UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

Universidade Federal de Pernambuco

Centro de Tecnologia e Geociências

Departamento de Eletrônica e Sistemas

Detecção de Intrusão em Redes de Computadores usando Autoencoders

Trabalho de Conclusão de Curso de Graduação

por

Arthur de Andrade Almeida

Orientador: Prof. Cleber Zanchettin, Ph.D.

Recife, Novembro / 2022

UNIVERSIDADE
FEDERAL
DE PERNAMBUCO

DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES USANDO AUTOENCODERS

Arthur de Andrade Almeida

Trabalho de Conclusão de Curso de Graduação apresentado à Coordenação do Curso de Graduação em Engenharia Eletrônica, DES/CTG, da Universidade Federal de Pernambuco, como parte dos requisitos necessários à obtenção do título de Bacharel em Engenharia Eletrônica.

Orientador(a): Cleber Zanchettin Ph.D.

Recife
Novembro de 2022

Ficha de identificação da obra elaborada pelo autor,
através do programa de geração automática do SIB/UFPE

Almeida, Atrhur de Andrade.
Detecção de Intrusão em Redes de Computadores usando Autoencoders / Atrhur de Andrade Almeida. - Recife, 2022.
30 : il., tab.

Orientador(a): Cleber Zanchettin
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal de Pernambuco, Centro de Tecnologia e Geociências, Engenharia Eletrônica - Bacharelado, 2022.

1. Detecção de Anomalia. 2. Aprendizagem de Máquina. 3. Aprendizagem Profunda. 4. Sistemas de Detecção de Instrusos. 5. Semi-supervisionado. I. Zanchettin, Cleber . (Orientação). II. Título.

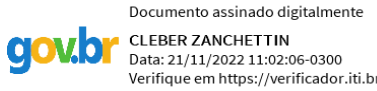
620 CDD (22.ed.)

DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES USANDO AUTOENCODERS

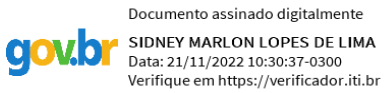
Arthur de Andrade Almeida

TRABALHO DE CONCLUSÃO DE CURSO SUBMETIDO AO CORPO DOCENTE DO DEPARTAMENTO DE ELETRÔNICA E SISTEMAS DO CENTRO DE TECNOLOGIAS E GEOCIÊNCIAS DA UNIVERSIDADE FEDERAL DE PERNAMBUCO COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE BACHAREL EM ENGENHARIA ELETRÔNICA.

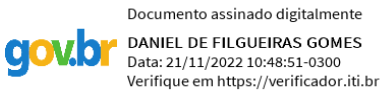
Examinado por:



Prof. Cleber Zanchettin, Ph.D.



Prof. Sidney Marlon Lopes de Lima, Ph.D.



Prof. Daniel de Filgueiras Gomes, Ph.D.

RECIFE, PE – BRASIL
NOVEMBRO DE 2022

Por seus padrões intencionais na criação, Deus gerou a ciência e a tecnologia. A ciência descobre os padrões. A tecnologia explora os padrões. Deus é a gênese de toda tecnologia humana

---

Tony Reinke

Agradecimentos

Em primeiro lugar, a Deus por minha vida e por me ajudar a passar por todos os obstáculos enfrentados não apenas neste trabalho, mas também ao longo da graduação.

A minha família, por me darem a melhor educação desde sempre, ao incentivo durante as fases difíceis e a compreensão nos momentos que eu precisei me fazer ausente para me dedicar neste e em outros trabalhos.

Aos amigos, que sempre estiveram ao meu lado, pelos momentos de diversão e pelo apoio demonstrado desde que iniciei a minha graduação.

Ao professor Cleber Zanchettin, por ter sido meu orientador, mesmo mudando de tema mais de duas vezes, e ter me ensinando tanto sobre aprendizagem de máquina.

A todos que participaram, direta ou indiretamente do desenvolvimento deste trabalho de pesquisa, enriquecendo o meu processo de aprendizado.

Resumo do Trabalho de Conclusão de Curso apresentado ao DES/UFPE como parte dos requisitos necessários para a obtenção do grau de Bacharel em Engenharia Eletrônica (Eng.)

DETECÇÃO DE INTRUSÃO EM REDES DE COMPUTADORES USANDO AUTOENCODERS

Arthur de Andrade Almeida

Novembro/2022

Orientador(a): Cleber Zanchettin Ph.D.

Programa: Engenharia Eletrônica

Sistemas de detecção de intrusos (IDS) têm se tornado um mecanismo essencial na segurança primária de uma rede de computadores protegendo-a contra ciber-ataques. Um IDS é geralmente baseado em assinaturas de padrões de intrusão conhecidas ou na detecção de anomalias no comportamento dos usuários na rede. Com o crescente avanço da aprendizagem de máquina e das redes neurais profundas, diversas soluções surgiram destas áreas para atacar esse problema. Neste estudo, foi investigado uma solução eficiente para detecção de intrusão usando uma arquitetura baseada em um *autoencoder*, treinado de forma semi-supervisionada, para detectar anomalias no trafego de uma rede de computadores. O modelo proposto utiliza a função de erro de reconstrução como limiar de detecção, a qual assume um papel fundamental na descoberta de uma eventual anomalia. O modelo foi avaliado na base de dados NSL-KDD, um *benchmark* bastante conhecido e utilizado na tarefa de detecção de intrusão em redes de computadores. A taxa de acurácia e de *F1-score* da solução é de 90.49% e 91.80% respectivamente, apresentando um desempenho superior em relação a outros métodos considerados estado-da-arte no problema investigado.

Palavras-chave: IDS, Detecção de Anomalia, Aprendizagem de Máquina, Aprendizagem Profunda, Semi-supervisionada .

Abstract of Course Conclusion Work presented to DES/UFPE as a partial fulfillment of the requirements for the degree of Bachelor of Electronic Engineering (Eng.)

INTRUSION DETECTION IN COMPUTER NETWORKS USING AUTOENCODERS

Arthur de Andrade Almeida

November/2022

Advisor: Cleber Zanchettin Ph.D.

Course: Electronic Engineering

Intrusion detection systems (IDS) have become an essential mechanism in the primary security of a computer network, protecting it against cyber-attacks. An IDS is usually based on signatures of known intrusion patterns or on detecting anomalies in the behavior of users on the network. With the increasing advancement of machine learning and deep neural networks, several solutions have emerged from these areas to tackle this problem. In this study, we investigated an efficient solution for intrusion detection using an architecture based on an autoencoder trained in a semi-supervised way, to detect anomalies in the traffic of a computer network. The proposed model uses the reconstruction error function, which plays a fundamental role in the discovery of an eventual anomaly. The model was evaluated in the NSL-KDD database, a well-known benchmark used in the task of intrusion detection in computer networks. The accuracy and F1-score rate of the solution is 90.49% and 91.80% respectively, presenting a superior performance in relation to other methods considered state-of-the-art for the problem.

Keywords: IDS; Anomaly Detection; Machine Learning; Deep Learning; Semi-supervised

Sumário

Lista de Figuras **ix**

Lista de Tabelas **x**

1 Introdução **1**
1.1 Objetivo Geral 4
1.1.1 Objetivos específicos 5
1.2 Organização do TCC 5

2 Fundamentação Teórica **6**
2.1 Aprendizagem de Máquina 6
2.2 Aprendizagem profunda 7

3 Metodologias e Dados **8**
3.1 Conjunto de Dados 8
3.2 Pré-processamento 9
3.2.1 Codificação das Variáveis Categóricas 9
3.2.2 Separação dos Dados 10
3.2.3 Normalização 10

4 Método de Detecção de Anomalia Baseado em Autoencoder **12**
4.1 Autoencoders 12
4.2 Distribuição Normal 13
4.3 Regra Empírica 14
4.4 Método Proposto 14

5 Experimentos **17**
5.1 Métricas de Validação 17
5.2 Implementação 18
5.3 Resultados 19
5.3.1 Diferentes arquiteturas 19
5.3.2 Representação dos dados no espaço latente 20

5.3.3 Definindo o Limiar de Separação 22
5.3.4 Comparação com outros métodos 23

6 Conclusões **26**

Referências Bibliográficas **27**

Lista de Figuras

1.1 Estrutura Geral de um *IDS*. Imagem adaptada de [1] 3

3.1 Cada exemplo categórico é transformado em um vetor de dimensão igual a quantidade de atributos únicos do coluna. 9

4.1 Distribuição Gaussiana. Neste exemplo, tem-se uma distribuição normal padrão com $\mu = 0$ e $\sigma = 1$. 14

5.1 Esquema geral da arquitetura proposta. Nesta figura, a camada de entrada e a camada de saída representam os vetores da entrada e da sua versão reconstruída, respectivamente, ambos possuindo uma dimensão $\mathbb{R}^{122}$. O codificador é definido por vetores de dimensões $\mathbb{R}^{[64,32,16]}$. O decodificador é espelhado a fim de realizar a reconstrução. O espaço latente de dimensão $\mathbb{R}^8$ representa a versão condensada da entrada que será reconstruída pelo decodificador. 20

