Determinação Automática de Limiar de Detecção de Ataques em Redes de Computadores Utilizando Autoencoders

Luan Gonçalves Miranda, Pedro Ivo da Cruz e Murilo Bellezoni Loiola

***Resumo*— Atualmente, mecanismos de segurança digital, como os sistemas de detecção por anomalia utilizando *Autoencoders* (AE) mostram grande potencial por desviar de problemas intrínsecos aos dados como, por exemplo, o desbalanceamento. Pelo fato dos AE utilizarem um limiar de separação, de definição não trivial e não padronizada, para classificar o erro de reconstrução extraído, a definição deste limiar impacta diretamente o desempenho do sistema de detecção. Portanto, este trabalho propõe a definição automática deste limiar utilizando alguns algoritmos de aprendizado de máquina. Para tanto, foram avaliados três algoritmos: o *K-Nearst Neighbors*, o *K-Means* e a *Support Vector Machine*.**

***Palavras-Chave*— Detecção de Anomalia, Auto Codificadores, Ataques em Redes.**

***Abstract*— Currently, digital security mechanisms like Anomaly Detection Systems using Autoencoders (AE) show great potential for bypassing problems intrinsic to the data, such as data imbalance. Because AE use a non-trivial and non-standardized separation threshold to classify the extracted reconstruction error, the definition of this threshold directly impacts the performance of the detection process. Thus, this work proposes the automatic definition of this threshold using some machine learning algorithms. For this, three algorithms were evaluated: the K-Nearst Neighbors, the K-Means and the Support Vector Machine.**

***Keywords*— Anomaly Detection, Autoencoders, Network Attacks.**

I. Introdução

Atualmente, a segurança digital pode ser vista como uma área de extrema importância econômica, tendo em vista o ativo de altíssimo valor que se tornou a informação, denominando a época atual como a "Era da Informação". Com isso, mecanismos de proteção de dados e sistemas vêm sendo desenvolvidos para suprir esta necessidade. Uma das abordagens estudadas é a dos Sistemas de Detecção de Intrusão (IDS, do inglês *Intrusion Detection Systems*), responsáveis por monitorar e inspecionar as atividades em rede com o intuito de detectar ameaças que tentem violar a segurança [1], [2].

A estrutura clássica de um IDS baseia-se em quatro componentes: o Decodificador, que traduz os dados de tráfego codificados pelo protocolo IP e tem como missão decodificar estes dados para serem utilizados posteriormente; o Pré-processador, onde é feita a extração, criação, redução, conversão ou normalização dos dados decodificados; o Sistema de Decisão, onde se realiza a detecção de ataque; e o Mecanismo de Defesa, sendo executado ao detectar um ataque [1], [2].

Em um IDS, o componente mais importante é o sistema de decisão, pois é nele que todo o processo de detecção de ataque ocorre. Duas abordagens podem ser utilizadas em sua construção: a abordagem por Mau-Uso, dando origem aos Sistemas de Detecção por Mau-Uso (MDS, do inglês *Misuse Detection Systems*) e a abordagem por Anomalia, dando origem aos Sistemas de Detecção por Anomalia (ADS, do inglês *Anomaly Detection Systems*) [3], [4]. Nos MDS, são utilizadas características previamente conhecidas dos ataques para realizar a detecção dos mesmos. Já nos ADS, é utilizado o desvio de padrão nos dados, sendo este padrão determinado pelo uso normal da rede. A hipótese é de que um ataque, ou uso anômalo da rede, provocará alteração nos dados coletados pelo IDS, permitindo assim a sua identificação. Uma vantagem dos ADS sobre os MDS é a não necessidade de conhecimento específico dos ataques a serem detectados [3], [4].

