ROBSON VITOR MENDONÇA

CARACTERIZAÇÃO DE TRÁFEGO DE INTRUSÕES POR MEIO DE ALGORITMO DE APRENDIZAGEM PROFUNDA

LAVRAS – MG

2021

ROBSON VITOR MENDONÇA

CARACTERIZAÇÃO DE TRÁFEGO DE INTRUSÕES POR MEIO DE ALGORITMO DE APRENDIZAGEM PROFUNDA

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores e Sistemas Embarcados, para a obtenção do título de Mestre.

Prof. DSc. Demóstenes Zegarra Rodriguez
Orientador

Prof[a]. DSc. Renata Lopes Rosa
Coorientadora

LAVRAS – MG
2021

Ficha catalográfica elaborada pelo Sistema de Geração de Ficha Catalográfica da Biblioteca Universitária da UFLA, com dados informados pelo próprio autor.

Mendonça, Robson Vitor

Caracterização de tráfego de intrusões por meio de algoritmo de aprendizagem profunda / Robson Vitor Mendonça. – Lavras : UFLA, 2021.

92 p. :

Dissertação (mestrado acadêmico)–Universidade Federal de Lavras, 2021.

Orientador: Prof. DSc. Demóstenes Zegarra Rodriguez.

Bibliografia.

1. Sistemas de Detecção de Intrusão. 2. Aprendizado de Máquina. 3. Aprendizado Profundo. I. Rodríguez, Demóstenes Zegarra. II. Rosa, Renata Lopes. III. Título.

ROBSON VITOR MENDONÇA

CARACTERIZAÇÃO DE TRÁFEGO DE INTRUSÕES POR MEIO DE ALGORITMO DE APRENDIZAGEM PROFUNDA

Dissertação apresentada à Universidade Federal de Lavras, como parte das exigências do Programa de Pós-Graduação em Ciência da Computação, área de concentração em Redes de Computadores e Sistemas Embarcados, para a obtenção do título de Mestre.

APROVADA em 16 de abril de 2021.

Profª. DSc. Katia Cilene Neles da Silva	FMF
Prof. DSc Jadson Castro Gertrudes	UFOP

Prof. DSc. Demóstenes Zegarra Rodriguez
Orientador

Profª. DSc. Renata Lopes Rosa
Co-Orientadora

LAVRAS – MG
2021

Dedico este trabalho à minha família, em especial à minha esposa Juliana e meus filhos Geovana e Eduardo, para que nunca desistam de seus sonhos.

AGRADECIMENTOS

Agradeço a Deus pelo dom da vida e por guiar meus passos. A Nossa Senhora Aparecida e Santa Rita de Cássia a quem tantas vezes pedi auxílio para superar as minhas dificuldades.

Aos meus pais Ondina (*in memorian*) e Francisco (*in memorian*) e meus irmãos e irmãs, por me ensinarem os caminhos da verdade e honestidade, e terem me feito o homem que sou hoje.

À minha esposa Juliana, por todo apoio e compreensão para que eu conseguisse terminar este trabalho. A cada dia se renova a certeza de que Deus colocou você ao meu lado, para que pudéssemos caminhar juntos e vencermos nossos desafios.

À professora Renata Rosa e ao professor Demóstenes Rodriguez, por compartilharem seus conhecimentos comigo, pela paciência e apoio em momentos difíceis, e por tudo que fizeram por mim para que eu pudesse chegar até aqui.

Aos colegas de mestrado pela convivência agradável e compartilhamento de conhecimentos, pela descontração nas cansativas viagens a Lavras.

Ao Instituto Federal de Educação, Ciência e Tecnologia do Sul de Minas Gerais - IFSULDEMINAS, que através do Programa de Incentivo à Qualificação, e afastamento parcial das atividade laborais, me proporcionou participar das atividades presenciais do mestrado.

À Universidade Federal de Lavras, em especial ao Programa de Pós Graduação em Ciência da Computação pela oportunidade de realizar esse sonho.

"Não há lugar para sabedoria onde não há paciência."
Santo Agostinho

RESUMO

Algoritmos de aprendizagem de máquina, com destaque aos de aprendizado profundo, estão sendo aplicados em diversas áreas de conhecimento, como processamento de imagem, vídeo, voz, texto e análise de tráfego de rede de computadores. As redes de computadores e os serviços ofertados aos usuários em geral tem atraído a atenção de invasores, gerando um aumento significativo de danos potenciais a tais serviços. Para resolver tal problema, Sistemas de Detecção de Intrusão, são utilizados na prevenção aos ataques. Porém, ainda há falhas na detecção, com alto índice de falsos positivos. Neste contexto, este trabalho propõe um modelo de aprendizado profundo, denominado *Tree-CNN (SRS)*, que efetua detecções de tráfegos anômalos, aumentando a acurácia na classificação de ataques *DDoS, Infiltration, Web* e *Brute force*, citados como os principais ataques a redes de computadores. Para tal, os resultados obtidos com o modelo proposto foram comparados aos resultados obtidos com os algoritmos *Naive Bayes, Random Forest, Decision Tree, Support Vector Machine, Multilayer Perceptron, Convolutional Neural Networks* e *Deep Belief Networks*, onde o modelo proposto obteve resultados superiores em todos os cenários, reduzindo assim o número de falsos positivos na classificação do tráfego de rede. Uma base de tráfego capturado em uma universidade foi utilizada para treinamento e teste do modelo *Tree-CNN (SRS)*. No tráfego capturado, analisou-se algumas características dos ataques mais comuns, usando técnicas de seleção de atributos e Análise de Componentes Principais para redução de dimensionalidade.

Palavras-chave: Sistema de Detecção de Intrusão em Redes. Aprendizado de Máquina. Aprendizado profundo.

ABSTRACT

Machine learning algorithms, especially deep learning algorithms, are being applied in several areas of knowledge, such as image processing, video, voice, text and computer network traffic analysis. Computer networks and services offered to users in general have attracted the attention of attackers, generating a significant increase in potential damage to such services. To solve this problem, Intrusion Detection Systems are used to prevent attacks. However, there are still flaws in the detection, with a high false positive index. In this context, this work proposes a Deep Learning model, called Tree-CNN (SRS), which detects anomalous traffic, increasing the accuracy in the classification of DDoS attacks, Infiltration, Web and Brute force, cited as the main attacks on computer networks. For this, the results obtained with the proposed model were compared to the results obtained with the algorithms Naive Bayes, Random Forest, Decision Tree, Support Vector Machine, Multilayer Perceptron, Convolutional Neural Networks and Deep Belief Networks, where the proposed model obtained superior results in all scenarios, thus reducing the number of false positives in the classification of network traffic. In the captured traffic, some characteristics of the most common attacks were analyzed, using attribute selection techniques and Principal Component Analysis to reduce dimensionality.

Keywords: Network Intrusion Detection System. Machine Learning. Deep learning.

LISTA DE FIGURAS

Figura 2.1 – Árvore de Decisão 24
Figura 2.2 – Random Forest 27
Figura 2.3 – Classificação por meio de SVM 30
Figura 2.4 – Modelo matemático simples do neurônio biológico 31
Figura 2.5 – Rede alimentada adiante com camada única 33
Figura 2.6 – Exemplo de RNA multicamadas típica 33
Figura 2.7 – Fluxo de processo do MLP 34
Figura 2.8 – Rede Neural Recorrente 35
Figura 2.9 – Redes Neurais *feedforward* e recorrente 36
Figura 2.10 – *Boltzmann Machine* e *Restricted Boltzmann Machine* 37
Figura 2.11 – *Restricted Boltzmann Machine* 38
Figura 2.12 – Modelo estrutural de uma DBN 40
Figura 2.13 – Arquitetura tradicional de uma CNN 42
Figura 2.14 – Arquitetura LeNet 43
Figura 2.15 – Tree-CNN 44
Figura 2.16 – Tree-CNN na classificação de tráfego de rede 46
Figura 2.17 – Processo de seleção de atributos 50
Figura 3.1 – Comparação da eficiência dos *datasets* 56
Figura 3.2 – Taxonomia IDS 59
Figura 4.1 – Metodologia de análise do tráfego 63
Figura 4.2 – Cenário 1 - Ambiente de obtenção do *dataset* 65
Figura 4.3 – Pacote do tráfego capturado 67
Figura 4.4 – Trecho do *dataset* após redução de dimensionalidade 68
Figura 4.5 – Cenário 2 - Ambiente de avaliação do modelo 71
Figura 5.1 – Comparação de acurácia entre o Tree-CNN (SRS) e DBNIDS 78

LISTA DE TABELAS

Tabela 2.1 – CIC-IDS2017 registros por arquivo 22

Tabela 2.2 – CIC-IDS2017 quantidade de registros por classificação 22

Tabela 3.1 – Comparação entre *datasets* públicos 54

Tabela 3.2 – Acurácia na detecção de ataques DDoS 57

Tabela 4.1 – Distribuição de máquinas entre departamentos 64

Tabela 4.2 – Matriz de Confusão 72

Tabela 4.3 – Matriz de Confusão - Ataque *Distributed Denial-of-Service* (DDoS) 73

Tabela 5.1 – Resultados para detecção de ataque **DDoS** em *dataset* próprio 75

Tabela 5.2 – Resultados para detecção de ataque ***Infiltration*** em *dataset* próprio 76

Tabela 5.3 – Resultados para detecção de ataque ***Web*** em *dataset* próprio 76

Tabela 5.4 – Resultados para detecção de ataque ***Brute Force*** em *dataset* próprio 76

Tabela 5.5 – Métricas do Tree-CNN (SRS) na fase de teste em *dataset* próprio 77

Tabela 5.6 – Resultados do modelo Tree-CNN (SRS) em ambiente corporativo 77

Tabela 5.7 – Acurácia obtida pelo Tree-CNN (SRS) e o DBNIDS no *dataset* CIC-IDS2017 79

Tabela 5.8 – Tempo de execução para a classificação dos ataques pelo *Deep Belief Network Intrusion Detection System* (DBNIDS) e o Tree-CNN (SRS), em fase de testes, utilizando o *dataset* CIC-IDS2017 79

LISTA DE ACRÔNIMOS

SVM	*Support Vector Machine*
RBM	*Restricted Boltzmann Machine*
MB	*Máquina de Boltzmann*
DBN	*Deep Belief Network*
RNN	*Recurrent Neural Network*
FNN	*Feed-foward Neural Network*
SNN	*Spiking Neural Network*
CNN	*Convolutional Neural Network*
DNN	*Deep Neural Network*
KNN	*K–Nearest Neighbors*
RNA	Redes Neurais Artificiais
MLP	*Multilayer Perceptron*
Adaline	*Adaptive Linear Element*
Bagging	*Bootstrap Aggregating*
NB	*Naive Bayes*
DT	*Decision Tree*
RF	*Random Forest*
IDS	*Intrusion Detection System*
IoT	*Internet of Things*
DoS	*Denial-of-Service*
DDoS	*Distributed Denial-of-Service*
DL	*Deep Learning*
PCA	*Principal Component Analysis*
BF	*Brute Force*
ICMP	*Internet Control Message Protocol*
TCP	*Transmission Control Protocol*

UDP	*User Datagram Protocol*
IP	*Internet Protocol*
R2L	*Remote-to-Local*
U2R	*User-to-Root*
IA	Inteligência Artificial
AM	Aprendizado de Máquina
RAM	*Random-access Memory*
GB	*Gigabytes*
WEKA	*Waikato Environment for Knowledge Analysis*
SRS	*Soft-Root-Sign*
Tree-CNN	*Tree Convolutional Neural Network*
MC	Matriz de Confusão
LAN	*Local Area Network*
HTTP	*Hypertext Transfer Protocol*
FTP	*File Transfer Protocol*
DBNIDS	*Deep Belief Network Intrusion Detection System*
PCAP	*Package Capture*
ARFF	*Attribute-Relation File Format*
DCNN	*Deep Convolution Neural Networks*
BP	*Back-propagation*
ReLU	*Rectified Linear Unit*
RLReLU	*Randomized Leaky Rectified Linear Activation*
CART	*Classification And Regression Tree*
LMT	*Logistic Model Tree*
OSS	*One-Sided Selection*
SMOTE	*Synthetic Minority Oversampling Technique*
BiLSTM	*Bi-directional Long Short-Term Memory*
MB	Máquina de Boltzman

VP	Verdadeiro Positivo
VN	Verdadeiro Negativo
FP	Falso Positivo
FN	Falso Negativo
Nmap	*Network Mapper*
DR	*Detection Rate*
XSS	*Cross-Site Scripting*

SUMÁRIO

1 INTRODUÇÃO 14
1.1 Objetivo 16
1.2 Objetivos específicos 16
1.3 Justificativa 16
2 REFERENCIAL TEÓRICO 18
2.1 Sistemas de Detecção de Intrusão 18
2.2 Principais tipos de ataques a redes de computadores 19
2.3 *Datasets* 20
2.4 Aprendizado de Máquina 23
2.4.1 *Decision Tree* 23
2.4.2 *Random Forest* 26
2.4.3 *Naive Bayes* 27
2.4.4 *Support Vector Machine* 28
2.5 Redes Neurais Artificiais 30
2.5.1 Redes Alimentadas Adiante com Camada Única 32
2.5.2 Redes Alimentadas Diretamente com Múltiplas Camadas 33
2.5.3 Redes Neurais Recorrentes 35
2.6 *Deep Learning* 36
2.6.1 *Restricted Boltzmann Machine* 36
2.6.2 *Deep Belief Networks* 39
2.6.3 *Convolutional Neural Network* 41
2.6.4 *Tree Convolutional Neural Network* 43
2.7 Funções de Ativação 46
2.7.1 Softmax 47
2.7.2 *Rectified Linear Unit* 47
2.7.3 *Soft-Root-Sign* 48
2.8 Métodos multivariados para redução de dados 48
2.8.1 Seleção de atributos 49
2.8.2 Agregação 51
2.8.3 Análise de Componentes Principais 51
3 TRABALHOS RELACIONADOS 53
3.1 Sistema de Detecção de Intrusão 53

3.2 ***Datasets*** 54
3.3 **Aprendizado de Máquina** 56
3.4 ***Deep Learning*** 58
3.5 **Considerações** 61
4 **METODOLOGIA** 62
4.1 **Análise do tráfego de rede** 62
4.2 **Construção do *dataset*** 63
4.2.1 **Execução do ataque** 65
4.2.2 **Captura de tráfego** 66
4.3 **Obtenção do modelo proposto** 68
4.3.1 **Implementação** 69
4.3.2 **Treinamento** 69
4.3.3 **Validação: comparação com outros modelos de Aprendizado de Máquina** 70
4.3.4 **Validação: aplicação do modelo em um ambiente corporativo** 71
4.3.5 **Validação: comparação do modelo proposto com outro modelo em DL** . 72
4.4 **Métricas de avaliação de desempenho** 72
5 **RESULTADOS** 75
5.1 **Avaliação de performance do Tree-CNN (SRS) em *dataset* próprio** . . . 75
5.2 **Avaliação do Tree-CNN (SRS) em outro cenário** 77
5.3 **Avaliação do Tree-CNN (SRS) com o modelo DBNIDS** 78
6 **CONCLUSÃO** 80
6.1 **Publicações** 81
REFERÊNCIAS 82

1 INTRODUÇÃO

O crescimento constante do número de dispositivos conectados à Internet tem atraído criminosos virtuais em busca de vulnerabilidades nas redes de computadores e sistemas. De acordo com o relatório Cisco (2019), a expectativa é que até 2023 o número de dispositivos conectados à Internet seja mais de três vezes o número de habitantes global, que segundo a ONU (2019) será de aproximadamente 8 bilhões de habitantes, logo, serão cerca de 3,6 dispositivos por pessoa.

Nesse contexto de crescimento acelerado de dispositivos conectados à Internet, a segurança é um fator importante a ser estudado e trabalhado. Portanto, a utilização de sistemas de proteção se torna indispensável.

Sistemas de Detecção de Intrusão, comumente conhecidos pelo termo em inglês *Intrusion Detection System* (IDS) (LIPPMANN et al., 2000), desempenham um papel muito importante na detecção de ataques a redes de computadores, tais como intrusões e *malwares*. Em grande parte, esses sistemas ainda trabalham somente com detecções conhecidas, ou seja, que foram identificadas anteriormente. Em ataques como do tipo *zero-day* (KIM et al., 2009), em que pode ser explorada uma falha não conhecida para um ataque do tipo *Distributed Denial-of-Service* (DDoS), o IDS baseado em assinaturas o classifica como tráfego normal. Para resolver este tipo de problema, a utilização de algoritmos de Aprendizado de Máquina (AM) está sendo estudada neste contexto, a fim de caracterizar tráfegos anômalos por meio de padrões.

Nos trabalhos de Chu, Lin e Chang (2019) e Bindra e Sood (2019), algoritmos de AM como *Decision Tree* (DT), *Naive Bayes* (NB), *Random Forest* (RF) e *Support Vector Machine* (SVM) são estudados na busca por melhor precisão na detecção de ataques por meio de IDSs. Contudo, a acurácia obtida por esses algoritmos, tal como no trabalho de Balakrishnan, K e A (2014), usando o *dataset* CSE-CIC-IDS2018 (SHARAFALDIN; LASHKARI; GHORBANI, 2018) não ultrapassa os 80% na detecção de ataques do tipo *Denial-of-Service* (DoS).

Com a evolução dos algoritmos de *Deep Learning* (DL), esta técnica também vem sendo utilizada no contexto de IDSs. De acordo com os resultados obtidos no trabalho de Yin et al. (2017), onde foi criado um modelo baseado em *Recurrent Neural Network* (RNN), observa-se que DL é muito adequado para obtenção de um modelo de classificação com alta precisão. Além disso, evidenciou-se que o desempenho de DL é superior aos métodos tradicionais de classificação em AM, como DT, RF e NB, tanto na classificação binária, quanto na classificação multi-classe.

Em Manimurugan et al. (2020), os autores desenvolveram um modelo DL baseado em *Deep Belief Network* (DBN) para classificação de ataques, denominado DBNIDS. Para os testes,

usaram o *dataset* público CIC-IDS2017, onde obtiveram resultados muito promissores, como *Brute force:* 97.71%, DoS/DDoS: 96.67%, *Infiltration:* 96.37% e *Web:* 98.37%.

Neste sentido, a busca por modelos de AM e DL que sejam mais acurados estão sempre no foco das pesquisas, principalmente para assuntos relacionados à segurança da informação, como é caso de IDSs. Assim, este trabalho propõe um modelo de classificação de ataques *DDoS, Infiltration, Web* e *Brute force*, por meio de algoritmo de DL, de forma a incrementar a acurácia nas detecções de ataques.

É importante destacar que devido à evolução dos *malwares* e, as mudanças contínuas nas estratégias de ataque, as bases de dados existentes precisam ser atualizadas periodicamente (NEHINBE, 2011), para que o treinamento de modelos em AM não sejam prejudicados.

Devido ao crescente número de atividades intrusivas, muitas pesquisas, como em Gharib et al. (2016), Ferreira e Shinoda (2016), Maciá-Fernández et al. (2018), Thakkar e Lohiya (2020), vêm sendo feitas com o intuito de encontrar um *dataset* abrangente e válido na detecção de ataques (KOCH; GOLLING; RODOSEK, 2014). Porém, obter um *dataset* adequado nas detecções é um grande desafio (NEHINBE, 2011), pois as bases de dados existentes que caracterizam possíveis ataques, em sua maioria, não refletem as tendências atuais. Há uma falta de variedade de tráfego e diversidade de ataques nas bases atuais (SHARAFALDIN; LASHKARI; GHORBANI, 2018).

Diversos trabalhos, tais como o de Heidemann e Papadopoulos (2009), Tavallaee et al. (2009), Nehinbe (2011), Shiravi, Shiravi e Ghorbani (2012) e Gharib et al. (2016), propõem um novo *dataset* para o aumento de desempenho de um IDS. De acordo com tais trabalhos, algumas características são muito importantes na construção de um *dataset* consistente e abrangente acerca de ataques a redes de computadores. Entre estas características, Gharib et al. (2016) citam que é preciso conhecer a diversidade de ataques, os protocolos utilizados pelos atacantes e a rotulagem dos mesmos.

Como parte desta pesquisa, um novo *dataset* será construído para treinamento e teste do modelo de DL proposto para detecção de ataques, e avaliação com outros modelos de AM. Este *dataset* será composto por registros obtidos a partir de uma rede real, na qual a caracterização dos ataques *DDoS, Infiltration, Web* e *Brute force* será realizada.

Para que seja obtido um *dataset* atual e objetivo, técnicas de pré-processamento são aplicadas. Como a *Principal Component Analysis* (PCA), que permite a seleção de atributos mais representativos, a fim de caracterizar os tipos de ataques já citados e formar novos modelos em tempos curtos. As bases de dados existentes, como o CIC-IDS2017, trabalham com 86

atributos, e isso causa uma maior latência na formação de um modelo de aprendizagem de máquina para caracterizar os possíveis ataques.

1.1 Objetivo

O objetivo deste trabalho é desenvolver um modelo para identificação de ataques do tipo *DDoS, Infiltration, Web* e *Brute force* a redes de computadores, por meio de técnicas de Aprendizado de Máquina (AM), aumentando os valores de acurácia, sensibilidade, precisão e medida-F em relação aos trabalhos já existentes.

1.2 Objetivos específicos

- Construir um *dataset* mais abrangente e atual, com ataques *DDoS, Infiltration, Web* e *Brute force* em um ambiente real;
- ~~Aplicar técnicas de seleção de características, a fim de reduzir a dimensionalidade do conjunto, aumentando a velocidade de identificação dos ataques;~~

1.3 Justificativa

Técnicas de detecção de ataques existentes não estão preparadas para o alto volume de tráfego atual. Portanto, cada vez mais são necessárias soluções que sejam capazes de analisar o tráfego de maneira automática, evitando gargalos na rede que ocasionam lentidão e falsos positivos nas análises.

Buscando acompanhar este crescimento, estão sendo estudadas formas de utilização de AM nessas análises, de forma que os modelos de caracterização de ataques possuam maior acurácia. Uma das técnicas de aprendizagem de máquina que tem obtido resultados superiores às técnicas existentes, são as de DL (YIN et al., 2017). Portanto, pretende-se estudar a performance de DL na detecção dos tipos de ataques *DDoS, Infiltration, Web* e *Brute force*, e criar um modelo que obtenha acurácia, sensibilidade, precisão e Medida-F superiores a algoritmos propostos atualmente, pois de acordo com Sharafaldin, Lashkari e Ghorbani (2018), os ataques *DDoS, Infiltration, Web* e *Brute force* são os tipos de ataques mais comuns.

A classificação de tráfego de redes de computadores por meio de DL é uma técnica recente. Alguns estudos, como em Shone et al. (2018), Dawoud, Shahristani e Raun (2019), Karatas, Demir e Sahingoz (2019) e Yang et al. (2019a), desenvolveram modelos em DL, porém

ainda há a necessidade de mais estudos, principalmente com *datasets* mais recentes e algoritmos mais sofisticados, criando modelos mais eficientes para essa tarefa.

Nos trabalhos de Sharafaldin, Lashkari e Ghorbani (2018) e Yang et al. (2019a), os autores mostram que a aplicação de algoritmos de DL em análise de tráfego de rede, especificamente em detecção de intrusão, tem sido feita utilizando bases estáticas, como KDDCup99 (LABORATORY, 1999), NSL-KDD (TAVALLAEE et al., 2009), CIC-IDS2017 (SHARAFALDIN; LASHKARI; GHORBANI, 2018) e CSE-CIC-IDS2018. Neste trabalho os algoritmos de DL serão aplicados a um *dataset* próprio contendo tráfego de ataques recentes, obtidos durante esta pesquisa.

Neste contexto, serão estudados alguns algoritmos de DL, e com o ajuste de hiperparâmetros do modelo proposto, pretende-se aumentar a acurácia na detecção de ataques *DDoS, Infiltration, Web* e *Brute force*. Para validação, os algoritmos serão aplicados em *dataset* próprio e públicos.

A utilização de um *dataset* próprio é necessário para reproduzir um ambiente de pequeno e médio porte, como em universidades brasileiras. Tal *dataset* foi construído, pelo fato destes tipos de ataques estudados serem mais comuns neste tipo de ambiente.

2 REFERENCIAL TEÓRICO

Neste capítulo serão apresentadas as teorias abordadas neste trabalho, bem como assuntos relacionados ao trabalho que se fazem necessários para entendimento do contexto.

2.1 Sistemas de Detecção de Intrusão

Sistemas de Detecção de Intrusão executam papel fundamental na análise de tráfego em uma rede de computadores, pois são eles que monitoram os dados que passam pela rede e aplicam algoritmos de análise em cada amostra. Este tipo de sistema não efetua bloqueio de tráfego, ele apenas analisa e dispara um alerta para outro sistema efetuar alguma ação em determinado tipo de tráfego. Uma analogia feita por Koziol (2003), entre sistemas de detecção de intrusão e alarmes contra roubos, é que ambos monitoram o ambiente em que estão inseridos e, ao detectar atividade suspeita, emitem alertas.

Segundo Rehman (2003), um sistema de detecção de intrusão é um conjunto de técnicas e métodos para detecção de atividades suspeitas na rede, e são divididos em duas principais categorias, conforme descrito a seguir.

(a) **Baseado em assinatura**: Nesta categoria o reconhecimento de ataques é limitado ao conhecimento armazenado em uma base de dados, ou em regras definidas manualmente pelo administrador de rede e segurança, ou automaticamente por sistemas do tipo *honeypots*. De acordo com Kaur e Kaur (2013), a detecção por meio desta categoria, corresponde às assinaturas de ataques já conhecidos. Tais ataques são armazenados em um banco de dados onde o IDS consulta para analisar se o tráfego é um ataque ou não. Esta categoria é similar aos *softwares* antivírus, em que há vacinas para os vírus conhecidos. Esse modelo tem um custo computacional baixo, porém, não consegue detectar ataques desconhecidos (SILVA, 2011). Uma grande desvantagem desse tipo de IDS é em relação a ataques do tipo *zero-day*, que acontecem tão logo seja divulgada uma vulnerabilidade, pegando o sistema de surpresa, pois ainda não existem assinaturas para o ataque.

(b) **Baseado em anomalia**: Nesta categoria os ataques são identificados através da análise do tráfego na rede, comparando os atributos de uma conexão e identificando conteúdos que não estejam previamente definidos por um modelo aceitável de tráfego normal. Nessa análise, tráfegos que não são previamente tratados pelo administrador da rede, são detectados como anomalias. De acordo com Warzynski e Kolaczek (2018), IDSs baseados em anomalia, procuram desvios dos comportamentos normais no tráfego de rede. Caso seja

encontrada tal situação, há o indício de um ataque em curso. Apesar da autonomia desta categoria, há a questão do maior custo computacional para mantê-lo, pois demanda de mais processamento.

