Deep learning approach for Intelligent Intrusion Detection System

Niranjan Surya Prasad R P

July 2022

1 Introduction

2 Related works

Wei et al. [23] designed an optimization technique to optimize the basic neural network structure in the DBN. This optimization method underwent different optimization methods starting from the partial swarm optimization (PSO) and fish swarm clustering method where the generated optimal solution is further optimized using the genetic operators with self-adjusting crossover probability and mutation probability. The final optimal solution constructed by the aforementioned joint optimization algorithm was used as the network structure of IDS. Trials conducted on NSL-KDD suggested that this method is superior to other optimization algorithms with a reduction in detection time by around 26% and increased accuracy from 1.30% to 14.80%.

Vinayakumar et al. [22] proposed to develop a highly scalable and hybrid DNN framework called scale-hybrid-IDS-AlertNet which can be applied in real-time to efficiently surveil network activity and host-level events to preventatively warn against potential intrusions in order to combat the sharp rise in malicious attacks and the dynamic nature of malware. This paper also provided a detailed performance comparison between the DNNs and other machine learning classifiers tested on various publicly available benchmark IDS datasets. The results showed that the DNNs outperformed all of the classic machine learning classifiers in both HIDS and NIDS. Kasongo et al. [10] presented a deep learning based intrusion detection system that was implemented using a Feed forward deep neural networks (FFDNN) model with filter-oriented feature engineering. The FFDNN models employed in this study were coupled to an FEU using Information Gain (IG) in an effort to decrease the input dimension while boosting the classifier's accuracy. The scrutinization of the proposed IDS and other existing ML models was done on NSL-KDD datasets. The results suggested that the complexity of the intrusion detection classification problem increases with the number of attack classes we have, necessitating a greater number of neurons. The performance for multiclass classification using the FEU yielded an overall accuracy of 86.19% with a depth of three hidden layers and 150 neutrons. Kasongo et al. [11] introduced a new feature extraction framework called Wrapper-based feature extraction unit (WFEU) on Feedforward deep neural network (FFDNN). This model used the Extra Tree algorithm to generate a reduced optimal feature vector. This work was trained and evaluated on UNSW-NB15 and AWID datasets. The results indicated that, In the case of UNSW-NB15, this method produced overall accuracies for the binary and multiclass classification schemes of 87.10 and 77.16 percent, respectively. In the case of AWID, It achieved overall accuracies of 99.66% and 99.77% for the binary and multiclass classification schemes, respectively. The authors further stated that WFEU is better than filter-oriented selection. Major drawbacks of this model are computationally expensive and time consuming.

Imrana et al. [7] proposed a bidirectional Long-Short-Term-Memory (BiDLSTM) based intrusion detection system. Their model was trained and tested on the NSL-KDD dataset, a benchmark dataset for most IDSs. For Binary classification, the suggested model achieved a training accuracy of 99.95%, a testing accuracy of 94.26%, and 87.46% on the KDDTest^{+} and the KDDTest^{-21} datasets, respectively. For Multi-class classification, They carried out the studies using WEKA, BiDLSTM achieved an accuracy of 91.36% and 82.05% with a lower false alarm rate of 0.88% and 4.20% on KDDTest^{+} and KDDTest^{-21} respectively. However, the BiDLSTM model also has drawbacks, including a higher complexity and longer training time than the tested models to provide an overall effective anomaly detection.

Riyaz et al. [17] presented a new feature selection algorithm called Conditional random field and linear correlation coefficient-based feature selection (CRF-LCFS) and developed an IDS using the Convolutional neural network (CNN). To cut down on computation, their feature selection technique chose the features that had contributed the most. Further, enhanced the classification accuracy of CNN. Tenfold cross-validation was used to validate trials on KDD datasets using the TensorFlow library. Final results showed that the proposed model outperformed the existing CNN in terms of detection accuracy (98.8%), training time (0.57 s), and testing time (0.26 s), with a false alarm rate of less than 1%. Mendonça et al. [15] designed a novel IDS based on Tree-CNN and SRS activation function. Soft-Root-Sign (SRS) has shown a low time complexity over the other activation function due to its faster convergence rate and learning speed making it crucial in an environment where early detection is needed to avert damage. Trials were conducted on three datasets: CICIDS2017 and 2 novel datasets were built on real network data obtained from a medium-sized company and a common LAN network topology of a university campus. Tests on DDoS, Infiltration, Brute Force, and Web attacks showed around a 36% reduction in execution time and 98% detection accuracy. Outperformed DBN.