Apesar das diferenças entre as características dos dados de usuários normais e maliciosos serem pequenas, o uso de técnicas específicas, como as técnicas de aprendizado de máquina, possibilita a realização desta distinção. Em [5], por exemplo, três técnicas foram investigadas para fazer a detecção de um ataque: *Random Forests*, árvore de decisão e *REPTree*. O trabalho utilizou uma base de dados SNMP-MIB para treinar e avaliar as técnicas. Esta mesma base de dados também foi utilizada em [6], que avaliou o *AdaboostM1*, uma *RandomForest* e uma Rede Neural Perceptron Multicamadas na detecção de ataques. Já em [7] foi avaliado o uso de Redes Neurais Profundas, na base de dados CICIDS2017.

Embora amplamente utilizados em problemas de classificação, estes algoritmos não apresentam bom desempenho quando os dados disponíveis para treinamento são desbalanceados, isto é, quando há uma predominância considerável de determinadas classes em relação a outras. Este é um problema comum em IDS, visto que os dados de ataques, normalmente, aparecem em menor proporção. Neste sentido, os Auto Codificadores (AE, do inglês *Autoencoders*) têm sido utilizados por permitir o treinamento usando apenas a classe mais comum [8], [9]. Assim, o AE reduzirá o Erro de Reconstrução (RE, do inglês *Reconstruction Error*) apenas desta classe dominante, sendo que uma classe anômala produzirá um RE de maior amplitude, permitindo a distinção entre às duas classes.

Esta característica de treinamento de AE possibilita sua

Luan Gonçalves Miranda, CECS, UFABC, Santo André-SP, e-mail: luan.miranda@ufabc.edu.br; Pedro Ivo da Cruz, CECS, UFABC, Santo André-SP, e-mail: pedro.cruz@ufabc.edu.br; Murilo Bellezoni Loiola, CECS, UFABC, Santo André-SP, e-mail: murilo.loiola@ufabc.edu.br. Este trabalho foi parcialmente financiado pela CAPES (001).

utilização como um classificador através do RE extraído dele, sendo RE considerado pequeno para a classe de treinamento e maior para qualquer outra. A decisão sobre qual classe determinado RE pertence é feita através da sua comparação com um limiar. Em geral, este limiar é calculado utilizando testes de hipótese, resultando em diferentes critérios para o seu cálculo [10]–[12]. Alguns trabalhos, como os dos artigos [10]–[12], que utilizam o RE para realizar a detecção de anomalia, utilizam formas diferentes de definição desse limiar. Em [10], foi utilizado um AE eliminador de ruído e também o algoritmo de Análise de Componentes Principais (PCA, do inglês *Principal Component Analysis*). O limiar de separação foi obtido de forma a minimizar a taxa de falso positivos, i.e., a taxa do sistema identificar um ataque quando este não está ocorrendo. Em [11], foram utilizadas duas classes de AE, o AE básico e o AE Variacional, comparadas a uma Máquina de Vetor Suporte (SVM, do inglês *Support Vector Machine*). O limiar não é explicitamente calculado. Ao invés disto, vários valores de limiares diferentes foram escolhidos arbitrariamente e o desempenho das técnicas foram avaliadas também em função da escolha do limiar. Já em [12], foi considerado um ADS utilizando AE quando havia Contaminação Adversária (AC, do inglês *Adversarial Contamination*), ou seja, quando existe, nos dados normais utilizados no treinamento do AE, uma porção de dados anômalos. Isto prejudica o desempenho do ADS. Para uma análise mais profunda, proporções de AC foram consideradas no trabalho. Também foi estabelecido um critério para o cálculo do limiar considerando esta proporção. No entanto, esta proporção nem sempre é previamente conhecida, dificultando a obtenção do limiar em uma situação mais prática.

Como visto, a obtenção de um limiar para conseguir fazer a detecção utilizando AE continua sendo uma questão em aberto. Neste trabalho, ao invés de realizar o cálculo explícito do limiar, que depende muitas vezes de variáveis desconhecidas pelo projetista do sistema, propomos que esta decisão seja tomada por outro algoritmo de aprendizado de máquina. Foram considerados os algoritmos K-Vizinhos Mais Próximos (KNN, do inglês *K-Nearest Neighbors*), K-means e SVM para realizar esta tarefa. Como estes algoritmos também precisam passar por uma etapa de treinamento, também é proposto um *pipeline* para o treinamento do sistema completo, contendo o AE e o algoritmo de decisão.