Nesta pesquisa será utilizada a abordagem por anomalia, pois o objetivo é a detecção em tempo real ou aproximado e consequentemente a notificação para interrupção da conexão.

2.2 Principais tipos de ataques a redes de computadores

A seguir são elencados os perfis de ataques mais comuns. Contudo, para cada perfil desse, existe uma ou muitas variações.

DoS: O principal objetivo desse tipo de ataque, é fazer com que um serviço na rede fique lento ou indisponível, passando a não responder solicitações de usuários ou outros sistemas (REAVES; MORRIS, 2009). Nesse tipo de ataque, são exploradas vulnerabilidades dos sistemas operacionais ou algum serviço. Para conseguir o feito, o atacante realiza muitas solicitações ao serviço de rede a fim de esgotar os recursos da máquina alvo. Este tipo de ataque é facilmente detectável e bloqueável pelo *firewall*, quando é usado apenas um endereço de origem para o ataque. Porém, pode ficar complexo para a defesa quando o atacante usa uma lista de endereços para disfarçar o ataque.

DDoS: Semelhante ao DoS, este ataque tem o mesmo objetivo, porém, utiliza de sistemas distribuídos para realizar as solicitações, a fim de esgotar os recursos do alvo. Como exemplo, Honda et al. (2015) citam o protocolo *Internet Control Message Protocol* (ICMP) e as solicitações disparadas repetidamente contra um alvo, por meio do *software ping*. O atacante coordena o ataque por meio de máquinas *zombies* distribuídas. A distribuição das fontes de ataque é um estratégia de dificultar o bloqueio, pois pode ocorrer por parte do *firewall* o bloqueio de solicitações que não são referentes aos ataques. Para essa distribuição dos ataques, são usadas *botnets*, que são redes auto-espalháveis e auto-organizáveis de dispositivos comprometidos (*bots*) que podem ser usados para realizar atividades maliciosas de maneira coordenada sob o controle de um *botmaster* (PIJPKER; VRANKEN, 2017). Devido à quantidade de dispositivos desprotegidos conectados à internet, os invasores conseguem implantar sistemas em diversos computadores e, a partir desses computadores infectados, disparam ataques coordenados às suas vítimas.

Infiltration: Este tipo de ataque acontece de dentro da rede, geralmente por meio da exploração de alguma falha de *software* em um dos dispositivos da rede. Conforme explicam

Sharafaldin, Lashkari e Ghorbani (2018), após o sucesso na exploração da vulnerabilidade, é aberta uma *backdoor* para conduzir diversos ataques na rede, tais como varredura de endereços *Internet Protocol* (IP), *Port Scan* e enumeração de serviços, que podem ser realizados usando o *software* de código aberto *Network Mapper* (Nmap).

Web: Esse tipo de ataque está cada vez mais frequente. Por meio dele o atacante insere códigos maliciosos em formulários de páginas *Web* e submetem esses códigos para o servidor. A prevenção a esse tipo de ataque deve ser tomada pelo desenvolvedor da aplicação, sanitizando as entradas de dados e tratando requisições ao servidor. Alguns tipos comuns dessa classe de ataque são *SQL Injection* e *Cross-Site Scripting* (XSS). No ataque *SQL Injection* o atacante passa uma *string* que teoricamente seria uma consulta ao banco de dados da aplicação forçando o servidor a retornar um valor verdadeiro, quebrando a consulta de autenticação. No XSS, o atacante insere *scripts* na base de dados do sistema, para que posteriormente sejam executados pelo navegador de internet do cliente, dessa forma, quando o usuário acessa a aplicação o *script* é executado.

Brute force: Neste tipo de ataque o invasor tenta de forma ilegal obter credenciais de acesso a sistemas (HONDA et al., 2015). Além do roubo de credenciais, um invasor pode tentar descobrir documentos ocultos por meio de força bruta (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Devido a quantidade de tentativas, este é um dos métodos menos inteligentes de ataque, pois é facilmente detectável pelos sistemas de monitoramento de segurança.

Probing/Port scan: Este tipo de ataque tem como objetivo obter informações da rede, para que sejam utilizados posteriormente por algum outro tipo de ataque. Neste caso é executada a varredura pela rede em busca de portas *Transmission Control Protocol* (TCP) e *User Datagram Protocol* (UDP) abertas, efetuando assim o mapeamento delas para posterior exploração de vulnerabilidades de algum serviço nessas portas . Entre as técnicas de *Probing* utilizadas, estão: *UDPScan e SYNScan.*

2.3 *Datasets*

Para pesquisas em AM, os *datasets* são essenciais, pois é através deles que são treinados e testados os algoritmos. Os principais *datasets* encontrados na literatura com registros de tráfego de rede, e que são utilizados no contexto de IDSs para classificação, são brevemente descritos a seguir.

KDDCUP'99: A primeira versão do *dataset* KDDCUP foi criado em 1998 pelo laboratório *MIT Lincon.* Conforme descrito por Vinayakumar et al. (2019), na ocasião foram utilizadas

milhares de máquinas *UNIX* e, aproximadamente cem usuários acessando essas máquinas. O tráfego de rede foi capturado por dez semanas e armazenado no formato *Package Capture* (PCAP). Os ataques foram classificados em DoS (Negação de serviço), *Remote-to-Local* (R2L) (Acesso não autorizado a máquina remota), *User-to-Root* (U2R) (acesso não autorizado a privilégios de superusuário) e *Probing* (Varredura de portas e serviços). Apesar deste ser um dos *datasets* mais usados, ele possui alguns problemas que afetam os resultados dos algoritmos de AM. Segundo Tavallaee et al. (2009), o principal problema do KDDCUP'99 é o alto número de registros redundantes. Esse problema influencia no treinamento do algoritmo, pois com uma grande quantidade de registros redundantes o algoritmo é induzido ao erro, aprendendo somente com registros mais frequentes.

NSL-KDD: Este *dataset* é considerado uma evolução do KDDCUP'99. Para criar este conjunto, os autores selecionaram registros não duplicados da base KDDCUP'99. Com essa seleção, os autores conseguiram que o *dataset* pudesse representar melhor a realidade, fazendo com que o algoritmo conhecesse mais registros diferentes, assim, aprendendo padrões diferentes no momento do treinamento. McHugh (2000) detalha os problemas encontrados no *dataset* KDDCUP'99, bem como a solução implementada no *dataset* NSL-KDD.

CIC-IDS2017: Este *dataset* é considerado o mais realista desde o lançamento do KDDCUP'99 (1999) até 2017, ano em que foi criado. Além de manter um formato de arquivo da biblioteca PCAP, com informações de alto nível, o tráfego capturado reflete o mundo real de rede, pois a biblioteca consegue capturar pacotes de toda a rede, inclusive os não endereçados à máquina onde está sendo realizada a captura, ou seja, atuando em modo promíscuo. De acordo com Sharafaldin, Lashkari e Ghorbani (2018) este *dataset* contém os tipos de ataques atualizados mais comuns, representando desta forma uma rede de computadores no mundo real. A análise do tráfego registrado foi feita através do *software CICFlowMeter*, atribuindo nomes aos fluxos com base nos atributos: data e hora, endereço IP de origem e destino, porta de origem e destino, protocolos e ataques.

O *dataset* é composto por oito arquivos, sendo divididos pelos períodos de captura e tipos de ataque, conforme pode ser visto na Tabela 2.1.

Tabela 2.1 – CIC-IDS2017 registros por arquivo

Arquivo	Registros
Monday-WorkingHours.pcap_ISCX.csv	529.919
Tuesday-WorkingHours.pcap_ISCX.csv	445.910
Wednesday-workingHours.pcap_ISCX.csv	692.704
Thursday-WorkingHours-Morning-WebAttacks.pcap_ISCX.csv	458.969
Thursday-WorkingHours-Afternoon-Infilteration.pcap_ISCX.csv	288.603
Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv	286.468
Friday-WorkingHours-Morning.pcap_ISCX.csv	191.034
Friday-WorkingHours-Afternoon-DDos.pcap_ISCX.csv	225.746

Fonte: Do autor (2021)

Na Tabela 2.2 são exibidas as quantidades de registros de ataques e tráfego benigno em todo *dataset*.

Tabela 2.2 – CIC-IDS2017 quantidade de registros por classificação

Tipo	Quantidade
Benign	2.273.097
DoS Hulk	231.073
Port Scan	158.930
DDoS	128.027
DoS GoldenEye	10.293
FTP Patator	7938
SSH Patator	5897
DoS Slow Loris	5796
DoS Slow HTTP Test	5499
Botnet	1966
Web Attack: Brute Force	1507
Web Attack: XSS	652
Infiltration	36
Web Attack: SQL Injection	21
HeartBleed	11

Fonte: Do autor (2021)

CSE-CIC-IDS2018: Este *dataset* surgiu da colaboração entre duas instituições *Communications Security Establishment (CSE)* e *Canadian Institute for Cybersecurity (CIC)*. Ele foi desenvolvido para ser o mais realista possível, de forma que possam ser detectadas anomalias na rede antes que os ataques ocorram de fato. Para que esse *dataset* cumpra as necessidades mais próximas de análise de uma rede real, o tráfego é rotulado como intrusões dos tipos *Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks e Infiltration.* Este último ataque trata inclusive invasões de infiltração pelo lado de dentro da rede.

2.4 Aprendizado de Máquina

Segundo Rezende (2003), Aprendizado de Máquina (AM) é uma área de Inteligência Artificial (IA) cujo objetivo é o desenvolvimento de técnicas computacionais sobre o aprendizado, bem como a construção de sistemas capazes de adquirir conhecimento de forma automática.

Em vista do aumento de poder computacional obtido nos últimos anos, os algoritmos de AM tem ganhado espaço, pois em muitos casos estes algoritmos exigem grande capacidade de processamento e armazenamento de dados. Com esse ganho, estão sendo desenvolvidas diversas pesquisas com vistas a melhorar o desempenho destes algoritmos, tornando-os cada vez mais precisos e rápidos.

Para Lima, Pinheiro e Santos (2016), AM pode ser classificado em Aprendizado Supervisionado e Aprendizado Não Supervisionado.

(a) **Aprendizado Supervisionado**

Neste método são fornecidos modelos ao algoritmo, para que ele consiga identificar uma entrada exatamente igual ao modelo predefinido. Segundo Faceli et al. (2011) o Aprendizado Supervisionado é considerado um modelo preditivo. Neste caso, o aprendizado pode ocorrer por classificação ou regressão.

(b) **Aprendizado Não Supervisionado**

Neste caso, não é fornecido nenhum modelo. O algoritmo lê todas as entradas e classifica-as, de acordo com o nível de similaridade entre os dados encontrados.

Em AM os computadores são programados para reconhecerem padrões e realizar inferências, com base em experiências anteriores do algoritmo. Antes de realizar inferências, são necessárias coleções de dados (*datasets*) ou modelos de dados, para que o algoritmo aprenda sobre o problema.

Nesta pesquisa estão sendo aplicados os algoritmos de classificação NB, SVM, RF, *Multilayer Perceptron* (MLP), Tree-CNN (ReLU), Tree-CNN (Softmax) e Tree-CNN (SRS), portanto, faz-se necessário entender os princípios de cada um conforme é apresentado no decorrer do texto.

2.4.1 *Decision Tree*

As *Decision Tree*s (DTs) em computação são estruturas de dados compostas por elementos denominados nós. Seguindo um modelo hierárquico, a árvore possui um nó raiz, por onde a

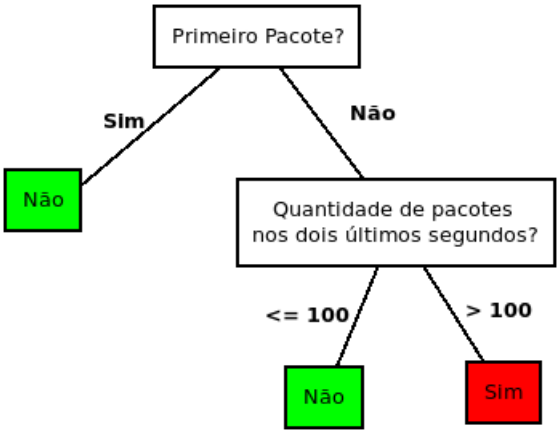
árvore é iniciada, sequencialmente a árvore é composta por nós do tipo filho, onde cada nó pode ter outros filhos ou sub-árvores. Quando um nó não tem filhos, ele é denominado nó folha ou terminal. O dado inicial entra pela raiz da árvore e passa pelos nós de decisão até chegar ao nó folha, que, por sua vez, apresenta o resultado do processamento.

Segundo Kotsiantis (2013) as DT são modelos sequenciais, que combinam logicamente uma sequência de testes simples. Cada teste compara um atributo numérico com um valor limite, ou um atributo nominal com um conjunto de valores possíveis.

De acordo com Faceli et al. (2011), a força desse modelo de classificação, está na capacidade de dividir um problema complexo em problemas mais simples, aplicando-se a mesma estratégia recursivamente até obter-se a classificação, ou seja, o resultado em um nó folha.

Conforme pode ser observado, a Figura 2.1 representa uma DT, para detecção de ataque do tipo DDoS. Com base na quantidade de pacotes por intervalos que estão entrando na rede, é possível determinar se o tráfego é um ataque distribuído de negação de serviço ou não.

Figura 2.1 – Árvore de Decisão



Fonte: Adaptado de Viegas (2016)

Como pode ser observado, a árvore é composta por condicionais do tipo *Se-Então*, que são os nós de decisão. Após percorrer o caminho na DT, o tráfego é então classificado, como ataque ou não. Apresentando, assim, a classificação no nó folha. Essa simplicidade de entendimento faz com que os algoritmos de árvore de decisão sejam amplamente estudados, pois permitem uma visão simplificada do problema.

Existem vários algoritmos de implementação de DT, tais como: ID3, C4.5, J48, *Logistic Model Tree* (LMT), *Classification And Regression Tree* (CART) e BFTree (SAHU; MEHTRE, 2015). Apesar de implementarem DT, cada algoritmo desse tem especificidades, fazendo com

que sejam melhores em determinadas tarefas. Embora existam muitos algoritmos desse tipo, Faceli et al. (2011), ressalta que os que mais se destacam são C4.5 e CART.

Segundo Rezende (2003) o critério de escolha dos atributos utilizados no particionamento de cada iteração é o ponto crucial de sucesso ou não para algoritmos de DT. De acordo com Quilici-Gonzalez e Zampirolli (2015) o critério de seleção de atributos na composição da DT, está embasado na Teoria da Informação de Claude Shannon, que foi introduzida por Quinlan (1986) em *Introducion of Decision Trees*.

Os métodos de divisão de uma DT, bem como a definição do nó raiz, é calculada de diferentes formas em cada algoritmo. Alguns dos métodos mais utilizados são: Ganho da Informação, utilizado no algoritmo C4.5 e o Índice de Gini, usado no CART (FACELI et al., 2011).

O Ganho da Informação, aplicado no algoritmo C4.5, tem como fundamento o cálculo de entropia. Este cálculo é usado para determinar a aleatoriedade do atributo alvo, ou seja, a dificuldade em classificar determinado item por conta da incerteza.

A entropia deve ser calculada para cada atributo do *dataset*. Quanto menor a entropia, mais rápida é a classificação na árvore. A seguir, na Equação 2.1 é apresentada a equação para calculo da entropia da variável A, considerando um conjunto S de dados.

$$Entropia(A) = -\sum_{i=1}^{n} p_i \ \log_2 \ p_i \tag{2.1}$$

Segundo Faceli et al. (2011) o ganho de informação é dado pela diferença entre a entropia do *dataset* e a soma ponderada da entropia de cada variável, conforme pode ser visto na Equação 2.2.

$$Entropia(A,p,q) = \sum_{i=1}^{v} \frac{p_i + q_i}{p+q} H(p_i, q_i) \tag{2.2}$$

O ganho de informação é encontrado conforme Equação 2.3.

$$GI(A,p,q) = I(p,q) - E(A,p,q) \tag{2.3}$$

As DT devem ser monitoradas para que não cresçam desordenadamente e retornem dados incorretos, aumentando a taxa de erro e prejudicando a eficiência do algoritmo. Para que isso não ocorra, são aplicadas técnicas de controle da árvore.

O superajustamento da árvore acontece, principalmente quando há dados ruidosos no *dataset* de treinamento que induz a árvore a classificações não confiáveis. O problema de classificação incorreta, ocorre principalmente em nós mais profundos. Para evitar essa situação, recomenda-se a poda da árvore. Para uma poda precisa, é necessário encontrar os nós menos relevantes e eliminá-los, deixando a árvore menor e mais inteligente (RUSSELL; NORVIG, 2016). A poda pode ser realizada antes ou depois de construído o modelo. E, na literatura o processo é conhecido como pré-poda e pós-poda.

A pré-poda impede a construção de ramos que não são bons preditores. A vantagem dessa técnica é que não se perde tempo construindo uma árvore que não gerará bons resultados, tendo de ser aplicada posteriormente a poda.

A pós-poda é a prática mais comum na construção de DT, e produz maior confiabilidade das predições. Apesar deste método ser mais lento, ele acaba sendo mais confiável (QUINLAN, 1987) .

As principais vantagens e desvantagens na utilização de DT são elencadas por (FACELI et al., 2011):

(i) **Vantagens**: Flexibilidade, Robustez, Seleção de atributos, Interpretabilidade e Eficiência;

(ii) **Desvantagens**: Replicação, Valores ausentes, atributos contínuos e Instabilidade.

2.4.2 *Random Forest*

De acordo com Breiman (2001), as *Random Forest*s (RFs) são uma combinação de *Decision Tree*s (DTs) em prol de um resultado mais preciso. É considerado um algoritmo de aprendizado supervisionado e pode ser utilizado tanto para classificação como para regressão.

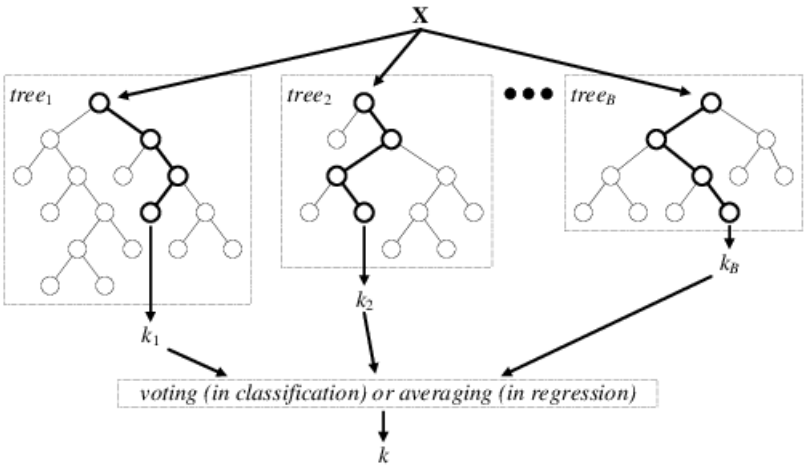
O algoritmo RF, foi criado por Breiman (2001), conforme é relatado em seu trabalho intitulado *Random Forests*. Segundo Breiman (2001), as RF são formadas por uma combinação de preditores de árvores, e cada árvore depende dos valores de um vetor amostrado aleatoriamente. A classificação final é a classe que recebeu mais votos de todas as árvores da floresta.

Neste algoritmo o classificador é baseado no método *Bootstrap Aggregating* (Bagging), criado por Breiman (1996). No método *bagging*, cada árvore gerada recebe um conjunto de treinamento diferente, formado por n instâncias de treinamento escolhidas aleatoriamente. Cada nó da árvore gerada, recebe m atributos escolhidos aleatoriamente, que orientam o direcionamento do nó, baseado na melhor discriminação de classes do conjunto de treinamento (DINIZ et al.,

2013). De acordo com Oshiro (2013), a cada nó da árvore um subconjunto de m atributos é selecionado aleatoriamente e avaliado. O atributo que demonstrar melhor resultado é escolhido para o nó.

Na Figura 2.2 pode ser observado o fluxo de uma RF, Onde o *dataset* X é distribuído para a floresta e cada árvore faz a classificação independente.

Figura 2.2 – Random Forest



Fonte: Verikas et al. (2016)

No contexto de IDS, uma amostra de tráfego é inserida no nó principal da RF e distribuído para as árvores da floresta. Cada árvore irá analisar as características do tráfego, comparando-o a fim de identificar se é um ataque ou não. Esse processo é feito por todas as árvores. No final, cada árvore da floresta apresenta seu voto, que são então computados e a classificação é dada pela maior pontuação da soma de todas as árvores.

2.4.3 *Naive Bayes*

O algoritmo classificador *Naive Bayes* (NB) é um método probabilístico de classificação baseado no teorema de Thomas Bayes. Por meio dele se calcula e analisa a relação entre cada atributo e a classe da amostra, auxiliando no método de indução (QIN et al., 2011).

O principal objetivo do algoritmo é calcular a probabilidade condicional, utilizando como base o teorema de Bayes e aplicar regras de decisão (MAIA, 2005). O NB desconsidera a correlação entre as variáveis na classificação, ou seja, mesmo que os objetos analisados tenham atributos idênticos, ele tratará cada um como único. Dessa forma, para classificar uma nova instância, o algoritmo determina a classe mais provável, dados os atributos da instância (MITCHELL, 1997).

O classificador NB tem se mostrado comparável a outros métodos de aprendizado, como exemplo as Redes Neurais Artificiaiss (RNAs) e, em alguns domínios, sendo superior à elas (MITCHELL, 1997). Contudo, uma das dificuldades práticas enfrentadas nesse método de classificação, é ter o conhecimento prévio de muitas probabilidades, o que nem sempre é possível (MITCHELL, 1997).

O cálculo realizado pelo teorema de Thomas Bayes pode ser visualizado na Equação 2.4.

$$P(A|B) = \frac{P(B|A).P(A)}{P(B)} \tag{2.4}$$

Onde, $P(A)$ é a probabilidade *a priori* de A. E, $P(A|B)$ é a probabilidade *a posteriori* do evento A, dada a ocorrência de B. Portanto, $P(B|A)$ é chamada de probabilidade condicional do evento B, dada a ocorrência do evento A, quando conhecemos A e B. Se conhecemos B mas não conhecemos A, então é chamada de verossimilhança.

O NB é considerado um dos algoritmos mais rápidos em AM, pois necessita de poucos dados para realizar o treinamento, além de demandar pouco tempo para que se tenha resultados. Estas características fazem com que em muitas situações ele seja a melhor opção.

Segundo Domingos e Pazzani (1997), o NB demonstra ter um bom desempenho em domínios com evidentes dependências entre os atributos, além de ser um classificador robusto, pois se comporta bem mesmo na presença de ruídos, ou seja, mesmo com dados incompletos ele consegue mostrar resultados satisfatórios (KONONENKO, 1991).

O classificador NB é encontrado com muita frequência em pesquisas em diversas áreas do conhecimento, como na análise de sentimentos (SARKAR, 2018), sistemas de recomendações (TROUSSAS et al., 2013) e na detecção de ataques a redes de computadores (TSAI et al., 2009; MUKHERJEE; SHARMA, 2012). Em Tsai et al. (2009) os pesquisadores observaram que levando em consideração que é possível explorar a relação estrutural entre variáveis do sistema, pode-se usar um modelo de gráfico probabilístico denominado Naive Bayesian Networks, para responder qual a probabilidade de ser um certo tipo de ataque, observados alguns eventos anteriores no sistema.

2.4.4 *Support Vector Machine*

O *Support Vector Machine* (SVM) foi apresentado por Boser, Guyon e Vapnik (1992), porém, os estudos iniciaram com Vapnik e Chervonenkis (1971). Esse algoritmo se baseia na

Teoria de Aprendizado Estatístico (TAE) para obtenção de uma boa capacidade de generalização envolvendo regras estruturais de minimização de riscos (AL-QATF et al., 2018).

As SVM são consideradas classificadores lineares, mas também podem ser vistas como classificadores multi-classe, pois quando aplicadas funções matemáticas denominadas *kernel*, obtêm-se a separação em hiperplanos, permitindo assim a separação em mais de duas classes (SCHOLKOPF; BURGES; SMOLA, 1998).

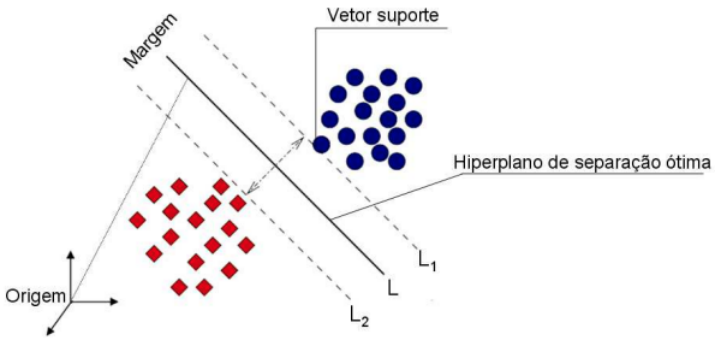
Este algoritmo é do tipo Aprendizado Supervisionado e pode ser utilizado tanto em problemas de classificação ou regressão. Ele tem como objetivo mapear e classificar um conjunto de dados num espaço multidimensional. Para atingir este objetivo, utiliza-se funções matemáticas denominadas *kernels*, que permitem a criação de hiperplanos que representem os dados em alta dimensão, aplicando o conceito de separação ótima de classes. Alguns dos *kernels* mais comuns, são: linear, polinomial, gaussiano, radial e sigmoide.

Onde não é possível realizar a separação satisfatória dos padrões do conjunto de dados, é feito o mapeamento do domínio do espaço de entrada para o espaço de características, aplicando a função *kernel* apropriada. De acordo com Takahashi (2012), a função *kernel* recebe dois pontos x_i e x_j do espaço de entrada e computa o produto escalar $\varphi^T(x_i).\varphi(x_j)$ no espaço de características.

SVMs são muito robustas e possuem grande capacidade de generalização, além de lidar muito bem com objetos de grande dimensão. Diferente de outras técnicas que geram classificadores sub ou superajustados (FACELI et al., 2011). Assim, de acordo com Faceli et al. (2011), a classificação não linear de um problema, ou seja, a convexidade, garante às SVMs resultados superiores às RNA do tipo MLP, pois tratam de um único mínimo global, diferente das RNA, onde há mínimos locais na função objetivo minimizada.

Na Figura 2.3 é representada a separação ótima entre classes (NASCIMENTO et al., 2009). Por meio de um hiperplano condicional denominado L, tal que, é orientado para maximizar a margem (a distância entre as bordas $L1$ e $L2$) pelo ponto mais próximo de cada classe.