Sadaf et al. [19] proposed a novel approach to IDS using Autoencoder and Isolation Forest in Fog computing. This approach simply focuses on binary classification, making it more appropriate for real-time applications where it is more important to distinguish attacks from regular packets. This was a two-stage technique, where the output of stage 1 (AE) acts as input for stage 2 (IF). AE was trained only on normal data, therefore a threshold value is assigned if the reconstruction loss is higher than the threshold it is labeled as "attack" otherwise as "normal". Trials on NSL-KDD revealed a high accuracy rate of 95.4%. Al-Qatf et al. [3] presented a novel unsupervised NIDS based on the self-taught learning (STL) framework. This framework was mainly used for feature learning and dimensionality reduction. This model was built by combining the sparse autoencoder with SVM. The SAE was an effective unsupervised learning approach for reconstructing the new feature representation. Further, the new characteristics are fed into SVM to improve the classification precision and detection ability. This model was tested and trained on NSL-KDD datasets. The outcomes showed that this strategy improved the SVM's performance. Additionally, it has outperformed all shallow learning techniques. Time spent on testing and training was drastically cut. Li et al. [13] proposed to develop an innovative self-learning 3-layer Auto-Encoder-based IDS that used random forest feature selection to select effective from the original dataset. The author claimed that the sample imbalance issue, which was highly prevalent in the network environment, can be effectively resolved using this technique. The random forest feature selection helped to reduce the computational complexity making it lightweight and this also reduced the detection speed of the model. The experimental findings demonstrated that in terms of simple training, robust adaptation, and high detection accuracy, the proposed method outperformed conventional machine learning-based intrusion detection methods.

3 Methodology

3.1 Background

3.1.1 Definition

An intrusion detection system (IDS) is a security framework that keeps track of computer systems and network traffic and examines that data for potential hostile assaults coming from the outside as well as system abuse or attacks coming from within the organisation [20].

3.1.2 Working Mechanism

A monitor-only application called an intrusion detection system is made to spot and report irregularities before hackers may harm your network infrastructure. Either your network or a client system (host-based IDS) has IDS installed on them. The majority of intrusion detection systems scan for well-known attack signatures or unusual departures from predetermined standards. The protocol and application levels of the OSI (Open Systems Interconnection) model are then contacted to further investigate these unusual patterns in the network traffic.

Your network infrastructure incorporates an IDS to serve as a detection system outside of the real-time communication band (a path connecting the information transmitter and receiver). Instead, it employs a SPAN or TAP port for network monitoring and analyses a copy of inline network packets to make sure the streaming traffic is not counterfeit or otherwise fabricated (acquired by port mirroring). The IDS effectively detects compromised components including corrupted data packets, DNS poisonings, Xmas scans, and more that have the ability to affect the performance of your entire network.

3.1.3 Classification Of IDS

1. **Intrusion Detection Based on Point of Detection**

   The IDS's first fundamental classification is based on how it is configured within the cybersecurity architecture and where it discovers intrusions.

   (a) **Network-Based IDS (NIDS):** A whole protected network can be monitored by a network-based IDS system. NIDSs are frequently installed in data chokepoints, typically in a demilitarised zone (DMZ) or network border, to record network traffic and examine individual packets for harmful content. Without degrading performance, a strategically positioned NIDS protocol may effectively monitor all network traffic. Because it does not increase traffic volume, it also has no impact on network throughput and availability.

   (b) **Host-Based IDS (HIDS):** Host IDS, on the other hand, is set up on every client computer connected to your network. Such an IDS might be able to watch network traffic to and from the machine, keep tabs on processes, and look through the system logs. As it is placed on networked computers, HIDS is able to identify malicious network packets transferred within the company, including any infected host attempting to access other machines. NIDS typically fails to achieve it.