5.2 Visualização do conjunto de validação usando o *t-SNE* antes e depois do *AE*. 21

5.3 O eixo horizontal define parcelas variadas de desvios padrão, enquanto que o eixo vertical apresenta os *scores* de classificação no conjunto de validação. A linha vertical tracejada é determinada pelo valor de *alpha* que apresenta maior *F1-score* e acurácia. 22

5.4 *Best threshold* foi o melhor limiar encontrado através do algoritmo 1 para separar distribuição dos erros de reconstrução dos dados de teste. 23

Lista de Tabelas

3.1 Número de registros do conjunto NSL-KDD. 9
3.2 Distribuição dos dados de treinamento, validação e teste. 10

5.1 Performance do AE utilizando diferentes arquiteturas. Todos os modelos foram treinados usando o KDDTrain+ e testados no KDDTest+. 19
5.2 Comparação do modelo proposto com soluções do estado da arte. Todos os modelos foram treinados usando os dados do KDDTrain+ e testados no KDDTest+. 24
5.3 Distribuição dos dados de treinamento, validação e teste 24
5.4 Comparação do modelo proposto com outras soluções do estado-da-arte. Todos os modelos foram treinados e testados usando o KDD-Train+. 25

Capítulo 1

Introdução

A informação é um dos ativos mais valiosos de uma organização. Portanto, como qualquer outro ativo, a informação precisa ser protegida. A ISO/IEC 27001 traz uma abordagem sistemática para identificar e gerenciar os riscos de segurança a fim de garantir a segurança das informações.

Serviços de segurança da informação requiridos por quase todas as redes de comunicações são construídos sobre três princípios básicos, comumente conhecidos como CID: confidencialidade, integridade e disponibilidade.

A confidencialidade tem como objetivo garantir que os dados de uma organização estejam seguros. Para alcançar isso, se deve controlar o acesso às informações a fim de evitar o compartilhamento não autorizado das mesmas, seja de forma intencional ou acidental. Um componente importante para manter a confidencialidade é garantir que aqueles que necessitam ter acesso a um determinado sistema possuam os privilégios necessários para entrar no sistema. Existem diversas formas de comprometer a confidencialidade dos dados de uma instituição. Isso pode envolver ataques diretos com o objetivo de conseguir acesso a sistemas restritos, os quais o intruso não tem o direito.

A integridade dos dados é garantida somente se estes são confiáveis e livres de adulteração. Se alguém tiver interesse em distorcer a reputação de uma organização é possível que essa possa tentar corromper imagens, títulos, ou mesmo informações pessoais de líderes, podendo manchar todo o prestigio da instituição. Também pode ser que isso aconteça por descuido de algum usuário que esteja acessando essas informações e acabe por alterar os dados de forma a prejudicar a sua integridade.

Mesmo que os dados de uma empresa sejam confidenciais e íntegros, se eles não estiverem disponíveis serão inúteis para aqueles que precisam acessá-los. A disponibilidade pode ser comprometida se, por exemplo, houver uma queda de energia e a organização não tiver um sistema de recuperação de energia, os usuários poderão ficar sem acesso a informações significativas.

Neste contexto, a criptografia, por exemplo, é um tipo de mecanismo para evi-

tar ataques, modificando a informação enviada de uma forma que não pode ser utilizada pelo intruso [2]. No entanto, considerando a infraestrutura de mais alto nível, é possível que alguns ataques prejudiquem a disponibilidade das informações de uma rede de computadores, como em um ataque conhecido como inundação *SYN* [3] do tipo *DoS* (*Denial-of-service*). Nesse tipo de ataque, o invasor envia muitas requisições TCP (*Transmission Control Protocol*) com origens falsas para a máquina da vítima. Depois de muitas solicitações, os recursos do usuário destino são esgotados, fazendo com que o sistema fique lento e, eventualmente, pare.

Quando atividades maliciosas como essas ocorrem, sistemas de detecção de intrusos (*IDS*) têm como objetivo, detectar e alertar que algo de errado aconteceu para que se possa tomar as medidas adequadas. Portanto *IDS* não substituem, mas complementam abordagens de segurança, como criptografia no exemplo acima.

Os *IDS* podem ser categorizados em dois grupos [4]:

1. Detecção de intrusos baseado em assinatura;
2. Detecção de intrusos baseado em anomalia;

Sistemas baseados em assinatura possuem um banco de dados contendo informações sobre formas de ataques conhecidos (assinaturas). Eles funcionam de forma reativa, ou seja, tráfegos de uma rede são monitorados pelo sistema para encontrar padrões que coincidam com assinaturas de ataques, emitindo um alerta caso o sistema detecte corretamente o ataque, como é demonstrado na parte esquerda da Figura 1.1.

Uma das vantagens desse tipo de sistema é que ele possui uma taxa de alarmes falsos [5] muito baixa, ou seja, conexões normais deverão ser classificadas como normais visto que apenas os padrões maliciosos estão armazenados no banco de dados. Entretanto, manter a base de dados alimentada exige um trabalho intensivo, pois é possível que não se consiga detectar ataques que não foram vistos previamente pelo sistema.

Sistemas de detecção de intrusos baseados em anomalias têm despertado interesse em pesquisadores devido a sua capacidade preventiva, diferentemente do método anterior. Apresentam um bom funcionamento na detecção de ataques desconhecidos. Em contrapartida, tal abordagem pode apresentar problemas, como uma alta taxa de falsos positivos.

Métodos baseados em anomalias primeiro modelam dados de comportamento normal, para que qualquer atividade que não possui comportamento semelhante ao modelo estabelecido seja classificada como anomalia, como apresentado na parte direita da Figura 1.1.

A aprendizagem de máquina é uma das abordagens que mais tem crescido no campo da cibersegurança, se tornando um dos principais métodos na detecção de

anomalias [6], dado a sua propriedade em de detectar intrusos usando o conhecimento prévio.

Técnicas de aprendizagem de máquina para detecção de intrusos podem ser divididas em três grupos: supervisionado, semi-supervisionado e não-supervisionado.

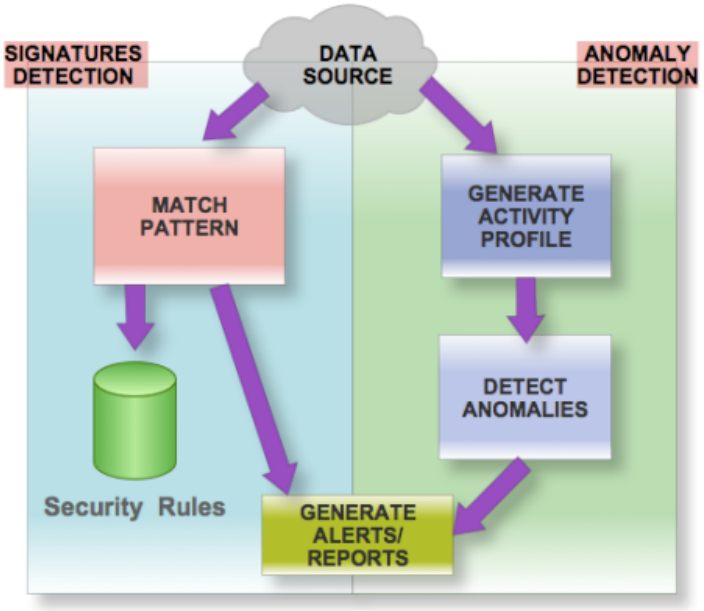


Figura 1.1: Estrutura Geral de um *IDS*. Imagem adaptada de [1]

Métodos supervisionados requerem que dados estejam rotulados com os dados normais e anomalias do tráfego de rede durante o treinamento do modelo [7]. Isso permite aprender as características de ambos os tipos de dados e realizar a previsão de maneira eficiente.

Obter dados com rótulos é uma tarefa difícil e muitas vezes chega a ser impossível de conseguir. A análise do tráfego de rede consome muito tempo e, geralmente, apenas uma pequena parte dos dados disponíveis pode ser rotulada. Além disso, em uma aplicação real nunca se pode ter certeza que um conjunto de exemplos abrange todos os ataques possíveis. Se um novo ataque ocorrer, pode ser que este não tenha sido visto nos dados de treinamento. Em segurança da informação, um sistema pode sofrer grandes danos se ataques raros não forem detectados rapidamente. Por exemplo, se ataques do tipo *U2R* (*User to Root*) não são detectados, o intruso pode obter acesso ao sistema como se fosse um usuário normal [8].

Algoritmos treinados sem supervisão podem ser usados para identificar um tráfego malicioso entre os fluxos de pacotes normais. Métodos não supervisionados geralmente lidam com formação de agrupamentos com objetivo de identificar o conjunto de dados que apresentam dissimilaridade para identificar uma intrusão. O desempenho de um algoritmo não supervisionado não é afetado por ataques desconhecidos e é semelhante ao desempenho do aprendizado supervisionado nesse

cenário. Porém, algoritmos supervisionados se mostram mais eficientes e com resultados melhores nos ataques conhecidos se comparados com os não supervisionados [9]. Além disso, problemas de dados de teste sendo extraídos de uma distribuição diferente não pode ser resolvido dentro de técnicas puramente supervisionadas ou não supervisionadas.