Figura 2.3 – Classificação por meio de SVM



Fonte: Nascimento et al. (2009)

As SVMs podem apresentar grande capacidade de generalização, porém, o desempenho depende diretamente dos parâmetros de *kernel* selecionados. Caso sejam escolhidos parâmetros inadequados, pode-se obter resultados baixos de acurácia (BONESSO, 2013). Neste caso, para estabelecer o equilíbrio e evitar um modelo complexo ou erro de treinamento alto, são ajustados os valores de C. Onde C é uma constante de regularização. Quanto maior o valor de C, maior peso ao número de erros e menor peso à margem do hiperplano (OGURI, 2006).

Há diversos parâmetros que devem ser ajustados para que o SVM produza resultados equilibrados. Portanto, faz-se necessário testes e levantamento para cada aplicação do algoritmo, ajustando os parâmetros corretos para que se obtenha melhores resultados.

As SVMs tem sido estudadas por muitos pesquisadores, especialmente na detecção de intrusão de redes de computadores, como em: (JUSTIN; MARATHE; DONGRE, 2017; ARTHUR, 2018).

2.5 Redes Neurais Artificiais

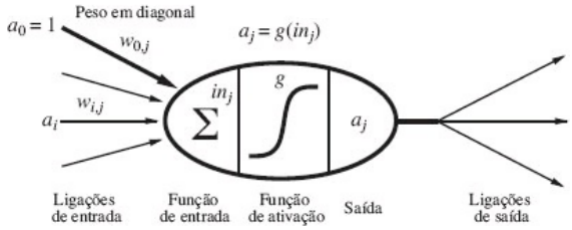
As Redes Neurais Artificiaiss (RNAs) são consideradas a forma simplificada de representação das conexões de neurônios do cérebro humano por meio de modelos matemáticos, formando uma rede onde as camadas são interconectadas simulando os neurônios biológicos, realizando os processamentos camada a camada por meio de algoritmos computacionais. Com este processo, a RNA aprende e consegue realizar inferências no futuro com base no que aprendeu no passado.

Estas são as propulsoras da tecnologia de AM utilizada atualmente. Por meio das RNA é possível simular o comportamento do cérebro humano em determinadas situações, como exemplo na tomada de decisão. Haykin (2007) explica que na sua forma mais geral, uma rede neural

é uma máquina que é projetada para modelar a maneira como o cérebro realiza uma tarefa particular ou função de interesse.

Segundo Faceli et al. (2011), os primeiros registros de RNA foram realizados por Mcculloch e Pitts (1943), ocasião em que foi construído o modelo matemático simples, conforme pode ser visto na Figura 2.4 que representa o neurônio biológico e na Equação 2.5 que representa matematicamente o cálculo de processamento do neurônio artificial.

Figura 2.4 – Modelo matemático simples do neurônio biológico



Fonte: Norvig e Russell (2014)

O modelo apresentado na Figura 2.4, pode ser expressado conforme Equação 2.5, onde, a_i é a ativação de saída da unidade i e, $w_{i,j}$ é o peso sobre a ligação da unidade i com essa unidade (RUSSELL; NORVIG, 2016).

$$a_j = g(\sum_{i=0}^{n} w_{i,j} a_i) \tag{2.5}$$

Este modelo tinha como objetivo a execução de funções lógicas simples e cada neurônio executava funções diferentes. Em seguida, Hebb (1949) contribuiu com estudos sobre o aprendizado e posteriormente Rosenblatt (1958) contribuiu com a teoria sobre os perceptrons. Apesar dos avanços, em 1970 o ritmo das pesquisas diminuiu após a publicação do livro de Minsky e Papert (1969), onde foi levantado o problema da rede tratar somente problemas linearmente separáveis. Contudo, o tema ganhou força novamente em 1980, com a chegada de computadores mais rápidos, construção de computadores paralelos e propostas de novas arquiteturas de RNA, permitindo assim a construção de redes com capacidade de resolver problemas mais complexos.

O poder de uma RNA está na paralelização maciça e distribuída do processamento, bem como na generalização, possibilitando o aprendizado da rede (HAYKIN, 2007). Por meio da decomposição de um problema complexo em tarefas simples, atribui-se à rede as tarefas que ela está capacitada a resolver. Para Ferneda (2006) a capacidade de aprendizado em exemplos,

inferência com base no aprendizado e ganho no desempenho, são as principais características de uma RNA.

Faceli et al. (2011) afirmam que uma RNA é caracterizada pelos aspectos de arquitetura e aprendizado, onde a arquitetura está ligada à quantidade de unidades de processamento, e o aprendizado está ligado às regras de ajuste dos pesos e informações que são utilizadas. A rede recebe várias entradas e o algoritmo calcula os pesos sinápticos de forma ordenada para alcançar o objetivo do projeto (BRAGA; FERREIRA; LUDERMIR, 2007). O processo de aprendizado da RNA é determinado por um algoritmo de AM, que tem como tarefa ajustar os pesos de suas conexões.

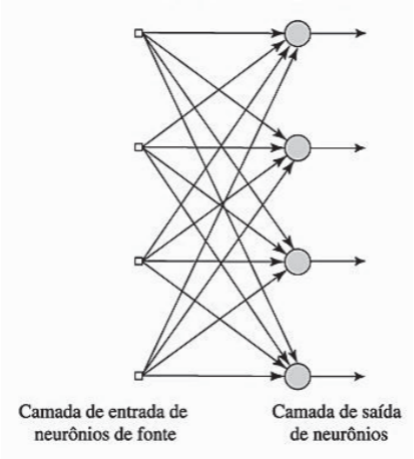
De acordo com a complexidade dos problemas a serem resolvidos pelas RNAs, foram propostas diferentes funções de ativação. Inicialmente as funções propostas por Mcculloch e Pitts (1943), retornavam apenas 0 ou 1. Essa limitação foi contornada com funções que retornam intervalos, como no caso da função Tangente Hiperbólica que assume valores entre (-1 a 1).

Assim, as RNAs podem ser classificadas arquiteturalmente, basicamente, em três tipos: Redes Alimentadas Adiante com Camada Única, Redes Alimentadas Diretamente com Múltiplas Camadas e Redes Recorrentes (HAYKIN, 2007).

2.5.1 Redes Alimentadas Adiante com Camada Única

As Redes Alimentadas Adiante com Camada Única são comumente encontradas na literatura com o nome em inglês *Single Layer Feed Forward Network*. De acordo com Haykin (2007), este tipo de rede é acíclica, ou seja, é estritamente alimentada adiante. Este é um tipo de rede com camada única, pois neste caso não há computação na camada de entrada, então, para o cálculo da quantidade camadas, leva-se em consideração somente a camada de saída. Na Figura 2.5 é ilustrada uma rede com quatro nós computacionais.

Figura 2.5 – Rede alimentada adiante com camada única



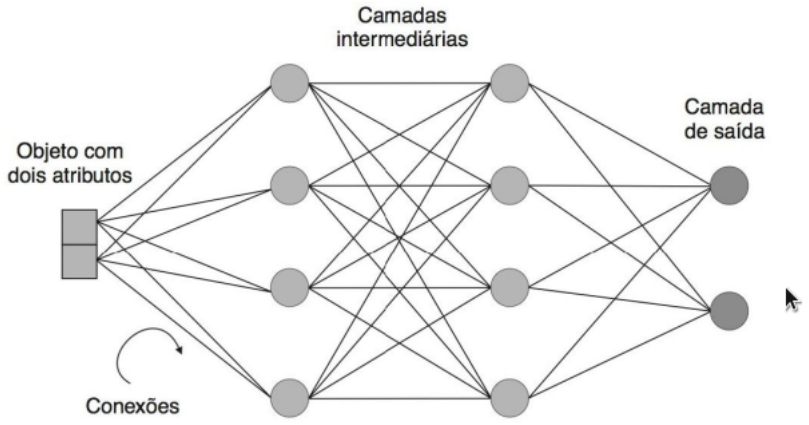
Fonte: Haykin (2007)

2.5.2 Redes Alimentadas Diretamente com Múltiplas Camadas

Este tipo de rede se distingue pela presença de camadas ocultas, cujos os nós computacionais são chamados de neurônios ocultos, e sua função é intervir entre a entrada externa e a saída da rede de uma maneira útil (HAYKIN, 2007).

Segundo Luger (2013), nesse tipo de rede, na ativação, o neurônio n passa a função de ativação para o neurônio $n+1$. Com isso, os erros no interior da rede podem se espalhar com maior facilidade e se tornarem muito complexos. A retropropagação neste caso, ajuda no equilíbrio da rede, por meio de um algoritmo que atribui aos neurônios sua parcela de culpa pelo erro, ajustando os pesos correspondentes.

Figura 2.6 – Exemplo de RNA multicamadas típica



Fonte: Faceli et al. (2011)

Dentro desta arquitetura de RNA foi desenvolvido o modelo *Multilayer Perceptron* (MLP). De acordo com Portugal e Fernandes (2013), o modelo MLP foi criado por Paul Werbos, David Parker e David Rumelhart, em 1974. E, em 1986 foi detalhado por McClelland et al. (1986) onde foi publicado no livro *Parallel Distributed Processing* um capítulo inteiro dedicado ao MLP.

Neste modelo, além das camadas de entrada e saída, ele possui uma ou mais camadas ocultas, permitindo a resolução de problemas não lineares. Diferentemente da rede *Adaptive Linear Element* (Adaline) e *Perceptron*, que trabalham com somente uma camada e não conseguem resolver problemas não lineares (FACELI et al., 2011).

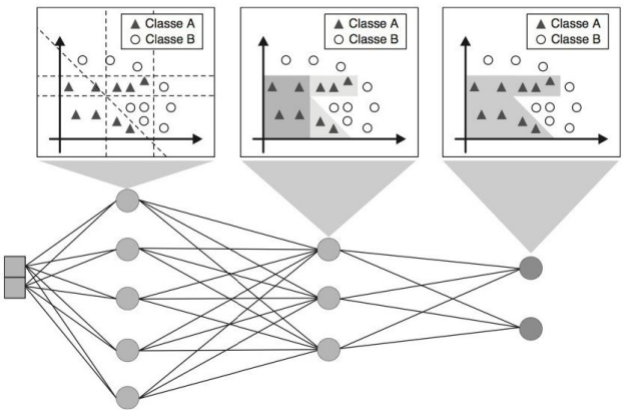
Segundo Haykin et al. (2009), o processo de treinamento deste modelo é composto por dois passos.

1. Na fase *forward*, os pesos sinápticos da rede são fixos e o sinal de entrada é propagado pela rede, camada por camada, até atingir a saída. Assim, nesta fase, as mudanças estão confinadas aos potenciais de ativação e saídas dos neurônios na rede.

2. Na fase de *backward*, um sinal de erro é produzido comparando a saída da rede com uma resposta desejada. O sinal de erro resultante é propagado pela rede, novamente camada por camada, mas desta vez a propagação é executada na direção inversa. Nesta segunda fase, ajustes sucessivos são feitos nos pesos sinápticos da rede. O cálculo dos ajustes para a camada de saída é direto, mas é muito mais desafiador para as camadas ocultas.

Na Figura 2.7 é demonstrado o fluxo de uma rede MLP, onde as camadas fazem a separação não linear das classes.

Figura 2.7 – Fluxo de processo do MLP



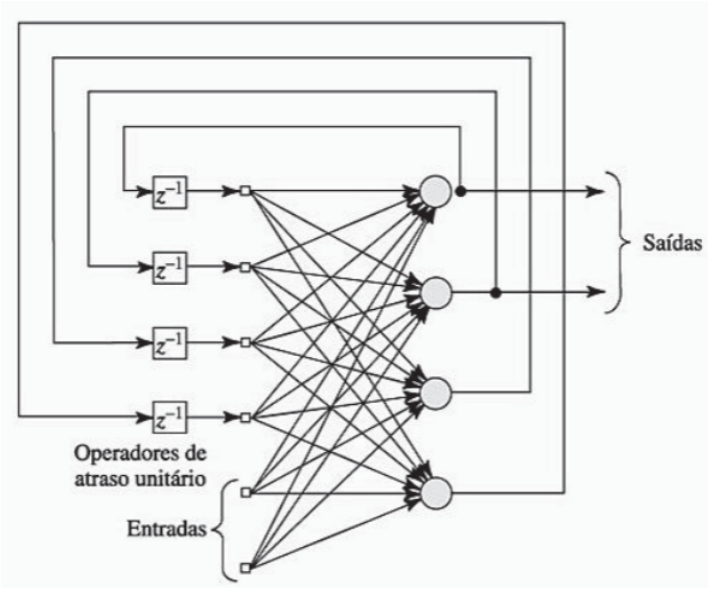
Fonte: Faceli et al. (2011)

2.5.3 Redes Neurais Recorrentes

As *Recurrent Neural Network*s (RNNs) são tipos de Redes Neurais Artificiaiss (RNAs) que possuem retroalimentação. Nesse modelo, a saída de um neurônio pode ser usada na entrada de outro neurônio da mesma camada como alimentação. De acordo com Faceli et al. (2011), as redes com retropropagação são indicadas para aplicações em que é necessário processar informações sequenciais e na simulação de sistemas dinâmicos. Uma breve descrição de como é o funcionamento das RNN, é realizada por Yin et al. (2017); segundo eles, as RNN incluem unidades de entrada, unidades de saída e unidades ocultas onde está centrado o papel mais importante da rede.

Diferente do modelo tradicional de RNA que são denominadas *Feed-foward Neural Network* (FNN) onde a saída da camada atual é a alimentação das camadas subsequentes, em RNN é aplicado um *loop* direcional onde informações produzidas anteriormente são aplicadas na saída atual, formando assim uma análise cíclica. Na Figura 2.8 é representada uma RNN com camadas ocultas.

Figura 2.8 – Rede Neural Recorrente

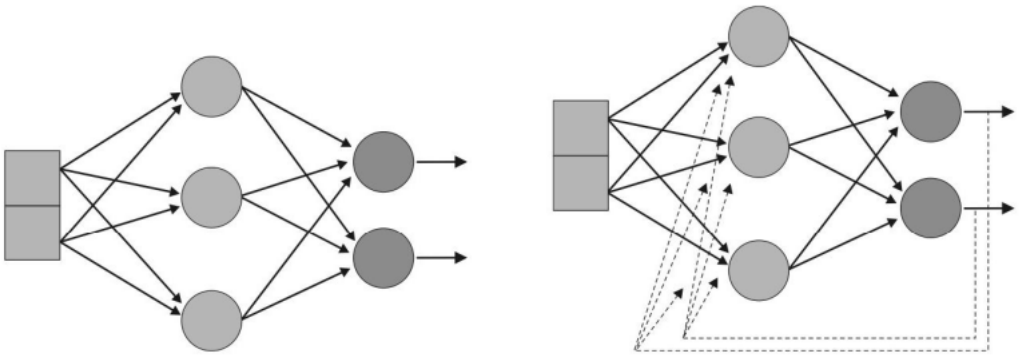


Fonte: Haykin (2007)

Na Figura 2.9 pode-se comparar os dois modelos de redes (a) *feedforward* e (b) recorrente. Pode-se observar que na figura (b), a saída dos neurônios mais adiante, servem de alimentação para os neurônios anteriores, ou seja, a realimentação. Enquanto que na rede *feedforward* a

alimentação é sempre adiante, ou seja, a saída de uma camada é conectada a outra mais adiante, e assim sucessivamente até o resultado final da rede.

Figura 2.9 – Redes Neurais *feedforward* e recorrente



Fonte: Faceli et al. (2011, p. 113)

2.6 *Deep Learning*

Deep Learning (DL) é uma área de pesquisa de Aprendizado de Máquina (AM) (ZACCONE; KARIM; MENSHAWY, 2017) e pode ser considerado como um passo a mais em AM, pois trabalha com múltiplas camadas no aprendizado (LECUN; BENGIO; HINTON, 2015). Segundo Lab (2015), essa característica remete à sua origem, que é a Inteligência Artificial (IA). Na definição de Patterson e Gibson (2017), DL é como uma rede neural com mais de duas camadas.

Em DL são criados modelos de aprendizagem em vários níveis, supervisionado ou não supervisionado, formando então um processo de aprendizado em camadas, onde as camadas inferiores se comunicam com as camadas superiores de forma não linear.

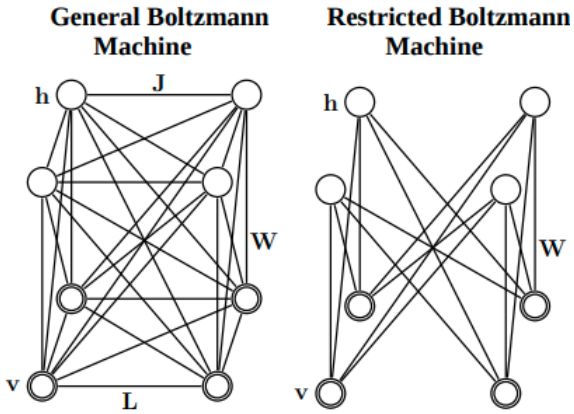
2.6.1 *Restricted Boltzmann Machine*

A *Restricted Boltzmann Machine* (RBM) é baseada na Máquina de Boltzman (MB), que foi criada em 1985 por Geoffrey Hinton e Terry Sejnowski, baseada na distribuição de Boltzmann, também conhecida como distribuição Gibbs. Em homenagem ao físico austríaco Ludwig Boltzmann, Hinton e Sejnowski, deram ao método o nome de Máquina de Boltzmann, pois é baseada numa área da Mecânica Estatística, em que se relaciona a entropia e termodinâmica, áreas de estudo de Boltzmann (HINTON; SEJNOWSKI, 1983; ACKLEY; HINTON; SEJNOWSKI, 1985).

Uma MB é considerada uma Redes Neurais Artificiais (RNA) estocástica generativa de duas camadas. As camadas são compostas por nós visíveis e ocultos, e estes nós são conectados entre si por meio de ligações bidirecionais, onde os nós visíveis representam as entradas e saídas, enquanto os nós ocultos são usados para representação interna, não realizando nenhum contato externo (FACION, 2019). Segundo Norvig e Russell (2014) as MB usam pesos simétricos e uma função de ativação estocástica, de modo que a probabilidade de a saída ser 1, é em função da entrada ponderada total. Elas passam por transições de estado, semelhante a uma pesquisa simulada de recozimento, em busca do melhor resultado com base no conjunto de treinamento.

A Figura 2.10 exibe uma comparação entre uma MB e uma RBM. Observa-se que a MB está com todos os nós interconectados, enquanto na RBM não há conexão entre camadas ocultas e visíveis.

Figura 2.10 – *Boltzmann Machine* e *Restricted Boltzmann Machine*

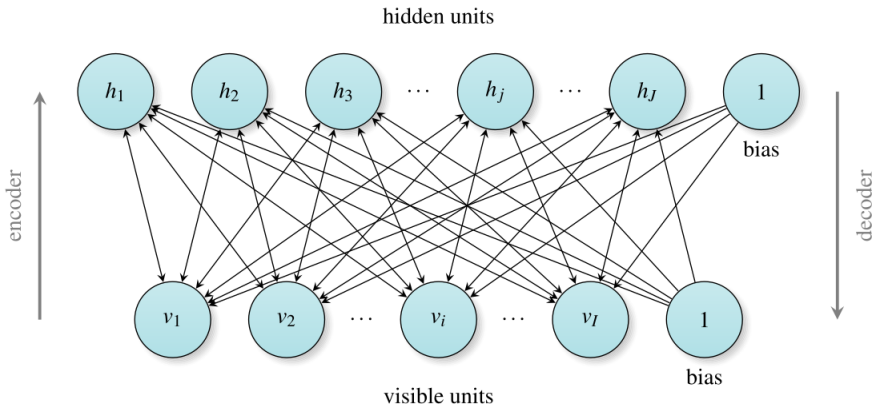


Fonte: Salakhutdinov e Hinton (2009)

Dessa forma, uma RBM é considerada uma MB mais fácil de ser treinada, pois são impostas restrições que tornam o processo de aprendizado e inferência mais rápidos, como a ausência de ligação entre nós da mesma camada (FACION, 2019).

Na Figura 2.11, podem ser observadas, com mais clareza, as conexões entre os nós de cada camada da rede em uma RBM.

Figura 2.11 – *Restricted Boltzmann Machine*



Fonte: Lopes e Ribeiro (2015)

Conforme detalhado por Facion (2019), as RBM são modelos baseados em energia, e uma configuração conjunta (v,h) das unidades visíveis e ocultas possui uma energia fornecida por:

$$E(v,h) = -\sum_{i \in visivel} a_i v_i - \sum_{j \in oculta} b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{2.6}$$

Onde, v_i, h_j, são os estados binários da unidade visível i e, unidade oculta j, a_i, b_j são seus *bias* e, w_{ij} é o peso entre eles.

A probabilidade que a rede atribui a um vetor visível, v, é dada pela soma de todos os possíveis vetores ocultos:

$$p(v) = \frac{1}{Z} \sum_{h} e^{-E(v,h)} \tag{2.7}$$

Onde Z é a função de partição e é somada por todos os pares possíveis de vetores visíveis e ocultos:

$$Z = \sum_{v,h} e^{-E(v,h)} \tag{2.8}$$

Resultando em:

$$p(v) = \frac{\sum_h e^{-E(v,h)}}{\sum_{v,h} e^{-E(v,h)}} \tag{2.9}$$

Nesse contexto, para estabilizar o sistema, a energia $E(v,h)$ deve ser a menor possível; em contrapartida, a probabilidade marginal $p(v)$ deve ser grande suficientemente (ZHANG; CHEN,

2017). Hinton, Osindero e Teh (2006), propuseram o algoritmo *Contrastive Divergence (CD)*, que tem como objetivo atingir a probabilidade máxima no menor tempo possível.

De acordo com Zhang e Chen (2017), este algoritmo utiliza os dados de treinamento para inicializar a camada visível e obter a camada oculta por distribuição condicional, sendo o resultado uma reconstrução da entrada.

A Equação 2.10 representa os cálculos realizados pelo algoritmo.

$$\Delta w_{ij} = \varepsilon(\langle v_j h_j \rangle_{data} - \langle v_j h_j \rangle_{recon}) \tag{2.10}$$

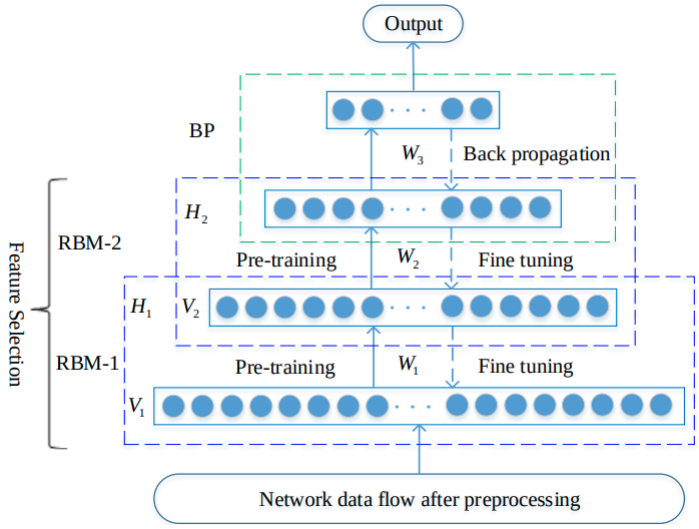
Existem diversos estudos aplicando as RBM na detecção de intrusão de rede, como é o caso de Zhang e Chen (2017), que comparam as RBM, DBN e SVM na detecção de intrusão de rede em grandes volumes de dados. Neste estudo concluíram que o modelo apresentado, RBM-DBN, é superior quando comparado ao modelo de AM tradicional como a RNA, sendo mais rápido e treinado em menor tempo.

Outro trabalho abordando a utilização de RBM é o de Elsaeidy et al. (2019), onde utilizaram as RBM para detecção de intrusão em uma *smart city*. De acordo com os autores, foram utilizadas as RBM devido a capacidade de aprender recursos de alto nível a partir de dados brutos, de maneira não supervisionada. E, além disso, lidam com representação real de dados gerados a partir de medidores e sensores inteligentes. Nos resultados da metodologia proposta, eles puderam comprovar a eficiência e alta precisão na detecção de ataques.

2.6.2 *Deep Belief Networks*

As *Deep Belief Network*s (DBNs) são Redes Neurais Artificiaiss (RNAs) multicamadas, de aprendizado não supervisionado, compostas por diversas *Restricted Boltzmann Machine*s (RBMs) em cada camada e um classificador na camada superior, conforme pode ser observado na Figura 2.12 (HINTON; OSINDERO; TEH, 2006). Após o pré-treinamento da rede, são ajustados os parâmetros para obter melhores resultados (YANG et al., 2019b).

Figura 2.12 – Modelo estrutural de uma DBN



Fonte: Yang et al. (2019a)

Em uma DBN os dados são inseridos nas RBM pela camada inferior, passando pelas camadas intermediárias que possuem apenas conexão unidirecional, até chegar a camada superior que possui conexão bidirecional (ZHAO; ZHANG; ZHENG, 2017). Dessa forma, o aprendizado em uma DBN é dividido em pré-treinamento e ajuste fino, onde o pré-treinamento é o processo de aprendizado não supervisionado, em que são inseridos os primeiros dados na RBM inferior e repassada por cada RBM até a camada final. Após essa fase de pré-treinamento, os parâmetros de cada camada são ajustados com o algoritmo de aprendizado supervisionado *Back-propagation* (BP). Contudo, Lopes e Ribeiro (2015), dizem que o BP pouco altera os pesos aprendidos na fase de pré-treinamento, e que a maioria dos ganhos de desempenho são obtidos nesta fase, onde o algoritmo é não supervisionado.

A característica de aprender a reproduzir a distribuição de probabilidade das entradas, sem nenhuma supervisão, é a mesma para as RBM e DBN (GERON, 2019). Contudo, as DBN são muito melhores nesta tarefa, pois geralmente os dados no mundo real são organizados em padrões hierárquicos. Assim, as RNA profundas aproveitam desta mesma característica em detrimento das RNA rasas.

Por apresentarem resultados positivos, as DBN estão presentes em várias pesquisas, como em Qu et al. (2017), onde foi desenvolvido um modelo de detecção de intrusão, aplicando como entrada o *dataset* NSL-KDD. Eles concluíram que o algoritmo satisfaz os quesitos de alta eficiência e confiabilidade nas detecções de intrusão de rede, superando algoritmos como SVM e

BP. Ainda ressaltam que a DBN tem grande capacidade de extração de características de um grande volume de dados, além de possuir um tempo de treinamento baixo.

2.6.3 *Convolutional Neural Network*

As *Convolutional Neural Network*s (CNNs) são consideradas um tipo de RNA, porém, especializadas em grandes volumes de dados. Para Khan et al. (2018), a CNN é uma das categorias mais populares de RNA.

