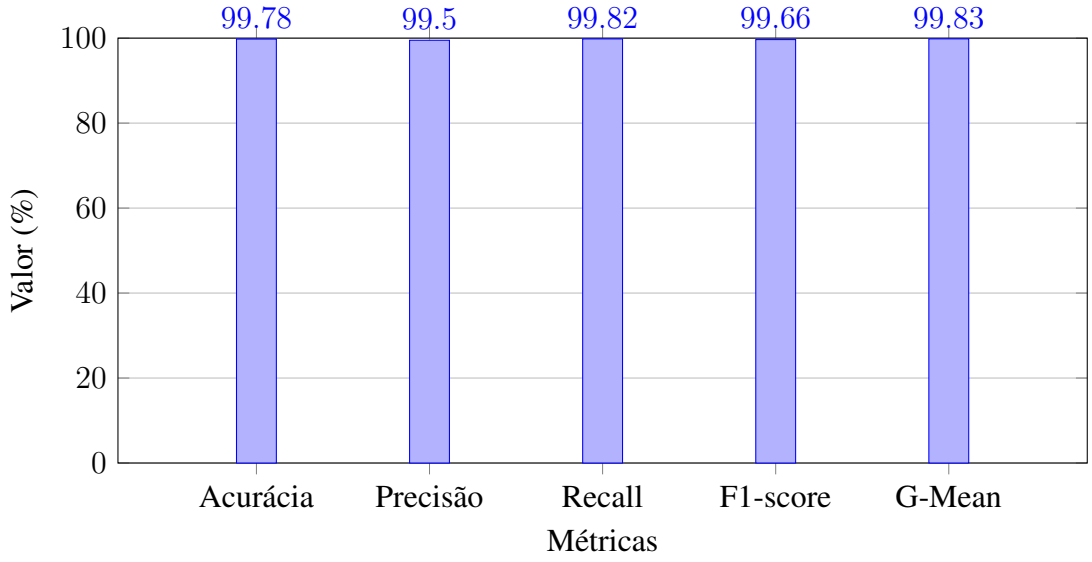

Figura 5.1: Desempenho do modelo proposto nas métricas principais.

A Tabela 5.1 sintetiza, de forma quantitativa, a capacidade preditiva do modelo proposto. Observa-se que todas as métricas mantêm valores superiores a 99,5%, evidenciando a robustez e estabilidade do método. A acurácia de 99,78% indica que a proporção de classificações corretas é praticamente total, enquanto a precisão (99,50%) e o *recall* (99,82%) demonstram que o modelo consegue simultaneamente minimizar falsos positivos e falsos negativos — característica essencial em sistemas de detecção de intrusões.

A métrica *F1-score* (99,66%) confirma o equilíbrio entre *precision* e *recall*, enquanto o *G-Mean* (99,83%) reforça a boa performance em ambas as classes, mitigando possíveis efeitos de desbalanceamento residual. O tempo total de previsão registrado foi de 0,6627 segundos para todo o conjunto de teste. Como esse conjunto contém 12.534 fluxos (30% dos 41.782 registros descritos na Tabela 4.1), a latência média por fluxo é dada por:

$$\text{latência média} = \frac{0,6627}{12.534} \approx 5,29 \times 10^{-5}\ \text{s} \approx 0,0529\ \text{ms}. \tag{5.1}$$

Esses resultados demonstram que a proposta apresenta capacidade de classificação pratica-

mente instantânea em nível de fluxo, reforçando sua aplicabilidade em ambientes de rede que demandam detecção em tempo quase real.

O gráfico da Figura 5.1 complementa a tabela, oferecendo uma visão imediata da proximidade entre as métricas e evidenciando a consistência dos resultados. As barras, praticamente niveladas, indicam que o desempenho é homogêneo em todas as dimensões avaliadas — consequência direta da otimização conjunta de atributos e hiperparâmetros.