Uma forma de superar a limitação de um modelo supervisionado e apresentar resultados mais promissores do que os métodos não supervisionados, é utilizando métodos de aprendizagem semi-supervisionada. Métodos semi-supervisionados usam um pequeno número de dados rotulados no conjunto de dados de treinamento para reduzir tarefas de rotulação realizadas pelo humano e melhorar o desempenho com o suporte de dados não rotulados no conjunto de dados de treinamento.

Treinar métodos de detecção de intrusos de maneira semi-supervisionada é treinar determinado modelo apenas com tráfego de pacotes normais [10]. Dessa forma, o modelo em questão consegue aprender padrões inerentes aos fluxos normais, sem ter a necessidade de possuir alguma informação sobre as anomalias, podendo resultar em um método generalizado capaz de agir melhor contra novos tipos de intrusos.

Métodos baseado em aprendizagem de máquina necessitam de um especialista para extrair importantes informações dos dados. Por outro lado, soluções baseadas em aprendizagem profunda melhoram efetivamente a detecção de intrusos em redes em comparação com métodos tradicionais [11] e possuem a capacidade de aprender a partir dos dados.

Os *Autoencoders* (AE) têm se tornado bastante popular na tarefa de detecção de anomalias, devido ao seu funcionamento simples e efetivo de reconstrução dos dados de entrada [12]. É possível utilizar um AE para capturar o perfil dos dados normais, com isso é possível detectar os dados que não correspondem a esse perfil previamente estabelecido.

1.1 Objetivo Geral

Esta pesquisa tem como principal contribuição, investigar e desenvolver um método de aprendizagem profunda treinado de forma semi-supervisionada para detectar intrusos em um tráfego de rede. A ideia é obter uma solução eficiente que seja treinada com poucas iterações e apresente uma boa performance em comparação com outros métodos do estado da arte. Ademais, também é proposto uma nova forma de determinar um limiar de separação entre as amostras normais e anômalas, considerando a regra empírica baseada em uma distribuição normal.

1.1.1 Objetivos específicos

- Propor um modelo de aprendizagem profunda semi-supervisionado para detectar anomalias com alta performance;
- Propor um novo algorítimo para encontrar um limiar de separação entre amostras normais e anomalias;
- Comparar o modelo proposto com outras soluções do estado-da-arte para validar a sua efetividade;

1.2 Organização do TCC

O conteúdo deste TCC está dividido em seis capítulos. As referências encontram-se nas páginas finais. A seguir, um resumo dos capítulos seguintes.

Capítulo 2. Apresenta uma revisão literária dos principais métodos de detecção de intrusos encontrados na literatura, bem como daqueles que serviram como base para este projeto.

Capítulo 3. São apresentados os detalhes fundamentais dos dados utilizados nos experimentos e da fase de pré-processamento destes dados.

Capítulo 4. É abordada uma visão geral das técnicas que foram investigadas neste TCC e todo o processo de construção do modelo proposto.

Capítulo 5. Oferece uma visão geral da implementação do modelo, dos experimentos realizados e como o mesmo foi validado.

Capítulo 6. Apresenta as considerações finais do trabalho, trazendo um resumo do que foi proposto e dos principais resultados encontrados, além de sugestões para possíveis trabalhos futuros.

Capítulo 2

Fundamentação Teórica

2.1 Aprendizagem de Máquina

Detectar anomalias usando técnicas de aprendizagem de máquina ganhou popularidade nos últimos anos, devido a sua robustez contra ataques desconhecidos em comparação com métodos tradicionais como os baseados em assinatura [13]. Dependendo da disponibilidade de rótulos nos dados, os métodos de aprendizagem de máquina podem ser categorizados em três abordagens: supervisionados, semi-supervisionados e sem supervisão.

Nos métodos supervisionados, o problema se torna uma tarefa de classificação. Para identificar se uma amostra de tráfego representa um ataque ou não é necessário ter uma quantidade suficiente de dados rotulados para que se possa alimentar e treinar determinadas estruturas, como, por exemplo, modelos baseados em redes neurais.

Existem diversos tipos de algoritmos que fazem uso de um aprendizado supervisionado, como: *K-Nearest Neighbor* (*KNN*), *Support Vector Machine* (*SVM*), *Decision Tree* (*DT*) [14] e até mesmo modelos como o *Multilayer Perceptron* (*MLP*).

Tavallaee *et al.* (2009) [15] usou o método *Naive Bayes Tree* (*NBTree*) nos dados do NSL-KDD e obteve 82.02% de acurácia. Ingre *et al.* (2015) [16] propôs uma Rede Neural Artificial para detectar anomalias e reportou uma taxa de acurácia de 81.2% no NSL-KDD. Kevric *et al.* (2017) [17] mostrou que ao combinar dois algoritmos de árvores, o *Random Tree* e o *NBTree*, obtém-se uma taxa de acurácia de 89.24% na detecção de intrusos, superando cada um dos algoritmos individualmente. Ashfaq *et al.* (2017) [18] propôs uma abordagem semi-supervisionada baseada em uma lógica difusa, utilizando dados não rotulados. O modelo é definido por uma rede neural com uma única camada interna e a saída é uma rede de associação difusa. Um valor difuso (baixo, médio ou alto) é usado para categorizar uma amostra não rotulada. O classificador é retreinado após incorporar cada categoria separadamente no conjunto

de treinamento original, obtendo uma taxa de acerto de 84.12% no conjunto de teste.

2.2 Aprendizagem profunda

Nos últimos anos, a aprendizagem profunda tem se tornado um dos temas mais populares em sistemas de detecção de intrusos. Hodo *et al.* (2017) [19] apresenta um estudo de redes neurais rasas e profundas em sistemas de detecção de intrusos e mostra a importância da extração de características na detecção de anomalias.

Em modelos híbridos é comum ter um *Autoencoder* (*AE*) como estágio inicial, para se gerar uma representação condensada da entrada original dos dados. Azar *et al.* (2017) [20] utiliza *AE* para aprender características mais discriminativas dos dados. Em seguida, ele usa algoritmos supervisionados como o *SVM* e *KNN* para a tarefa de classificação e obtém 83.34% de acurácia. Com estrutura similar, Javaid *et al.* (2016) [21] propôs um método de aprendizagem profunda em dois estágios: (1) *AE* esparso para aprender as *features* mais expressivas e (2) uma rede neural baseada em regressão *softmax* para classificação e, por fim, obtiveram um taxa de acurácia e *F1-score* de 88.39% e 90.4%, respectivamente.

Métodos híbridos também são, no final das contas, abordagens supervisionadas, ou seja, sua performance depende bastante na quantidade de rótulos existentes e do equilibro de classes no dados, isso significa que só conseguem classificar amostras desconhecidas treinando com uma grande quantidade de dados semelhantes e com rótulos correspondentes.

Em sistemas de detecção de intrusão, existe uma quantidade pequena de dados apropriados disponíveis publicamente, devido a razões como confidencialidade dos dados de uma empresa [22]. Para lidar com esse problema, métodos treinados sem supervisão (e.g., *Autoencoder*) são utilizados em detecção de anomalia, usando apenas dados representados por tráfego de pacotes normais durante o treinamento, no contexto de redes de computadores, por exemplo.

Em Sadaf *et al.* (2020) [23] é proposto um algoritmo de detecção de anomalia baseado em um limiar de separação. Se o erro de reconstrução de uma amostra for maior que esse limiar, ela é rotulada como ataque, caso contrário como normal. Esse valor é definido com base na função de perda durante o treinamento do *AE*, obtendo uma taxa de acurácia de 88.98%. Em Gharib *et al.* (2019) [24]é apresentado o *AutoIDS*, uma solução treinada de forma semi-supervisionada, que consegue distinguir entre amostras normais e anômalas de fluxos de pacotes utilizando um *AE* esparso seguido por um *AE*. O *AutoIDS* possui um valor de acurácia de 90.17%.

Capítulo 3

Metodologias e Dados

Nesta seção são descritos detalhes do conjunto de dados utilizados, bem como a metodologia empregada no pré-processamento destes dados.

3.1 Conjunto de Dados

As informações sobre os critérios usados para analisar pacotes de dados de rede de computadores em produtos comerciais não são fáceis de encontrar publicamente, devido a questões de privacidade e estratégia de negócio. No entanto, existem alguns conjuntos de dados disponíveis publicamente. O KDDCUP'99, por exemplo, foi construído com base no DARPA'98 [4]. O maior problema desta base de dados é o grande número de gravações duplicadas. Em [15] foi analisado que no conjunto de treinamento aproximadamente 78% dos pacotes da rede estão duplicados. Isso faz com que os classificadores se tornem enviesados em relação aos dados mais frequentes. Ademais, o KDDCUP'99 está desatualizado, pois não contém registros de ataques recentes. [15] propôs o a base de dados NSL-KDD, ao remover pacotes de redes duplicados no KDDCUP'99.