O que torna esse modelo de rede neural tão eficiente é a convolução. A convolução é uma equação matemática que tem como produto uma nova função. Essa nova função pode ser interpretada como uma função modificada das funções coeficientes da equação incógnita. De acordo com Sovierzoski (2011) a convolução opera com duas funções, ou dois sinais, $x(t)$ e $h(t)$, para gerar uma terceira função ou sinal como resultado da operação, $y(t)$.

De acordo com Ferreira (2017), a convolução pode ser representada pela Equação 2.11. Sejam as funções f e g para uma variável contínua x, onde $*$ é o simbolo da convolução.

$$f(x) * g(x) = \int_{-\infty}^{+\infty} f(\tau).g(x-\tau)d\tau \tag{2.11}$$

Ainda Ferreira (2017), quando x está no conjunto de inteiros $\mathbb{Z}$ a equação da evolução discreta é definida, como na Equação 2.12.

$$f[x] * g[x] = \sum_{n=-\infty}^{\infty} f[n].g(x-n) \tag{2.12}$$

Segundo Tobergte e Curtis (2013) no processamento de imagens, onde a imagem é definida como uma função bidimensional, a convolução é útil para detecção de bordas, suavização de imagem, extração de atributos, entre outras aplicações. Apesar das CNN serem muito utilizadas na análise de imagens (FERREIRA, 2017), elas também tem sido utilizadas na análise de tráfego de rede, como é apresentado nos trabalhos Vinayakumar, Soman e Poornachandrany (2017), Wu, Chen e Li (2018), Naseer e Saleem (2018).

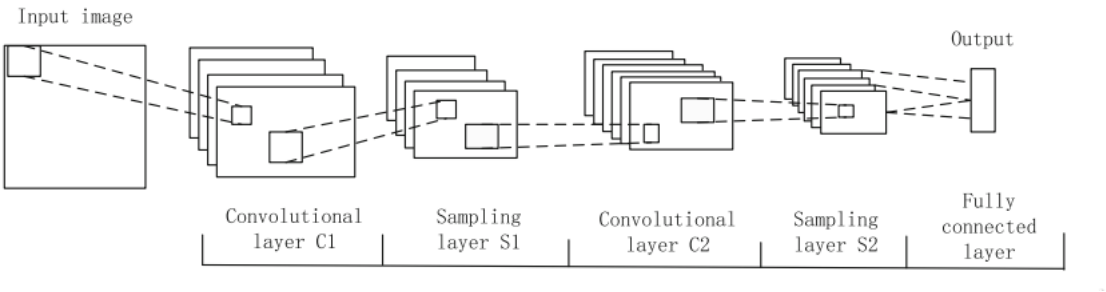
Conforme Lihao e Yann (2018) afirmam, o modelo CNN foi proposto pelo professor Yann LeCun. E foi utilizado, principalmente na classificação para reconhecimento de imagens. As CNN são muito úteis quando deseja-se que a rede aprenda padrões a partir de entradas de alta dimensão, como é o caso de imagens e vídeos (KHAN et al., 2018).

Uma CNN possui cinco camadas principais, sendo elas: entrada, convolução, amostragem inferior (também chamada de agrupada), conexão e saída (LIHAO; YANN, 2018). Em uma CNN

não há limite para o número de camadas, permitindo que a rede se adapte ao conteúdo de entrada para cada camada.

Na Figura 2.13 é apresentado um modelo arquitetural para uma CNN tradicional, neste caso para classificação de imagens. Nela é possível observar a aplicação dos filtros e geração de entradas para os camadas subsequentes, até a apresentação do resultado na saída da CNN.

Figura 2.13 – Arquitetura tradicional de uma CNN



Fonte: Zhang et al. (2016)

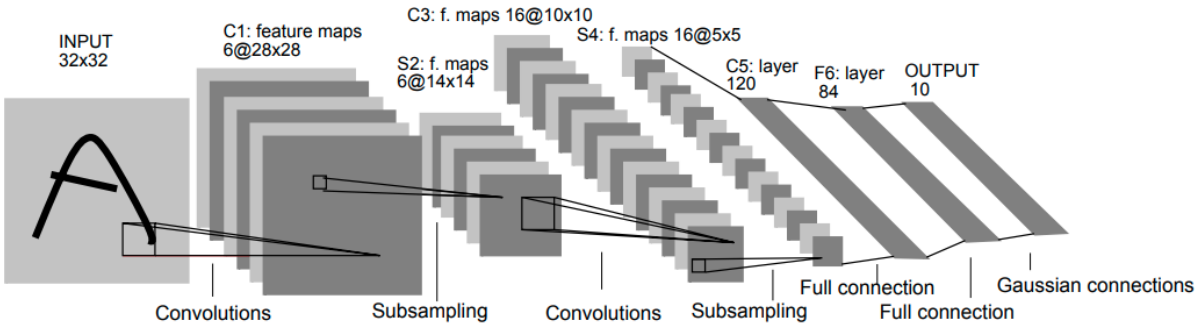
Conforme descrito por Zhang et al. (2016), na Figura 2.13, a CNN recebe uma imagem em sua entrada. Esta imagem é passada por quatro filtros treináveis, para produzir quatro mapas de características na camada convolucional C1. Cada bloco 2x2 é adicionado, calculado os pesos, passa por um *bias* e então é enviado a uma função sigmoide para produção dos quatro mapas de recursos, que serão entregues à camada de amostragem S1. Esse processo é repetido conforme a hierarquia, até que é produzido na saída da CNN um vetor de uma dimensão.

Diante de resultados promissores, novas arquiteturas baseadas em CNN foram construídas, dentre estas, algumas que tiveram destaque são: LeNet (LECUN et al., 1998), AlexNet (KRIZHEVSKY; SUTSKEVER; HINTON, 2012), VGGNet (SIMONYAN; ZISSERMAN, 2015), GoogLeNet (SZEGEDY et al., 2015), ResNet (HE et al., 2016) e ZFNet (ZEILER; FERGUS, 2013).

Segundo Gulli e Pal (2017), LeNet foi a primeira *Deep Convolution Neural Networks* (DCNN). Nesta ocasião, ela foi utilizada para a classificação de imagens em manuscritos.

A arquitetura proposta por LeCun et al. (1998) é apresentada na Figura 2.14.

Figura 2.14 – Arquitetura LeNet



Fonte: LeCun et al. (1998)

Em grande parte dos estudos, CNNs são aplicadas no contexto de classificação de imagens, porém, elas também estão sendo estudadas em outros contextos, como em IDSs para redes de computadores (MELIBOEV; JUMABEK; KIM, 2020), (LIU; TANG; YANG, 2019), (JIANG et al., 2020), (CHOWDHURY et al., 2017), (MANIKANDAN et al., 2020), (CHOCKWANICH; VISOOTTIVISETH, 2019). Neste contexto, na Subseção 2.6.4 é apresentada uma arquitetura de CNN hierárquica.

2.6.4 *Tree Convolutional Neural Network*

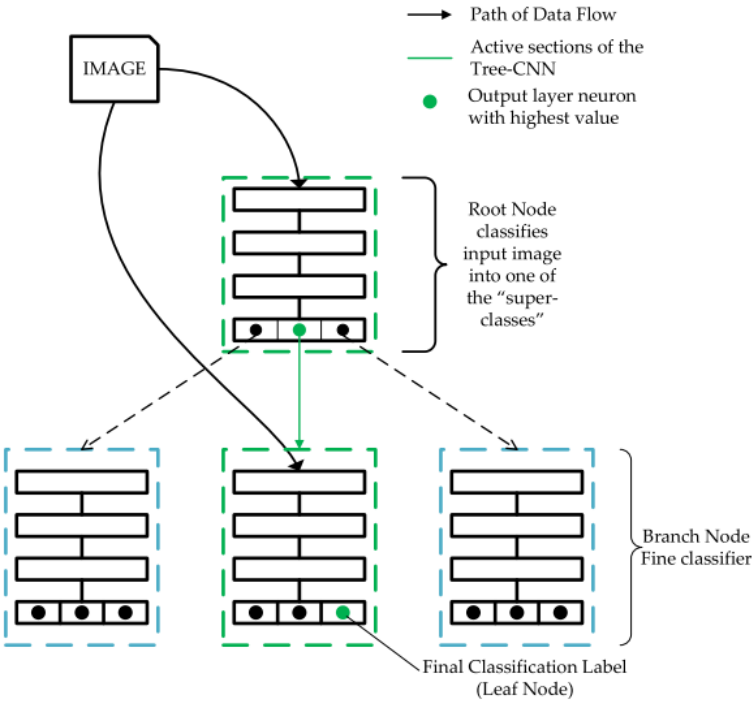
As *Tree Convolutional Neural Network*s (Tree-CNNs) são um tipo de *Deep Convolution Neural Networks* (DCNN) que tem como característica principal a disposição hierárquica das camadas convolucionais, sendo assim, semelhante a uma estrutura de árvore. De acordo com Jiang et al. (2018), a Tree-CNN tem como foco a especialização, enquanto a CNN comum é mais generalista. Dessa forma, a Tree-CNN consegue analisar sub-categorias que uma CNN não é capaz.

Como exemplo, na classificação de imagens, na categoria aeronaves, a CNN reconheceria muito bem o que é uma aeronave, porém, teria dificuldades ao identificar se é um avião de passageiros, helicóptero ou avião militar. Neste segundo caso, a Tree-CNN é mais precisa, pois após treinada, consegue produzir resultados assertivos em sub-categorias.

Segundo Yan et al. (2015), modelos de CNN hierárquicas demonstraram ter resultados melhores quando comparados aos de CNN padrão. Em uma CNN hierárquica, a complexidade na fase de treinamento é reduzida, pois os nós superiores executam a classificação utilizando recursos básicos (ROY; PANDA; ROY, 2020).

Na Figura 2.15 é exibida uma ilustração genérica de uma Tree-CNN de dois níveis, onde é possível observar o fluxo da classificação em uma CNN hierárquica para imagens. Neste caso, observa-se que o nó *root* executa a classificação mais generalizada da imagem. Posteriormente, é inserido no nó folha a classificação feita pelo nó *root*, que então executa a classificação final.

Figura 2.15 – Tree-CNN



Fonte: Roy, Panda e Roy (2020)

Em Xiao et al. (2014) é possível observar uma arquitetura semelhante, onde as classes totais são divididas gradualmente em direção ao nó folha, em um modelo guiado por um processo pragmático e orientado a erro. Vieira, Rosa e Rodríguez (2020) descrevem de forma sucinta os passos de uma Tree-CNN:

1° A rede é treinada para classificar os dados em N classes. Os dados pertencentes a uma nova classe são apresentados à rede, então, a rede cresce para acomodar a nova classe;

2° Posteriormente são adicionados nós folhas ou ramificações à estrutura atual;

3° O objetivo de redução do esforço de treinamento é composto por 2 componentes; o número de pesos atualizados e o número de exemplos, antigos ou novos, necessários para o treinamento;

4° Finalmente, as atualizações são apresentadas em uma seção da árvore.

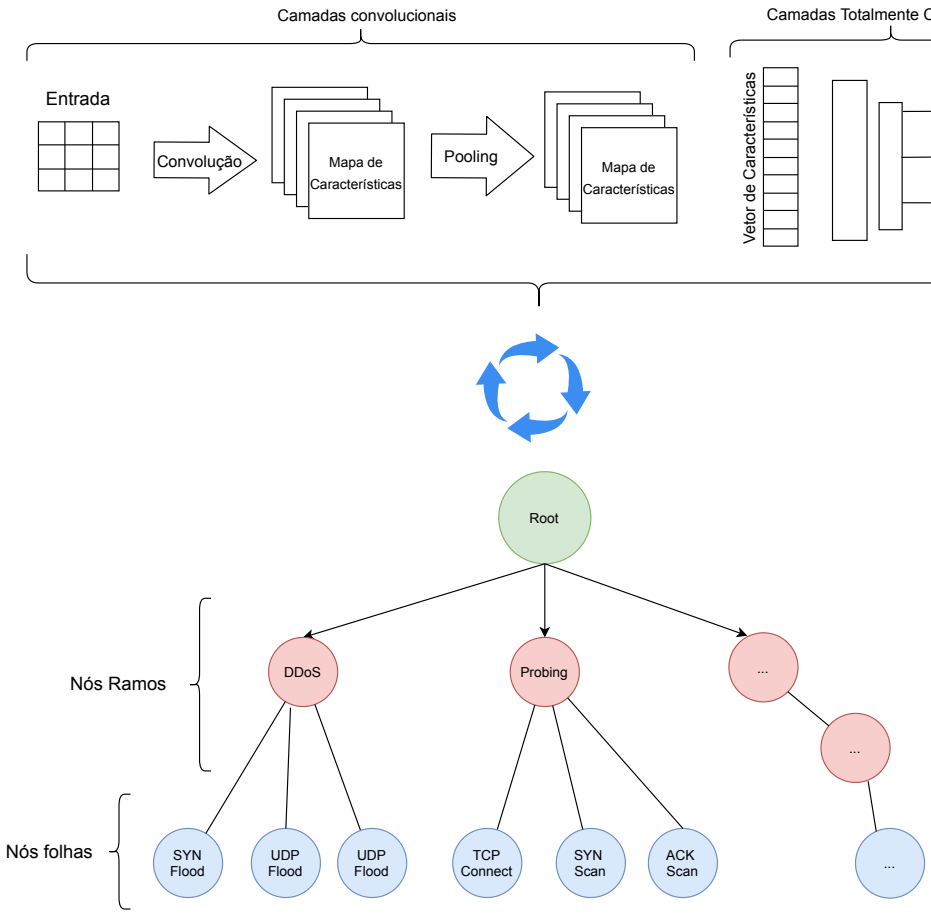
Esses passos são detalhados por Jiang et al. (2018), onde a rede é composta por um tronco principal (Trunk-Net) por onde os dados são recebidos e classificados de forma mais generalizada. Posteriormente, os dados são encaminhados aos demais nós (*Branch-CNN*), onde estes então realizam a tarefa mais pesada que é a classificação mais especializada dos dados de entrada.

Em uma Tree-CNN, composta estruturalmente pelo tronco principal denominado *Trunk-Net* e ramos (*Branch-Net*), a probabilidade de os dados serem classificados corretamente pelo tronco da rede é dada por $P(x_t, Tr)$. Tal qual, existem duas sub-redes onde uma é usada para codificar a função de entrada em um número fixo de classes, folha ou ramificação, e outra para codificar os locais para as funções de saída, a *Trunk-Net*. Já a função $P(C_i, Tr)$, representa a probabilidade dos dados de entrada serem classificados na i-ésima categoria correta. E, $P(x_t, Br)$, é a probabilidade dos dados de entrada serem classificados na t-ésima categoria correta por um tronco da rede.

Dessa forma, em que $P(x_t, Br) > P(x_t, Tr)$, pois a *Branch-Net* é responsável por distinguir a t-ésima categoria de outras categorias semelhantes. Sendo assim, assume-se que $P(C_i, Tr)$ é igual a 1, então a probabilidade da Tree-CNN $P(x_t, Br)$ $P(C_i, Tr)$ classificar corretamente é maior do que a probabilidade da *Trunk-NET*, $P(x_t, Tr)$.

Dentro do contexto de IDS, na Figura 2.16 é representado, de forma sucinta, a classificação de um ataque DDoS por meio de uma Tree-CNN.

Figura 2.16 – Tree-CNN na classificação de tráfego de rede



Fonte: Do autor (2021)

2.7 Funções de Ativação

Uma função de ativação, tem o propósito de manter a saída do neurônio artificial dentro de seu próprio domínio, podendo ser diferenciável ou parcialmente diferenciável (SILVA, 2015). Por meio de funções de ativação é possível ajustar os pesos e *bias* para que a saída da rede seja melhorada. A saída de um neurônio artificial é o produto da função de ativação aplicada à sua entrada, que se torna a entrada para o próximo neurônio da rede.

Em *Deep Learning* (DL), funções de ativação são aplicadas para melhorar o desempenho das redes. Tais funções, bem como as transformações lineares, estão no cerne das *Deep Neural Networks* (DNNs) (RAMACHANDRAN; ZOPH; LE, 2017), desempenhando papel fundamental no treinamento dessas redes.

Para incrementar os resultados das classificações por um algoritmo de Aprendizado de Máquina (AM), são trabalhadas as funções de ativação, que são um passo importante em uma DNN fornecendo uma propriedade não linear para esse tipo de rede.

Nos últimos anos, muitas funções de ativação foram propostas no intuito de substituir funções já conhecidas, como a *Rectified Linear Unit* (ReLU) (NAIR; HINTON, 2010), incluindo *Randomized Leaky Rectified Linear Activation* (RLReLU), Swish (RAMACHANDRAN; ZOPH; LE, 2017), Maxout (GOODFELLOW et al., 2013) entre outras (HE et al., 2015).

Portanto, a função de ativação é crucial para um bom comportamento e desempenho de aprendizado (LECUN; BENGIO; HINTON, 2015).

2.7.1 Softmax

A função de ativação *Softmax*, proposta por Bridle (1990), tem como objetivo a normalização das saídas das RNAs, a fim de proporcionar a classificação por meio de probabilidades. A equação para esta função é apresentada na Equação 2.13.

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad for\ i = 1, 2, \ldots, K \tag{2.13}$$

Esta função atua na classificação binária e multi-classe. Na classificação multi-classe, os valores de saída são ajustados para que fiquem entre 0 e 1, sendo que a soma desses valores deve ser 1. Dessa forma, a função retorna as probabilidades das entradas, sendo a classe objeto a que possui maior probabilidade. A *Softmax* é geralmente utilizada em camadas de saída da rede em DL (NWANKPA et al., 2018).

2.7.2 *Rectified Linear Unit*

A função de ativação *Rectified Linear Unit* (ReLU), proposta por Nair e Hinton (2010), é umas das funções de ativação mais usadas para aplicação em DL, e está presente em camadas ocultas das redes. Segundo Nwankpa et al. (2018), a justificativa por ser uma das funções de ativação mais usadas atualmente está no seu baixo custo computacional quando comparado a outras funções de ativação.

Na Equação 2.14 é exibida a expressão matemática da função.

$$Relu(z) = max(0, z) \tag{2.14}$$

A função trabalha com valores entre $[0, \infty]$, sendo que qualquer valor negativo de entrada é tomado como 0, reduzindo a quantidade de cálculos realizados pela função. De acordo com Geron (2019), o que faz com que a ReLU seja tão rápida é o fato de não ter operações matemáticas complexas.

2.7.3 *Soft-Root-Sign*

Como alternativa às funções de ativação citadas anteriormente, a *Soft-Root-Sign* (SRS) (LI; ZHOU, 2020), tem mostrado resultados promissores para a obtenção de modelos de treinamento mais rápidos. A SRS pode ajustar a saída de forma adaptativa por meio de um par de parâmetros treináveis independentes, que apresentam melhor desempenho de generalização e velocidade de aprendizagem mais rápida. É importante ressaltar que atualmente existem poucos estudos com a utilização da função de ativação SRS, principalmente em uma Tree-CNN.

De acordo com Li e Zhou (2020), a função de ativação SRS é definida pela Equação 2.15, onde α e β são um par de parâmetros treináveis não negativos.

$$SRS(x) = \frac{x}{\frac{x}{\alpha} + e^{\frac{x}{\beta}}} \tag{2.15}$$

E, a derivada é obtida pela Equação 2.16.

$$SRS'(x) = \frac{(1 + \frac{x}{\beta})e^{-\frac{x}{\beta}}}{(\frac{x}{\alpha} + e^{-\frac{x}{\beta}})^2} \tag{2.16}$$

Em contraste com a ReLU, a SRS possuí região não monotônica quando $x < 0$, fortalecendo assim o aprendizado da rede. Outro ponto destacado por Li e Zhou (2020), é que a SRS possui saída limitada quando $x > 0$ o que evita e retifica a distribuição de saída a ser espalhada no espaço de números reais não negativos.

Considerando as características favoráveis acerca da SRS, nesta pesquisa, optou-se por utilizá-la visando assim o aumento na velocidade de aprendizado do modelo proposto.

2.8 Métodos multivariados para redução de dados

Algoritmos de AM são utilizados frequentemente para extrair informações de fontes de dados diversas. Porém, para que isso aconteça com precisão, faz-se necessário que os dados sejam tratados, pois caso tenham ruídos, podem afetar o desempenho dos algoritmos ou até mesmo apresentar informações imprecisas (FACELI et al., 2011). Para resolver esse problema, são

aplicadas técnicas de pré-processamento de dados, a fim de eliminar ou minimizar os problemas dos *datasets*.

Segundo Estrela, Boente e Goldschmidt (2006) a tarefa de pré-processamento dos dados é dividida em quatro etapas:

- **Seleção de dados:** Por meio desta função, identifica-se os dados que são relevantes para o processo de construção do modelo;
- **Limpeza de dados:** Aqui são realizados tratamentos dos dados, removendo dados ruidosos por meio de definições de conhecimento dos dados;
- **Codificação dos dados:** A codificação é o processo de conversão de dados para um tipo específico, como no caso de algoritmos que trabalham somente com valores numéricos, mas a base tem valores categóricos;
- **Enriquecimento dos dados:** Neste processo, podem ser incluídos dados para complementar os registros manualmente, inserindo informações que são de conhecimento prévio ou relacionando a base de trabalho com outras bases externas.

Em grandes volumes de dados, a redução de dimensionalidade é tarefa primordial para o bom desempenho dos algoritmos de aprendizagem de máquina. De acordo com Borges e Nievola (2012), caso o *datasets* tenha um grande número de atributos, esse fator pode atrapalhar o processo de aprendizagem na base. Faceli et al. (2011) explicam que, caso cada atributo seja visto como uma coordenada em um espaço d-dimensional, onde d é a quantidade de atributos, o hipervolume desse espaço cresce exponencialmente com a adição de novos atributos, levando ao que se conhece na literatura como maldição da dimensionalidade.

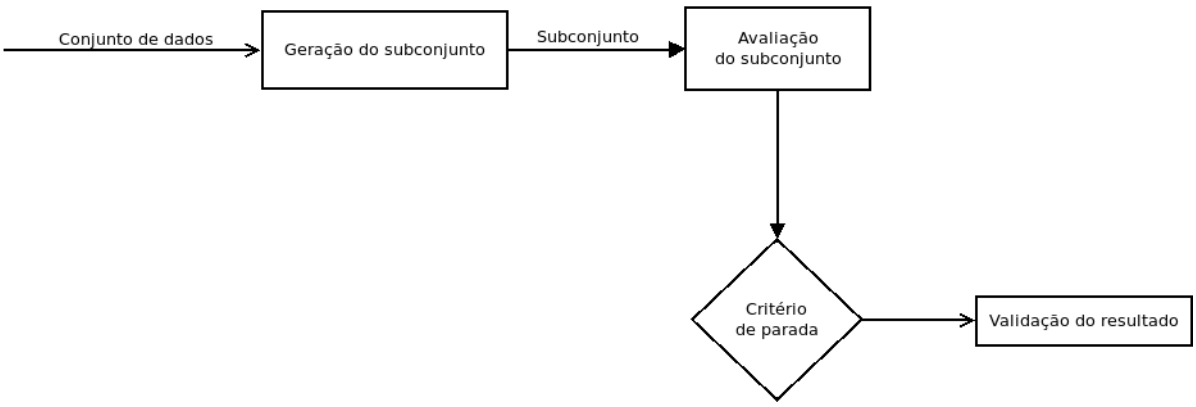
De acordo com Faceli et al. (2011) existem duas abordagens diferentes para redução de dados, são elas: Seleção de Atributos e Agregação.

2.8.1 Seleção de atributos

A seleção de atributos é essencial em bases de dados com muitos atributos, pois uma parte desses atributos geralmente é irrelevante, redundante ou ruidosos (FACELI et al., 2011). Nessa abordagem, podem ser aplicadas técnicas de ordenação para seleção de atributos ou seleção por subconjunto.

Um subconjunto é criado a partir de um conjunto de características, de acordo com determinado critério de seleção (BORGES; NIEVOLA, 2012). Na Figura 2.17, Dash e Liu (1997) apresentam um modelo gráfico do fluxo de seleção de atributos.

Figura 2.17 – Processo de seleção de atributos



Fonte: Dash e Liu (1997)

Segundo Lee (2005) a seleção de atributos pode ser particularmente importante, em casos em que a medição de atributos é custosa, neste caso, cria-se um subconjunto do conjunto original.

Algumas das vantagens na aplicação da técnica de seleção de atributos, são: identificar atributos importantes, otimizar técnicas de AM no *dataset*, redução de memória e tempo de processamento, eliminação de ruído e atributos irrelevantes, lidar melhor com a maldição da dimensionalidade (FACELI et al., 2011) .

Em *datasets* pequenos e com poucos atributos, a revisão manual pode ser aplicada para que sejam removidos os atributos que não são importantes para o modelo a ser gerado. Porém, em casos onde há uma grande quantidade de dados, com muitos atributos e dificuldade de relacionamento entre estes atributos, são aplicadas técnicas para reduzir o número de atributos.

De acordo com Lee (2005), estas técnicas são:

- **Embutida:** A seleção dos atributos é realizada pelo próprio algoritmo de AM, geralmente por meio de uma árvore de decisão;
- **Baseada em filtro:** Os atributos irrelevantes são filtrados antes da aplicação do algoritmo de AM, seguindo algum critério pré-estabelecido;
- **Baseada em *wrapper*:** A seleção ocorre antes de ser aplicado o algoritmo de AM, porém, em paralelo os subconjuntos são submetidos ao algoritmo para avaliação.

Otimizar o *dataset* é tarefa fundamental para aplicação de algoritmos de AM, pois reduz o tempo de treinamento e gera inferência mais confiável. Portanto, a aplicação de técnicas de redução de dimensionalidade é um processo intrínseco da geração de modelos para treinamento dos algoritmos.

2.8.2 Agregação

A agregação é a combinação de atributos originais por meio de funções lineares ou não lineares. A técnica de Análise de Componentes Principais, em inglês PCA, é uma das mais conhecidas para a tarefa de agregação. Esta técnica correlaciona estatisticamente os dados do modelo, reduzindo a dimensionalidade do conjunto. O processo consiste em remover atributos redundantes, tornando o modelo mais inteligente (FACELI et al., 2011).

Ao combinar os atributos, removendo redundâncias, a técnica pode levar o *dataset* à perda de valores originais. Para algumas áreas de pesquisa, esses valores são essenciais para interpretar resultados, como em: finanças, biologia, monitoramento ambiental e medicina. Portanto, nessas áreas são usadas técnicas de seleção de atributos ao invés de agregação.

2.8.3 Análise de Componentes Principais

Segundo Lyra et al. (2010), a *Principal Component Analysis* (PCA) é uma das mais importantes ferramentas da análise multivariada, sendo a base para a maioria dos métodos multivariados de análise de dados. Esta técnica tem sido amplamente usada em diversas áreas de pesquisa, como: análise de tráfego de rede, economia, processamento de imagem e genética (TENÓRIO, 2013).