O restante do trabalho é organizado da seguinte forma: a Seção II descreve as técnicas utilizadas e a Seção III detalha as etapas e ferramentas propostas neste trabalho. Já a Seção IV apresenta e discute os resultados obtidos e, por fim, a Seção V conclui o trabalho.

II. Fundamentação Teórica

A. Autoencoders

AE são redes neurais que, diferente do objetivo clássico de classificação de dados, visam reconstruir na saída o sinal de entrada após este passar por camadas intermediárias de compressão e descompressão. Estas camadas de compressão consistem em camadas com menos neurônios que as de entrada e de saída [8]. Com isso, a particularidade dos AE se dá pela estrutura da rede neural utilizada, onde a rede apresenta um gargalo em sua Camada Intermediária (IL, do inglês *Intermediate Layer*), representando uma compressão dos dados. A estrutura consiste em dois blocos, o codificador (do inglês *Encoder*) e o decodificador (do inglês *Decoder*). Codificador refere-se às camadas iniciais até a IL da rede, tendo a função de comprimir os dados de entrada. Já o decodificador refere-se às camadas finais da rede, objetivando à descompressão dos dados comprimidos e, consequentemente, à reconstrução dos dados da entrada. Estas etapas de compressão e descompressão só são possíveis pelo fato de existirem correlações entre as características inseridas na entrada [8].

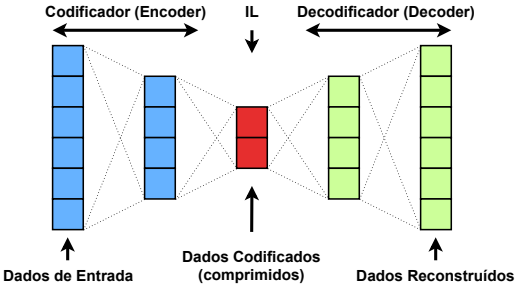

Fig. 1. Exemplo de Representação de um AE com duas camadas no codificador e decodificador.

Um exemplo de um AE pode ser visualizado na Fig. 1. Neste exemplo, o AE é composto por 5 camadas: duas de codificação (neurônios azuis), uma IL (neurônios vermelhos) e duas de decodificação (neurônios verdes).

Duas aplicações diretas dos AE são a redução de dimensionalidade e a remoção de ruído dos dados [8], [9], sendo uma alternativa ao PCA [9], algoritmo clássico utilizado para redução de dimensionalidade, uma vez que realiza mapeamentos não lineares, podendo fazer projeções mais adequadas dos dados [13]. Uma aplicação diferente dos AE refere-se a sua utilização para classificação de dados através da utilização de alguns parâmetros extraídos e utilizados como características, como no caso do RE [10], [11], [14]. Apesar da eficiência da reconstrução dos dados, a compressão existente na IL da rede provoca um pequeno RE. A tentativa de construção de um AE eficiente consiste em reduzir este erro de tal forma que a saída seja a mais próxima possível à entrada. Esta forma de utilizá-lo possibilita a construção de ADS baseado em AE.

B. Autoencoders em ADS

AE podem ser utilizados em ADS por permitirem a extração de algumas características, sendo a principal delas o RE. A premissa por trás da utilização do RE tem a ver com o fato de que, dado que o AE foi treinado para minimizar o RE com dados da classe majoritária, o RE desta classe sempre estará próximo a zero. Por outro lado, ao receber dados anômalos na sua entrada, o RE resultante será maior. Com isso, o RE é, portanto, uma característica extraída pelo AE e que pode ser utilizada como base para decidir se aqueles dados na entrada correspondem a um ataque ou não.

A principal questão, nesta categoria de utilização, é a escolha eficiente do limiar de separação, como visto anteriormente. O presente trabalho propõe a utilização de alguns

algoritmos de aprendizado de máquina para tomar esta decisão, sendo realizada abaixo uma breve descrição desses algoritmos utilizados.