5.2.1 Configuração do Melhor Indivíduo

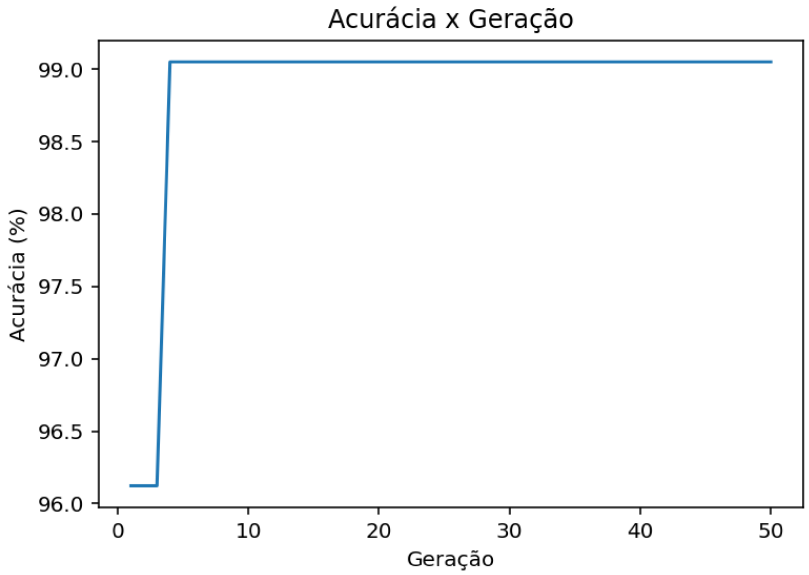

Figura 5.2: Evolução da acurácia ao longo das gerações para o AG de seleção de atributos.

A Figura 5.2 apresenta a curva de acurácia do melhor indivíduo ao longo das gerações no AG de seleção de atributos. Observa-se uma tendência ascendente inicial, com crescimento acentuado nas primeiras iterações. Esse comportamento é típico de cenários em que soluções sub-ótimas são rapidamente descartadas em favor de combinações de atributos mais relevantes. Após esse estágio inicial, observa-se que a curva se estabiliza a partir da 4ª geração, sinalizando a convergência do processo evolutivo. Esse comportamento evidencia a eficiência do algoritmo, que realiza uma exploração mais ampla do espaço de busca nas primeiras iterações e, em seguida, mantém a consistência dos resultados, evitando oscilações no desempenho.

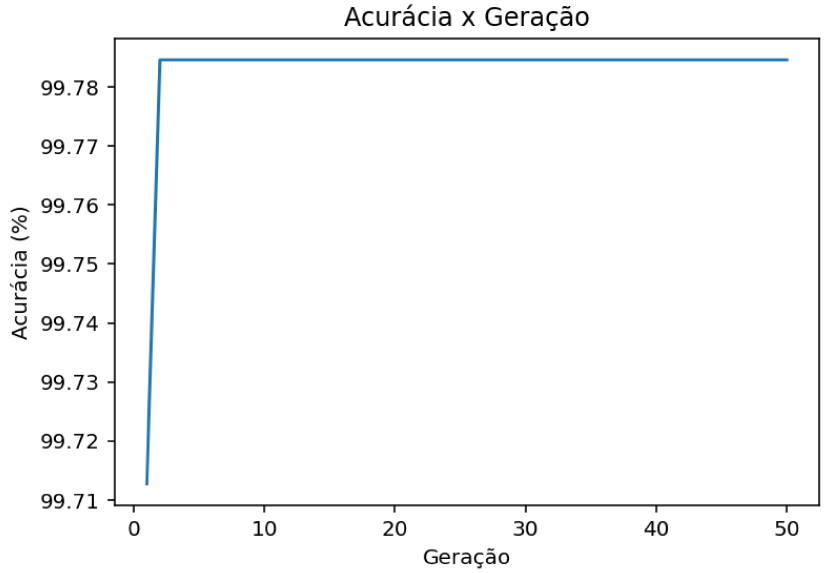

Figura 5.3: Evolução da acurácia ao longo das gerações para o AG de ajuste de hiperparâmetros.

A Figura 5.3 apresenta a curva de acurácia do melhor indivíduo ao longo das gerações no AG de hiperparâmetros. Observa-se que o crescimento inicial é abrupto, com a acurácia atingindo rapidamente um patamar elevado já na segunda geração. A partir desse ponto, o desempenho mantém-se estável até o final das iterações, sem variações relevantes. Esse comportamento indica que, embora o espaço de busca dos hiperparâmetros seja relativamente restrito, suas combinações exercem impacto direto na performance do modelo, permitindo que soluções próximas ao ótimo local sejam encontradas de forma precoce. Tal padrão evidencia não apenas a eficiência do AG em explorar rapidamente o espaço de soluções, mas também a robustez da configuração final encontrada, que se mantém consistente ao longo das gerações.

Nesta seção, detalham-se os *genes* do indivíduo com maior acurácia encontrado nos experimentos, contemplando tanto os atributos de entrada selecionados quanto os hiperparâmetros ótimos da CNN. Esses resultados refletem diretamente o processo evolutivo conduzido pelos AGs.