Comumente utilizada para redução de dados em reconhecimento estatístico de padrões (HAYKIN, 2007), a técnica de PCA aplica a abordagem de agregação para redução da dimensionalidade. Segundo Hong, Ruohe e Kunhui (2008), a PCA é considerada como um dos algoritmos de análise de linearidade mais bem sucedidos.

De acordo com Siwek e Osowski (2017), a PCA é uma transformação ortogonal de dados convertendo as observações representadas pelos vetores $\mathbf{x}$ em um conjunto de vetores $\mathbf{y}$ de menor dimensão. Por meio desta técnica é possível criar modelos menores de dados, simplificando as análises.

Para reduzir a dimensionalidade, o algoritmo de PCA extrai do *dataset* original um novo *dataset* menor, então, esse subconjunto menor é denominado componentes. A ordem de maior variabilidade é definida pela ordem dos componentes no subconjunto. De forma crescente,

o primeiro componente é o que possui maior variabilidade, o segundo componente a segunda maior variabilidade, e assim sucessivamente.

Para Rossi (2017), o algoritmo de PCA se resume em:

1. Organize os dados das medições em uma matriz n x m , onde m é o número de variáveis medidas ou dimensões, e n é o número de amostras;

2. Caso seja necessário, divida as medições de cada dimensão pelo seu desvio padrão para normalizá-las e evitar a sensibilidade da PCA à diferença de escala entre dimensões;

3. Calcule a matriz de covariância da matriz resultante dos passos anteriores (caso o passo 2 tenha sido efetuado, essa matriz será a de correlações);

4. Calcule os autovetores e autovalores associados à matriz de covariância;

5. Ordene os autovetores de acordo com os autovalores associados. Desse modo, o primeiro autovetor é a componente principal, o segundo é a segunda componente principal e assim por diante;

6. Descarte as componentes de menor relevância. Para isso, defina o percentual da variância original que deve ser mantido, e escolha as componentes principais de modo que a soma dos autovalores associados seja maior ou igual a esse percentual.

3 TRABALHOS RELACIONADOS

Nesta seção são apresentados os trabalhos mais relevantes relacionados a essa pesquisa.

O estudo acerca da detecção de intrusão em redes de computadores é composto por vários componentes, tais como: sistemas de detecção de intrusão, algoritmos que analisam o tráfego e *datasets* para treinamento dos algoritmos. Para melhor organização, esta seção foi segmentada em IDS, *Datasets*, AM e DL.

Pesquisas em AM tem sido impulsionadas pelo grande salto de poder computacional que foi agregado nas últimas décadas. Este poder computacional faz com que sejam aplicados algoritmos complexos, além de permitir o trabalho com *datasets* muito grandes. O AM tem sido estudado não só em computação, mas também em diversas áreas do conhecimento, como processos fabris, medicina, agricultura e economia.

3.1 Sistema de Detecção de Intrusão

A detecção automática de ameaças a redes de computadores é pautada em diversos estudos, pois a segurança da informação é assunto primordial para organizações.

No trabalho de Khraisat et al. (2019), pode-se observar uma visão abrangente dos componentes de um IDS, abordando técnicas, *datasets* e desafios. Na abordagem, os autores citam a dificuldade de um IDS baseado em assinaturas em detectar ataques como do tipo *zero-day*, além disso, como os *malwares* estão cada vez mais sofisticados, para um IDS detectar um tipo de ataque pode ser necessário a inspeção de diversos pacotes, inclusive pacotes já passados pelo IDS para que seja feita uma correlação. Como alternativa, apresentam o IDS baseado em anomalias, pois trabalham com modelos pré-estabelecidos de aprendizado, onde é definido um padrão de tráfego normal e o algoritmo faz a comparação desses modelos com o tráfego passante, classificando conforme o modelo definido.

Para Khraisat et al. (2019), os métodos podem ser classificados em três tipos: baseado em estatística (LIN; KE; TSAI, 2015), baseado em conhecimento (ELHAG et al., 2015; CAN; SAHINGOZ, 2015), e baseado em AM (BUCZAK; GUVEN, 2016; MESHRAM; HAAS, 2017). Em IDSs baseados em anomalias, os ataques do tipo *zero-day* podem ser identificados no momento do ataque, pois não precisam de um reconhecimento prévio daquele tipo de ataque. Eles citam as técnicas usadas pelos atacantes para evitar a detecção pelo IDS, tais como: fragmentação, inundação, ofuscação e criptografia, e, reforçam que essas técnicas representam um desafio para o IDS, pois no caso de criptografia, o IDS não consegue inspecionar o tráfego

encapsulado. Os autores ainda fizeram um breve levantamento de características dos principais *datasets* públicos, como pode ser visto na Tabela 3.1. Nesta comparação observa-se a evolução dos *datasets*, principalmente no quesito ataque do tipo *zero-day*, que é quando não tem nenhum conhecimento do ataque, apenas padrões de tráfego que sugerem um ataque em curso.

Tabela 3.1 – Comparação entre *datasets* públicos

Dataset	Realist Traffic	Label data	IoT traces	Zero-day attacks	Full packet captured	Year
DARPA 98	✓	✓	x	x	✓	1988
KDDCUP 99	✓	✓	x	x	✓	1999
CAIDA	✓	x	x	x	x	2007
NSL-KDD	✓	✓	x	x	✓	2009
ISCX 2012	✓	✓	x	x	✓	2012
ADFA-WD	✓	✓	x	✓	✓	2014
ADFA-LD	✓	✓	x	✓	✓	2014
CICIDS2017	✓	✓	x	✓	✓	2017
Bot-IoT	✓	✓	✓	✓	✓	2018

Fonte: Khraisat et al. (2019)

O fator seleção de atributos também foi estudado pelos autores. Neste caso, executaram o algoritmo de árvore de decisão C4.5 no conjunto NSL-KDD completo (41 atributos) e após a redução, com 13 atributos. Observou-se que quando é aplicada a seleção de atributos, como no caso da técnica Ganho de Informação, o tempo gasto para classificação reduziu de 2,76 para 0,84 segundos. Dados esses valores, pode-se observar que há vantagem considerável quando é aplicada alguma técnica de redução de dimensionalidade.

Ao concluírem a pesquisa os autores reforçam a necessidade de *datasets* mais novos para que as pesquisas sejam realizadas, pois os ataques estão cada vez mais sofisticados e os atacantes estão cada vez mais ardilosos. Portanto, um IDS eficaz deve ser capaz de detectar com precisão diferentes tipos de ataques, incluindo invasões que incorporam técnicas de evasão.

3.2 *Datasets*

Como peça chave para treinar os algoritmos de Aprendizado de Máquina (AM), os *datasets* são estudados por diversos pesquisadores, como em (FERREIRA; SHINODA, 2016; MACIÁ-FERNÁNDEZ et al., 2018; THAKKAR; LOHIYA, 2020).

Em Gharib et al. (2016) é realizada uma avaliação abrangente de alguns *datasets*, onde os autores propõe um método para avaliar os conjuntos existentes, aplicando um cálculo de eficiência com base nos atributos: Diversidade de ataques, Anonimato, Protocolos disponíveis, Captura completa, Interação completa, Configuração completa da rede, Tráfego completo, Conjunto de

características, Heterogeneidade, Dados rotulados, e Metadados. No estudo, eles compararam os *datasets* DARPA98, KDD99, ISC2012 e ADFA13, usados para desenvolvimento de pesquisas em detecção de intrusão em redes de computadores. O problema levantado pelos autores refere-se a falta de um *dataset* realista para pesquisas científicas. Os autores destacam que recentemente a quantidade de ataques passou de um problema de baixa escala para um problema de larga escala mais sofisticado, portanto, faz-se necessário um conjunto mais realista para treinamento dos algoritmos.

A dificuldade em ter um conjunto mais realista está atrelado a problemas legais e de privacidade, pois os dados capturados em um ambiente real, contém informações confidenciais ou protegidas por leis. Portanto, eles ressalvam que grande parte dos estudos nessa área, apresentam resultados com base em *datasets* públicos, já estudados, o que acarreta em muitos casos de propagação de erros. Eles citam as formas para criar um novo *dataset*, uma delas é gerando tráfego artificial por meio de alguma ferramenta via *software*, ou utilizando simuladores onde todo ambiente é simulado por algum *software*; ou por meio de emuladores, onde o ambiente é composto por um misto de *software* e *hardware*; ou ainda, onde monta-se uma bancada com vários equipamentos, como roteadores, *switches* e computadores, para geração do tráfego.

Para avaliar o *framework* proposto, Gharib et al. (2016) propõem a Equação 3.1. A equação, tem W como coeficiente de flexibilidade (peso de cada recurso), que pode ser definido com base no tipo de IDS selecionado para teste. No trabalho foram selecionados onze recursos, logo $W = 11$. O coeficiente de cada subfator é definido por V, definido com base em experiências ou distribuição de subfatores em diferentes cenários. Neste caso, existem dois recursos com subfatores: ataques e protocolos. Além disso, F é a aparência do fator e subfator específico no *dataset* que pode ser binário (0 ou 1) ou com vários valores.

$$\sum_{i=1}^{n} W_i (\sum_{j=1}^{m} V_j * F_j) \tag{3.1}$$

Onde n é o número de recursos da estrutura proposta, no caso deste trabalho, são 11, e, m é o número de coeficientes para cada fator. Na estrutura proposta, para dois fatores o valor de ataques e protocolos de m é 7 e 5 respectivamente, mas para os outros fatores $m = 1$.

Como exemplo, na Figura 3.1 é exibido o resultado da aplicação do cálculo nas bases *KDD99* e *KYOTO*. Observa-se que a base *KYOTO* a pontuação está acima do obtido com a base *KDD99*.

Figura 3.1 – Comparação da eficiência dos *datasets*

Dataset	Calculation	Score
KDD	0.05*1+ 0.05*0 + 0.1*1 + 0.05*1 + 0.05*1 + 0.25 * (0.1 + 0.0 + 0.04 + 0.08 + 0.04) + 0.25 * (0.0 + 0.19 + 0.16 + 0.03 + 0.0 + 0.0 + 0.2) + 0.05*0 + 0.05*0 + 0.05*1 + 0.05*1	0.56
KYOTO	0.05*1 + 0.05*0 + 0.1*1 + 0.05*1 + 0.05*1 + 0.25 * (0.1 + 0.74 + 0.04 + 0.08 + 0.04) + 0.25 * (0.36 + 0.19 + 0.16 + 0.03 + 0.03 + 0.03 + 0.2) + 0.05*0 + 0.05*0 + 0.05*1 + 0.05*1	0.85

Fonte: Gharib et al. (2016)

Neste estudo os autores concentraram na criação do método para avaliação de *datasets* de ataques a redes, porém, colocaram como próxima etapa para este trabalho, a geração de um *dataset* e aplicar a ele o método de avaliação desenvolvido.

3.3 Aprendizado de Máquina

Há diversas pesquisas em Aprendizado de Máquina (AM) com aplicação de algoritmos rasos, como é o caso de (CHU; LIN; CHANG, 2019; BINDRA; SOOD, 2019), onde demonstram que quando comparados aos *Intrusion Detection Systems* (IDSs) baseados em assinatura, AM é superior.

No trabalho de Chu, Lin e Chang (2019), eles realizaram experimentos e compararam dos resultados apresentados pelos algoritmos SVM, NB, DT e uma RNA usando o MLP. O objetivo do estudo é a identificação precoce de ataques persistentes avançados, que são organizados e orquestrados por grupos de *hackers* distribuídos, formando um ataque complexo e direcionado. Para alcançar os objetivos, os autores usaram a técnica de PCA para redução de dimensionalidade e aprimorar o *dataset*, a fim de extrair as características para otimizar a execução dos algoritmos de AM na detecção de ataques. Os tipos de ataques analisados foram DoS, *Probe*, R2L, e U2R.

Chu, Lin e Chang (2019) utilizaram o *dataset* NSL-KDD e criaram subconjuntos para utilização na ferramenta *Waikato Environment for Knowledge Analysis* (WEKA). Conforme relatado pelos autores, os classificadores SVM e MLP tiveram resultados próximos (97,22% e 97,82%) respectivamente. Para atingir esses resultados foram ajustados parâmetros C e *gamma* no SVM, chegando aos valores $C = 1,0$ e $gamma = 0,0$. Já no MLP o parâmetro ajustado foi a quantidade de camadas ocultas, nesse caso encontraram o resultado melhor com 4 camadas. Para os demais parâmetros foi utilizado o padrão. Uma questão importante detectada pelos autores foi em relação a velocidade de detecção, quando se usa a redução de dimensionalidade.

Apesar de não mostrar muita diferença na precisão de detecção, os autores puderam observar um ganho muito grande na velocidade com que os ataques foram reconhecidos após a aplicação da técnica de PCA.

Outro trabalho que faz comparações da eficiência de classificação pelos algoritmos de AM é o de Bindra e Sood (2019). Abordando algoritmos de aprendizado supervisionado, evidenciaram que esse tipo de aprendizado tem melhor precisão na taxa de detecção de ataques do tipo DDoS. Eles usaram o *dataset* CIC-IDS2017 e aplicaram os algoritmos e técnicas de AM, como *Logistic Regression*, *K–Nearest Neighbors* (KNN), NB, RF, SVM e *Linear Discriminant Analysis*. É possível observar nos resultados apresentados na Tabela 3.2, que o algoritmo RF obteve melhor precisão na detecção do tipo de ataque DDoS.

Tabela 3.2 – Acurácia na detecção de ataques DDoS

Machine Learning Models	Mean Accuracy	Standard Deviation
Logistic Regression	0.824770	0.003
K–Nearest Neighbors	0.943640	0.001
Gaussian NB	0.810424	0.002
Random Forest	0.961341	0.001
Linear SVM	0.823536	0.002
K–Nearest Neighbors with n_neighbors=3	0.951	0.001
Random Forest with n_estimator=20	.965	0.001
Linear Discriminan Analysis	.821	0.003

Fonte: Adaptado de Bindra e Sood (2019)

Bindra e Sood (2019) citam que precisaram reduzir a dimensionalidade do *dataset*, pois em primeiro momento, ao tentar executar os algoritmos sem a redução, perceberam que seria necessário tempo surpreendentemente alto para execução de cada algoritmo. Eles reforçam a necessidade de pré-processamento dos dados para que o algoritmo possa ter melhor desempenho e precisão. Outra ação que fizeram para melhorar o *dataset* foi a transformação de dados categóricos em numéricos, como no caso dos rótulos *BENIGN* e DDoS, eles substituíram por 0 e 1, respectivamente.

Os parâmetros de cada algoritmo também foram alterados para melhorar os resultados. No caso do KNN o valor padrão do número de vizinhos é de 5, porém, quando é reduzido para 3 e 1, o desempenho é aprimorado. No caso do RF o valor padrão é 10, e quando aumentaram o valor para 20, o desempenho também melhorou. Outro parâmetro importante para RF é a quantidade de *jobs*, por padrão é 1, porém, pode ser ajustado para a quantidade de *CPUs* disponíveis na máquina, cada incremento de 1 é a utilização de mais uma *CPU* para o processamento do

algoritmo. Com o aumento de *CPUs* consegue-se paralelizar as tarefas e, consequentemente, reduz-se o tempo para classificação.

Os resultados apontam que o algoritmo RF demonstrou ser o mais eficiente na detecção de ataques do tipo DDoS, atingindo 96,2% de precisão. Um dos fatores a que os autores atribuem os resultados alcançados, é referente a validação cruzada utilizada por eles, pois essa técnica evita o problema de *underfitting*. O *underfitting* em AM, indica que o modelo é fraco, e possui baixa capacidade de generalização e não representa o mundo real (KIRK, 2014). Como destaque, os autores citam que os resultados podem ser melhores com a utilização de outras técnicas de pré-processamento de dados.

3.4 *Deep Learning*

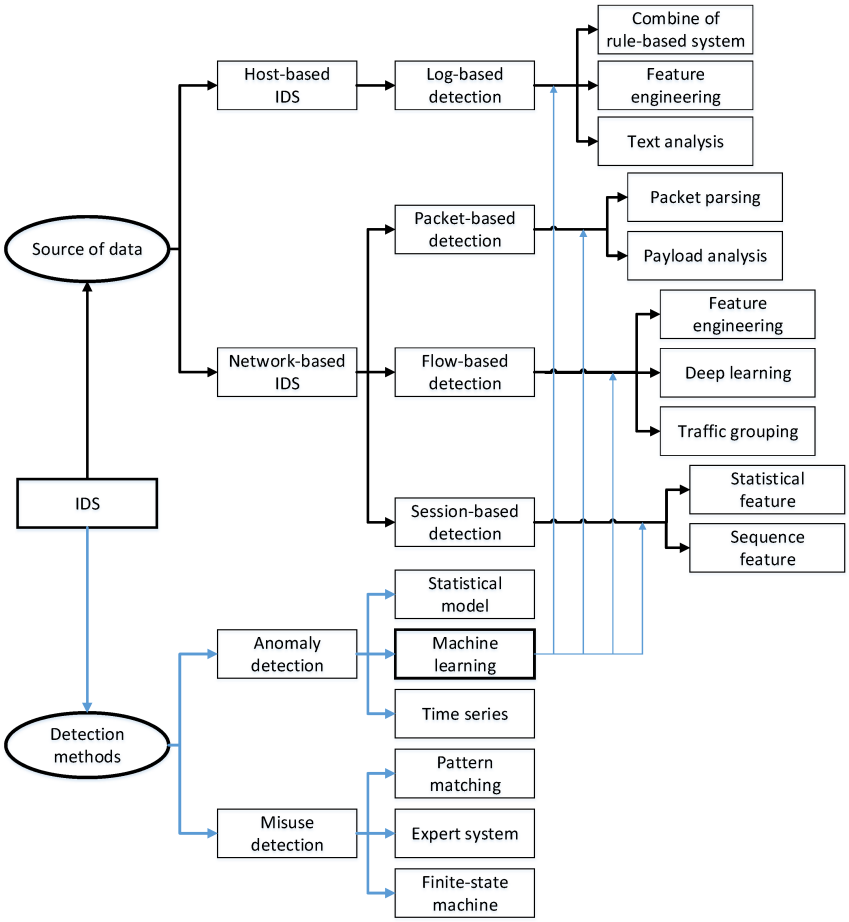
Em virtude de resultados promissores com a utilização de Aprendizado de Máquina (AM) na classificação de tráfego de rede, por meio de algoritmos tradicionais, a técnica de *Deep Learning* (DL) está presente em muitas pesquisas (AKSU; AYDIN, 2018; LIU; LANG, 2019).

Em Aksu e Aydin (2018) os autores abordam a detecção de varreduras de portas por meio de algoritmos de AM tradicionais e DL. Os autores compararam os resultados obtidos com o algoritmo *Support Vector Machine* (SVM) e uma *Deep Neural Network* (DNN). Para a DNN, foram aplicadas sete camadas ocultas com um número diferente de neurônios para cada camada, por fim, aplicaram a função ReLU ao modelo em DL.

Para realização dos experimentos, Aksu e Aydin (2018) utilizaram o *dataset* CIC-IDS2017. Onde, primeiramente realizaram a normalização do tráfego contido no arquivo *"Friday-WorkingHours-Afternoon-PortScan.pcap_ISCX.csv"*, que conta com 286.467 registros, sendo 158.930 tentativas de varredura de portas, e 127.537 comportamentos benignos. Os autores dividiram o *dataset* para treino e teste, sendo 67% e 33% respectivamente. Então, foram criados os modelos para SVM e DNN com base nos dados de treinamento e, por fim, aplicaram o modelo ao conjunto de teste. Dessa forma, os autores obtiveram acurácia de 97,80% com DNN e 69,79% com SVM. Os autores concluem o trabalho apontando para novos testes com mais tipos de ataques e outros algoritmos de AM e DL.

Um levantamento dos desafios enfrentados pelos IDSs baseados em AM é apresentado por Liu e Lang (2019), onde os autores propõem uma taxonomia para os IDSs, conforme Figura 3.2. Eles abordaram os métodos de detecção usados pelos IDSs e realizaram comparações entre estes métodos, apontando as vantagens e desvantagens nos modelos baseados em assinaturas e anomalias.

Figura 3.2 – Taxonomia IDS



Fonte: Liu e Lang (2019)

Neste trabalho os autores comparam os modelos de AM em IDS. Classificam em modelos profundos e rasos, sendo os rasos, os algoritmos de AM tradicionais como NB, SVM e DT. Já para modelos profundos, eles comparam algoritmos como RBM, DBN, CNN, RNN e DNN.

As comparações, entre algoritmos rasos e profundos, são feitas em torno dos aspectos: tempo de execução, pois em DL, gasta-se mais tempo para treino e teste; número de parâmetros (aprendidos pelo algoritmo) e hiperparâmetros (definidos antes da execução do algoritmo), em DL os parâmetros superam em muito em modelos rasos, porém, precisam de mais tempo; representação de recursos, capacidade de aprendizagem e interpretabilidade.

Os autores destacam que há vários problemas enfrentados pelos pesquisadores em IDSs baseados em AM. Entre eles está a falta de *datasets*, pois a maioria das pesquisas estão sendo conduzidas com *datasets* antigos ou muito específicos. A representatividade de uma rede real é

outro ponto levantado por eles, pois alguns algoritmos apresentam ótima precisão em *datasets*, porém quando aplicados a uma rede real apresentam resultados muito abaixo.

Para eles DL tem desempenhado um papel cada vez mais importante, principalmente nos estudos em IDSs. Quando comparados aos modelos de AM rasos, os modelos de DL possuem habilidades mais fortes de adaptação e generalização. Além disso, as abordagens de DL são independentes da engenharia de recursos e do conhecimento do domínio, o que leva uma vantagem excepcional sobre os modelos de AM rasos.

Em Manimurugan et al. (2020) os autores propuseram o modelo DBNIDS para detecção de intrusões em redes de computadores. O estudo foi conduzido no contexto de proteção de dispositivos de *Internet of Things* (IoT). O modelo proposto é composto por uma DBN e um algoritmo do tipo guloso, utilizado para a fase de treinamento não supervisionado. De acordo com os autores, a combinação deste tipo de algoritmo com a DBN, proporciona a otimização do modelo, pois o algoritmo otimiza camada por camada da DBN. Também foi implementada a função de ativação Softmax na fase de ajustes finos na camada superior, a fim de aprimorar os recursos de amostras rotuladas. Para avaliação do modelo foi utilizado o *dataset* CIC-IDS2017, e para comparação dos resultados, foram utilizadas outras técnicas e modelos, como SVM, RNN, *Spiking Neural Network* (SNN) e FNN. Conforme afirmam os autores, foram obtidos resultados ótimos pelo modelo DBNIDS. Nos resultados apresentados, o modelo alcançou acurácia de 99,37% para classe de tráfego normal (BENIGNO), 97,93% para *Botnet*, 97,71% para *Brute Force* (BF), 96,67% para DoS e DDoS, 96,37% para *Infiltration*, 97,71% para *Ports Scan* e 98,37% para a *Web attack*.

Em Jiang et al. (2020), os autores criaram um modelo para detecção de intrusões em redes, baseado em CNN e *Bi-directional Long Short-Term Memory* (BiLSTM) (SIAMI-NAMINI; TAVAKOLI; NAMIN, 2019), denominado CNN-BiLSTM. Neste caso, a CNN foi utilizada para extrair características espaciais, enquanto a BiLSTM foi usada para extrair características temporais. Para o treinamento e teste do modelo os autores criaram um *dataset* e utilizaram as técnicas *One-Sided Selection* (OSS) (KUBAT; MATWIN, 1997) e *Synthetic Minority Oversampling Technique* (SMOTE) (CHAWLA et al., 2002) para normalizá-lo e balancear as classes. A técnica OSS reduz os registros ruins do *dataset*, enquanto a SMOTE incrementa o *dataset* inserindo mais amostras de classes com poucas amostras, fazendo com que o *dataset* fique balanceado. Para avaliar o modelo CNN-BiLSTM, os autores aplicaram-o aos *datasets* públicos NSL-KDD (TAVALLAEE et al., 2009) e UNSW-NB15 (MOUSTAFA, 2015). Neste trabalho os autores avaliaram as classes tráfego Normal e ataques DoS, *Probe*, U2R e R2L. O modelo

proposto, CNN-BiLSTM, obteve acurácia de 83,58% no *dataset* NSL-KDD, e 77,16% no UNSW-NB15, superando outros algoritmos, como RF, AlexNet, LeNet-5, CNN e BiLSTM.

3.5 Considerações

Após análise dos trabalhos citados neste Capítulo, é possível observar que técnicas de AM e DL tem sido utilizadas com sucesso em várias pesquisas no contexto de IDSs.

Para a construção de um modelo de AM para detecção de ataques a redes de computadores, é imprescindível que se tenha bons *datasets* para realizar o treino e teste do algoritmo. Estes *datasets* devem possuir tráfegos reais e recentes para que o algoritmo possa ser preparado para um ambiente real.

Além disso, observa-se que técnicas de pré-processamento dos dados do *dataset* são imprescindíveis na redução de dimensionalidade, pois isso melhora o desempenho do algoritmo, ensinando-o quais os atributos são determinantes na detecção de ataques. Outro ganho no *dataset*, é a conversão de tipos de atributos, transformando, como exemplo, texto em numérico.

No quesito algoritmos, há muitas propostas promissoras, que apontam para modelos de DL cada vez mais sofisticados para esta tarefa. Os cuidados com hiperparâmetros são fundamentais no treinamento de um modelo, pois evita um modelo superajustado ou subajustado.

Diante disso, este trabalho propõe um modelo que seja rápido, com baixo custo computacional e eficiente na classificação correta de tráfego de rede, principalmente em ataques.

4 METODOLOGIA

Neste capítulo serão descritos os passos para alcançar os objetivos da pesquisa. Em um primeiro momento, foram simulados ataques a uma rede de computadores em uma Universidade. Esta simulação teve como objetivo a obtenção de um *dataset*, contendo tráfego malicioso e benigno.

De posse dos dados, foram aplicadas técnicas de redução de dimensionalidade, a fim de obter-se um *dataset* mais enxuto, mas que representasse o tráfego e pudesse ser analisado em menor tempo pelos algoritmos. Após a redução de dimensionalidade, foram aplicados ao *dataset* os algoritmos estudados.

Inicialmente aplicou-se os algoritmos NB, SVM, RF, MLP, Tree-CNN (ReLU), Tree-CNN (Softmax) e Tree-CNN (SRS) ao *dataset* próprio, a fim de validar o modelo proposto nesta pesquisa com outros algoritmos de AM e DL.