2. **Intrusion Detection Based on Method of Detection**

   The IDS's second fundamental classification is determined by how it detects intrusions and by the precise monitoring and analytical techniques it employs.

   (a) **Signature-Based IDS (SIDS):** The identification of known threats is done by signature-based IDS solutions. An IDS solution generates a signature and adds it to its list of signatures to evaluate incoming traffic after identifying malware or other hazardous content. Due to the fact that all alerts are produced in response to such discovery of known harmful content, an IDS can achieve a high threat detection rate with no false positives. An IDS that relies on signatures, however, cannot detect zero-day vulnerabilities and is only capable of spotting known threats.

(b) **Anomaly-Based IDS (AIDS):** The "normal" behavior of the protected system is modeled by anomaly-based IDS. Any irregularities in future behavior are classified as potential risks and cause alerts by being compared to this model. While this method can uncover unique or zero-day threats that the Signature-based detection method couldn't easily spot, these systems must balance false positives with false negatives due to the challenge of developing an accurate model of "normal" behavior. This method has a significant disadvantage in that it may exaggerate the threat posed by a specific irregularity and wrongly classify an activity as an intrusion, squandering money on unnecessary mitigation efforts [21].

3.1.4 Workflow Diagram

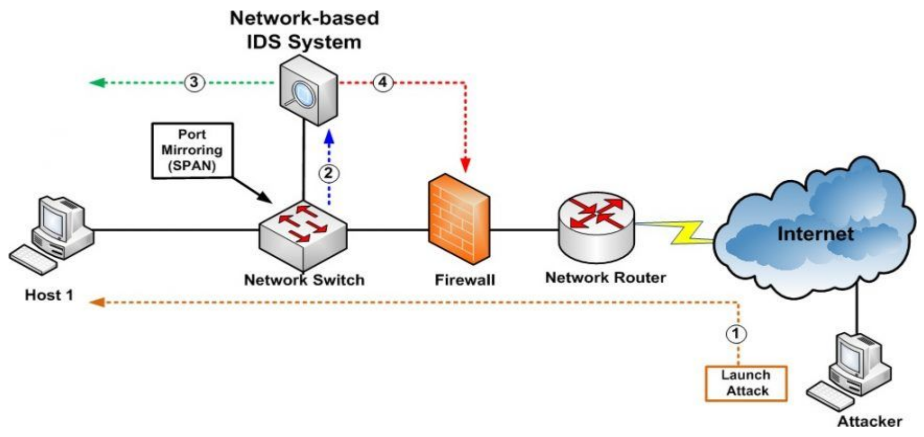

Figure 1: Intrusion Detection Systems [8].

3.1.5 Software or tools available to implement IDS

1. **SolarWinds Security Event Manager**: This tool integrates HIDS and NIDS systems. It gathers network-based real-time log data. Like this, SEM is widely used to create customizable intrusion detection techniques that may instantly disable accounts and disconnect connected devices when it detects any intrusion.

2. **Suricata**: A free open-source IDS is Suricata. It can perform network security monitoring, inline intrusion prevention, and real-time intrusion detection. It is a powerful security tool that combines IDS and IPS.

3. **Snort**: The most widely used free NIDS software for Windows is called Snort. In addition to being an intrusion detector, Snort also functions as a packet logger and a packet sniffer.

4. **OSSEC**: The OSSEC programme is effective at compiling lists of crucial files and periodically checking them. This enables the tool to notify the network administrator right away if anything suspect is discovered.

5. **Sagan:** It is a HIDS with certain NIDS features, the Sagan Log analysis tool may incorporate reports produced on snort data.

3.1.6 Role of Deep learning in IDS

Detecting intrusions and novel assaults, supporting the ability to self-learn, and spotting zero-day attacks make deep learning more potent than machine learning at handling massive data. To find higher-level features, it applies numerous layers of modifications. On massive datasets, deep learning is being used to tackle classification, image recognition, self-driving cars, and speech recognition issues. It uses numerous unidentified layers to automatically choose the features or mine attributes before performing training and testing on the supplied data to obtain the classification results. In contrast to conventional machine learning, which requires features to first be retrieved before training and testing, deep learning involves feature selection and training through the same process. Deep learning is available in a variety of forms [19].