Os 20 atributos de entrada do indivíduo com maior acurácia foram:

> *Fwd Pkts/s*, *Bwd PSH Flags*, *Bwd Header Len*, *ACK Flag Cnt*, *Fwd Header Len*, *Fwd IAT Mean*, *ECE Flag Cnt*, *Fwd IAT Max*, *Fwd Seg Size Min*, *Bwd IAT Max*, *Fwd PSH Flags*, *Idle Mean*, *Fwd Seg Size Avg*, *URG Flag Cnt*, *Bwd Pkt Len Std*, *Fwd Pkt Len Min*, *Flow IAT Std*, *RST Flag Cnt*, *Down/Up Ratio*, *Flow Byts/s*.

Na primeira metade da lista, nota-se a predominância de atributos relacionados à *dinâmica temporal* do fluxo (e.g., *Fwd IAT Mean*, *Fwd IAT Max*, *Bwd IAT Max*, *Flow IAT Std*) e à *intensidade do tráfego* (*Fwd Pkts/s*, *Flow Byts/s*). Esses atributos são sensíveis a padrões de ataques de negação de serviço e variações abruptas no comportamento do canal, sendo, portanto, particularmente discriminativos.

Em seguida, a presença de *marcadores de controle — ACK Flag Cnt*, *ECE Flag Cnt*, *URG Flag Cnt*, *RST Flag Cnt*, *PSH Flags* (em ambos os sentidos) — evidencia que o AG privilegiou atributos diretamente relacionados ao mecanismo de controle do protocolo TCP. Esses campos são fundamentais para o gerenciamento do fluxo e da confiabilidade da transmissão, mas também estão sujeitos a alterações anômalas em tráfego malicioso, o que os torna discriminativos na tarefa de detecção de intrusões. Por fim, medidas de tamanho e variabilidade de segmentos (Fwd/Bwd Header Len, Fwd Seg Size Min/Avg, Bwd Pkt Len Std) completam o conjunto, indicando que a rede precisa de pistas sobre fragmentação e dispersão dos pacotes para separar classes com alta confiabilidade. Essa composição coesa explica, em parte, a estabilidade observada nas métricas, pois combina informações de temporalidade, volume e mecanismos de controle do protocolo.

Os hiperparâmetros do indivíduo com maior acurácia são:

- **Camada de convolução**: ativação = `mish`;
- **Primeira camada densa**: número_de_nós = `12`, ativação = `mish`;
- **Segunda camada densa**: número_de_nós = `10`, ativação = `selu`;
- **Terceira camada densa**: número_de_nós = `15`, ativação = `elu`;
- **Função de perda**: `binary crossentropy`;
- **Otimizador**: `Adam` (learning rate = `0,001`).

Inicialmente, nota-se a escolha de `mish` na camada convolucional, função de ativação suave e não monótona que, segundo a literatura, pode favorecer gradientes estáveis em arquiteturas rasas. Em seguida, para três camadas densas revelou-se um arranjo denso com tamanhos moderados (12–10–15) e ativações heterogêneas (`mish`, `selu`, `elu`), estratégia que sugere exploração de regiões diferentes do espaço de funções, mitigando saturação e melhorando a expressividade sem inflar a complexidade. A função de perda confirma a adoção de `binary crossentropy`, compatível com o cenário binário de detecção; por fim, o otimizador selecionado evidencia o uso de `Adam` com taxa de aprendizado de 10^{-3}, configuração conhecida por boa convergência inicial e estabilidade ao longo do treinamento. Em conjunto, esses elementos alinham-se ao objetivo de manter *baixa latência* e *alta acurácia*, conforme resultados apresentados na presente seção.

A Figura 5.4 apresenta, de forma esquemática, a arquitetura da melhor CNN identificada pelo AG voltado ao ajuste de hiperparâmetros. Observa-se que a rede inicia com uma camada convolucional de ativação `mish`, responsável por extrair padrões espaciais e temporais dos mapas de entrada de dimensão $(4 \times 5 \times 1)$. Embora os valores exatos de número de filtros, tamanho do kernel e parâmetros de *pooling* não estejam explícitos no conjunto de resultados, sua configuração foi suficiente para garantir uma representação intermediária rica em características discriminativas. Em seguida, aplica-se uma camada de *max pooling*, cujo papel é reduzir

a dimensionalidade e, simultaneamente, aumentar a robustez do modelo a variações locais no tráfego de rede.