Posteriormente, o modelo proposto foi aplicado em um ambiente corporativo, a fim de comparar os resultados em ambientes distintos.

Como parte final da avaliação do modelo proposto nesta pesquisa, foi realizada a comparação com o modelo apresentado por Manimurugan et al. (2020), que é composto por uma DBN, onde foi utilizado o *dataset* CIC-IDS2017. Ambos modelos, tanto o proposto nesta pesquisa quanto o proposto por Manimurugan et al. (2020), foram aplicados aos *datasets* próprio e CIC-IDS2017.

A título de informação, os procedimentos procedimentos de treino e teste dos algoritmos, foram realizados em um computador equipado com processador Intel(R) Xeon(R) CPU E5-2620 v3 ®, 16 *Gigabytes* (GB) de memória *Random-access Memory* (RAM).

4.1 Análise do tráfego de rede

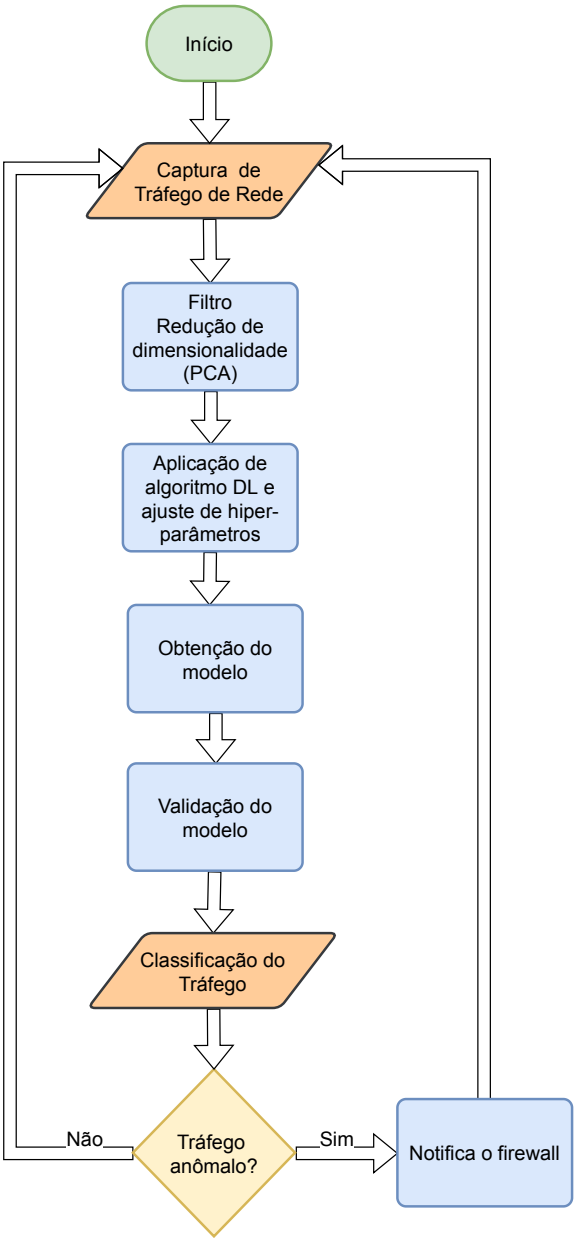
O fluxo de análise do tráfego de rede até a classificação e comunicação ao *firewall*, pode ser visto na Figura 4.1.

A princípio o coletor de tráfego é colocado em paralelo ao tráfego de rede, para que não cause lentidão ou interrupção da rede. Então, o tráfego coletado é passado pelo filtro, que tem como objetivo selecionar os atributos mais relevantes. Posteriormente, é submetido ao algoritmo para classificação, que emitirá um alerta de ataque, caso detecte tráfego anômalo.

Caso o modelo detecte que há tráfego malicioso, então, o *firewall* é notificado para que aquela conexão seja interrompida. Na comunicação ao *firewall* são enviados atributos que iden-

tificam somente a conexão maliciosa, evitando assim o bloqueio de conexões com características similares, como por exemplo, tamanho de pacotes iguais, porém com conteúdos distintos.

Figura 4.1 – Metodologia de análise do tráfego



Fonte: Do autor (2021)

4.2 Construção do *dataset*

Para obtenção do *dataset* foi utilizado um ambiente de uma Universidade. Este ambiente é composto por uma rede em topologia *Local Area Network* (LAN) comum, composta

por equipamentos como *Switches*, Roteador e *Firewall*, segmentado em sub-redes para cada departamento.

O ambiente é composto por 8 departamentos e uma sala de servidores. Na Tabela 4.1 são exibidos os nomes e quantidades de computadores.

Tabela 4.1 – Distribuição de máquinas entre departamentos

Departamento	Quantidade de Computadores
Dep1	50
Dep2	30
Dep3	45
Dep4	40
Dep5	40
Dep6	15
Dep7	45
Dep8	45
Servidores	15

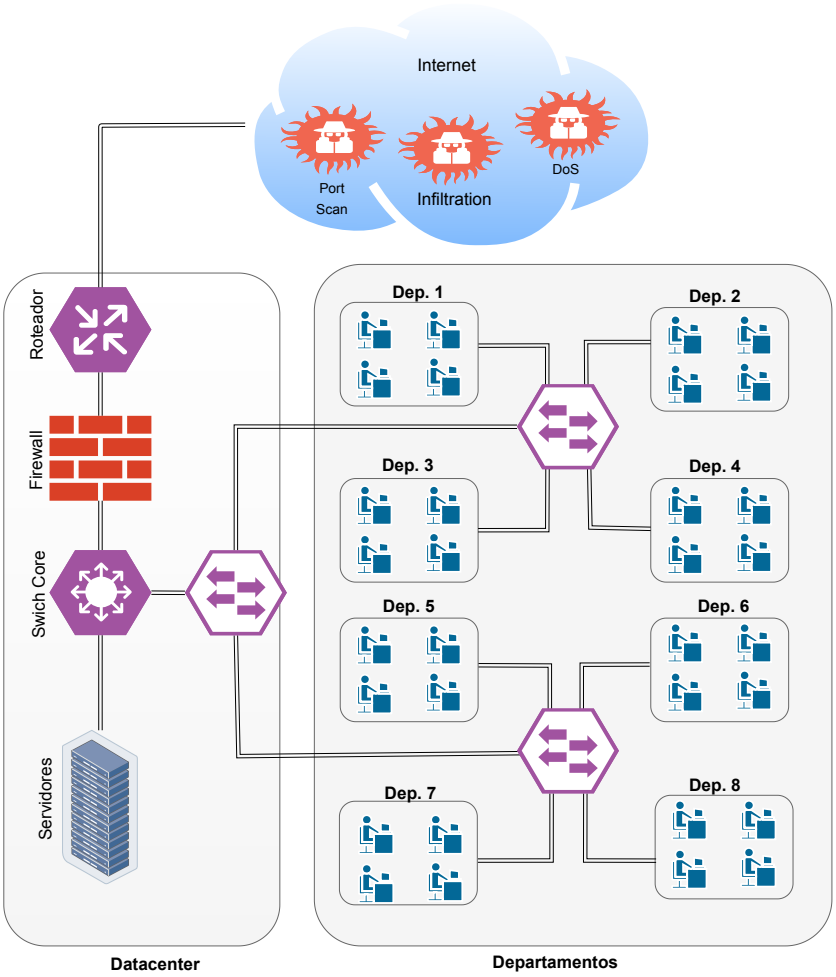
Fonte: Do autor (2021)

No ambiente dos departamentos, há uma diversidade de Sistemas Operacionais instalados nos computadores, tais como: Microsoft (Windows 7, Windows 8.1 e Windows 10) e a distribuição Linux Ubuntu. Além do Sistema Operacional, os computadores possuem clientes de *e-mail*, suíte de escritório (textos, planilhas, etc), navegadores de acesso à Internet e demais *softwares* básicos necessários ao ambiente.

Na sala de servidores, foram instalados em cada equipamentos o Sistema Operacional Microsoft Windows versões 2012 e 2016. Nestes servidores foram instalados serviços como autenticação centralizada, compartilhamento de diretórios, servidores de *e-mail*, *File Transfer Protocol* (FTP) e *Hypertext Transfer Protocol* (HTTP).

A representação gráfica da topologia pode ser observada na Figura 4.2.

Figura 4.2 – Cenário 1 - Ambiente de obtenção do *dataset*



Fonte: Do autor (2021)

4.2.1 Execução do ataque

Para execução dos ataques foi utilizada a distribuição Linux denominada Kali Linux. Esta distribuição é voltada para auditoria e segurança de computadores e redes. Ela é muito usada na área de segurança, pois conta com um conjunto de ferramentas pré instaladas que facilitam o processo de auditoria.

Na simulação do ataque do tipo *Infiltration*, a dinâmica do ataque ocorreu com o envio de um *worm* por *e-mail*. Neste caso o usuário que recebeu o *e-mail* executou este *worm*, que após instalado no computador alvo, criou-se uma *backdoor* no computador remoto, permitindo assim o acesso a ele. Além disso, este *worm* se espalhou pela rede por meio de compartilhamentos de diretórios, expondo vulnerabilidades de outras máquinas ligadas a rede.

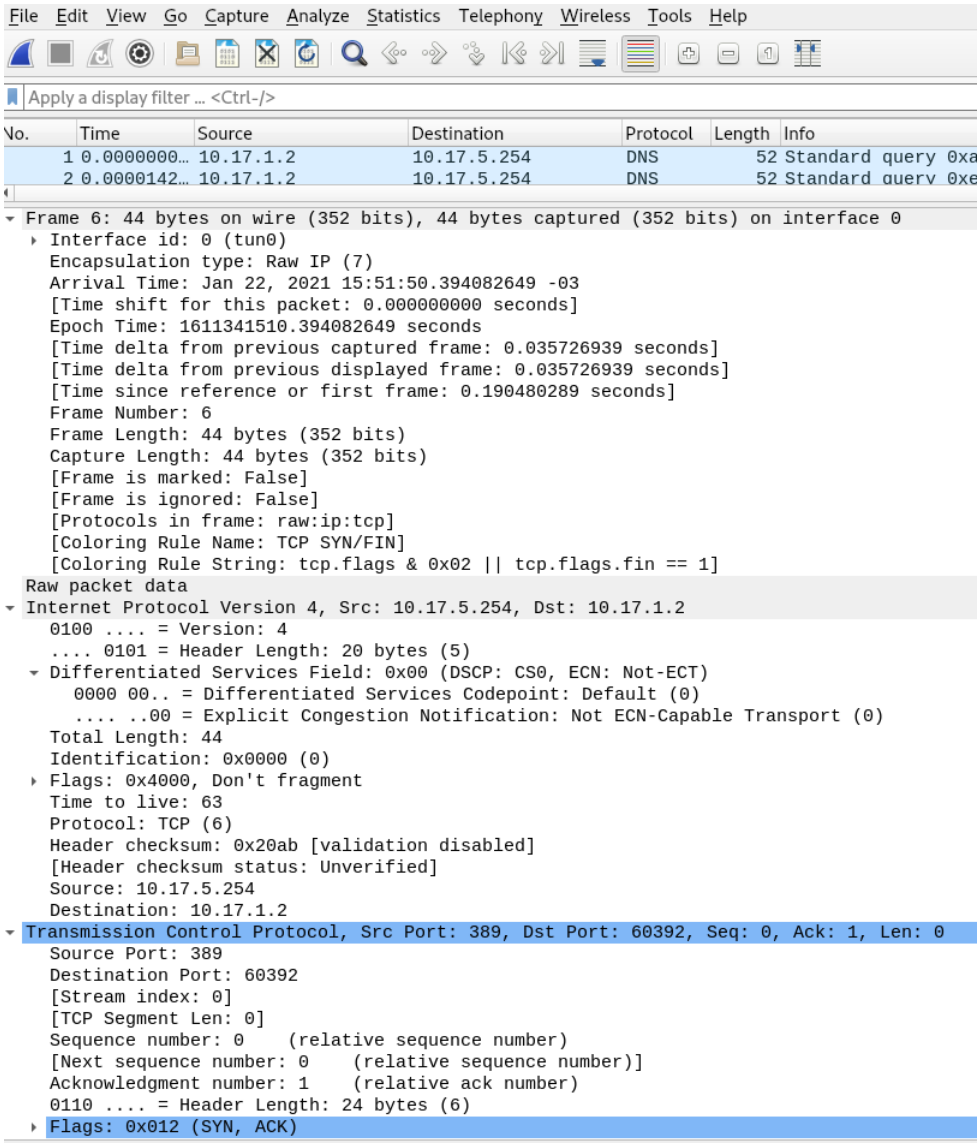
Posteriormente, efetuou um *Port Scan*, para realizar o mapeamento de portas TCP e UDP abertas nos servidores. Para o *scan* foi utilizada a ferramenta Nmap, presente no Kali Linux. Foi executada a varredura das sub-redes em busca de portas abertas que pudessem ser utilizadas para o controle remoto de alguma máquina na rede.

Após o mapeamento das vulnerabilidades, o ataque então foi concretizado.

4.2.2 Captura de tráfego

Todo o tráfego do período de ataque foi capturado e armazenado com a ferramenta *Wireshark* (COMBS, 1998). No total foram capturados 276.643 registros, distribuídos entre normais e ataques. Conforme pode ser observado na Figura 4.3, em apenas uma conexão existem muitos atributos.

Figura 4.3 – Pacote do tráfego capturado



Fonte: Do autor (2021)

A fim de obter um registro da conexão de forma enxuta e objetiva, para que o algoritmo possa analisar de forma mais rápida, faz-se necessário a aplicação de técnicas de redução de dimensionalidade do tráfego capturado. Para esta tarefa, a técnica de PCA foi aplicada ao tráfego coletado, permitindo assim a seleção de características fundamentais para classificação do tráfego.

Dessa forma, após a aplicação da PCA, o número de atributos do *dataset* foi reduzido de 41 para 12.

Em seguida, o tráfego foi classificado manualmente no *dataset*, e inserido o atributo *Label*, assumindo valores como *BENIGN* para quando o pacote pertence a uma conexão normal e *DDoS, Infiltration, Web* e *Brute force*, conforme cada tipo de ataque simulado.

Na Figura 4.4 é exibido um trecho do *dataset* no formato *Attribute-Relation File Format* (ARFF), posteriormente ao tratamento para redução de dimensionalidade. É possível observar que, com exceção do atributo *Label*, todos outros são numéricos. Essa conversão de texto para numérico auxilia o algoritmo na análise do tráfego, fazendo com que o processamento seja mais ágil.

Figura 4.4 – Trecho do *dataset* após redução de dimensionalidade

```
@attribute ' Destination Port' numeric
@attribute ' Flow Duration' numeric
@attribute ' Total Fwd Packets' numeric
@attribute ' Total Backward Packets' numeric
@attribute 'Total Length of Fwd Packets' numeric
@attribute ' Total Length of Bwd Packets' numeric
@attribute ' Min Packet Length' numeric
@attribute ' Max Packet Length' numeric
@attribute ' Packet Length Mean' numeric
@attribute ' Packet Length Std' numeric
@attribute ' Packet Length Variance' numeric
@attribute ' Label' {BENIGN,Infiltration}

@data
443,188931,9,7,528,4513,0,1430,296.529412,523.33344,273877.8897,BENIGN
443,213931,10,8,563,4549,0,1430,269.052632,500.368028,250368.1637,BENIGN
138,28,13,0,2808,0,216,216,216,0,0,BENIGN
80,119961783,17,16,668,780,0,384,42.588235,110.693165,12252.97683,BENIGN
53,61146,1,1,47,79,47,79,57.666667,18.475209,341.333333,BENIGN
53,1202457,6,2,222,304,37,152,62.555556,50.710233,2571.527778,BENIGN
53,160,2,2,72,136,36,68,48.8,17.527122,307.2,BENIGN
53,61607,4,2,168,310,42,155,74.285714,55.138354,3040.238095,BENIGN
444,42664070,5,6,141,291,0,267,36,77.399789,5990.727273,Infiltration
443,426640,5,6,141,291,0,267,36,77.399789,5990.727888,Infiltration
444,4070,5,6,142,291,0,267,36,77.399789,5990.727279,Infiltration
444,17765,5,6,141,291,0,267,36,77.399789,5933.727273,Infiltration
444,302776,5,6,141,291,0,267,36,77.399789,990.727273,Infiltration
444,66545,5,6,141,291,0,267,36,77.399789,1190.727273,Infiltration
444,878888,5,6,141,291,0,267,36,77.399789,3345.727273,Infiltration
123,134,2,2,96,96,48,48,48,0,0,BENIGN
53,205,2,2,142,242,71,121,91,27.386128,750,BENIGN
389,120,2,2,326,326,163,163,163,0,0,BENIGN
443,195381,13,12,575,14015,0,4380,561.153846,1058.446473,1120308.935,BENIGN
443,195240,14,14,575,14053,0,2920,504.413793,821.489871,674845.6084,BENIGN
443,195362,13,13,575,14015,0,2920,540.37037,872.321269,760944.396,BENIGN
443,12877,1,2,0,44,0,38,11,18.220867,332,BENIGN
```

Fonte: Do autor (2021)

4.3 Obtenção do modelo proposto

O modelo proposto nesta pesquisa para detecção de intrusão em redes de computadores, é composto pela combinação de uma *Convolutional Neural Network* (CNN) hierárquica, deno-

minada Tree-CNN, e a função de ativação *Soft-Root-Sign* (SRS). No contexto deste trabalho, o modelo é denominado Tree-CNN (SRS).

Por meio deste modelo, pretende-se obter resultados de acurácia acima dos demais modelos. Além disso, pretende-se obter menor tempo de treinamento e evitar o problema de esquecimento de dados na fase de retreinamento de novos modelos, conforme relatado em (ROY; PANDA; ROY, 2020). As etapas de obtenção do modelo são descritas nas subseções a seguir.

4.3.1 Implementação

Na implementação dos modelos DL foi utilizada a linguagem de programação Python (ROSSUM; JR, 1995) e as bibliotecas TensorFlow (ABADI et al., 2015), Keras (CHOLLET et al., 2015), Numpy (OLIPHANT, 2006) e Pandas (MCKINNEY et al., 2010).

A linguagem Python foi escolhida por ser uma linguagem de alto nível e ser muito utilizada no meio acadêmico para desenvolvimento de pesquisas em AM e DL. Além disso, possui bibliotecas de alto nível para implementação dos modelos, como é o caso da Keras que faz interface com a TensorFlow.

Nesta fase, é escrito o código base para execução dos modelos, ou seja, o estrutura e disposição dos métodos no bloco de código. Também podem ser aplicados hiperparâmetros iniciais para os modelos, porém, os definitivos somente são definidos após o treino, que é onde ajusta-se os valores, para que o modelo não fique superajustado ou subajustado.

Para cada modelo de DL foi utilizada a respectiva função de ativação. As funções Softmax e ReLU são nativas na Keras, portanto, não foi necessário escrevê-las.

Contudo, o modelo proposto, utiliza a função de ativação *Soft-Root-Sign* (SRS), portanto, foi necessária a implementação.

Na Subseção 4.3.2, a seguir, são descritos os procedimentos do treinamento do modelo Tree-CNN (SRS).

4.3.2 Treinamento

Para obtenção do modelo, no treinamento e teste foi utilizado o *dataset* próprio. Dessa forma, o *dataset* foi dividido em 60% para treino, 20% para teste e 20% para validação. Após definidos os parâmetros de divisão do *dataset*, então, são definidos os hiperparâmetros de treinamento da rede. Conforme ressalta Pina et al. (2019), o desempenho da rede é diretamente afetado pelos valores definidos para os hiperparâmetros, portanto, a definição de hiperparâmetros com valores mais justos, proporcionam resultados de classificação melhores.

Desta forma, após exaustivas tentativas de ajustes, foi possível obter os valores apresentados a seguir.

(a) Taxa de aprendizado (learning rate): 0.1

(b) Épocas (*epochs*): 50

(c) Tamanho de lote (*batch_size*): 10

(d) Taxa de controle de *overfitting* (*dropout*): 0.5

(e) Taxa de controle dos pesos da rede (*momentum*): 0.8

Após o modelo pronto, foram realizados os testes com os demais modelos e algoritmos, conforme descritos nas sub-seções a seguir.

4.3.3 Validação: comparação com outros modelos de Aprendizado de Máquina

Nesta etapa de testes foram executados os algoritmos NB, SVM, RF, MLP, Tree-CNN (ReLU), Tree-CNN (Softmax) e Tree-CNN (SRS) no *dataset* próprio, a fim de obter métricas de comparação para validação do modelo proposto.

Para a experimentação podem ser usadas ferramentas que já incluem uma vasta gama de algoritmos, como WEKA (WITTEN; FRANK; HALL, 2011). O WEKA é uma das ferramentas mais utilizadas em AM, pois permite a execução dos algoritmos de forma simplificada, sem a necessidade de implementações. O *software* possui um ambiente de trabalho com diversos algoritmos implementados em linguagem de programação Java, além de contar com uma grande quantidade de ferramentas de pré-processamento de dados.

De acordo com Frank, Hall e Witten (2017) o WEKA foi projetado para que seja possível testar de forma mais flexível, os métodos existentes em novos *datasets*, sem a necessidade de codificar nenhum programa.

Contudo, em casos mais específicos, há a necessidade de implementação de alguns modelos em código, como é o caso do modelo proposto neste trabalho. Portanto, neste trabalho, para os algoritmos NB, SVM, RF, MLP foi usado o WEKA, enquanto que para os algoritmos DL foi realizada a implementação em código, conforme descrito na Subseção 4.3.1.

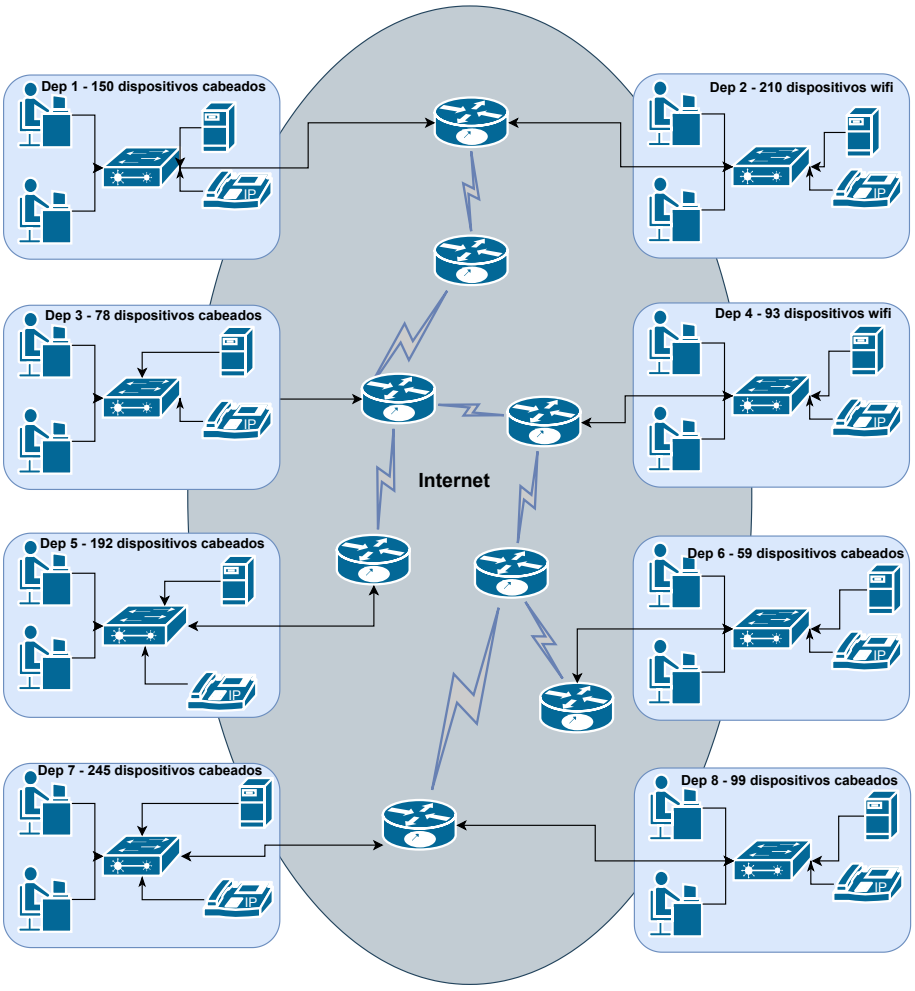
Os resultados obtidos nesta fase, são apresentados na Seção 5.1, onde são exibidos os valores das métricas que cada algoritmo obteve.

4.3.4 Validação: aplicação do modelo em um ambiente corporativo

Seguindo o fluxo de validação do modelo proposto, nesta fase, o modelo foi aplicado a um ambiente corporativo, a fim de avaliar os resultados do modelo num ambiente real.

Uma ilustração do ambiente de validação pode ser observada na Figura 4.5. O ambiente é composto por vários departamentos dispersos geograficamente, porém, interligados em rede.

Figura 4.5 – Cenário 2 - Ambiente de avaliação do modelo



Fonte: Do autor (2021)

A validação ocorreu com a aplicação do modelo em uma rede corporativa, com captura do tráfego em tempo real, seguindo a metodologia proposta. Para que pudesse ser realizada essa operação, foi necessária a colaboração da equipe de segurança da informação do ambiente corporativo.

Dessa forma, foram obtidos os resultados apresentados na Seção 5.2.

4.3.5 Validação: comparação do modelo proposto com outro modelo em DL

Como parte final da validação do modelo proposto, nesta fase, foi utilizado o modelo em DL, denominado DBNIDS, proposto por Manimurugan et al. (2020).

A escolha desse modelo se deu por conta de ser um modelo em DL, tal como o proposto neste trabalho. Além disso, o modelo é recente e obteve resultados muito promissores na detecção dos mesmos tipos de ataques que são estudados neste trabalho.

Nesta fase, o modelo DBNIDS foi aplicado a dois cenários, onde o primeiro cenário é o *dataset* próprio, construído neste trabalho. E, o segundo cenário é o ambiente corporativo.

Posteriormente o Tree-CNN (SRS) foi aplicado ao *dataset* CIC-IDS2017, para obtenção das métricas de desempenho no mesmo *dataset* usado pelo DBNIDS.

Após obtidos os resultados de cada modelo na detecção dos ataques *DDoS, Infiltration, Web* e *Brute force*, foram então gerados os resultados, conforme pode ser observado com mais detalhes no Capítulo 5.

4.4 Métricas de avaliação de desempenho

Para avaliar os resultados obtidos, foi utilizada neste trabalho a Matriz de Confusão (MC) proposta por Pearson (1903). Por meio dela, é possível avaliar a qualidade dos resultados obtidos, utilizando como parâmetro as métricas Acurácia, Precisão, Sensibilidade e Medida-F (F1).