O NSL-KDD é dividido em dois conjuntos: KDDTrain+ e KDDTest+. Ambos contêm amostras de fluxos normais e anormais de pacotes de rede. As especificações do NSL-KDD podem ser vistas na Tabela 3.1. Os tipos de ataques contidos nesta base de dados são *Denial of Service* (DoS), *Probing*, *User to Root* (U2R), e *Root to Local* (R2L). O KDDTrain+ contém um total de 24 ataques, enquanto que o KDDTest+ possui 17 tipos de ataques que não estão inclusos no outro. Essa variabilidade de fluxos maliciosos que são exclusivos no conjunto de teste, servirão para mostrar como o modelo desenvolvido generaliza bem para ataques não vistos.

Categoria	Tipo	KDDTrain+	KDDTest+
Normal	–	67.343	9.711
Ataques	DoS	45.927	7.458
	Probing	11.656	2.421
	R2L	995	2.754
	U2R	52	200
Total	–	125.973	22.544

Tabela 3.1: Número de registros do conjunto NSL-KDD.

3.2 Pré-processamento

3.2.1 Codificação das Variáveis Categóricas

Dado que o NSL-KDD contêm *features* não numéricas, i.e., *protocol_type*, *service* e *flag*. Essas três colunas são codificadas utilizando o método *one-hot encoding* para cada valor não numérico em um vetor $\mathbf{v} = [v_1, v_2, \cdots, v_{d-1}, v_d]$, no qual d representa o número total de atributos da coluna original e $v_i \in [0, 1]$. Por exemplo, na variável *protocol_type* há três tipos de protocolos diferentes "tcp", "udp" e "icmp", após aplicado o *one-hot encoding* obtém-se os seguintes vetores: $[1, 0, 0]$, $[0, 1, 0]$, $[0, 0, 1]$, como está indicado na Figura 3.1.

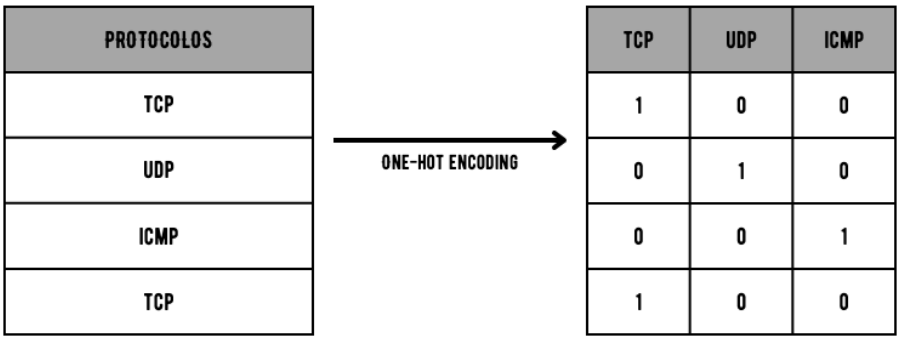


Figura 3.1: Cada exemplo categórico é transformado em um vetor de dimensão igual a quantidade de atributos únicos do coluna.

As colunas *services* e *flags* são codificadas da mesma forma. Cada uma com 70 e 11 categorias diferentes, respectivamente. Após o conversão das variáveis categóricas

apresentadas, tem-se um total de 84 *features*, que com adição das 38 numéricas resulta, então, em um total de 122 *features*.

Ademais, neste trabalho, o modelo será treinado utilizado uma abordagem de classificação binária, ou seja, não será levado em conta qual o tipo do ataque referente aquele tráfico de pacotes. Para isto, os rótulos serão transformados em valores binários, 0 para categoria normal e 1 referente a algum ataque.

3.2.2 Separação dos Dados

Após realizar o pré-processamento, os dados são divididos em conjuntos de treinamento, validação e teste. Primeiramente, o KDDTrain+ é separado em dois subconjuntos:

- **A**: Contém apenas as amostras normais;
- **B**: Contém as amostras referente aos ataques;

No conjunto de treinamento são usados apenas 80% dos dados do subconjunto A. Os dados anômalos do subconjunto B serão adicionados aos 20% restantes do A, definindo o conjunto de validação. O KDDTest+ será empregado apenas na fase de teste [25], pois contém tipos de ataques diferentes dos que estão presentes no KDDTrain+, servindo como uma forma de validar a generalização do modelo. Essa divisão pode ser melhor entendida observando a Tabela 3.2.

	KDDTrain+		**KDDTest+**
Categoria	**Treino**	**Validação**	**Teste**
Normal	53874	13469	9711
Ataque	–	58630	12833
Total	53874	72099	22544

Tabela 3.2: Distribuição dos dados de treinamento, validação e teste.

3.2.3 Normalização

Normalização ou padronização é uma das técnicas mais utilizadas em pré-processamento de dados. O objetivo da padronização é reescalonar as *features* numéricas para que possuam a mesma escala, sem perder informação e mantendo as proporções dos dados de origem. Além disso, alguns algoritmos são beneficiados pela normalização prévia dos dados como as redes neurais artificiais e aqueles baseados em distâncias como o *K-Nearest Neighbors*. O método de padronização utilizado é descrito pela Equação 3.1, o qual mapeia todas as *features* numéricas no intervalo $[0, 1]$, ou seja, todas as 122.

$$X_{scaled} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{3.1}$$

É importante notar que o reescalonamento das amostras deverá ser feito após a divisão dos dados, usando apenas o conjunto de treinamento e salvando os parâmetros obtidos (X_{min} e X_{max}). Em seguida, deve-se normalizar os conjuntos de validação e teste, usando os parâmetros obtidos anteriormente. Usar qualquer informação do conjunto de teste para realizar a normalização é um potencial viés na avaliação da performance do modelo.

Capítulo 4

Método de Detecção de Anomalia Baseado em Autoencoder

4.1 Autoencoders

O *Autoencoder* (AE) é um tipo de rede neural treinado com pouca ou nenhuma supervisão, com o objetivo de aprender a representar as características mais relevantes de uma entrada [26]. Isto pode ser obtido ao reconstruir o conjunto de entrada de maneira eficiente.

Um AE pode ser dividido em duas partes: A primeira parte é definida por um codificador $z = E_\phi(x)$, que mapeia a entrada $x \in \mathcal{X}$ em uma variável latente $z \in \mathcal{Z}$. A segunda parte é descrita por um decodificador, $x' = D_\theta(z)$, responsável pela reconstrução da entrada, i.e., $x' \in \mathcal{X}$.

A princípio pode parecer que reconstruir um dado qualquer seja pouco útil, porém ao treinar o AE para realizar tal tarefa, a variável latente (z) passa a conter de forma automática os atributos mais importantes da entrada. Uma forma de obter isso é reduzindo a dimensão inicial do conjunto de entrada ($z < x$). Uma estrutura desse tipo é conhecida como *autoencoder undercomplete*, como apresentado na Figura 5.1.

Para treinar um AE é necessário medir a qualidade da reconstrução através de uma função de perda $\mathcal{L} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$, de tal forma que $\mathcal{L}$ mede o quão dissimilar é a entrada x da sua suposta reconstrução x'. Na maioria das vezes a função de perda escolhida é a distância euclidiana, com isso o AE deverá ser treinado de forma a minimizar a Equação 4.1 de acordo com os parâmetros θ e ϕ.

$$\mathcal{L}(\theta, \phi) = \|X - D_\theta(E_\phi(X)\|_2^2 \tag{4.1}$$

Deve-se tomar cuidado com redes muito profundas, pois é possível que o modelo "memorize" os dados de treinamento realizando a tarefa de reconstrução sem apren-

der um espaço latente com características úteis acerca da distribuição dos dados, esse tipo de problema é mais conhecido como *overfitting*.

Um *AE* é capaz de descrever os dados através de uma relação não-linear, isto pode ser visto como uma poderosa generalização do *PCA* [27]. Para dados de alta dimensionalidade, *autoencoders* conseguem aprender a representar dados de maneira complexa, diferentemente do *PCA*, que projeta os dados em um espaço de dimensão reduzida por meio de uma transformação linear.

4.2 Distribuição Normal

Antes de estimar uma variável, a primeira tarefa que deve ser realizada é entender como a mesma está distribuída, e em seguida atribuir probabilidades aos eventos. A teoria da probabilidade descreve quão provável é que uma variável aleatória aconteça em determinado espaço amostral. Uma função densidade de probabilidade $f(x)$ (fdp) é responsável por atribuir a probabilidade de uma variável aleatória contínua em um intervalo infinitesimal dx ao invés de um valor absoluto. Para se determinar a probabilidade de um espaço de pontos, integra-se a fdp sobre esse mesmo espaço. Considerando o caso de uma variável unidimensional, i.e., $x \in \mathbb{R}$, a probabilidade de x estar dentro de um intervalo $[a, b]$ é dado por

$$P(a \leq x \leq b) = \int_a^b f(x)\, dx \tag{4.2}$$

Uma distribuição normal, também conhecida como Gaussiana $X \sim \mathcal{N}(\mu, \sigma^2)$ é uma distribuição simples que depende apenas de dois parâmetros do conjunto de dados: a média $\mu \in \mathbb{R}$ (que é igual a mediana e a moda), e o desvio padrão $\sigma \in (0, \infty)$. A fdp de uma distribuição normal é

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\left(\frac{x-\mu}{\sigma}\right)^2}. \tag{4.3}$$

O parâmetro μ define onde se encontra o pico central da distribuição, enquanto que a largura é controlada pelo desvio padrão σ. Veja a plotagem de uma função densidade de probabilidade na Figura 4.1.