3.2 Problem Statement

According to [2], over the previous three years, researchers have moved their concentration to DL tools for developing the IDS system. Of the ways that have been suggested, 60% are solely DL-based approaches, 20% employ a hybrid approach that combines ML and DL-based algorithms, and just 20% are ML-based methods.

The learning process will be more demanding in terms of CPU resources and computational time due to the huge dataset and deep nature of the DL methods. The NIDS model will be increasingly effective at detecting intrusions the more it is taught. This is one of the primary concerns with DL models. [17], [15] and [3] discuss a potential remedy for this flaw. Another significant problem is the reduced detection rate for the R2L and U2R attack classes. However, only a few researchers [7], [15] have been able to offer a solution that will lead to an improvement. Inferring from the aforementioned typical problems, we can say that the training and testing times will be reduced by carefully choosing useful features for the model. We can cut back on computation resources in this way.

3.3 Proposed Solution

Here, we go into further detail about our proposed network intrusion detection system (NIDS), which is built on the Convolution neural network (CNN) and the Correlation feature selection (CFS). The suggested NIDS is shown in Fig. 2 as having three main parts. The cybersecurity database CICIDS2017, which is utilised to train the suggested model, is first introduced and examined. Before the data is supplied into intrusion detection, the data is preprocessed to guarantee its quality. The second step involves applying the correlation-based feature selection method to extract the dataset's best feature subset. Finally, the architecture and mathematical model of the CNN, which receives the processed data for classification, are shown and explained.