Na Tabela 4.2, é representada uma MC, como exemplo genérico na detecção de intrusão em redes de computadores. Dessa forma, são obtidos os valores de cada variável para composição das equações e obtenção dos resultados para cada métrica.

Tabela 4.2 – Matriz de Confusão

		Resultado	
		Ataque	Benigno
Predição	Ataque	Verdadeiro Positivo (VP)	Falso Positivo (FP)
	Benigno	Falso Negativo (FN)	Verdadeiro Negativo (VN)

Fonte: Do autor (2021)

No contexto deste trabalho, o tráfego é classificado como ataque ou não, sendo assim, a conexão pertencente a um ataque é rotulada com o nome do ataque, enquanto a conexão normal é rotulada como benigno. Dessa forma o modelo gera o resultado para cada classe e então, o resultado é comparado com o predito, obtendo-se assim os valores para as classificações.

Na Tabela 4.3 é possível observar um exemplo, onde, de um total de 150 registros de ataque DDoS, o modelo classificou 100 corretamente e 50 como tráfego normal. Enquanto que para 100 registros de conexão normal, foram classificados corretamente 70, os demais foram falsos positivos.

Tabela 4.3 – Matriz de Confusão - Ataque DDoS

	DDoS	Benigno
DDoS	**100**	30
Benigno	50	**70**

Fonte: Do autor (2021)

Com base nesses resultados, observa-se que, quanto mais acurado o modelo for, maior é a quantidade de VP e VN. Ou seja, o quanto predito que se confirmou.

Após construção da MC, são calculadas as métricas. A obtenção das métricas tem como objetivo apontar o quão eficiente está sendo o modelo, ou seja, se ele está acertando as predições.

Na literatura são encontradas diversas métricas para aferição, porém, para esta pesquisa foram selecionadas as métricas apresentadas a seguir, pois são amplamente utilizadas em trabalhos que tem como objetivo a avaliação de modelos de AM.

(a) Acurácia

A acurácia representa o quanto o modelo foi assertivo nas predições. O valor é obtido por meio da Equação 4.1.

$$Acurácia = \frac{VP + VN}{VP + VN + FP + FN} \tag{4.1}$$

(b) Precisão

Esta métrica tem como objetivo extrair a porcentagem de classificações corretas que se confirmam, com base nas classificações que o modelo apontou como corretas. Sendo assim, Precisão é obtida conforme Equação 4.2.

$$Precisão = \frac{VP}{VP + FP} \tag{4.2}$$

(c) Sensibilidade

É a quantidade de sentenças positivas corretamente identificadas, ou seja, quantas vezes a expressão completa procurada foi encontrada na análise. De acordo com Sharafaldin,

Lashkari e Ghorbani (2018), Sensibilidade é a razão dos fluxos de ataque classificados corretamente (VP) sobre a soma das predições dadas como corretas (VP + FN). Essa métrica também é reconhecida como *Detection Rate* (DR) (LIU; LANG, 2019). Ela é obtida por meio da Equação 4.3.

$$Sensibilidade = \frac{\text{VP}}{\text{VP} + \text{FN}} \tag{4.3}$$

(d) Medida-F (F1)

Esta métrica foi proposta por Rijsbergen (1979), e tem como objetivo uma combinação harmônica de Precisão e Sensibilidade em uma única medida (SHARAFALDIN; LASHKARI; GHORBANI, 2018). Caso o valor de F1 seja muito baixo, é um indicativo que a Precisão ou Sensibilidade estão baixos. Na Equação 4.4, é apresentado o cálculo para encontrar esta métrica.

$$F1 = 2 \times \frac{Precisão \times Sensibilidade}{Precisão + Sensibilidade} \tag{4.4}$$

As medidas de desempenho são calculadas de acordo com os valores de Acurácia, Precisão, Sensibilidade e Medida-F. A partir desses resultados, é possível identificar qual algorítimo obteve melhor desempenho.

5 RESULTADOS

Neste capítulo são apresentados os resultados obtidos conforme os experimentos descritos no Capítulo 4. Estes resultados são fundamentais para determinar se o modelo proposto nesta pesquisa é vantajoso em relação aos demais, e promissor na identificação de ataques a redes de computadores.

Inicialmente, foram obtidos os resultados dos algoritmos aplicados ao *dataset* próprio. Após validado o modelo proposto, este então foi aplicado em um ambiente corporativo diferente do ambiente em que o *dataset* próprio foi criado.

Em seguida, o modelo proposto neste trabalho, *Tree-CNN (SRS)*, foi comparado ao modelo proposto por Manimurugan et al. (2020), baseado em DBN. Neste caso, o modelo proposto neste trabalho foi aplicado ao *dataset* CIC-IDS2017, utilizado pelo modelo DBNIDS. E, por fim, o modelo DBNIDS foi aplicado ao *dataset* próprio.

Para avaliar os modelos, foram tomadas como base as métricas Acurácia, Sensibilidade, Precisão e Medida-F. Neste contexto, o algoritmo que obtém o valor mais alto dessas métricas, indica maior assertividade na detecção dos tipos de ataques estudados.

5.1 Avaliação de performance do Tree-CNN (SRS) em *dataset* próprio

Nesta etapa todos os modelos foram aplicados ao *dataset* próprio, a fim de avaliar o desempenho de cada um deles.

Para que os modelos NB, SVM, RF, MLP, Tree-CNN (ReLU), Tree-CNN (Softmax) e Tree-CNN (SRS) sejam avaliados, esta fase é composta pelas etapas de treino e teste.

Os resultados apresentados nas Tabelas 5.1, 5.2, 5.3 e 5.4, são relativos a fase de treino de todos algoritmos NB, SVM, RF, MLP, Tree-CNN (ReLU), Tree-CNN (Softmax) e Tree-CNN (SRS) no *dataset* próprio.

Tabela 5.1 – Resultados para detecção de ataque **DDoS** em *dataset* próprio

Modelo	Acurácia	Sensibilidade	Precisão	Medida-F
NB	0,79	0,83	0,80	0,80
SVM	0,82	0,84	0,83	0,82
RF	0,85	0,86	0,85	0,85
MLP	0,89	0,89	0,88	0,89
Tree-CNN (ReLU)	0,93	0,94	0,95	0,93
Tree-CNN (Softmax)	0,95	0,94	0,94	0,94
Tree-CNN (SRS)	0,99	0,98	0,97	0,98

Fonte: Do autor (2021)

Tabela 5.2 – Resultados para detecção de ataque ***Infiltration*** em *dataset* próprio

Modelo	Acurácia	Sensibilidade	Precisão	Medida-F
NB	0,78	0,83	0,79	0,79
SVM	0,82	0,84	0,81	0,82
RF	0,85	0,86	0,83	0,85
MLP	0,89	0,89	0,87	0,89
Tree-CNN (ReLU)	0,94	0,93	0,94	0,93
Tree-CNN (Softmax)	0,96	0,94	0,93	0,94
Tree-CNN (SRS)	0,98	0,98	0,99	0,98

Fonte: Do autor (2021)

Tabela 5.3 – Resultados para detecção de ataque ***Web*** em *dataset* próprio

Modelo	Acurácia	Sensibilidade	Precisão	Medida-F
NB	0,77	0,82	0,80,	0,79
SVM	0,81	0,82	0,83	0,81
RF	0,84	0,86	0,84	0,84
MLP	0,88	0,89	0,88	0,88
Tree-CNN (ReLU)	0,91	0,94	0,95	0,92
Tree-CNN (Softmax)	0,94	0,94	0,94	0,94
Tree-CNN (SRS)	0,99	0,98	0,96	0,98

Fonte: Do autor (2021)

Tabela 5.4 – Resultados para detecção de ataque ***Brute Force*** em *dataset* próprio

Modelo	Acurácia	Sensibilidade	Precisão	Medida-F
NB	0,80	0,82	0,80,	0,80,
SVM	0,83	0,88	0,84	0,84
RF	0,85	0,89	0,85	0,86
MLP	0,89	0,92	0,87	0,87
Tree-CNN (ReLU)	0,94	0,96	0,94	0,94
Tree-CNN (Softmax)	0,95	0,94	0,94	0,91
Tree-CNN (SRS)	0,98	0,99	0,99	0,98

Fonte: Do autor (2021)

Conforme pode ser observado, o modelo proposto Tree-CNN (SRS), alcançou resultados superiores aos demais modelos na detecção dos ataques *DDoS, Infiltration, Web* e *Brute force* na fase de treino, obtendo resultados promissores de acurácia de (0,99/0,98/0,99/0,98), Sensibilidade (0,98/0,98/0,98/0,99), Precisão (0,97/0,99/0,96/0,99) e Medida-F (0,98/0,98/0,98/0,98), respectivamente para cada ataque citado.

Como destaque nesta fase, observa-se que os modelos profundos Tree-CNN (ReLU), Tree-CNN (Softmax) e Tree-CNN (SRS) apresentaram resultados superiores aos modelos rasos NB, SVM, RF, MLP.

Após a etapa de treino, o Tree-CNN (SRS) foi aplicado ao *dataset* próprio para teste de performance, e os resultados são apresentados na Tabela 5.5.

Tabela 5.5 – Métricas do Tree-CNN (SRS) na fase de teste em *dataset* próprio

	Acurácia	Sensibilidade	Precisão	Medida-F
DDoS	0,98	0,98	0,96	0,98
Infiltration	0,97	0,97	0,98	0,97
Web attack	0,98	0,97	0,95	0,97
Brute force	0,97	0,98	0,98	0,97

Fonte: Do autor (2021)

Na fase de teste do Tree-CNN (SRS) no *dataset* próprio, os resultados positivos se mantiveram, ocorrendo pouca variação, conforme pode ser observado na Tabela 5.5. Portanto, a configuração do modelo foi mantida e usada para aplicação em outros ambientes, pois considerando os ajustes realizados, foram obtidos altos valores com essa configuração.

Após a apresentação dos resultados que indicam a eficiência do modelo Tree-CNN (SRS), este então foi submetido a outra fase de teste. Porém, desta vez em um ambiente corporativo diferente, conforme descrito na Subseção 4.3.4.

5.2 Avaliação do Tree-CNN (SRS) em outro cenário

Nesta fase o Tree-CNN (SRS) foi avaliado no contexto de uma empresa de médio porte, conforme cenário descrito na Figura 4.5, onde foi aplicado ao tráfego em tempo real e os resultados obtidos são apresentados na Tabela 5.6.

Tabela 5.6 – Resultados do modelo Tree-CNN (SRS) em ambiente corporativo

	Acurácia	Sensibilidade	Precisão	Medida-F
DDoS	0,97	0,96	0,97	0,96
Infiltration	0,98	0,97	0,97	0,97
Web attack	0,99	0,98	0,98	0,98
Brute Force	0,98	0,97	0,98	0,97

Fonte: Do autor (2021)

Conforme pode ser observado o Tree-CNN (SRS) obteve valores próximos aos obtidos quando aplicado ao *dataset* próprio. Por ser um ambiente distinto, esperava-se que o modelo

apresentasse resultados inferiores, porém a variação entre resultados em ambos ambientes foi muito baixa, o que indica que o Tree-CNN (SRS) está sendo bem assertivo na identificação de ataques *DDoS, Infiltration, Web* e *Brute force*.

5.3 Avaliação do Tree-CNN (SRS) com o modelo DBNIDS

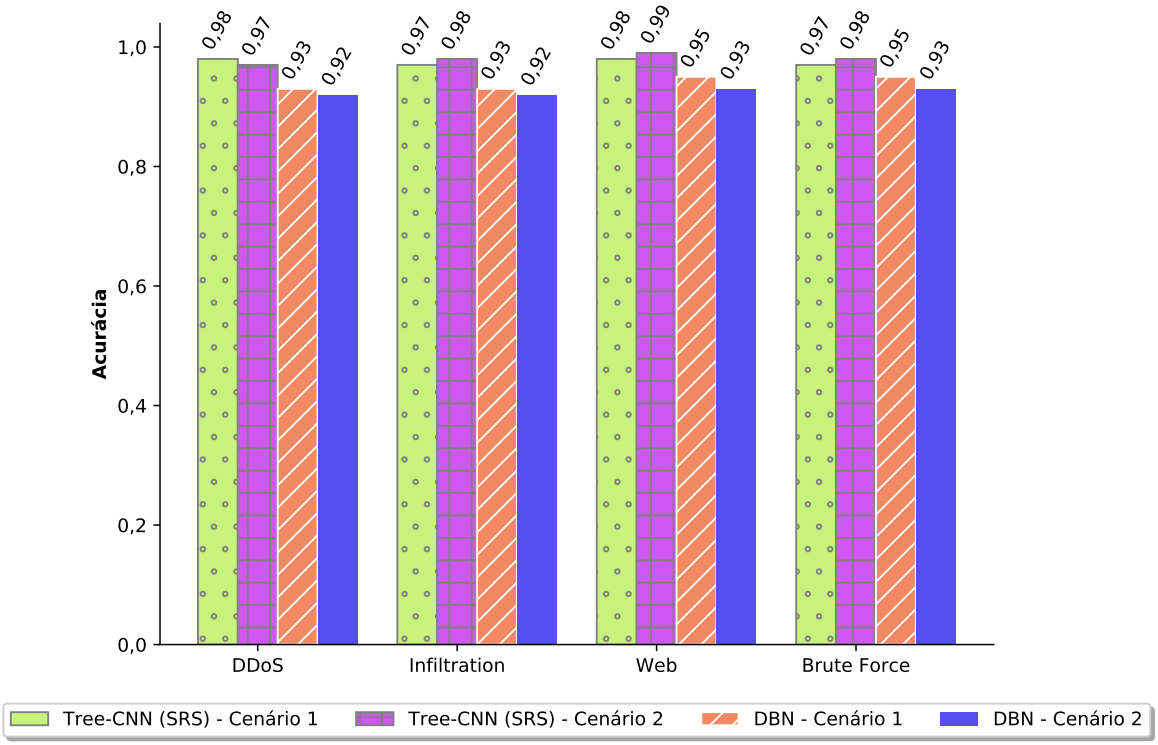
Nesta última fase de avaliação, o modelo proposto Tree-CNN (SRS) foi comparado ao modelo proposto por Manimurugan et al. (2020), denominado DBNIDS.

O modelo DBNIDS foi escolhido para comparação por ser uma pesquisa recente onde estudou-se os mesmos ataques estudados nesta pesquisa. Além disso, o DBNIDS obteve métricas expressivas de acurácia na detecção dos ataques *DDoS, Infiltration, Web* e *Brute force*.

Para realizar a comparação, o modelo DBNIDS foi testado nos mesmos cenários em que o Tree-CNN (SRS) foi testado, com os mesmos tipos de ataques.

Os resultados obtidos podem ser visualizados no gráfico apresentado na Figura 5.1. Onde, Cenário 1 refere-se ao *dataset* próprio, e Cenário 2 refere-se ao ambiente corporativo.

Figura 5.1 – Comparação de acurácia entre o Tree-CNN (SRS) e DBNIDS



Fonte: Do autor (2021)

Analisando os resultados, observa-se que o Tree-CNN (SRS) apresentou melhor acurácia que o modelo DBNIDS em ambos cenários. Em ataques do tipo *Infiltration* no cenário 2, o DBNIDS obteve acurácia de 0,92 enquanto o Tree-CNN (SRS) obteve 0,98.

Além dos valores de acurácia, o Tree-CNN (SRS) apresentou melhores resultados que o DBNIDS em Medida-F, onde o melhor resultado alcançado pelo DBNIDS foi de 0,95 para os ataques *Brute Force* e *Web*, enquanto o Tree-CNN (SRS) alcançou Medida-F de 0,98 nos demais ataques.

Como parte final do processo de validação, o Tree-CNN (SRS) também foi aplicado ao *dataset* CIC-IDS2017 para análise de desempenho e comparação ao DBNIDS, pois os resultados deste modelo foram obtidos neste *dataset*.

Os resultados são apresentados na Tabela 5.7, onde pode ser observado que o Tree-CNN (SRS) obteve resultados de acurácia superiores aos do DBNIDS.

Tabela 5.7 – Acurácia obtida pelo Tree-CNN (SRS) e o DBNIDS no *dataset* CIC-IDS2017

	DDoS	Infiltration	Brute Force	Web
DBN	0,96	0,96	0,97	0,98
Tree-CNN (SRS)	0,99	0,98	0,98	0,99

Fonte: Do autor (2021)

Outra comparação realizada é em relação ao tempo para classificação. Neste caso, foi observado que o Tree-CNN (SRS) obteve uma detecção mais rápida, aumentando a velocidade de detecção dos ataques em 36% em média quando comparado ao DBNIDS, conforme pode ser visto na Tabela 5.8.

Tabela 5.8 – Tempo de execução para a classificação dos ataques pelo DBNIDS e o Tree-CNN (SRS), em fase de testes, utilizando o *dataset* CIC-IDS2017

Modelo	Tempo (segundos)
DBNIDS	1.903
Tree-CNN (SRS)	1.201

Fonte: Do autor (2021)

6 CONCLUSÃO

Esse trabalho procurou obter um novo modelo de detecção de intrusão em redes de computadores, por meio de algoritmos de *Deep Learning* (DL). Para isso foi proposto o modelo Tree-CNN (SRS), que é composto por *Deep Convolution Neural Networks*s (DCNNs) e a função de ativação *Soft-Root-Sign* (SRS).

A função de ativação desempenha um papel crítico em redes neurais profundas, e o uso de uma função de ativação mais eficaz foi explorado neste trabalho. Assim, a *Soft-Root-Sign* (SRS) apresentou bons resultados quanto à Acurácia, Sensibilidade, Precisão e Medida-F. Além disso, o ajuste de hiperparâmetros no treinamento do modelo, mostrou-se muito eficaz para que fosse possível obter um modelo equilibrado, evitando *overfitting* ou *underfitting*.

Com base nos resultados obtidos nos testes em diferentes ambientes, o modelo proposto neste trabalho, Tree-CNN (SRS), demonstrou melhor desempenho que os demais algoritmos testados, inclusive em relação ao modelo DBNIDS, proposto por (MANIMURUGAN et al., 2020). Além disso, observou-se que o modelo proposto foi, em média, 36% mais rápido na detecção dos ataques, em comparação ao modelo DBNIDS. Esse resultado é muito significativo, principalmente em um ambiente crítico, onde a detecção precoce de ataques é essencial para evitar danos.

A complexidade de tempo utilizando a função de ativação SRS foi baixa em comparação com outros algoritmos, pois a SRS possui melhor desempenho de generalização, além de maior velocidade de aprendizado para geração do modelo, através da normalização em lote, acelerando o treinamento da rede profunda. Este trabalho valida o uso da função de ativação SRS no algoritmo Tree-CNN. Os experimentos mostraram taxas de aprendizado significativamente maiores, e comprovaram as propriedades da SRS, como suavidade, não monotonicidade e delimitação. Além disso, a propriedade limitada da função de ativação SRS difere de outras funções de ativação.

A taxa de assertividade do Tree-CNN e SRS apresentou valores maiores em relação às métricas obtidas com outros algoritmos, validando a proposta de utilização desta solução.

Conforme pode ser observado nos resultados apresentados, o modelo proposto se mostra superior aos demais modelos, indicando assim um modelo promissor na detecção de intrusões em redes de computadores.

Dessa forma, demonstrou-se nesta pesquisa, por meio dos resultados obtidos, que com o modelo proposto foi possível reduzir o número de falsos positivos na detecção de intrusão em tráfegos de redes de computadores, validando assim a proposta deste trabalho.

Para trabalhos futuros, sugere-se que o modelo seja avaliado em mais ambientes, principalmente com alto fluxo de tráfego, onde seja possível avaliar a classificação de outros tipos de ataques e o tempo para classificação nesses ambientes.

6.1 Publicações

Como resultado desta pesquisa, o artigo *"Intrusion Detection System Based on Fast Hierarchical Deep Convolutional Neural Network"*, foi produzido e aceito para publicação, conforme dados bibliográficos a seguir:

- MENDONCA, Robson V.; TEODORO, Arthur A. M.; ROSA, Renata L.; SAADI, Muhammad; MELGAREJO, Dick Carrillo; NARDELLI, Pedro H. J.; RODRIGUEZ, Demostenes Z.. Intrusion Detection System Based on Fast Hierarchical Deep Convolutional Neural Network. IEEE Access, [S.L.], v. 9, p. 61024-61034, 2021. Institute of Electrical and Electronics Engineers (IEEE). http://dx.doi.org/10.1109/access.2021.3074664.

REFERÊNCIAS

ABADI, M. et al. **TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems**. 2015. Software available from tensorflow.org. Disponível em: <http://tensorflow.org/>.

ACKLEY, D. H.; HINTON, G. E.; SEJNOWSKI, T. J. A learning algorithm for boltzmann machines. **Cognitive Science**, v. 9, n. 1, p. 147–169, 1985. ISSN 03640213.

AKSU, D.; AYDIN, M. A. Detecting port scan attempts with comparative analysis of deep learning and support vector machine algorithms. In: **2018 International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism (IBIGDELFT)**. [S.l.: s.n.], 2018. p. 77–80.

AL-QATF, M. et al. Deep Learning Approach Combining Sparse Autoencoder With SVM for Network Intrusion Detection. **IEEE Access**, IEEE, v. 6, p. 52843–52856, 2018.

ARTHUR, M. P. An SVM-based Multiclass IDS for Multicast Routing Attacks in Mobile Ad Hoc Networks. In: **2018 International Conference on Advances in Computing, Communications and Informatics (ICACCI)**. [S.l.: s.n.], 2018. p. 363–368.

BALAKRISHNAN, S.; K, V.; A, K. Intrusion Detection System Using Feature Selection and Classification Technique. **International Journal of Computer Science and Application**, Science and Engineering Publishing Company, v. 3, n. 4, p. 145, 2014.

BINDRA, N.; SOOD, M. Detecting DDoS Attacks Using Machine Learning Techniques and Contemporary Intrusion Detection Dataset. **Automatic Control and Computer Sciences**, 2019. ISSN 1558108x.

BONESSO, D. Estimação dos Parâmetros do Kernel em um Classificador SVM na Classificação de Imagens Hiperespectrais em uma Abordagem Multiclasse. p. 108, 2013.

BORGES, H. B.; NIEVOLA, J. C. Comparing the dimensionality reduction methods in gene expression databases. **Expert Systems with Applications**, Elsevier, v. 39, n. 12, p. 10780–10795, 2012.

BOSER, B. E.; GUYON, I. M.; VAPNIK, V. N. A training algorithm for optimal margin classifiers. In: **Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory**. [S.l.]: ACM Press, 1992. p. 144–152.

BRAGA, A. de P.; FERREIRA, A. C. P. de L.; LUDERMIR, T. B. **Redes Neurais Artificiais : Teoria e Aplicações**. Rio de Janeiro: LTC Editora, 2007. ISBN 978-8521615644.

BREIMAN, L. Bagging Predictors. **Machine Learning**, v. 24, n. 2, p. 123–140, ago. 1996. ISSN 1573-0565.

BREIMAN, L. Random Forest draft. **Machine Learning**, p. 1–33, 2001.

BRIDLE, J. S. Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition. In: **Neurocomputing**. [S.l.]: Springer, 1990. p. 227–236.

BUCZAK, A. L.; GUVEN, E. A Survey of Data Mining and Machine Learning Methods for Cyber Security Intrusion Detection. **IEEE Communications Surveys and Tutorials**, 2016. ISSN 1553877x.

CAN, O.; SAHINGOZ, O. K. A survey of Intrusion Detection Systems in wireless sensor networks. In: **6th International Conference on Modeling, Simulation, and Applied Optimization, ICMSAO 2015 - Dedicated to the Memory of Late Ibrahim El-Sadek**. [S.l.: s.n.], 2015. ISBN 9781467366014.

CHAWLA, N. V. et al. Smote: Synthetic minority over-sampling technique. **Journal of Artificial Intelligence Research**, 2002. ISSN 10769757.

CHOCKWANICH, N.; VISOOTTIVISETH, V. Intrusion Detection by Deep Learning with TensorFlow. In: **2019 21st International Conference on Advanced Communication Technology (ICACT)**. [S.l.: s.n.], 2019. p. 654–659.

CHOLLET, F. et al. **Keras**. GitHub, 2015. Disponível em: <https://github.com/fchollet/keras>.

CHOWDHURY, M. M. U. et al. A few-shot deep learning approach for improved intrusion detection. In: **2017 IEEE 8th Annual Ubiquitous Computing, Electronics and Mobile Communication Conference (UEMCON)**. [S.l.: s.n.], 2017. p. 456–462.

CHU, W. L.; LIN, C. J.; CHANG, K. N. Detection and classification of advanced persistent threats and attacks using the Support Vector Machine. **Applied Sciences (Switzerland)**, 2019. ISSN 20763417.

CISCO. **Cisco annual internet report (2018–2023) white paper**. 2019. <https://www.cisco.com/c/en/us/solutions/collateral/executive-perspectives/annual-internet-report/white-paper-c11-741490.html>. (Acessado em 04/03/2021).

COMBS, G. **Wireshark**. 1998. <https://www.wireshark.org/>. (Acessado em 22/01/2021).

DASH, M.; LIU, H. Feature selection for classification. **Intelligent Data Analysis**, v. 1, n. 3, p. 131–156, 1997. ISSN 1088467x.

DAWOUD, A.; SHAHRISTANI, S.; RAUN, C. Deep Learning for network anomalies detection. **Proceedings - International Conference on Machine Learning and Data Engineering, iCMLDE 2018**, IEEE, p. 117–120, 2019.

DINIZ, F. A. et al. RedFace: um sistema de reconhecimento facial baseado em técnicas de análise de componentes principais e autofaces. **Revista Brasileira de Computação Aplicada**, v. 5, n. 1, p. 42–54, 2013. ISSN 2176-6649.

DOMINGOS, P.; PAZZANI, M. On the Optimality of the Simple Bayesian Classifier underZero-One Loss. Machine Learning. **Machine Learning**, v. 29, p. 103–130, 1997.

ELHAG, S. et al. On the combination of genetic fuzzy systems and pairwise learning for improving detection rates on Intrusion Detection Systems. **Expert Systems with Applications**, 2015. ISSN 09574174.

ELSAEIDY, A. et al. Intrusion detection in smart cities using Restricted Boltzmann Machines. **Journal of Network and Computer Applications**, Elsevier Ltd, v. 135, n. September 2018, p. 76–83, 2019. ISSN 10958592.

ESTRELA, V. V.; BOENTE, A. N. P.; GOLDSCHMIDT, R. R. Uma metodologia para apoio à realização do processo de descoberta de conhecimento em bases de dados. **II Workshop de Computação Científica da UENF**, 09 2006.

FACELI, K. et al. **Inteligência Artificial. Uma Abordagem de Aprendizado de Máquina**. [S.l.]: Ltc, 2011. ISBN 8521618808.