1) Limiar de Referência: Como base de comparação, utilizou-se o método de cálculo do limiar proposto em [12], onde foi utilizada a equação da Pontuação Z (ZS, do inglês *Z-Score*) para definição do limiar θ, sendo dada por

$$\theta = \mu_{RE} + Z_{AC} * \sigma_{RE} \tag{1}$$

sendo μ_{RE} a média e σ_{RE} o desvio padrão dos REs obtidos, e o Z_{AC} o valor do *ZS* relacionado com a porcentagem de AC no treinamento. Como aqui não está sendo utilizada AC, o limiar resulta apenas na média do RE. Com isso, para a classificação dos dados entre normal e anômalo, a seguinte equação foi utilizada:

$$\text{Referência} = \begin{cases} \text{normal} & \text{se} \quad RE \leq \mu_{RE} \\ \text{anomalo} & \text{se} \quad RE > \mu_{RE} \end{cases} \tag{2}$$

2) K-Nearest Neighbors: O algoritmo KNN [15] é um dos clássicos algoritmos de aprendizado supervisionados. A classificação do KNN é realizada através da similaridade entre dados, ou melhor, a partir da distância entre uma instância de dado e seus vizinhos. Existem diversas formas de calcular esta distância, sendo a mais comum a Distância Euclidiana (ED, do inglês *Euclidean Distance*).

3) KMeans: O algoritmo *K-Means* [16] é um algoritmo que faz parte da classe de técnicas não-supervisionada, ou seja, que não necessitam de apresentação das classes dos dados em seu treinamento. Este algoritmo é baseado na formação de grupos (do inglês *clusters*) dos dados em torno de um centroide.

4) SVM: SVM [17] é um algoritmo supervisionado baseado na teoria de aprendizado estatístico, cujo objetivo é encontrar um hiperplano ideal que separe as classes de dados diferentes. Para isso, são utilizados os chamados Vetores de Suportes, sendo eles os dados localizados nas extremidades das classes. Com isso, é tomado o hiperplano na média destes vetores.

III. Técnica Proposta

Para o objetivo deste trabalho, foram extraídos dois conjuntos de características do AE: o RE e as valores da Saída da IL (SIL). Com isso, três modelos de ADS foram avaliados: um utilizando como característica somente o RE, outro utilizando somente a SIL e, por fim, um modelo onde a SIL e o RE foram utilizados em conjunto como característica do ADS. A SIL foi utilizada, diferente dos trabalhos mostrados anteriormente, para avaliação da codificação do AE como característica para detecção em um ADS.

Na Fig. 2, é possível visualizar o fluxo da técnica aqui proposta, sendo detalhada cada etapa nas subseções seguintes.

A. Conjunto de Dados

Para avaliação da técnica proposta, foi utilizado o conjunto de dados NSL-KDD (NSL-KDD, do inglês *National Security Laboratory - Knowledge Discovery and Data Mining Tools*) [18], que é uma derivação do conjunto desenvolvido em 1999 para a competição de Descoberta de conhecimento e Ferramentas de Mineração de Dados (KDD, do inglês *Knowledge Discovery and Data Mining Tools*) [19], corrigindo erros e informações redundantes.

B. Pré-processamento

O conjunto de dados NSL-KDD, já mencionado, é dividido em duas bases diferentes: uma para treinamento e outra para teste. Porém, como o método de treinamento utilizado aqui difere desta divisão pré realizada, inicialmente esses dois conjuntos de dados foram concatenados e, em seguida, foi realizada uma divisão mais adequada para a técnica aqui proposta, que se baseiam em três bases diferentes: uma para treinamento do AE, outra para treinamento dos algoritmos de definição de limiar e outra para teste. Pela utilização do método *k-fold* de validação cruzada na avaliação, novamente o conjunto é dividido em dois: um para treinamento (redividido para treino do AE e do algoritmo de limiar) e outro para teste. Baseado na quantidade de dados normais presentes no conjunto de treinamento de cada *fold*, foi realizada uma divisão aleatória de 50% destes dados para treinamento do AE, e os outros 50% para treinamento dos algoritmos de detecção do limiar, sendo este último balanceado com dados anômalos, tendo em vista a necessidade de mais de uma classe no treino desses algoritmos, diferentemente do AE.