3.3.1 Architecture

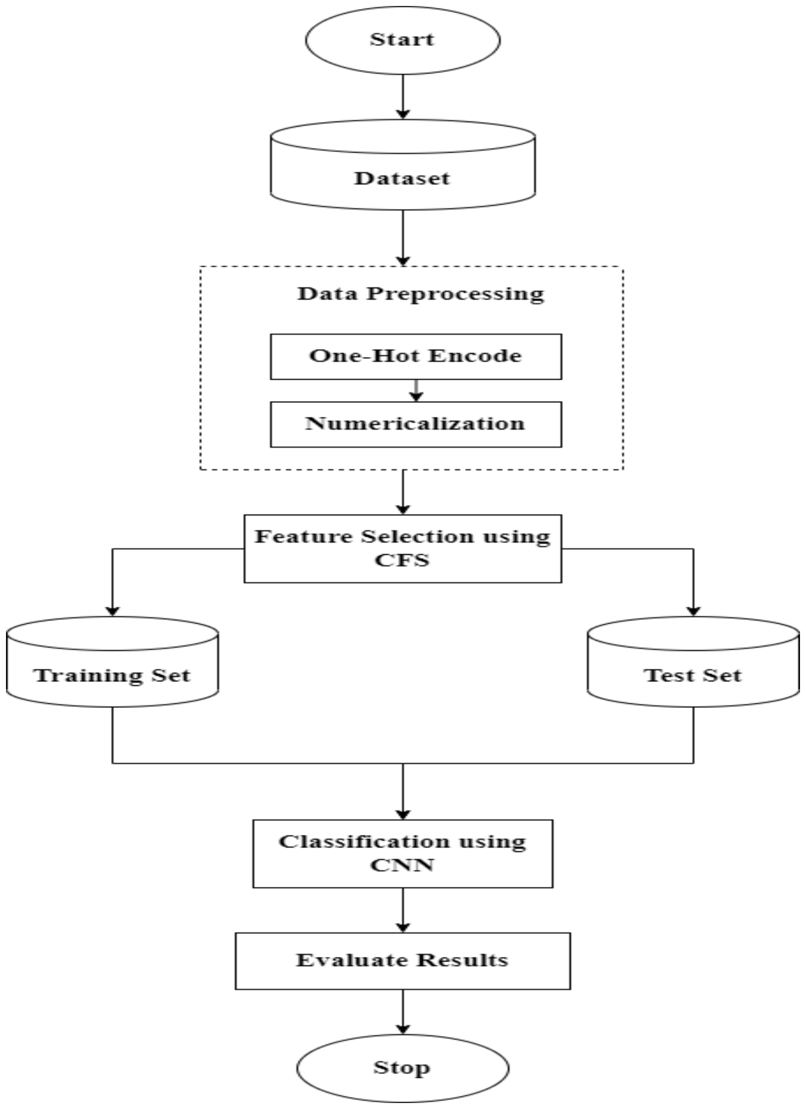

Figure 2: Proposed model architecture

1. **CSE-CIC-IDS2018 Dataset**

   Our model was evaluated using the CSE-CIC-IDS2018 dataset [14]. CIC-2018 is made up of 10 days worth of sub-datasets that were amassed through 16 different sorts of assaults. Brute-force, Heartbleed, Botnet, DoS, DDoS, Web attacks, and internal network infiltration are just a few of the seven attack scenarios that are included in the final dataset. The dataset contains 80 features that CICFlowMeter-V3 retrieved from the network traffic that was gathered, as well as the system logs and network traffic that were recorded on each machine [9].

2. **Data Preprocessing**

   Significant challenges for data modelling and knowledge discovery are posed by redundant, unpredictable, and diverse types of data. To prepare the data, we convert the dataset's categorical data to numerical data. The data preprocessing module cleans up the dataset, selects the best feature, combines comparable features, and provides sufficient information for anomaly detection.

   (a) **One-hot encode**

   Categorical variables can be transformed into a form that ML algorithms can use to make predictions more accurately through a process known as one hot encoding. Because many machine learning models require numeric input variables. The "Protocol" and "Label" characteristics will consequently be one-hot encoded.

   (b) **Numericalization**

   The CSE-CIC-IDS 2018 dataset is processed using Pandas. The percentage of samples in the entire data set that possess a certain feature must be counted if it has default values (NaN) or infinite values (Infinity). The occupancy ratio controls how NaN and Infinity are replaced. All feature values that contain Nan or Infinity are changed to 0's.

3. **Feature Selection**

   By picking a subset of features that contribute to the sample to represent, feature selection is a technique for reducing the dimensionality of the data. It also helps to cut down on model training time and processing costs. Our model uses correlation-based feature selection, which is the method we prefer to use. Being a filter strategy, the method is unrelated to the chosen classification model. As the name already implies, it exclusively considers correlations while evaluating feature subsets.

3.3.2 Implementation

Correlation-based Feature selection

General Principle: The objective is to identify a feature subset with low feature-feature correlation to prevent redundancy and high feature-class correlation to preserve or improve predictive power.

In order to do so, the algorithm uses the equation below to estimate the worth of a subset s with k features [5]:

$$Merit_s = \frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}}$$

- $\overline{r_{ff}}$**:** average feature-feature correlation.
- $\overline{r_{cf}}$**:** average feature-class correlation.
- k: number of features of that subset

The subset that produces the most merit is what we look for. Both a low feature-feature correlation in the denominator and a high feature-class correlation in the numerator can result in high merit.

Using merit as a heuristic, [6] suggests a **best-first-search** methodology. The search begins with an empty subset, and it assesses each feature's merit before adding it to the empty set. Since k=1, the denominator of the equation above can be simplified to 1 for this step, allowing the feature-feature correlation to be disregarded.

$$\frac{k\overline{r_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} = \frac{1\overline{r_{cf}}}{\sqrt{1 + 1(1-1)\overline{r_{ff}}}} = \frac{\overline{r_{cf}}}{\sqrt{1}} = \overline{r_{cf}}$$

Feature-class correlation is purely focused on the initial correlation for the initial iteration. The as-yet-empty subset receives the addition of the feature with the highest feature class correlation. Feature that forms the best subset with the one that was already included is preserved after all features are once again evaluated.

The algorithm loops back to the next best unexpanded feature subset when a feature expansion does not result in an improvement. This algorithm explores the entire feature subset space without any restrictions. The amount of backtracking must be controlled so that only the feature subset that produced the highest merit is explored [5].