FACION, N. **Arquitetura Hierárquica Profunda para Aprendizado com Poucos Dados**. 134 p. Dissertação (Mestrado) — Universidade Estadual de Campinas, Campinas, 2019.

FERNEDA, E. Redes Neurais e sua aplicação em sistemas de recuperação de informação. **Ciencia da Informacao**, v. 35, n. 1, p. 25–30, 2006. ISSN 01001965.

FERREIRA, A. d. S. Redes Neurais Convolucionais Profundas na Detecção de Plantas Daninhas em Lavoura de Soja. p. 70, 2017.

FERREIRA, E. W. T.; SHINODA, A. A. The Development and Evaluation of a Dataset for Testing of IDS for Wireless Networks. **IEEE Latin America Transactions**, v. 14, n. 1, p. 404–410, 2016.

FRANK, E.; HALL, M. A.; WITTEN, I. H. The WEKA workbench. **Data Mining**, p. 553–571, 2017.

GERON, A. **Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems**. [S.l.]: O'Reilly Media, 2019. ISBN 1492032646.

GHARIB, A. et al. An Evaluation Framework for Intrusion Detection Dataset. In: **2016 International Conference on Information Science and Security (ICISS)**. [S.l.]: IEEE, 2016. p. 1–6. ISBN 978-1-5090-5494-7.

GOODFELLOW, I. et al. Maxout Networks. In: DASGUPTA, S.; MCALLESTER, D. (Ed.). **Proceedings of the 30th International Conference on Machine Learning**. Atlanta, Georgia, USA: PMLR, 2013. (Proceedings of Machine Learning Research, 3), p. 1319–1327.

GULLI, A.; PAL, S. **Deep Learning with Keras: Implement Neural Networks with Keras on Theano and TensorFlow**. [S.l.]: Packt Publishing, 2017. ISBN 9781787128422.

HAYKIN, S. **Redes Neurais: Princípios e Prática**. [S.l.]: Artmed, 2007. ISBN 9788577800865.

HAYKIN, S. S. et al. **Neural networks and learning machines/Simon Haykin.** [S.l.]: New York: Prentice Hall, 2009.

HE, K. et al. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In: **2015 IEEE International Conference on Computer Vision (ICCV)**. [S.l.: s.n.], 2015. p. 1026–1034.

HE, K. et al. Deep residual learning for image recognition. In: **Proceedings of the IEEE conference on computer vision and pattern recognition**. [S.l.: s.n.], 2016. p. 770–778.

HEBB, D. O. **The Organization of Behavior; A Neuropsychological Theory**. [S.l.]: Wiley, 1949.

HEIDEMANN, J.; PAPADOPOULOS, C. Uses and challenges for network datasets. **Proceedings - Cybersecurity Applications and Technology Conference for Homeland Security, CATCH 2009**, p. 73–82, 2009.

HINTON, G. E.; OSINDERO, S.; TEH, Y. W. A fast learning algorithm for deep belief nets. **Neural Computation**, v. 18, n. 7, p. 1527–1554, 2006. ISSN 08997667.

HINTON, G. E.; SEJNOWSKI, T. J. Optimal perceptual inference. In: CITESEER. **Proceedings of the IEEE conference on Computer Vision and Pattern Recognition**. [S.l.], 1983. v. 448.

HONDA, S. et al. Topase: Detection of brute force attacks used disciplined ips from ids log. **Proceedings of the 2015 IFIP/IEEE International Symposium on Integrated Network Management, IM 2015**, IEEE, p. 1361–1364, 2015.

HONG, D.; RUOHE, Y.; KUNHUI, L. Research on face recognition based on pca. **Proceedings - 2008 International Seminar on Future Information Technology and Management Engineering, FITME 2008**, n. 2007, p. 29–32, 2008.

JIANG, K. et al. Network Intrusion Detection Combined Hybrid Sampling With Deep Hierarchical Network. **IEEE Access**, v. 8, p. 32464–32476, 2020.

JIANG, S. et al. Tree-CNN: from generalization to specialization. **Eurasip Journal on Wireless Communications and Networking**, EURASIP Journal on Wireless Communications and Networking, v. 2018, n. 1, p. 1–12, 2018. ISSN 16871499.

JUSTIN, V.; MARATHE, N.; DONGRE, N. Hybrid IDS using SVM classifier for detecting DoS attack in MANET application. In: **2017 International Conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)**. [S.l.: s.n.], 2017. p. 775–778.

KARATAS, G.; DEMIR, O.; SAHINGOZ, O. K. Deep Learning in Intrusion Detection Systems. **International Congress on Big Data, Deep Learning and Fighting Cyber Terrorism, IBIGDELFT 2018 - Proceedings**, IEEE, p. 113–116, 2019.

KAUR, T.; KAUR, S. **Comparative Analysis of Anomaly Based and Signature Based Intrusion Detection Systems Using PHAD and Snort**. 2013.

KHAN, S. et al. **A Guide to Convolutional Neural Networks for Computer Vision**. [S.l.: s.n.], 2018. v. 8. 1–207 p. ISSN 2153-1056. ISBN 9781681730219.

KHRAISAT, A. et al. Survey of Intrusion Detection Systems: techniques, datasets and challenges. **Cybersecurity**, 2019. ISSN 2523-3246.

KIM, I. et al. A case study of unknown attack detection against Zero-day worm in the honeynet environment. In: IEEE. **2009 11th International Conference on Advanced Communication Technology**. [S.l.], 2009. v. 3, p. 1715–1720.

KIRK, M. **Thoughtful Machine Learning: A Test-Driven Approach**. [S.l.]: O'Reilly Media, 2014. ISBN 9781449374105.

KOCH, R.; GOLLING, M.; RODOSEK, G. D. Towards Comparability of Intrusion Detection Systems: New Data Sets. **TERENA Networking Conference**, p. 7, 2014.

KONONENKO, I. Semi-naive bayesian classifier. In: KODRATOFF, Y. (Ed.). **Machine Learning — EWSL-91**. Berlin, Heidelberg: Springer Berlin Heidelberg, 1991. p. 206–219. ISBN 978-3-540-46308-5.

KOTSIANTIS, S. B. Decision trees: A recent overview. **Artificial Intelligence Review**, v. 39, n. 4, p. 261–283, 2013. ISSN 02692821.

KOZIOL, J. **Intrusion detection with Snort**. [S.l.]: Sams Publishing, 2003.

KRIZHEVSKY, A.; SUTSKEVER, I.; HINTON, G. E. Imagenet classification with deep convolutional neural networks. In: PEREIRA, F. et al. (Ed.). **Advances in Neural Information Processing Systems**. [S.l.]: Curran Associates, Inc., 2012. v. 25, p. 1097–1105.

KUBAT, M.; MATWIN, S. Addressing the curse of imbalanced training sets: one-sided selection. In: **Proc. 14th International Conference on Machine Learning**. [S.l.]: Morgan Kaufmann, 1997. p. 179–186.

LAB, L. Deep Learning Tutorials. University of Montreal, 2015.

LABORATORY, M. I. O. T. L. **1999 Darpa Intrusion Detection Evaluation Dataset**. 1999. Disponível em: <http://www.ll.mit.edu/r-d/datasets/1999-darpa-intrusion-detection-evaluation-dataset>. Acesso em: 16 jun. 2020.

LECUN, Y.; BENGIO, Y.; HINTON, G. Deep Learning. **Nature**, Nature Publishing Group, v. 521, n. 7553, p. 436–444, 2015.

LECUN, Y. et al. Gradient-based learning applied to document recognition. **Proceedings of the IEEE**, IEEE, v. 86, n. 11, p. 2278–2324, 1998.

LEE, H. D. Seleção de atributos importantes para a extração de conhecimento de bases de dados. **Tese (Doutorado) do Curso de Pós-Graduação em Ciências de Computação e Matemática Computacional - ICMC - USP**, p. 182, 2005.

LI, D.; ZHOU, Y. Soft-Root-Sign: A New Bounded Neural Activation Function. In: SPRINGER. **Chinese Conference on Pattern Recognition and Computer Vision (PRCV)**. [S.l.], 2020. p. 310–319.

LIHAO, W.; YANN, D. A Fault Diagnosis Method of Tread Production Line Based on Convolutional Neural Network. 2018.

LIMA, I.; PINHEIRO, C.; SANTOS, F. **Inteligência Artificial**. [S.l.]: Elsevier Brasil, 2016. ISBN 9788535278095.

LIN, W. C.; KE, S. W.; TSAI, C. F. CANN: An intrusion detection system based on combining cluster centers and nearest neighbors. **Knowledge-Based Systems**, 2015. ISSN 09507051.

LIPPMANN, R. P. et al. Evaluating intrusion detection systems: The 1998 darpa off-line intrusion detection evaluation. **Proceedings - DARPA Information Survivability Conference and Exposition, DISCEX 2000**, v. 2, n. May, p. 12–26, 2000.

LIU, H.; LANG, B. Machine learning and deep learning methods for intrusion detection systems: A survey. **Applied Sciences**, Multidisciplinary Digital Publishing Institute, v. 9, n. 20, p. 4396, 2019.

LIU, X.; TANG, Z.; YANG, B. Predicting Network Attacks with CNN by Constructing Images from NetFlow Data. In: **2019 IEEE 5th Intl Conference on Big Data Security on Cloud (BigDataSecurity), IEEE Intl Conference on High Performance and Smart Computing, (HPSC) and IEEE Intl Conference on Intelligent Data and Security (IDS)**. [S.l.: s.n.], 2019. p. 61–66.

LOPES, N.; RIBEIRO, B. **Deep Belief Networks (DBNs)**. Cham: Springer International Publishing, 2015. 155–186 p. ISBN 978-3-319-06938-8.

LUGER, G. F. **Inteligencia Artificial**. [S.l.]: Pearson, 2013. ISBN 8581435505.

LYRA, W. D. S. et al. Classificação periódica: Um exemplo didático para ensinar análise de componentes principais. **Quimica Nova**, v. 33, n. 7, p. 1594–1597, 2010. ISSN 01004042.

MACIÁ-FERNÁNDEZ, G. et al. UGR'16: A new dataset for the evaluation of cyclostationarity-based network IDSs. **Computers and Security**, Elsevier Ltd, v. 73, p. 411–424, 2018. ISSN 01674048.

MAIA, R. B. Detecção da Intrusão Utilizando Classificação Bayesiana. **Rio de Janeiro: COPPE/UFRJ**, 2005.

MANIKANDAN, V. et al. DRCNN-IDS Approach for Intelligent Intrusion Detection System. In: IEEE. **2020 International Conference on Computing and Information Technology (ICCIT-1441)**. [S.l.], 2020. p. 1–4.

MANIMURUGAN, S. et al. Effective Attack Detection in Internet of Medical Things Smart Environment using a Deep Belief Neural Network. **IEEE Access**, p. 1–1, 2020.

MCCLELLAND, J. L. et al. **Parallel distributed processing**. [S.l.]: MIT press Cambridge, MA, 1986. v. 2.

MCCULLOCH, W.; PITTS, W. A logical calculus of ideas immanent in nervous activity. **Bulletin of Mathematical Biophysics**, v. 5, p. 127–147, 1943.

MCHUGH, J. Testing intrusion detection systems: a critique of the 1998 and 1999 darpa intrusion detection system evaluations as performed by lincoln laboratory. **ACM Transactions on Information and System Security**, v. 3, n. 4, p. 262–294, 2000. ISSN 10949224.

MCKINNEY, W. et al. Data structures for statistical computing in python. In: AUSTIN, TX. **Proceedings of the 9th Python in Science Conference**. [S.l.], 2010. v. 445, p. 51–56.

MELIBOEV, A.; JUMABEK, A.; KIM, W. 1D CNN based network intrusion detection with normalization on imbalanced data. In: **2020 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)**. [S.l.: s.n.], 2020. p. 218–224.

MESHRAM, A.; HAAS, C. Anomaly Detection in Industrial Networks using Machine Learning: A Roadmap. In: **Machine Learning for Cyber Physical Systems**. [S.l.: s.n.], 2017.

MINSKY, M.; PAPERT, S. An introduction to computational geometry. **Cambridge tiass., HIT**, 1969.

MITCHELL, T. M. **Machine Learning**. [S.l.]: McGraw-Hill Education, 1997. ISBN 0070428077.

MOUSTAFA, N. **The UNSW-NB15 dataset**. UNSW, 2015. Disponível em: <http://handle.unsw.edu.au/1959.4/resource/collection/resdatac_445/1>.

MUKHERJEE, S.; SHARMA, N. Intrusion detection using naive bayes classifier with feature reduction. **Procedia Technology**, Elsevier, v. 4, p. 119–128, 2012.

NAIR, V.; HINTON, G. E. Rectified linear units improve restricted boltzmann machines. In: **Proceedings of the 27th International Conference on International Conference on Machine Learning**. Madison, WI, USA: Omnipress, 2010. (Icmĺ10), p. 807–814. ISBN 9781605589077.

NASCIMENTO, R. F. F. et al. O algoritmo SVM: avaliação da separação ótima de classes em imagens CCD-CBERS-2. **XIV Simpósio Brasileiro de Sensoriamento Remoto**, p. 2079–2086, 2009.

NASEER, S.; SALEEM, Y. Enhanced network intrusion detection using deep convolutional neural networks. **KSII Transactions on Internet and Information Systems**, v. 12, n. 10, p. 5159–5178, 2018. ISSN 22881468.

NEHINBE, J. O. A critical evaluation of datasets for investigating IDSs and IPSs researches. **Proceedings of 2011, 10th IEEE International Conference on Cybernetic Intelligent Systems, CIS 2011**, IEEE, p. 92–97, 2011.

NORVIG, P.; RUSSELL, S. **Inteligência artificial: Tradução da 3a Edição**. [S.l.]: Elsevier Editora Ltda., 2014. ISBN 9788535251418.

NWANKPA, C. et al. **Activation Functions: Comparison of trends in Practice and Research for Deep Learning**. 2018.

OGURI, P. **Aprendizado de Máquina para o Problema de Sentiment Classification**. Dissertação (Mestrado) — Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2006.

OLIPHANT, T. E. **A guide to NumPy**. [S.l.]: Trelgol Publishing USA, 2006. v. 1.

ONU. **World population prospects 2019**. [s.n.], 2019. ISSN 0337-307x. ISBN 9789211483161. Disponível em: <http://www.ncbi.nlm.nih.gov/pubmed/12283219>.

OSHIRO, T. M. Uma abordagem para a construção de uma única árvore a partir de uma Random Forest para classificação de bases de expressão gênica. p. 1–6, 2013.

PATTERSON, J.; GIBSON, A. **Deep Learning: A Practitioner's Approach**. [S.l.]: O'Reilly Media, 2017. ISBN 9781491914212.

PEARSON, K. I. Mathematical contributions to the theory of evolution. —XI. On the influence of natural selection on the variability and correlation of organs. **Philosophical Transactions of the Royal Society of London. Series A, Containing Papers of a Mathematical or Physical Character**, The Royal Society, v. 200, n. 321-330, p. 1–66, jan. 1903.

PIJPKER, J.; VRANKEN, H. The role of internet service providers in botnet mitigation. **Proceedings - 2016 European Intelligence and Security Informatics Conference, EISIC 2016**, IEEE, p. 24–31, 2017.

PINA, D. et al. Análise de hiperparâmetros em aplicações de aprendizado profundo por meio de dados de proveniência. In: **Anais do XXXIV Simpósio Brasileiro de Banco de Dados**. Porto Alegre, RS, Brasil: Sbc, 2019. p. 223–228. ISSN 0000-0000. Disponível em: <https://sol.sbc.org.br/index.php/sbbd/article/view/8827>.

PORTUGAL, M. S.; FERNANDES, L. G. L. Redes neurais artificiais e previsão de séries econômicas: uma introdução. **Nova Economia**, v. 6, n. 1, dez. 2013.

QIN, D. et al. Ip traffic classification based on machine learning. **International Conference on Communication Technology Proceedings, ICCT**, IEEE, p. 882–886, 2011.

QU, F. et al. An intrusion detection model based on deep belief network. **ACM International Conference Proceeding Series**, p. 97–101, 2017.

QUILICI-GONZALEZ, J.; ZAMPIROLLI, F. de A. **Sistemas Inteligentes e Mineração de Dados**. [S.l.: s.n.], 2015. ISBN 9788561175382.

QUINLAN, J. R. Induction of decision trees. **Machine Learning**, v. 1, n. 1, p. 81–106, 1986. ISSN 0885-6125.

QUINLAN, J. R. Simplifying decision trees. **International journal of man-machine studies**, Elsevier, v. 27, n. 3, p. 221–234, 1987.

RAMACHANDRAN, P.; ZOPH, B.; LE, Q. V. Searching for activation functions. **arXiv preprint arXiv:1710.05941**, 2017.

REAVES, B.; MORRIS, T. Discovery, infiltration, and denial of service in a process control system wireless network. **2009 eCrime Researchers Summit, eCRIME '09**, IEEE, p. 1–9, 2009.

REHMAN, R. **Intrusion Detection Systems with Snort: Advanced IDS Techniques Using Snort, Apache, MySQL, PHP, and ACID**. [S.l.]: Prentice Hall PTR, 2003. (Bruce Perens' open source series). ISBN 9780131407336.

REZENDE, S. O. **Sistemas inteligentes: fundamentos e aplicações**. [S.l.]: Editora Manole Ltda, 2003.

RIJSBERGEN, C. **Information Retrieval**. 2. ed. ed. London: Butterworth, 1979. ISBN 0408709294.

ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. **Psychological Review**, p. 65–386, 1958.

ROSSI, R. G. **Análise de componentes principais em data warehouses**. Tese (Doutorado) — Universidade de São Paulo, 2017.

ROSSUM, G. V.; JR, F. L. D. **Python reference manual**. [S.l.]: Centrum voor Wiskunde en Informatica Amsterdam, 1995.

ROY, D.; PANDA, P.; ROY, K. Tree-CNN: A hierarchical Deep Convolutional Neural Network for incremental learning. **Neural Networks**, 2020. ISSN 18792782.

RUSSELL, S.; NORVIG, P. **Artificial Intelligence: A Modern Approach**. [S.l.]: Pearson, 2016. (Always learning). ISBN 9781292153964.

SAHU, S.; MEHTRE, B. M. Network intrusion detection system using J48 Decision Tree. In: **2015 International Conference on Advances in Computing, Communications and Informatics (ICACCI)**. [S.l.: s.n.], 2015. p. 2023–2026.

SALAKHUTDINOV, R.; HINTON, G. Deep Boltzmann Machines. **Journal of Machine Learning Research**, v. 5, n. 3, p. 448–455, 2009. ISSN 15324435.

SARKAR, K. Using character n-gram features and multinomial Naïve Bayes for sentiment polarity detection in bengali tweets. In: **2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)**. [S.l.: s.n.], 2018. p. 1–4.

SCHOLKOPF, B.; BURGES, C.; SMOLA, A. Advances in kernel methods - support vector learning. **MIT Press**, 12 1998.

SHARAFALDIN, I.; LASHKARI, A. H.; GHORBANI, A. A. Toward generating a new intrusion detection dataset and intrusion traffic characterization. In: INSTICC. **Proceedings of the 4th International Conference on Information Systems Security and Privacy**. [S.l.]: SCITEPRESS - Science and Technology Publications, 2018. p. 108–116. ISBN 9789897582820.

SHIRAVI, H.; SHIRAVI, A.; GHORBANI, A. A. A survey of visualization systems for network security. **IEEE Transactions on Visualization and Computer Graphics**, IEEE, v. 18, n. 8, p. 1313–1329, 2012. ISSN 10772626.

SHONE, N. et al. A deep learning approach to network intrusion detection. **IEEE Transactions on Emerging Topics in Computational Intelligence**, v. 2, n. 1, p. 41–50, 2018.

SIAMI-NAMINI, S.; TAVAKOLI, N.; NAMIN, A. S. The performance of lstm and bilstm in forecasting time series. In: **2019 IEEE International Conference on Big Data (Big Data)**. [S.l.: s.n.], 2019. p. 3285–3292.

SILVA, C. A. d. A. **Implementação de uma matriz de neurônios dinamicamente reconfigurável para descrição de topologias de redes neurais artificiais multilayer perceptrons**. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, 2015.

SILVA, J. R. C. D. **Sistemas de Detecção de Intrusão com técnicas de Inteligência Artificial**. 143 p. Dissertação (Mestrado) — UFV Universidade Federal de Viçosa, Viçosa, 2011.

SIMONYAN, K.; ZISSERMAN, A. Very Deep Convolutional Networks for Large-Scale Image Recognition. In: BENGIO, Y.; LECUN, Y. (Ed.). **3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings**. [S.l.: s.n.], 2015.

SIWEK, K.; OSOWSKI, S. Autoencoder versus pca in face recognition. **Proceedings of 2017 18th International Conference on Computational Problems of Electrical Engineering, CPEE 2017**, 2017.

SOVIERZOSKI, M. A. Convolução de sinais: Definição, propriedades e ferramentas. **Revista Ilha Digital**, v. 2, p. 81–95, 2011.

SZEGEDY, C. et al. Going deeper with convolutions. **2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)**, IEEE, jun. 2015.

TAKAHASHI, A. **Máquina de vetores-suporte intervalar**. 61 p. Tese (Doutorado) — Universidade Federal do Rio Grande do Norte, 2012.

TAVALLAEE, M. et al. A detailed analysis of the kdd cup 99 data set. In: **2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications**. [S.l.]: IEEE, 2009. p. 1–6. ISBN 978-1-4244-3764-1. ISSN 2329-6275.

TENÓRIO, D. F. **DETECÇÃO CEGA DE TRÁFEGO MALICIOSO ATRAVÉS DA VARIAÇÃO TEMPORAL DO MAIOR AUTOVALOR**. Dissertação (Mestrado) — Universidade de Brasília, Brasília, 2013.

THAKKAR, A.; LOHIYA, R. A review of the advancement in intrusion detection datasets. **Procedia Computer Science**, Elsevier B.V., v. 167, n. 2019, p. 636–645, 2020. ISSN 18770509.

TOBERGTE, D. R.; CURTIS, S. **Algorithms for image prcessing and computer vision**. [S.l.: s.n.], 2013. v. 53. 1689–1699 p. ISSN 1098-6596. ISBN 9788578110796.

TROUSSAS, C. et al. Sentiment analysis of facebook statuses using naive bayes classifier for language learning. In: **Iisa 2013**. [S.l.: s.n.], 2013. p. 1–6. ISSN null.

TSAI, C. F. et al. Intrusion detection by machine learning: A review. **Expert Systems with Applications**, Elsevier Ltd, v. 36, n. 10, p. 11994–12000, 2009. ISSN 09574174.

VAPNIK, V. N.; CHERVONENKIS, A. Y. On the uniform convergence of relative frequencies of events to their probabilities. **Theory of Probability & Its Applications**, Society for Industrial & Applied Mathematics (SIAM), v. 16, n. 2, p. 264–280, jan. 1971.

VERIKAS, A. et al. Electromyographic patterns during golf swing: Activation sequence profiling and prediction of shot effectiveness. **Sensors (Switzerland)**, v. 16, n. 4, 2016. ISSN 14248220.

VIEGAS, E. K. Detecção de intrusão baseada em anomalia para ambientes de produção. 2016.

VIEIRA, S. T.; ROSA, R. L.; RODRÍGUEZ, D. Z. A Speech Quality Classifier based on Tree-CNN Algorithm that Considers Network Degradations. **Journal of Communications Software and Systems**, Croatian Communications and Information Society (CCIS), v. 16, n. 2, p. 18–187, 6 2020. ISSN 1846-6079.

VINAYAKUMAR, R. et al. Deep Learning Approach for Intelligent Intrusion Detection System. **IEEE Access**, v. 7, p. 41525–41550, 2019.

VINAYAKUMAR, R.; SOMAN, K. P.; POORNACHANDRANY, P. Applying convolutional neural network for network intrusion detection. **2017 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2017**, v. 2017-Janua, p. 1222–1228, 2017.

WARZYNSKI, A.; KOLACZEK, G. Intrusion detection systems vulnerability on adversarial examples. **2018 IEEE (SMC) International Conference on Innovations in Intelligent Systems and Applications, INISTA 2018**, 2018.

WITTEN, I.; FRANK, E.; HALL, M. **Data Mining: Practical Machine Learning Tools and Techniques**. [S.l.]: Elsevier Science, 2011. (The Morgan Kaufmann Series in Data Management Systems). ISBN 9780080890364.

WU, K.; CHEN, Z.; LI, W. A Novel Intrusion Detection Model for a Massive Network Using Convolutional Neural Networks. **IEEE Access**, v. 6, p. 50850–50859, 2018.

XIAO, T. et al. Error-driven incremental learning in deep convolutional neural network for large-scale image classification. In: **MM 2014 - Proceedings of the 2014 ACM Conference on Multimedia**. [S.l.: s.n.], 2014. ISBN 9781450330633.

YAN, Z. et al. HD-CNN: Hierarchical Deep Convolutional Neural Network for Large Scale Visual Recognition. In: **ICCV'15: Proc. IEEE 15th International Conf. on Computer Vision**. [S.l.: s.n.], 2015.

YANG, K. et al. Adversarial Examples Against the Deep Learning Based Network Intrusion Detection Systems. **Proceedings - IEEE Military Communications Conference MILCOM**, IEEE, v. 2019-October, p. 559–564, 2019.

YANG, Y. et al. Building an effective intrusion detection system using the modified density peak clustering algorithm and deep belief networks. **Applied Sciences (Switzerland)**, v. 9, n. 2, 2019. ISSN 20763417.

YIN, C. et al. A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks. **IEEE Access**, v. 5, p. 21954–21961, 2017. ISSN 21693536.

ZACCONE, G.; KARIM, M.; MENSHAWY, A. **Deep Learning with TensorFlow**. [S.l.]: Packt Publishing, 2017. ISBN 9781786460127.

ZEILER, M. D.; FERGUS, R. **Visualizing and Understanding Convolutional Networks**. 2013.

ZHANG, X.; CHEN, J. Deep learning based intelligent intrusion detection. In: **2017 IEEE 9th International Conference on Communication Software and Networks (ICCSN)**. [S.l.: s.n.], 2017. p. 1133–1137.

ZHANG, Y. et al. Adaptive Convolutional Neural Network and Its Application in Face Recognition. **Neural Processing Letters**, Springer US, v. 43, n. 2, p. 389–399, 2016. ISSN 1573773x.

ZHAO, G.; ZHANG, C.; ZHENG, L. Intrusion detection using deep belief network and probabilistic neural network. In: **2017 IEEE International Conference on Computational Science and Engineering (CSE) and IEEE International Conference on Embedded and Ubiquitous Computing (EUC)**. [S.l.: s.n.], 2017. v. 1, p. 639–642.