Antes da divisão dos dados, a variável *service* foi retirada dos conjuntos de dados por existirem categorias na base de treinamento que não existiam na base de teste. Além disso, é realizado o pré-processamento *One-Hot Encoding*, onde as variáveis nominais são transformadas em variáveis numéricas.

Por fim, a normalização *MinMax* foi realizada em cada uma das três bases obtidas, tendo sido utilizado o mesmo fator de normalização para todas as bases, sendo este fator obtido a partir da base de treinamento dos algoritmos de definição do limiar.

C. Treinamento e Teste

O treinamento do modelo proposto neste trabalho é realizado em etapas. Inicialmente, o AE foi treinado utilizando o conjunto apropriado obtido no pré-processamento. Posteriormente, foi aplicado na entrada do AE o conjunto de dados de treinamento dos algoritmos de definição do limiar, sendo extraído o RE e/ou a SIL, utilizados como características. Estas características foram utilizadas para treinamento dos algoritmos (KNN, *K-Means* e SVM) finalizando o treinamento do modelo.

Com o modelo treinado, é seguido o mesmo fluxo dos dados no treinamento, porém utilizando o conjunto de teste. Com isso, os dados são inseridos na entrada do AE, seguido da extração do RE e/ou da SIL, sendo estes dados inseridos nos algoritmos para classificação entre dado Normal ou Anômalo.

D. Estrutura dos Algoritmos

O AE utilizado possuía 5 camadas, sendo 2 de codificação, a IL, e 2 de decodificação, sendo esta escolha de estrutura realizada de forma aleatória tendo em vista que esta definição é algo subjetivo no projeto de redes neurais. Porém, foi seguida a regra implícita de definição de neurônios em cada camada do AE, onde as camadas externas devem sempre ter uma quantidade maior de neurônios do que as camadas internas, representando a compressão dos dados. Com isso, considerando *N* como a quantidade de características do conjunto de dados

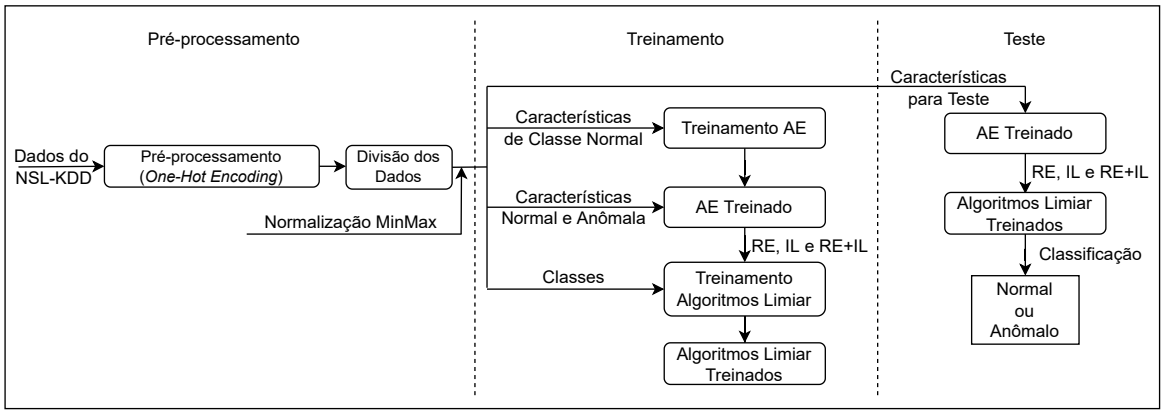

Fig. 2. Fluxo das etapas da técnica proposta.

utilizado, sendo *N* maior que o valor das camadas internas, a quantidade de neurônios de cada camada foi igual a *N*, 32, 16, 32 e *N*, seguindo da camada inicial até a final. Além disso, foi utilizada como função de ativação a função *relu* em todas as camadas.