De acordo com o Teorema do Limite Central (TCL), a soma e a média de um conjunto n ($n \to \infty$) de variáveis aleatórias é aproximadamente uma distribuição normal [28]. Com isso, é possível modelar sistemas complexos do mundo real, usando distribuições normais.

4.3 Regra Empírica

Em estatística, a regra empírica, também conhecida como, regra 68-95-99.7, é na realidade uma aproximação. Ela estabelece que se uma variável segue uma distribuição normal, 68%, 95%, 99,7% dos valores do conjunto se concentram no intervalo de um, dois e três desvios padrões da média [29], respectivamente.

Essas aproximações podem ser expressas, matematicamente, da seguinte forma,

$$P(\mu - 1\sigma \leq X \leq \mu + 1\sigma) \approx 68.27\%$$
$$P(\mu - 2\sigma \leq X \leq \mu + 2\sigma) \approx 95.45\%$$
$$P(\mu - 3\sigma \leq X \leq \mu + 3\sigma) \approx 99.73\%$$

em que P é a função de probabilidade, μ a média e σ o desvio padrão e X é uma variável aleatória.

A regra prática de três sigmas (ou regra 3σ) expressa uma heurística de que quase todos os elementos da distribuição normal está dentro do intervalo de $\pm 3\sigma$ da média. Dito isso, pode-se pensar que qualquer valor fora desse intervalo seja uma anomalia.

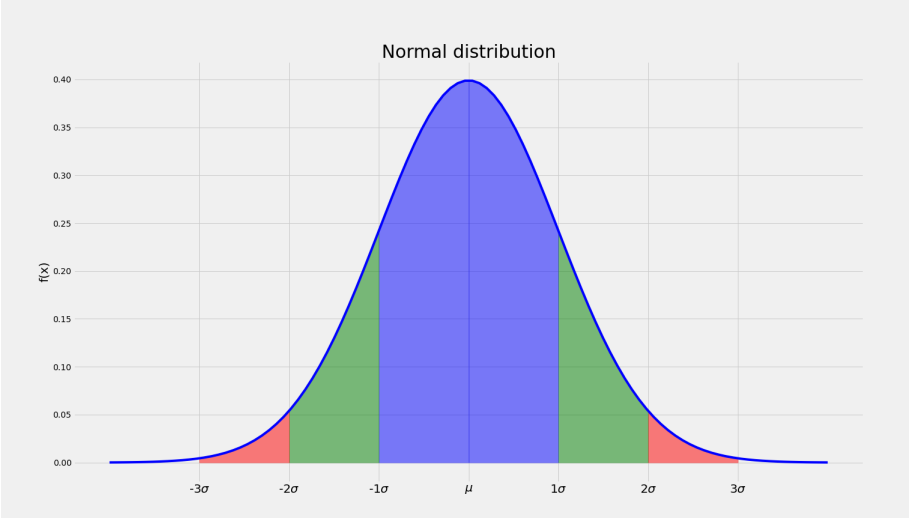


Figura 4.1: Distribuição Gaussiana. Neste exemplo, tem-se uma distribuição normal padrão com $\mu = 0$ e $\sigma = 1$.

4.4 Método Proposto

O modelo de detecção proposto usa como base uma estrutura de *Autoencoder*, que é um tipo de neural *feedforward* responsável por comprimir a entrada em uma representação latente e, em seguida, reconstruir a entrada a partir dessa representação. A arquitetura geral do *AE* proposto nesta pesquisa é representada na Figura 5.1. O

Autoencoder é treinado de forma a reduzir a função de perda, também conhecida, nesse contexto, como erro de reconstrução. O erro de reconstrução utilizado é a distância euclidiana, como é apresentado na Equação 4.4.

$$\mathcal{L}_{REC} = \left\| X - \hat{X} \right\|_2^2 \tag{4.4}$$

O modelo implementado utiliza o erro de reconstrução para detectar se existe alguma anomalia presente em um tráfego de pacotes na rede. Durante a fase de treinamento, o modelo é treinado apenas com dados que apresentam fluxos normais na rede, isso forçará o AE a aprender bem como codificar amostras normais em um espaço de dimensão menor do que o original. Uma vez treinado, é esperado que qualquer entrada que presente uma anomalia possua um alto erro de reconstrução, visto que o modelo é treinado para reconstruir unicamente exemplos normais.

Nesta pesquisa foram testadas diferentes arquiteturas, como será mostrado na próxima seção. Todas as camadas internas são totalmente conectadas (densas) e usam a tangente hiperbólica (*Tanh*) como função de ativação. Porém, na última camada do decodificador foi usado a função sigmoide para mapear os dados no intervalo entre [0, 1]. Os pesos do modelo são otimizados através do algoritmo de otimização *Adam*.

Os parâmetros do modelo são otimizados de forma a minimizar a função de perda $\mathcal{L}_{REC}$, tornando os dados que não possuem anomalia com baixo $\mathcal{L}_{REC}$. Durante a fase de teste, se alguma exemplo tiver um erro de reconstrução maior do que um limiar δ, este será marcado como um intruso ou anomalia, como é mostrado abaixo.

$$D_{AE}(x \in unknown) = \begin{cases} normal & \text{se } \mathcal{L}_{REC} \geq \delta \\ anomalia & c.c. \end{cases} \tag{4.5}$$

Um ponto importante a se notar é a forma de definir o limiar δ. Considerando que a distribuição é normal e considerando o que foi visto na Seção 4.2, pode se utilizar σ, 2σ ou 3σ como um possível δ para separar amostras normais das anômalas.

Neste estudo, porém, foi proposto uma abordagem para escolher um limiar a partir de um intervalo $\delta \in [0, 3\sigma)$, que melhor separe o conjunto de validação. O limiar ótimo é aquele que maximiza a Equação 5.1 e a Equação 5.4 ao mesmo tempo. Os detalhes sobre o algoritmo de definição do limiar são apresentados no algoritmo 1.

Algoritmo 1: Definir o melhor limiar de separação

Entrada: $\mathcal{L}(X_{train})$, X_{val}, y_{val}

Saída: Retorna o melhor limiar que obtém as melhores métricas no conjunto de validação.

início
  $bestF1 \leftarrow 0$
  $bestAcc \leftarrow 0$
  $bestAlpha \leftarrow 0$
  enquanto $\alpha \leq 3.0$ **faça**
    $\delta \leftarrow mean(\mathcal{L}_{REC}) + \alpha \cdot std(\mathcal{L}_{REC})$
    $pred \leftarrow (X_{val})$
    $f_1, acc \leftarrow \texttt{getScores}(y_{val}, pred, \delta)$
    se $(f_1 > bestF1)$ *and* $(acc > bestAcc)$ **então**
      $bestF1 \leftarrow f1$
      $bestAcc \leftarrow acc$
      $bestAlpha \leftarrow \alpha$
    fim
    $\alpha \leftarrow \alpha + 0.1$
  fim
fim
$\delta \leftarrow mean(\mathcal{L}_{REC}) + melhorAlpha \cdot std(\mathcal{L}_{REC})$
retorna δ

Capítulo 5

Experimentos

De início, são tratadas as métricas usadas para validar a performance do modelo proposto. Em seguida, serão abordados detalhes da implementação, bem como os resultados encontrados e a comparação com outros trabalhos da literatura.

5.1 Métricas de Validação

Para validar a performance do modelo proposto, foram usadas quatro métricas: *accuracy*, *precision*, *recall* e *F1-score*. Essas métricas são calculadas de acordo com os seguintes parâmetros:

- **True Positive (TP)**: Quantidade de anomalias que foram classificadas corretamente.
- **True Negative (TN)**: Quantidade de dados normais que foram classificados corretamente.
- **False Positive (FP)**: Quantidade de dados normais que foram classificados incorretamente como anomalias.
- **False Negative (FN)**: Quantidade de dados anômalos que foram classificados incorretamente como normais.

Accuracy determina a proporção total de predições corretas.

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.1}$$

Precision denota a taxa de anomalias classificadas corretamente dentre o total de amostras preditas como anômalas, como mostra a Equação 5.2.

$$precision = \frac{TP}{TP + FP} \tag{5.2}$$

De acordo com Equação 5.3, *recall* pode ser definida como o número de amostrar anômalas classificadas corretamente dentre todas as amostras são anomalias reais.

$$recall = \frac{TP}{TP + FN} \tag{5.3}$$

E, por último, *F1-score* é a média harmônica da Equação 5.3 e da Equação 5.2, como é mostrado na Equação 5.4.

$$f_1 = \frac{2 \cdot precision \cdot recall}{precision + recall} \tag{5.4}$$

5.2 Implementação

O modelo foi implementado usando a biblioteca *PyTorch*[1] para aprendizagem de máquina. Além deste, foi utilizado a biblioteca pública *Pytorch Lightning*[2], responsável por organizar o código *PyTorch* a fim de facilitar a leitura e a reprodutibilidade dos experimentos de aprendizagem profunda.