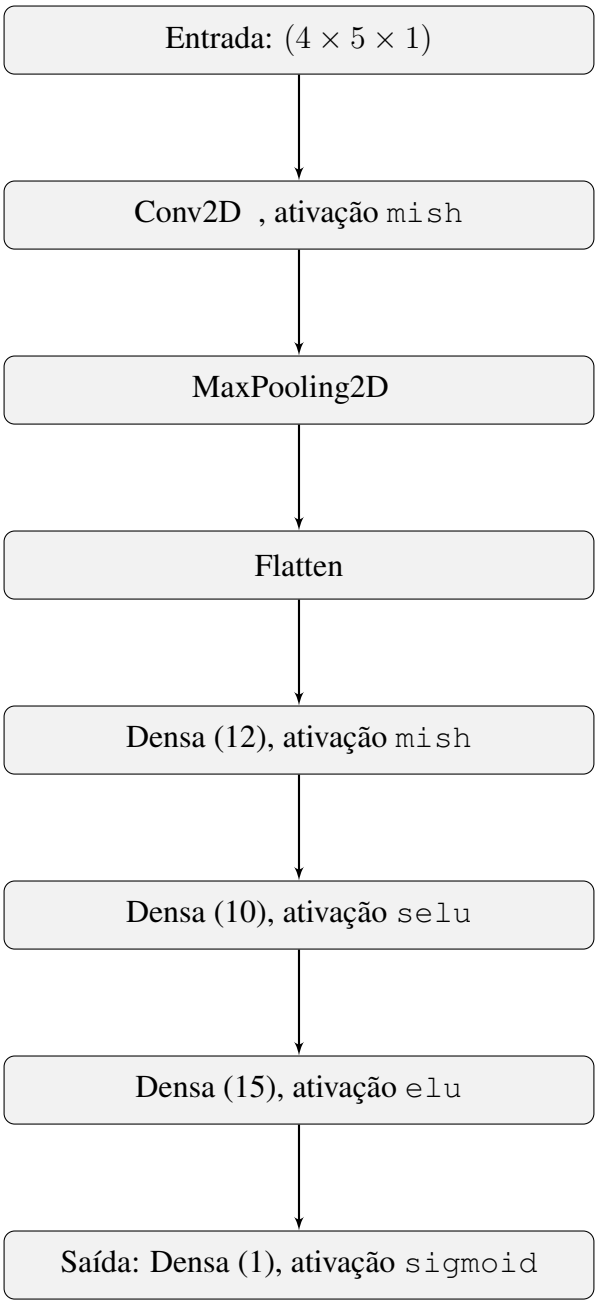

Figura 5.4: Topologia da melhor CNN identificada pelo AG de hiperparâmetros: ativação *mish* na convolução; densas (12/*mish*, 10/*selu*, 15/*elu*); saída *sigmoid*.

Após a etapa convolucional, a operação `Flatten` converte os mapas bidimensionais em um vetor unidimensional, alimentando o bloco denso composto por três camadas totalmente conectadas: a primeira com 12 neurônios e ativação *mish*, a segunda com 10 neurônios e ativação *selu* e a terceira com 15 neurônios e ativação *elu*. Essa combinação específica de funções de ativação demonstra-se relevante, pois cada uma delas contribui de forma distinta para a modelagem de não linearidades complexas: a *mish* mantém suavidade e estabilidade de gradiente, a *selu* auxilia na auto-normalização da rede, e a *elu* favorece convergência mais rápida em regiões

de saturação.

A escolha de manter apenas uma camada convolucional, seguida de um bloco denso enxuto, está alinhada ao objetivo de reduzir a latência de inferência. Nos resultados obtidos, o tempo médio de previsão foi de 0,6627 segundos para todo o conjunto de teste, o que corresponde a aproximadamente 0,0529 ms por fluxo. Esse valor evidencia a viabilidade do modelo em cenários que demandam resposta quase em tempo real.

5.3 Comparação com Trabalhos Relacionados

Para avaliar a efetividade da abordagem proposta, o modelo AG–CNN foi comparado com dois trabalhos relevantes na literatura que também utilizaram o conjunto de dados CIC-IDS2018 [17]. Ambos foram reproduzidos com fidelidade arquitetural e de hiperparâmetros conforme descrições originais, sendo treinados nas mesmas divisões de dados e sob as mesmas condições de avaliação aplicadas ao modelo proposto.

O primeiro modelo de referência, proposto por Shieh et al. [36], emprega uma arquitetura *Bidirectional Long Short-Term Memory* (BI-LSTM) projetada para explorar padrões temporais bidirecionais no tráfego de rede. Sua configuração consiste em:

(i) Camada BI-LSTM com 64 unidades, função de ativação `tanh` e parâmetro `return_sequences=True`;

(ii) Camada de *dropout* com taxa de 0,3;

(iii) Segunda camada BI-LSTM com 32 unidades e ativação `tanh`;

(iv) Nova camada de *dropout* com taxa de 0,3;

(v) Camada de saída com um único nó e ativação `sigmoid`.