CNN Model Design

Our model has been derived from [18] work. The most used deep learning algorithm for image training is CNN. The CIC-2018 dataset needs to be transformed into pictures in order to create a CNN-based intrusion model. Since each data set comprises 78 characteristics, except the 'Label' feature, we convert each labelled data set into 13x6 size images. For image classification, the 'Label' is utilised. Convolutional layers, max-pooling layers, a flatten layer and a fully connected layer (dense layers) make up a CNN model, which classifies network traffic as either benign (0) or malicious traffic (1). By arranging those layers and modelling parameters, we may determine the ideal CNN model. Table 1 displays our CNN model for CIC-2018 [12].

Layer (type)	**Output Shape**	**Param #**
input_1 (InputLayer)	[(None, 6, 6, 1)]	0
conv2d (Conv2D)	(None, 6, 6, 100)	500
max pooling2d (MaxPooling2D)	(None, 3, 3, 100)	0
conv2d_1 (Conv2D)	(None, 3, 3, 50)	20050
max pooling2d_1 (MaxPooling2D 1)	(None, 1, 1, 50)	0
flatten (Flatten)	(None, 50)	0
dense (Dense)	(None, 10)	510
dense_1 (Dense)	(None, 15)	165

Table 1: Our CNN Model

We deploy two convolution layers and two maxpooling layers one after the other. Although a CNN model need not include a max pooling layer, we do so since the converted images only include numerical data and not hidden signatures, which greatly reduces the risk of missing significant features. The activation functions for each convolutional layer and the final dense layer, respectively, are "relu" and "softmax." After each max pooling phase, dropout is implemented to lessen overfitting. Behind the last max-pooling layer, a completely connected layer is then deployed.

4 Evaluation

We discussed the assessment specifics in this section, including how to build up an experimental setting, the usage of the CSE-CIC-IDS 2018 dataset for anomaly detection, and the evaluation criteria we employed.

4.1 Environmental setup

We utilized Python 3.9 running on Windows 10 for this study. For nearly every type of learning approach, Python has a robust set of libraries. We utilised Keras, TensorFlow, and Scikit-learn [16] modules for our model. Keras [4] is a component of the open-source, GPU-enabled machine learning package Tensorflow [1]. On an ASUS laptop with the following characteristics, our simulations were run: 16GB of RAM with a 3.30GHz Intel Core i7-11370H processor. It utilizes a 4 GB dedicated GDDR6 VRAM GPU, the NVIDIA GeForce RTX 3050 Ti.

Datasets

A new dataset will be created by combining some of the CSV files from the CSE-CIC-IDS 2018 dataset. In the merger, the following files won't be used: 02-20-2018.csv has a different number of columns (84 against 80 in the other files); 02-22-2018.csv and 03-22-2018.csv only has the most common and safest types of traffic are included in them. These latter two hence won't really add any more pertinent data.

The top features chosen by the proposed correlation-based feature selection (CFS) algorithm are listed in Table 2. It identifies the feature-subset which yields the highest merit for finalizing the selected features.

S.No	**Features**
1	Fwd Seg Size Min
2	Flow IAT Min
3	Fwd IAT Min
4	Fwd IAT Tot
5	Flow Duration
6	Flow IAT Mean
7	Fwd IAT Mean
8	TotLen Fwd Pkts
9	Subflow Fwd Byts
10	Fwd Act Data Pkts
11	Tot Fwd Pkts
12	Subflow Fwd Pkts
13	Fwd Header Len
14	TotLen Bwd Pkts
15	Subflow Bwd Byts
16	Tot Bwd Pkts
17	Subflow Bwd Pkts
18	Bwd Header Len
19	Fwd URG Flags
20	CWE Flag Count
21	Active Min
22	Active Mean

Table 2: List of selected features

4.2 Evaluation Metrics

We determine the accuracy, recall, precision, and F-score in order to assess the effectiveness of our model. The explanation and derivation of each of these metrics are as follows:

(i) **Accuracy:** A method's accuracy on a given test set is measured by the proportion of test instances it properly recognises.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

(ii) **Recall:** The idea of the occurrences that the model properly classified as being positive is known as recall.