Nas estruturas dos algoritmos utilizados para definição do limiar foram utilizados 11 vizinhos no KNN, 2 *clusters* no *K-Means* e como *kernel* na SVM utilizou-se a função *rbf*.

IV. Resultados e Avaliação

Para desenvolvimento dos modelos foi utilizada a linguagem de programação *Python* desenvolvido no editor *Jupyter*, conjuntamente com as bibliotecas de aprendizado de máquina *Keras*, *TensorFlow* e *Scikit-Learn*.

Após a implementação de todos os algoritmos, foram obtidos os resultados com a utilização das seguintes métricas: acurácia, precisão, *recall* e *f1-score*, sendo eles apresentados nas subseções seguintes. Os resultados mostrados correspondem às médias e desvios padrão obtidos através da validação cruzada com *k-fold*, com $k = 10$. Além disso, foram utilizadas 50 épocas de treinamento, tamanho de *batch* de 256 e função de otimização *Adam* com taxa de aprendizagem de 0,0001.

A. Erro de Reconstrução

Utilizando somente o RE como característica extraída pelo AE e usada como entrada no algoritmo de classificação, os resultados obtidos são apresentados na Tabela I. Pode-se observar que o KNN apresenta o melhor desempenho, apresentando apenas um valor de precisão inferior ao da Referência. Comparando as matrizes de confusão média, apresentadas na Tabela II, pode-se ver que a Referência alcançou a média de 12550,2 acertos $(7368,3 + 5181,9)$ e 2302,3 erros $(337 + 1965,3)$, e o KNN 13478,7 acertos $(7159,6 + 6319,1)$ e 1374,8 erros $(547,7 + 827,1)$, tendo o KNN apresentado maior taxa de falsos positivos (quando a classe desejada é normal, mas é classificada como anômala) do que a referência, sendo esta taxa relacionada com a precisão. Este comportamento ocorre devido à pequena diferença entre as características dos usuários normais e anômalos, onde o RE de alguns dados podem destoar do RE padrão de sua classe, gerando confusão nos algoritmos de aprendizado de máquina, provocando o aumento de falsos negativos e positivos. Pode-se notar também que o KNN foi o mais estável ao comparar seu desvio padrão com os outros algoritmos.

O segundo melhor desempenho foi obtido pela SVM. No entanto, é possível observar o mesmo comportamento em relação à precisão. Em seguida, temos, respectivamente, a técnica da Referência e o *K-Means*.

TABELA I
MÉDIA DOS RESULTADOS DE CADA FOLD PARA O USO SOMENTE DO RE

	Referência	KNN	K-Means	SVM
Acurácia	0.845 ± 0.109	**0.907 ± 0.011**	0.311 ± 0.280	0.896 ± 0.014
Precisão	**0.956 ± 0.019**	0.929 ± 0.014	0.221 ± 0.373	0.926 ± 0.019
Recall	0.809 ± 0.103	**0.896 ± 0.011**	0.206 ± 0.300	0.880 ± 0.020
F1-Score	0.872 ± 0.065	**0.912 ± 0.010**	0.208 ± 0.333	0.902 ± 0.013

TABELA II
MATRIZES DE CONFUSÃO MÉDIAS

Referência			
Classes		Classificada	
		Normal	**Anômalo**
Desejada	**Normal**	7368,3	337
	Anômalo	1965,3	5181,9

KNN			
Classes		Classificada	
		Normal	**Anômalo**
Desejada	**Normal**	7159,6	547,7
	Anômalo	827,1	6319,1

B. Erro de Reconstrução e Camada Intermediária

Dois outros cenários foram avaliados: utilizando como características na entrada do classificador (i) o RE em conjunto com a SIL do AE; (ii) somente a SIL do AE. Os resultados obtidos são apresentados na Tabela III.

Nos dois cenários, o melhor resultado foi obtido com o KNN, seguido pela SVM e, por último, o *K-Means*. Pode-se

observar que ao se utilizar o RE em conjunto com a SIL não resultou em um melhor desempenho para o KNN e a SVM quando comparado com o caso que se utiliza somente a SIL.