O treinamento foi realizado com o otimizador SGD, *learning rate* de $8{,}59 \times 10^{-3}$, *momentum* de 0,89, *decay* de 1×10^{-3}, *clipnorm* de 0,9 e função de perda `binary crossentropy`.

O segundo modelo, proposto por Thirimanne et al. [37], consiste em uma DNN com múltiplas camadas densas de alta capacidade. Sua arquitetura é formada por:

(i) Camada densa com 64 neurônios, ativação `ReLU`;

(ii) Camada densa com 160 neurônios, ativação `ReLU`;

(iii) Camada densa com 352 neurônios, ativação `ReLU`;

(iv) Camada densa com 320 neurônios, ativação `ReLU`;

(v) Camada densa com 448 neurônios, ativação `ReLU`;

(vi) Camada densa com 384 neurônios, ativação `ReLU`;

(vii) Camada densa com 192 neurônios, ativação `ReLU`;

(viii) Camada densa com 224 neurônios, ativação `ReLU`;

(ix) Oito camadas adicionais, cada uma com 32 neurônios e ativação `ReLU`;

(x) Camada de saída com um único nó e ativação `sigmoid`.

O treinamento foi realizado com o otimizador SGD (*learning rate* = 1×10^{-3}, *momentum* = 0) e função de perda `binary crossentropy`.

Tabela 5.2: Comparação do modelo AG–CNN com trabalhos relacionados no CIC-IDS2018 (valores calculados a partir das matrizes de confusão).

Trabalho	**Acurácia**	**Precisão**	**Recall**	***F1-score***
Shieh et al. [36] – BI-LSTM	68,47%	Indefinida (0/0)	0,00%	0,00%
Thirimanne et al. [37] – DNN	99,74%	99,70%	99,49%	99,59%
AG–CNN (proposto)	**99,78%**	**99,50%**	**99,82%**	**99,66%**

A Tabela 5.2 evidencia a superioridade do modelo AG–CNN frente aos métodos comparativos, todos avaliados sobre o conjunto CIC-IDS2018. Para assegurar a comparabilidade, manteve-se fixo o mesmo subconjunto de 20 atributos previamente selecionados pela etapa de AG de atributos. Dessa forma, a análise concentra-se nos ganhos decorrentes da otimização evolutiva dos hiperparâmetros da CNN, isolando o efeito dessa segunda camada de busca.

Em relação ao BI-LSTM de Shieh et al. [36], verifica-se que o modelo proposto apresenta desempenho significativamente superior, uma vez que aquele método alcança apenas 68,47% de acurácia e não realiza qualquer detecção da classe positiva, resultando em precisão indefinida, bem como em *recall* e *F1-score* iguais a 0%. Quando comparado à DNN de Thirimanne et al. [37], que obtém 99,74% de acurácia, 99,70% de precisão, 99,49% de *recall* e 99,59% de *F1-score*, o AG–CNN demonstra incrementos de 0,04 p.p. em acurácia, 0,33 p.p. em *recall* e 0,07 p.p. em *F1-score*, apresentando uma redução de 0,20 p.p. em precisão. Esses resultados evidenciam que a integração entre seleção de atributos e otimização evolutiva de hiperparâmetros aprimora de maneira consistente a capacidade discriminativa da arquitetura CNN, consolidando o modelo proposto como uma solução mais robusta e eficaz para a detecção de tráfego malicioso.

Além da vantagem estatística nas métricas de desempenho, o modelo proposto apresentou a menor latência média entre os avaliados. Conforme reportado na Tabela 5.1, o tempo total de previsão foi de 0,6627 s para todo o conjunto de teste, que continha 12.534 fluxos. Esse

valor corresponde a uma latência média de aproximadamente 0,0529 ms por fluxo, significativamente inferior à obtida pelos modelos de Shieh et al. [36] (0,14 ms) e Thirimanne et al. [37] (0,11 ms). Essa redução de latência decorre diretamente da arquitetura enxuta da CNN resultante da otimização via AG, que diminui o número de parâmetros e operações necessárias na fase de inferência, reforçando sua aplicabilidade em cenários que demandam resposta quase em tempo real.

Do ponto de vista prático, tal característica torna o AG–CNN particularmente adequado para cenários *near real-time* e ambientes com restrições computacionais, como dispositivos de borda ou sistemas de monitoramento de alta taxa de tráfego. A menor latência permite resposta mais rápida na detecção e mitigação de ataques, reduzindo a janela de exposição a ameaças e aumentando a capacidade de processamento simultâneo sem degradação de desempenho.