$$Recall = \frac{TP}{TP + FN}$$

(iii) **Precision:** The model's precision is defined as the ratio of all positively labelled examples to the positive instances that are accurately identified by the model.

$$Precision = \frac{TP}{TP + FP}$$

(iv) **F-score:**By calculating the harmonic mean of recall and precision, the F-score—a metric for gauging accuracy—is calculated.

$$F\text{-}score = \frac{2 * Precision * Recall}{Precision + Recall}$$

5 Conclusion and Future work

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] Z. Ahmad, A. Shahid Khan, C. Wai Shiang, J. Abdullah, and F. Ahmad. Network intrusion detection system: A systematic study of machine learning and deep learning approaches. *Transactions on Emerging Telecommunications Technologies*, 32(1):e4150, 2021.

[3] M. Al-Qatf, Y. Lasheng, M. Al-Habib, and K. Al-Sabahi. Deep learning approach combining sparse autoencoder with svm for network intrusion detection. *Ieee Access*, 6:52843–52856, 2018.

[4] F. Chollet. Keras, 2015.

[5] J. S. Fischer. Correlation-based feature selection in python from scratch, Aug 2021.

[6] M. Hall. Correlation-based feature selection for discrete and numeric class machine learning. In *Proceedings of the 17th international conference on machine learning (ICML-2000)*, pages 359–366, 01 2000.

[7] Y. Imrana, Y. Xiang, L. Ali, and Z. Abdul-Rauf. A bidirectional lstm deep learning approach for intrusion detection. *Expert Systems with Applications*, 185:115524, 2021.

[8] A. Jaiswal. Intrusion detection systems, Jun 2017.

[9] V. Kanimozhi and T. P. Jacob. Artificial intelligence outflanks all other machine learning classifiers in network intrusion detection system on the realistic cyber dataset cse-cic-ids2018 using cloud computing. *ICT Express*, 7(3):366–370, 2021.

[10] S. M. Kasongo and Y. Sun. A deep learning method with filter based feature engineering for wireless intrusion detection system. *IEEE access*, 7:38597–38607, 2019.

[11] S. M. Kasongo and Y. Sun. A deep learning method with wrapper based feature extraction for wireless intrusion detection system. *Computers & Security*, 92:101752, 2020.

[12] J. Kim, Y. Shin, and E. Choi. An intrusion detection model based on a convolutional neural network. *Journal of Multimedia Information System*, 6(4):165–172, 2019.

[13] X. Li, W. Chen, Q. Zhang, and L. Wu. Building auto-encoder intrusion detection system based on random forest feature selection. *Computers & Security*, 95:101851, 2020.

[14] K. P. López. Cse-cic-ids2018, Feb 2022.

[15] R. V. Mendonça, A. A. Teodoro, R. L. Rosa, M. Saadi, D. C. Melgarejo, P. H. Nardelli, and D. Z. Rodríguez. Intrusion detection system based on fast hierarchical deep convolutional neural network. *IEEE Access*, 9:61024–61034, 2021.

[16] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[17] B. Riyaz and S. Ganapathy. A deep learning approach for effective intrusion detection in wireless networks using cnn. *Soft Computing*, 24(22):17265–17278, 2020.

[18] N. S. Ids-with-cnn/ids-cnn.ipynb at main · nivedha-s/ids-with-cnn, Sep 2021.

[19] K. Sadaf and J. Sultana. Intrusion detection based on autoencoder and isolation forest in fog computing. *IEEE Access*, 8:167059–167068, 2020.

[20] A. Sarmah. Intrusion detection systems: Definition, need and challenges, 2021.

[21] C. P. Software. What is an intrusion detection system (ids)?, Jul 2022.

[22] R. Vinayakumar, M. Alazab, K. Soman, P. Poornachandran, A. Al-Nemrat, and S. Venkatraman. Deep learning approach for intelligent intrusion detection system. *Ieee Access*, 7:41525–41550, 2019.

[23] P. Wei, Y. Li, Z. Zhang, T. Hu, Z. Li, and D. Liu. An optimization method for intrusion detection classification model based on deep belief network. *Ieee Access*, 7:87593–87605, 2019.