O *AE* proposto foi implementado e treinado em um computador Dell G15 com as seguintes especificações:

- Sistema operacional Ubuntu 22.04.5 LTS com uma arquitetura x86-64;
- Processador Intel Core i7 com 8 núcleos, memória cache de 24MB e uma frequência de *clock* de até 4.6GHz;
- Memória RAM de 16GB com uma frequência de 3200MHz
- Placa de vídeo NVIDIA GeForce RTX 3060 com 6GB disponíveis e CUDA na versão 11.4;

O modelo foi treinado durante 10 épocas (definidas de forma empírica), utilizando um *batch* de tamanho 256. Além disso, para otimização dos parâmetros utilizou-se uma taxa de aprendizado equivalente a 10^{-3}. Depois de aplicados todos os passos referentes a Seção 3.2, os dados de treino são misturados no começo de cada época para fazer com que o treinamento convirja mais rápido para um mínimo global. É importante ressaltar mais uma vez que durante o treino, os rótulos não são utilizados, dado que apenas dados sem anomalias passam pelo treinamento.

[1] `https://pytorch.org/`
[2] `https://www.pytorchlightning.ai/`

5.3 Resultados

Nesta seção serão apresentadas as análises realizadas com o objetivo de verificar a performance do modelo de detecção proposto. Além disso, será realizada a comparação entre métodos de detecção de intrusos da literatura considerados estado-da-arte para o problema e o método desenvolvido neste trabalho de graduação.

5.3.1 Diferentes arquiteturas

Nesta seção, foram examinadas diferentes arquiteturas e avaliadas de acordo com as métricas definidas. Os experimentos foram realizados alterando o número de camadas internas do AE. Na Tabela 5.1, os resultados mostram que uma arquitetura com 7 camadas internas apresenta uma melhor performance em comparação com as outras. Com isso, pode-se afirmar que a performance de um modelo é impactada pela quantidade de camadas usadas na sua implementação.

Arquitetura	**Accuracy**	**Precision**	**Recall**	**F1-Score**
[64]	87.20%	89.21%	88.22%	88.67%
[64, 32, 64]	87.80%	91.16%	87.04%	89.01%
[64, 32, 16, 32, 64]	87.86%	90.91%	87.43%	89.10%
[64, 32, 16, 8, 16, 32, 64]	**90.49%**	**89.86%**	**93.89%**	**91.80%**
[64, 32, 16, 8, 4, 8, 16, 32, 64]	88.17%	91.09%	87.84%	89.40%
[64, 32, 16, 8, 4, 2, 4, 8, 16, 32, 64]	84.34%	90.61%	80.88%	85.42%

Tabela 5.1: Performance do AE utilizando diferentes arquiteturas. Todos os modelos foram treinados usando o KDDTrain+ e testados no KDDTest+.

Desse modo, a arquitetura que apresentou melhor resultado pode ser vista graficamente através da Figura 5.1.

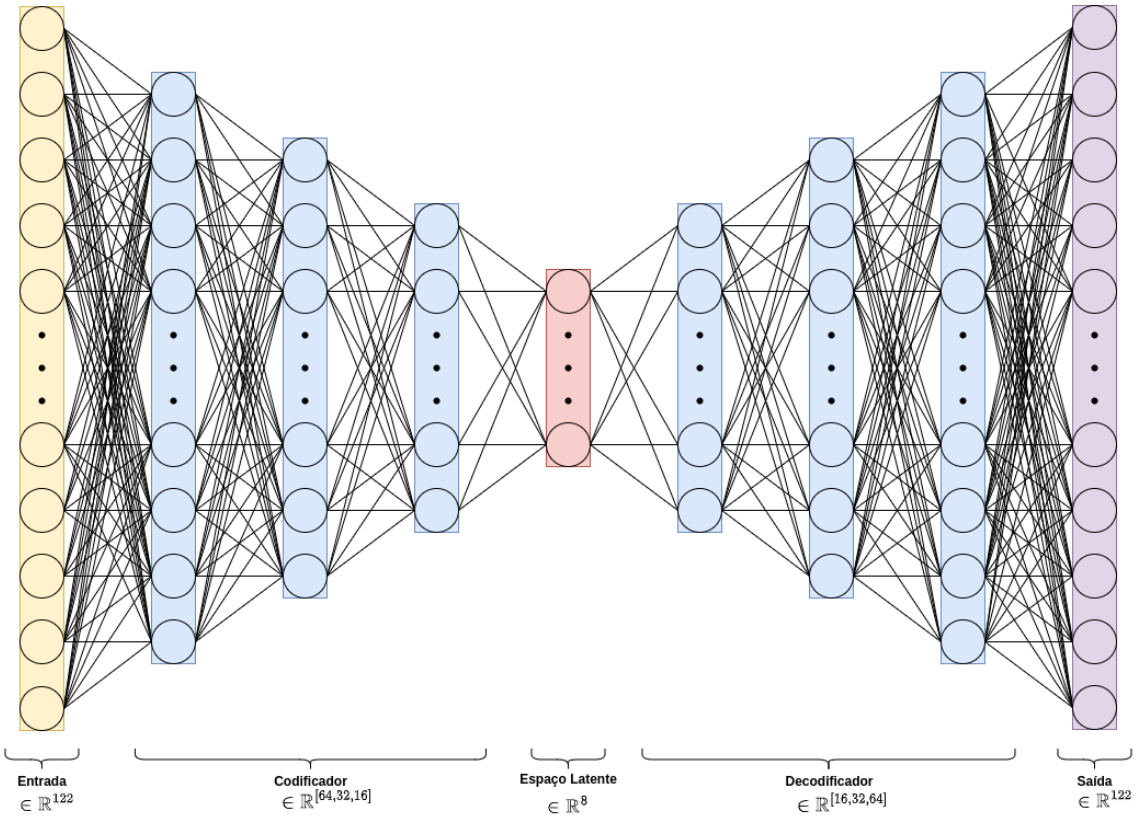


Figura 5.1: Esquema geral da arquitetura proposta. Nesta figura, a camada de entrada e a camada de saída representam os vetores da entrada e da sua versão reconstruída, respectivamente, ambos possuindo uma dimensão $\mathbb{R}^{122}$. O codificador é definido por vetores de dimensões $\mathbb{R}^{[64,32,16]}$. O decodificador é espelhado a fim de realizar a reconstrução. O espaço latente de dimensão $\mathbb{R}^{8}$ representa a versão condensada da entrada que será reconstruída pelo decodificador.

5.3.2 Representação dos dados no espaço latente

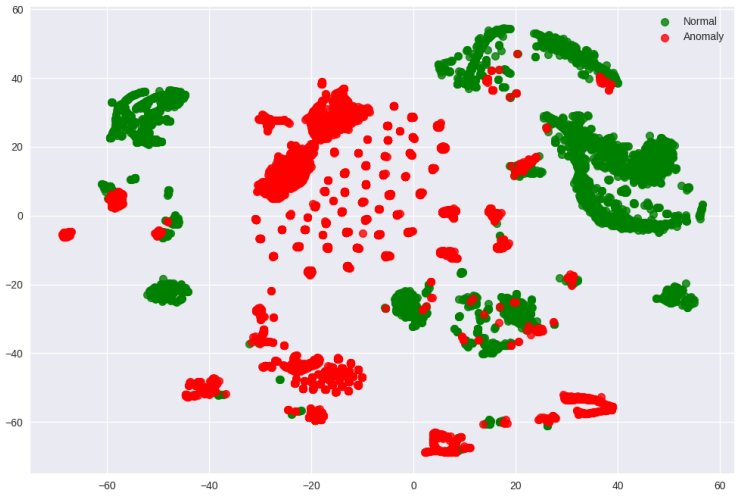
Para visualizar os dados no espaço latente foi utilizado o algoritmo *t-SNE*, o qual é bastante empregado para entender dados de alta dimensionalidade e projetá-los em um espaço de dimensão menor (como 2D ou 3D) [30]. O *t-SNE*[3] mede a similaridade entre pares de pontos em um espaço de alta dimensão e em um espaço de baixa de dimensão. Por fim, o algoritmo busca minimizar essa medida de similaridade através de uma função de custo usando o gradiente descendente.

Na Figura 5.2 estão apresentados dois gráficos referentes ao conjunto de validação após a aplicação do *t-SNE*. Na Figura 5.2a observa-se os pontos concernentes a cada exemplo do conjunto de validação depois de processados pelos algoritmo *t-SNE*. Nota-se que pontos concernentes a anomalias estão mais espalhados do que pontos normais.