Comparando os resultados obtidos para o *K-Means* nos dois cenários, utilizando somente a SIL foram obtidos valores mais altos do que com a utilização do RE em conjunto com a SIL. Porém, nos dois casos observa-se um desvio padrão alto, indicando uma grande variabilidade no desempenho do algoritmo em relação a sua inicialização.

Comparando os resultados obtidos nas seções IV-A e IV-B, nota-se que a utilização somente da SIL mostrou-se uma melhor opção, tendo em vista os resultados obtidos. Por exemplo, no KNN, todas as métricas ficaram em torno de 98%. Essa melhora no desempenho, no entanto, ocorre em detrimento de um aumento da complexidade do classificador, visto que o uso dos dados da SIL possuem uma maior dimensionalidade do que somente o RE e, portanto, exigem um maior esforço para realizar o treinamento e a classificação dos dados.

TABELA III
MÉDIAS DOS RESULTADOS PARA USO DO RE EM CONJUNTO COM A SIL E SOMENTE A SIL

RE e SIL			
	KNN	K-Means	SVM
Acurácia	**0.981 ± 0.001**	0.487 ± 0.325	0.960 ± 0.003
Precisão	**0.981 ± 0.001**	0.495 ± 0.441	0.975 ± 0.003
Recall	**0.981 ± 0.001**	0.415 ± 0.346	0.949 ± 0.004
F1-Score	**0.981 ± 0.001**	0.449 ± 0.388	0.962 ± 0.003

SIL			
	KNN	K-Means	SVM
Acurácia	**0.980 ± 0.001**	0.615 ± 0.294	0.957 ± 0.002
Precisão	**0.981 ± 0.002**	0.694 ± 0.434	0.975 ± 0.004
Recall	**0.981 ± 0.001**	0.531 ± 0.323	0.943 ± 0.005
F1-Score	**0.981 ± 0.001**	0.600 ± 0.369	0.959 ± 0.002

V. CONCLUSÕES

Este trabalho propõe a utilização de algoritmos de aprendizado de máquina para a detecção de ataques em redes. Estes algoritmos recebem dados extraídos por um AE, treinado somente utilizando dados normais, isto é, sem ataque, dando origem a um ADS baseado em AE. Para isso, alguns algoritmos de classificação foram analisados: o KNN, *K-Means* e a SVM. Os resultados também foram comparados com o método de obtenção de um limiar desenvolvido em [12].

Os resultados mostram que o algoritmo KNN teve um melhor desempenho comparado com a Referência em todos os cenários avaliados neste trabalho. O segundo melhor resultado foi obtido pela SVM, o terceiro foi a Referência e o pior foi ao utilizar o *K-Means*. Uma possível explicação para os resultados obtidos com o *K-Means*, pode ter relação com a convergência do algoritmo, que não foi atingida com os dados utilizados para seu treinamento, sendo esta uma hipótese que pode ser avaliada em trabalhos futuros.

Outro resultado importante foi que a utilização do RE em conjunto com a SIL, comparada somente com o uso da SIL, não proporcionou uma melhora dos resultados.

Também foi possível observar que utilizar somente a SIL traz um desempenho melhor do que utilizar somente o RE.

Com isso, analisando todos os resultados obtidos, nota-se que a utilização de um AE com o KNN para classificação da presença de um ataque na construção de um ADS, mostra-se muito eficiente comparado à utilização de somente um limiar sob o valor do RE obtido através de um AE.

Alguns trabalhos futuros poderão ser realizados com o foco na própria estrutura do AE para melhora dos resultados, alterando a quantidade de camadas, neurônios e parâmetros do AE.

REFERÊNCIAS

[1] N. Moustafa, J. Hu, and J. Slay, "A holistic review of Network Anomaly Detection Systems: A comprehensive survey," *Journal of Network and Computer Applications*, vol. 128, pp. 33–55, 2019.