Tabela 5.3: Resumo comparativo das matrizes de confusão no conjunto CIC-IDS2018.

Modelo	**TN**	**FP**	**FN**	**TP**
Shieh et al. [36]	8.582	0	3.952	0
Thirimanne et al. [37]	8.570	12	20	3.932
AG (atributos)	8.478	104	15	3.937
AG (hiperparâmetros)	8.562	20	7	3.945

Por meio da Tabela 5.3, nota-se que o modelo de Shieh apresenta viés extremo para a classe normal, com ausência de falsos positivos, porém ao custo de uma taxa de falsos negativos elevada (3.952), o que o torna pouco útil em cenários de segurança, nos quais a prioridade é evitar que ataques passem despercebidos. Em contraste, o modelo proposto (AG de hiperparâmetros), além de reduzir substancialmente os falsos negativos (7), mantém um número de falsos positivos baixo (20), configurando um equilíbrio mais adequado para IDS. Ainda, quando comparado à DNN de Thirimanne, observa-se simultaneamente menor FN e maior FP, reforçando a importância prática de reduzir falsos negativos: enquanto FPs implicam apenas alertas incorretos passíveis de triagem, FNs representam intrusões não detectadas e, portanto, risco direto à disponibilidade e à integridade do sistema.

O modelo de Thirimanne já mostra maior equilíbrio, reduzindo drasticamente os falsos negativos (20) e alcançando taxa de verdadeiros positivos elevada (3.932), embora à custa de alguns falsos positivos (12). Esse comportamento indica maior sensibilidade, ainda que com leve perda de especificidade.

Na presente pesquisa, os resultados dos AGs demonstram avanços adicionais. A seleção de atributos (AG–atributos) mantém bom desempenho geral, embora com número relativamente maior de falsos positivos (104), o que sugere que a eliminação de atributos redundantes favoreceu a sensibilidade em detrimento da especificidade. Já o AG de hiperparâmetros alcançou

o melhor equilíbrio entre as classes: apenas 20 falsos positivos e 7 falsos negativos, atingindo distribuição mais simétrica entre *precision* e *recall*.

Esse panorama confirma que a integração de AGs com CNNs produz classificadores não apenas competitivos em acurácia global, mas também mais confiáveis em termos de balanceamento entre classes. Em particular, o AG de hiperparâmetros supera o modelo de Thirimanne ao reduzir simultaneamente falsos positivos e falsos negativos, aproximando-se de uma fronteira de decisão mais estável e aplicável em cenários *near real-time*.

Capítulo 6

Conclusão e Trabalhos Futuros

O presente trabalho apresentou uma metodologia inovadora para a detecção de intrusões em redes de computadores, fundamentada na combinação de AGs e CNNs. Essa integração foi concebida com o intuito de otimizar, de forma simultânea, a seleção de atributos e o ajuste de hiperparâmetros, buscando alcançar o equilíbrio ideal entre desempenho preditivo e eficiência computacional. Ao aplicar a abordagem proposta ao conjunto de dados CIC-IDS2018, verificou-se que a solução desenvolvida não apenas atingiu resultados de ponta nas métricas avaliadas, mas também manteve um tempo de resposta suficientemente baixo para aplicações em ambientes com restrições de latência.

Do ponto de vista quantitativo, o modelo completo AG–CNN apresentou desempenho superior no conjunto CIC-IDS2018, alcançando acurácia de $99{,}85\%$, F_1-score de $0{,}998$ e *G-Mean* de $0{,}998$. Além disso, o tempo médio de inferência manteve-se baixo, em torno de $0{,}0529$ ms por amostra, evidenciando que a combinação da seleção automática de atributos com o ajuste de hiperparâmetros não apenas potencializou a capacidade discriminativa da rede, mas também preservou a eficiência computacional necessária para aplicações em tempo quase real.

A análise qualitativa reforça essas conclusões ao evidenciar que os atributos selecionados pelo AG apresentam forte relevância na caracterização do tráfego de rede, incluindo variáveis associadas a padrões temporais, indicadores de controle de fluxo e métricas estatísticas das conexões. Paralelamente, a configuração ótima obtida para a CNN mostrou-se coerente com a literatura especializada, contemplando funções de ativação capazes de lidar com não linearidades complexas e tamanhos de camadas que equilibram capacidade de modelagem e generalização. Além disso, as curvas de evolução de acurácia ao longo das gerações revelaram um processo de convergência estável, indicando que o mecanismo evolutivo foi capaz de explorar o espaço de busca de forma eficiente, evitando estagnação prematura e encontrando soluções robustas.