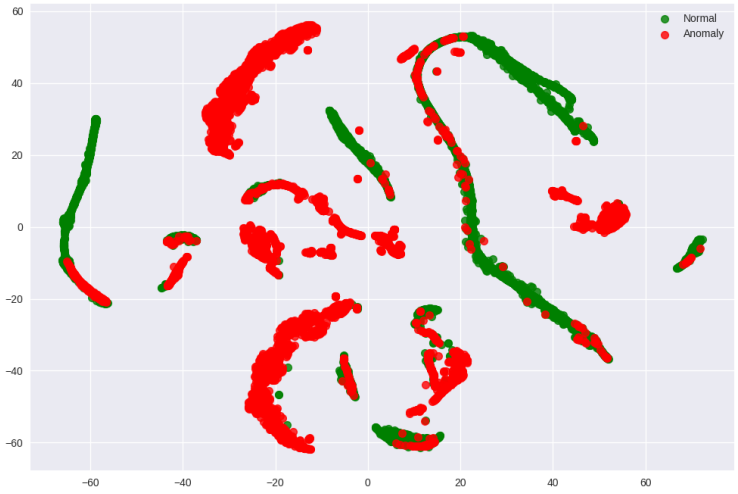
Após o modelo ser treinado, é obtido uma representação no espaço latente do

[3]Artigo explicando de maneira mais aprofundada o funcionamento do *t-SNE*.

conjunto de validação. O espaço latente obtido é usado como entrada no *t-SNE* e obtém-se a Figura 5.2b. Nesse caso, é possível perceber que os pontos que representam tráfego de pacotes normais estão mais condensados do que na Figura 5.2a e isto se deve ao fato de que através do *AE* é possível mapear o espaço original dos dados em um novo espaço com características mais expressivas para realizar a reconstrução dos dados em seguida.



(a) Representação dos dados originais.



(b) Representação dos dados no espaço latente.

Figura 5.2: Visualização do conjunto de validação usando o *t-SNE* antes e depois do *AE*.

Como foi dito na Subseção 3.2.2, o *AE* foi treinado apenas com dados referentes a tráfegos normais na rede. Dito isso, após o treinamento, mediu-se o erro de

reconstrução para cada uma das amostras de treinamento. Esse vetor de erros foi armazenado para definir o melhor limiar de separação entre amostras normais e anômalas por meio do algoritmo 1.

5.3.3 Definindo o Limiar de Separação

Como explicado na Seção 4.4, foram testados diferentes *alphas* (veja a Figura 5.3) que indicam frações de desvios padrões, ou seja, novos percentuais da regra empírica (ver a Seção 4.3) como está descrito no algoritmo 1.

Figura 5.3: O eixo horizontal define parcelas variadas de desvios padrão, enquanto que o eixo vertical apresenta os *scores* de classificação no conjunto de validação. A linha vertical tracejada é determinada pelo valor de *alpha* que apresenta maior *F1-score* e acurácia.

De acordo com a Figura 5.3, o valor *alpha* utilizado para definir o limiar foi de aproximadamente $\alpha = 0.9$, esse valor foi o que apresentou melhor separação entre dados normais e anômalos no conjunto de validação. Para verificar como o limiar escolhido separa o conjunto de teste foi gerado um histograma (veja a Figura 5.4) com base no erro de reconstrução dos dados de teste. É possível observar que boa parte parte dos dados anômalos se encontram a direita do limiar, porém é notável um pequeno percentual de falsos positivos, o que é normal ter acontecido, já que ao escolher o limiar a ideia era tentar maximizar as métricas da Figura 5.3 e não apenas minimizar a quantidade de falsos positivos.

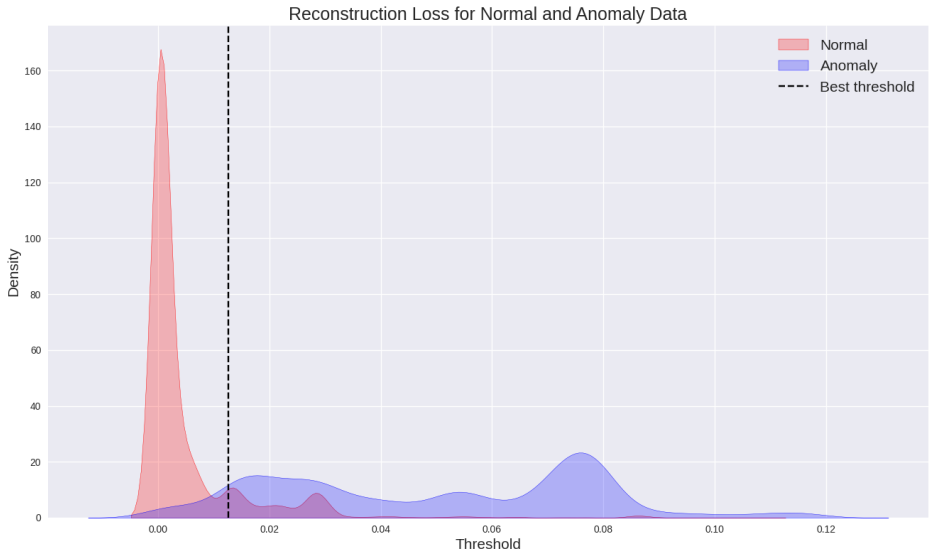


Figura 5.4: *Best threshold* foi o melhor limiar encontrado através do algoritmo 1 para separar distribuição dos erros de reconstrução dos dados de teste.

5.3.4 Comparação com outros métodos

Nesta seção o método proposto será comparado com abordagens supervisionadas bastante conhecidas na literatura [21], [31], bem como métodos semi-supervisionados baseados em *Autoencoders* [25], [24], [32], [23], uma vez que estes apresentam arquiteturas semelhantes. Todos os métodos utilizados para comparação foram avaliados através da base de dados NSL-KDD.

Além disso, é importante citar que todos os métodos presentes na Tabela 5.2 foram aplicados para problemas envolvendo apenas duas classes. Os métodos foram comparados através das métricas de *accuracy* e *F1-score*. A partir da Tabela 5.2 pode se perceber que o modelo desenvolvido neste projeto obteve uma *accuracy* maior que 90% superando outros métodos de detecção de intrusos, em termos de *accuracy* e *F1-score*. O *AE* proposto supera em performance até mesmo os métodos supervisionados, como o *LSTM* [33].

Método	Supervisionado	Accuracy	F1-Score
STL [21]	✓	88.39%	90.4%
Random tree + NBTree [17]	✓	89.24%	–
RNN-IDS [31]	✓	83.28%	–
AE [25]	–	88.28%	89.51%
Denoising AE [25]	–	88.65%	89.28%
LSTM [33]	✓	89.00%	–
AutoIDS [24]	–	90.17%	91.42%
AE [32]	–	84.21%	81.98%
AE [23]	–	88.98%	90.61%
Método Proposto	–	**90.49%**	**91.80%**

Tabela 5.2: Comparação do modelo proposto com soluções do estado da arte. Todos os modelos foram treinados usando os dados do KDDTrain+ e testados no KDD-Test+.

Para mostrar o processo de generalização do *AE* proposto, o treinamento e o teste são realizados uma segunda vez como em [25] e em [24] usando apenas o KDDTrain+. Neste caso, a fase de teste é realizada com base nas anomalias que estão presentes no conjunto KDDTrain+. Neste segundo experimento, o KDDTrain+ é dividido em três conjuntos: treinamento, validação e teste como em [24]. A distribuição dos conjuntos após a divisão é apresentada na Tabela 5.3.

	KDDTrain+		
Categoria	**Treino**	**Validação**	**Teste**
Normal	53874	6734	6735
Ataque	–	29315	29315
Total	53874	36049	36050

Tabela 5.3: Distribuição dos dados de treinamento, validação e teste

Com base no segundo experimento, o modelo proposto foi comparado com algumas das soluções estado-da-arte para o problema e apresentadas na Tabela 5.2, as quais também reportaram a performance referente ao conjunto KDDTrain+. De acordo com a Tabela 5.4, o *AE* desenvolvido apresenta um *F1-score* melhor em comparação com outros métodos semi-supervisionados [25], [24], no entanto, tem uma menor acurácia do que os modelos supervisionados [31], [21]. Métodos que utilizam supervisão aprendem melhor as características mais expressivas dos ataques do conjunto de treino. Com isso, quando os mesmos ataques aparecem no conjunto de teste, a taxa de acerto tende a ser bem maior.

Método	Accuracy	Precision	Recall	F1-Score
AE [25]	93.62%	91.39%	96.33%	93.80%
Denoising AE [25]	94.35%	94.26%	94.43%	94.35%
AutoIDS [24]	96.45%	95.56%	97.43%	96.49%
STL [21]	98.30%	–	–	**98.84%**
RNN-IDS [31]	**98.81%**	–	–	–
Método Proposto	95.83%	**97.18%**	**97.72%**	97.44%

Tabela 5.4: Comparação do modelo proposto com outras soluções do estado-da-arte. Todos os modelos foram treinados e testados usando o KDDTrain+.

De acordo com a Tabela 5.2 e a Tabela 5.4, a solução proposta supera a das soluções anteriores, se mostrando eficiente na descoberta de novos ataques.