[2] M. Ahmed, A. Naser Mahmood, and J. Hu, "A survey of network anomaly detection techniques," *Journal of Network and Computer Applications*, vol. 60, pp. 19–31, 2016.

[3] G. Fernandes, J. J. Rodrigues, L. F. Carvalho, J. F. Al-Muhtadi, and M. L. Proença, "A comprehensive survey on network anomaly detection," *Telecommunication Systems*, vol. 70, pp. 447–489, 2019.

[4] Kurniabudi, B. Purnama, Sharipuddin, Darmawijoyo, D. Stiawan, Samsuryadi, A. Heryanto, and R. Budiarto, "Network anomaly detection research: A survey," *Indonesian Journal of Electrical Engineering and Informatics*, vol. 7, pp. 36–49, 2019.

[5] A. Manna and M. Alkasassbeh, "Detecting network anomalies using machine learning and SNMP-MIB dataset with IP group," *2019 2nd International Conference on New Trends in Computing Sciences, ICTCS 2019 - Proceedings*, pp. 2–6, 2019.

[6] G. Al-Naymat, M. Al-Kasassbeh, and E. Al-Hawari, "Exploiting snmp-mib data to detect network anomalies using machine learning techniques," in *Proceedings of SAI Intelligent Systems Conference*. Springer, 2018, pp. 991–1004.

[7] P. Toupas, D. Chamou, K. M. Giannoutakis, A. Drosou, and D. Tzovaras, "An intrusion detection system for multi-class classification based on deep neural networks," *Proceedings - 18th IEEE International Conference on Machine Learning and Applications, ICMLA 2019*, pp. 1253–1258, 2019.

[8] D. Charte, F. Charte, S. García, M. J. del Jesus, and F. Herrera, "A practical tutorial on autoencoders for nonlinear feature fusion: Taxonomy, models, software and guidelines," *Information Fusion*, vol. 44, pp. 78–96, 2018.

[9] M. Sakurada and T. Yairi, "Anomaly detection using autoencoders with nonlinear dimensionality reduction," *ACM International Conference Proceeding Series*, vol. 02-Decembe, pp. 4–11, 2014.

[10] P. Madani and N. Vlajic, "Robustness of deep autoencoder in intrusion detection under adversarial contamination," *ACM International Conference Proceeding Series*, 2018.

[11] S. Zavrak and M. Iskefiyeli, "Anomaly-Based Intrusion Detection From Network Flow Features Using Variational Autoencoder," *IEEE Access*, vol. 8, pp. 108 346–108 358, 2020.

[12] H. Choi, M. Kim, G. Lee, and W. Kim, "Unsupervised learning approach for network intrusion detection system using autoencoders," *Journal of Supercomputing*, vol. 75, pp. 5597–5621, 2019.

[13] B. Hwang and S. Cho, "Characteristics of autoassociative MLP as a novelty detector," *Proceedings of the International Joint Conference on Neural Networks*, vol. 5, pp. 3086–3091, 1999.

[14] Y. Yu, J. Long, and Z. Cai, "Network Intrusion Detection through Stacking Dilated Convolutional Autoencoders," *Security and Communication Networks*, vol. 2017, pp. 1–10, 2017.

[15] O. Kramer, *Dimensionality Reduction with Unsupervised Nearest Neighbors*. Springer, 2013, vol. 51.

[16] J. Blömer, C. Lammersen, M. Schmidt, and C. Sohler, "Theoretical analysis of the k-means algorithm–a survey," in *Algorithm Engineering*. Springer, 2016, pp. 81–116.

[17] J. Yu, H. Lee, M.-S. Kim, and D. Park, "Traffic flooding attack detection with SNMP MIB using SVM," *Computer Communications*, vol. 31, no. 17, pp. 4212–4219, 2008.

[18] "NSL-KDD Dataset," https://www.unb.ca/cic/datasets/nsl.html, 2009 (accessed May 30, 2022).

[19] "The Third International Knowledge Discovery and Data Mining Tools Competition," http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html, 1999 (accessed May 30, 2022).