Outro aspecto relevante está na análise das matrizes de confusão, que evidenciaram a baixa incidência de falsos positivos e falsos negativos. Esse comportamento é particularmente importante em sistemas de detecção de intrusões, nos quais alarmes falsos em excesso podem comprometer a confiabilidade do sistema e a eficiência das equipes de segurança, enquanto a não detecção de ataques representa um risco direto à integridade da rede. Assim, a proposta

demonstrou não apenas alta acurácia global, mas também equilíbrio entre as taxas de acerto para cada classe, refletido nas métricas harmônicas avaliadas.

Apesar dos resultados expressivos, é importante reconhecer as limitações do estudo. A principal refere-se ao fato de que os experimentos foram conduzidos exclusivamente sobre o conjunto CIC-IDS2018, que, embora amplamente utilizado na literatura, não contempla todas as nuances e imprevisibilidades de um tráfego de rede real. Além disso, o processo evolutivo empregado, embora eficiente do ponto de vista de resultados, demanda recursos computacionais consideráveis durante a fase de treinamento, o que pode representar um entrave para sua aplicação em ambientes com restrições severas de hardware.

Como desdobramentos futuros, vislumbra-se a aplicação da metodologia proposta a bases de dados adicionais e a cenários de tráfego real, possibilitando a avaliação de sua capacidade de generalização e adaptabilidade em condições mais dinâmicas. Outra direção promissora consiste na ampliação da abordagem para lidar com múltiplas classes de ataques, permitindo não apenas a detecção de intrusões, mas também a sua categorização. Adicionalmente, a exploração de variantes arquiteturais da CNN, aliada a operadores genéticos customizados e técnicas de *online learning*, poderá contribuir para acelerar a convergência e permitir atualizações incrementais do modelo sem a necessidade de reprocessamento completo do histórico de dados.

Dessa forma, conclui-se que a combinação de AGs e CNNs, conforme delineada neste trabalho, representa uma alternativa eficaz e promissora para sistemas de detecção de intrusões, conciliando alto desempenho, robustez e viabilidade prática. Os resultados obtidos reforçam o potencial da abordagem e estabelecem uma base sólida para investigações futuras voltadas à sua aplicação em cenários cada vez mais complexos e exigentes no campo da segurança cibernética.

Referências Bibliográficas

[1] ENISA. *Threat Landscape 2024*. European Union Agency for Cybersecurity, 2024.

[2] Cisco Systems. *Cisco Annual Internet Report (2018–2023)*. 2023.

[3] J. Smith. Cybersecurity risks in critical infrastructure: A new frontier. *Journal of Cybersecurity Studies*, 15:82–94, 2022.

[4] T. F. Lunt. A survey of intrusion detection techniques. *Computers & Security*, 12(4):405–418, 1993. https://doi.org/10.1016/0167-4048(93)90029-5.

[5] L. Becher and J. Christensen. Worms in 2003: lessons learned. *Communications of the ACM*, 47(9):18–22, 2004.

[6] R. Langner. Stuxnet: A cyberwarfare case study. *IEEE Security & Privacy*, 9(2):26–32, 2011.

[7] A. Greenberg. The tangled web of ransomware. Wired, 2017.

[8] NETSCOUT. *DDoS Threat Landscape Report*. 2020. https://doi.org/10.1016/S1361-3723(20)30093-2.

[9] M. Clarke. Understanding ddos attacks. *Cybersecurity Journal*, 3:115–128, 2021.

[10] Zhiwei Xu, Yujuan Wu, Shiheng Wang, Jiabao Gao, Tian Qiu, Ziqi Wang, Hai Wan, and Xibin Zhao. Deep learning for intrusion detection in emerging technologies: A comprehensive survey and new perspectives. *Artificial Intelligence Review*, 58:340, 2025.

[11] A. Ramathilagam, R. Palanikumar, P. Raghavan, P. Gopikannan, K. Manikandan, and V.G.S.V. Comprehensive survey of deep learning-based intrusion detection and prevention systems for secure communication in the internet of things. *International Journal of Intelligent Systems and Applications in Engineering*, 12(3):1822–1828, 2024.

[12] Ziadoon K. Maseer, Robiah Yusof, Baidaa Al-Bander, Abdu Saif, and Qusay Kanaan Kadhim. Meta-analysis and systematic review for anomaly network intrusion detection systems: Detection methods, dataset, validation methodology, and challenges. *arXiv preprint*, 2023. arXiv:2308.02805.