Algumas das soluções [24], [25] possuem uma arquitetura menos complexas, com menos neurônios. Isso é uma forma de reduzir as chances de que o modelo tenha *overfitting*. Porém, o objetivo do modelo proposto é que ele se ajuste quase que perfeitamente aos dados normais. Isso garante um erro de reconstrução maior em amostras não usadas no treinamento (anomalias).

Autoencoders com mudanças abruptas na dimensão de uma camada para a outra tendem a fazer com que informações relevantes sejam perdidas no processo de reconstrução. Uma das vantagens do modelo proposto é a suavização de uma camada para outra, i.e., a dimensão da entrada não diminui abruptamente de uma canada para outra, mas a sua dimensão é reduzida pela metade até chegar no espaço vetorial de dimensão $\mathbb{R}^8$, como mostrado na Figura 5.1.

Capítulo 6

Conclusões

Neste projeto foi proposto um AE com 7 camadas internas, treinado de uma forma semelhante a um modelo semi-supervisionado para realizar a detecção anomalias em um tráfego de rede. Para classificar uma amostra como anomalia foi desenvolvido um algoritmo, usando como o base a regra empírica, a fim de de encontrar o melhor limiar de separação entre amostras diferentes.

Todos os passos mostrados ao longo do projeto foram resultados de um estudo intensivo acerca dos diversos métodos apresentados na literatura. O método implementado foi avaliado usando o conjunto de dados NSL-KDD. Além disso, algumas métricas foram utilizadas para reconhecer a superioridade do modelo apresentado em relação a algumas das abordagens do estado da arte. O método proposto obteve uma taxa de acurácia de 90.49% e um *F1-score* de 91.80% superando alguns modelos do estado da arte, como o [23].

Acredita-se que o modelo proposto ainda funcionará bem na detecção de intrusos em outros conjuntos. Porém, é necessário futuramente testar a eficiência do modelo proposto em ambientes de redes de de grande escala do mundo real, incorporando as limitações e capacidades dos sistemas reais. Também planeja-se estender o trabalho atual para detectar qual o tipo de ataque e não apenas se é um ataque. Uma proposta que pode ser sugerida para trabalhos futuros de detecção de intrusos é a utilização de redes neurais baseadas em grafos, campo que vem sendo bastante estudado na área de aprendizagem profunda.

Referências Bibliográficas

[1] GÓMEZ, J., GIL, C., PADILLA, N., et al. "Design of a snort-based hybrid intrusion detection system". In: *International Work-Conference on Artificial Neural Networks*, pp. 515–522. Springer, 2009.

[2] KRUEGEL, C., VALEUR, F., VIGNA, G. *Intrusion detection and correlation: challenges and solutions*, v. 14. Springer Science & Business Media, 2004.

[3] SCHUBA, C. L., KRSUL, I. V., KUHN, M. G., et al. "Analysis of a denial of service attack on TCP". In: *Proceedings. 1997 IEEE Symposium on Security and Privacy (Cat. No. 97CB36097)*, pp. 208–223. IEEE, 1997.

[4] KHRAISAT, A., GONDAL, I., VAMPLEW, P., et al. "Survey of intrusion detection systems: techniques, datasets and challenges", *Cybersecurity*, v. 2, n. 1, pp. 1–22, 2019.

[5] HUBBALLI, N., SURYANARAYANAN, V. "False alarm minimization techniques in signature-based intrusion detection systems: A survey", *Computer Communications*, v. 49, pp. 1–17, 2014.

[6] DUA, S., DU, X. *Data mining and machine learning in cybersecurity*. CRC press, 2016.

[7] CHAPANERI, R., SHAH, S. "A comprehensive survey of machine learning-based network intrusion detection", *Smart Intelligent Computing and Applications*, pp. 345–356, 2019.

[8] ALKASASSBEH, M., ALMSEIDIN, M. "Machine learning methods for network intrusion detection", *arXiv preprint arXiv:1809.02610*, 2018.

[9] LASKOV, P., DÜSSEL, P., SCHÄFER, C., et al. "Learning intrusion detection: supervised or unsupervised?" In: *International Conference on Image Analysis and Processing*, pp. 50–57. Springer, 2005.

[10] KIRAN, B. R., THOMAS, D. M., PARAKKAL, R. "An overview of deep learning based methods for unsupervised and semi-supervised anomaly detection in videos", *Journal of Imaging*, v. 4, n. 2, pp. 36, 2018.

[11] DONG, B., WANG, X. "Comparison deep learning method to traditional methods using for network intrusion detection". In: *2016 8th IEEE international conference on communication software and networks (ICCSN)*, pp. 581–585. IEEE, 2016.

[12] HAWKINS, S., HE, H., WILLIAMS, G., et al. "Outlier detection using replicator neural networks". In: *International Conference on Data Warehousing and Knowledge Discovery*, pp. 170–180. Springer, 2002.

[13] MISHRA, P., VARADHARAJAN, V., TUPAKULA, U., et al. "A detailed investigation and analysis of using machine learning techniques for intrusion detection", *IEEE communications surveys & tutorials*, v. 21, n. 1, pp. 686–728, 2018.

[14] AGRAWAL, S., AGRAWAL, J. "Survey on anomaly detection using data mining techniques", *Procedia Computer Science*, v. 60, pp. 708–713, 2015.

[15] TAVALLAEE, M., BAGHERI, E., LU, W., et al. "A detailed analysis of the KDD CUP 99 data set". In: *2009 IEEE symposium on computational intelligence for security and defense applications*, pp. 1–6. Ieee, 2009.

[16] INGRE, B., YADAV, A. "Performance analysis of NSL-KDD dataset using ANN". In: *2015 international conference on signal processing and communication engineering systems*, pp. 92–96. IEEE, 2015.

[17] KEVRIC, J., JUKIC, S., SUBASI, A. "An effective combining classifier approach using tree algorithms for network intrusion detection", *Neural Computing and Applications*, v. 28, n. 1, pp. 1051–1058, 2017.

[18] ASHFAQ, R. A. R., WANG, X.-Z., HUANG, J. Z., et al. "Fuzziness based semi-supervised learning approach for intrusion detection system", *Information sciences*, v. 378, pp. 484–497, 2017.

[19] HODO, E., BELLEKENS, X., HAMILTON, A., et al. "Shallow and deep networks intrusion detection system: A taxonomy and survey", *arXiv preprint arXiv:1701.02145*, 2017.

[20] YOUSEFI-AZAR, M., VARADHARAJAN, V., HAMEY, L., et al. "Autoencoder-based feature learning for cyber security applications". In: *2017 International joint conference on neural networks (IJCNN)*, pp. 3854–3861. IEEE, 2017.

[21] JAVAID, A., NIYAZ, Q., SUN, W., et al. "A deep learning approach for network intrusion detection system". In: *Proceedings of the 9th EAI International Conference on Bio-inspired Information and Communications Technologies (formerly BIONETICS)*, pp. 21–26, 2016.

[22] SOMMER, R., PAXSON, V. "Outside the closed world: On using machine learning for network intrusion detection". In: *2010 IEEE symposium on security and privacy*, pp. 305–316. IEEE, 2010.

[23] SADAF, K., SULTANA, J. "Intrusion detection based on autoencoder and isolation forest in fog computing", *IEEE Access*, v. 8, pp. 167059–167068, 2020.

[24] GHARIB, M., MOHAMMADI, B., DASTGERDI, S. H., et al. "Autoids: Auto-encoder based method for intrusion detection system", *arXiv preprint arXiv:1911.03306*, 2019.

[25] AYGUN, R. C., YAVUZ, A. G. "Network anomaly detection with stochastically improved autoencoder based models". In: *2017 IEEE 4th international conference on cyber security and cloud computing (CSCloud)*, pp. 193–198. IEEE, 2017.

[26] BANK, D., KOENIGSTEIN, N., GIRYES, R. "Autoencoders", *arXiv preprint arXiv:2003.05991*, 2020.

[27] LADJAL, S., NEWSON, A., PHAM, C.-H. "A PCA-like autoencoder", *arXiv preprint arXiv:1904.01277*, 2019.

[28] GOODFELLOW, I., BENGIO, Y., COURVILLE, A. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[29] CHANDOLA, V., BANERJEE, A., KUMAR, V. "Anomaly detection: A survey", *ACM computing surveys (CSUR)*, v. 41, n. 3, pp. 1–58, 2009.

[30] VAN DER MAATEN, L., HINTON, G. "Visualizing data using t-SNE." *Journal of machine learning research*, v. 9, n. 11, 2008.

[31] YIN, C., ZHU, Y., FEI, J., et al. "A deep learning approach for intrusion detection using recurrent neural networks", *Ieee Access*, v. 5, pp. 21954–21961, 2017.

[32] IERACITANO, C., ADEEL, A., MORABITO, F. C., et al. "A novel statistical analysis and autoencoder driven intelligent intrusion detection approach", *Neurocomputing*, v. 387, pp. 51–62, 2020.

[33] NASEER, S., SALEEM, Y., KHALID, S., et al. “Enhanced network anomaly detection based on deep neural networks”, *IEEE access*, v. 6, pp. 48231–48246, 2018.