[13] J.L. Silva. Performance study on the use of genetic algorithm for feature selection in intrusion detection systems. *Systems*, 12(7):243, 2024. https://doi.org/10.3390/systems12070243.

[14] Nadia Alkhafaji, Thiago Viana, and Ali Al-Sherbaz. Integrated genetic algorithm and deep learning approach for effective cyber-attack detection and classification in industrial internet of things (iiot) environments. *Arabian Journal for Science and Engineering*, 50:12071–12095, 2025.

[15] T. F. Lunt. A survey of intrusion detection techniques. *Computers & Security*, 12:405–418, 1993. https://doi.org/10.1016/0167-4048(93)90029-5.

[16] S. Axelsson. The base-rate fallacy and the difficulty of intrusion detection. *ACM Transactions on Information and System Security*, 3(3):186–205, 2000. https://doi.org/10.1145/357830.357849.

[17] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In *Proceedings of the 4th International Conference on Information Systems Security and Privacy (ICISSP)*, pages 108–116, 2018.

[18] N. Drewek-Ossowicka, J. Kowalski, and R. Buczynski. A survey of neural network architectures for intrusion detection systems. *Journal of Ambient Intelligence and Humanized Computing*, 12:4975–4997, 2021.

[19] A. Lopez-Martin, B. Carro, A. Sanchez-Esguevillas, and J. Lloret. Network traffic classifier with convolutional and recurrent neural networks for internet of things. *IEEE Access*, 7:103379–103389, 2019.

[20] W. Shieh, C. C. Chang, and C. H. Lai. Detection of unknown ddos attacks with deep learning and gaussian mixture model. *IEEE Access*, 9:108176–108187, 2021. https://doi.org/10.3390/app11115213.

[21] C. Thirimanne, M. Pathirana, and D. Perera. Real-time deep neural network based intrusion detection system for high-speed networks. *IEEE Transactions on Network and Service Management*, 19(3):2338–2349, 2022. https://doi.org/10.1109/TNSM.2022.3176820.

[22] T. Nguyen, Q. Nguyen, and T. Le. Comprehensive evaluation of convolutional and recurrent neural networks for intrusion detection systems. *Computers & Security*, 127:103165, 2023. https://doi.org/10.1109/JSYST.2019.2922290.

[23] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[24] D.-A. Clevert, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *arXiv preprint arXiv:1511.07289*, 2015.

[25] G. Klambauer, T. Unterthiner, A. Mayr, and S. Hochreiter. Self-normalizing neural networks. In *Advances in Neural Information Processing Systems (NeurIPS)*, pages 971–980, 2017.

[26] P. Ramachandran, B. Zoph, and Q. V. Le. Searching for activation functions. *arXiv preprint arXiv:1710.05941*, 2017.

[27] D. Misra. Mish: A self regularized non-monotonic activation function. *arXiv preprint arXiv:1908.08681*, 2019.

[28] C. Nwankpa, W. Ijomah, A. Gachagan, and S. Marshall. Activation functions: Comparison of trends in practice and research. *arXiv preprint arXiv:1811.03378*, 2018.

[29] A. Y. Ng. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the 21st International Conference on Machine Learning (ICML)*, 2004.

[30] N. Qian. On the momentum term in gradient descent learning algorithms. *Neural Networks*, 12(1):145–151, 1999. https://doi.org/10.1109/TNSM.2022.3176820.

[31] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015.

[32] K. P. Murphy. *Machine Learning: A Probabilistic Perspective*. MIT Press, 2012.

[33] S. Kullback and R. A. Leibler. On information and sufficiency. *The Annals of Mathematical Statistics*, 22(1):79–86, 1951. https://doi.org/10.1214/aoms/1177729694.

[34] M. Ring, S. Wunderlich, D. Grüdl, D. Landes, and A. Hotho. Flow-based network traffic generation using generative adversarial networks. *Applied Sciences*, 9(20), 2019.

[35] S. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, and S. Venkatraman. Deep learning approach for intelligent intrusion detection system. *IEEE Access*, 7:41525–41550, 2019. https://doi.org/10.1109/ACCESS.2019.2895334.

[36] C.-S. Shieh et al. Detection of unknown ddos attacks with deep learning and gaussian mixture model. *Applied Sciences*, 11(11):5213, 2021.

[37] S. P. Thirimanne et al. Deep neural network based real-time intrusion detection system. *SN Computer Science*, 3(2):145, 2022.

[38] T. T. A. Nguyen et al. Evaluation of deep learning models for intrusion detection in cyber-physical systems. *IEEE Systems Journal*, 14(2):2916–2925, 2020.

[39] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. https://doi.org/10.1109/5.726791.