Deep learning and machine learning methods for DDoS attacks detection

BSc. Computer Science

School of Informatics and Engineering, University of Sussex

candidate no. 230940

Project Supervisor: Dr Imran Khan

Statement of originality

This report is submitted as part requirement for the degree of BSc Computer Science at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged. I hereby give permission for a copy of this report to be loaned out to students in future years.

ABSTRACT

In this project we have conducted a research comparing the different ML and DL models in detecting Distributed Denial of Service attacks in order to compare their results, study their potential and develop our own models. Furthermore, to assess the potential of these models we have developed a set of notebooks comparing the performance of one of the most implemented techniques, K-nearest neighbor, and one that is often disregarded for the field of cybersecurity, Convolutional Neural Networks in hopes of finding new insights about these models. In addition, we addressed the potential of the combination of different ML and DL, aiming to demonstrate the versatility and the enhancement of models when various techniques are applied.

INDEX:

1. INTRODUCTION

With the improvement of Artificial Intelligence (AI), the use and adaptation of these new techniques have been critical for the development of technology. Therefore, a growth in fatal, disruptive and vicious cyber-attacks have increased at the same ratio. Some of the most common and disruptive are the Denial of Service (DoS) attacks. Their aim is to overrun or confuse the server to prevent legitimate users from accessing the services [1]. These types of attacks have been around for a long time due to their simplicity and their reliability in simple networking processes.

As technology advanced the attacks have been more sophisticated and cybercriminals started deploying these types of attacks from different machines at the same time, increasing the overload of servers and achieving a big delay in the responses of the services and even shutting them down. These attacks are called Distributed Denial of Service attacks (DDoS). DDoS attacks use a botnet to organize the broadcasting of false requests, making the server remain busy. They present one of the more laborious attacks to detect and mitigate since it is difficult to differentiate a legitimate traffic request from a malicious ones, as they use the same port and protocols. The nature of the attack may vary depending on the task, in figure 1 we can see some of the most common techniques from deploying them.

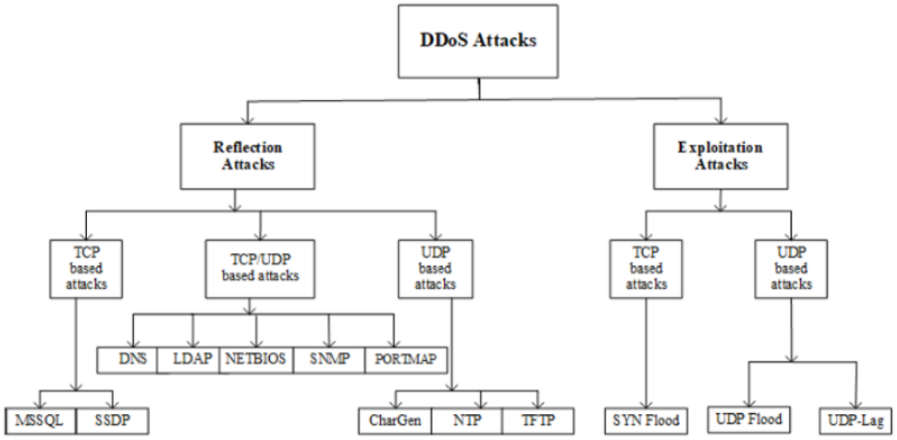

Figure 1:Flow chart of classification of DDoS attacks

The improvement of cyber-attacks correlates with the growth of defense mechanisms. Machine Learning and Deep Learning have proved to work effectively in attacks protection [2]. Its ability to detect patterns and analyze data, makes Artificial Intelligence a solid benchmark to study their application for intrusion detection systems (IDS) and assess their performance to unseen attacks.

2. OBJECTIVES

The purpose of this paper is to compare two Machine Learning (ML) and Deep Learning (DL) models used to detect, prevent, and mitigate DDoS attacks. K-nearest neighbor (KNN) and Convolutional Neural Networks (CNN) have been selected after reviewing more than thirty research papers related to the topic. We believe that CNNs are often disregarded since they are mostly used in the field computer vision, without considering the advantages of 1D-CNN for processing data in the form of arrays.

Our objectives are to compare and develop a model that is often overlooked with one widely implemented. Involving how well they differentiate malicious traffic against legitimate and their performance in metrics, such as precision, recall, F1-score and false positive/negative rates. To achieve this comparison, we have used the NSL KDD dataset and the CICDDoS2019 dataset which are widely used in intrusion detection products.

In addition, we aim to demonstrate the potential of the combination of different ML and DL techniques to address cybersecurity threats. We believe that the true potential of these methods relies on using them together with their respective tasks in the model. We are using Random Forest classifier to process our features and extract the ones that are the most influential in determining the attacks. Then we will combine this subset with our previous models to study their performance

3. PROFESSIONAL CONSIDERATION AND ETHICS

Since we are conducting research in cybersecurity, there could be aspects that are not ethical if the proper security procedures are not followed.

A DDoS attack could result in major disruption in the main online services of certain companies, as the average cost per hour of some enterprises server downtime could lead to millions of dollars in losses [Figure 2]. The use of DDoS attacks has increased over the past years at an alarming rate. The biggest attack was recorded in February of 2020 to AWS with an outstanding 2.3 Tbps of data coming into its servers.

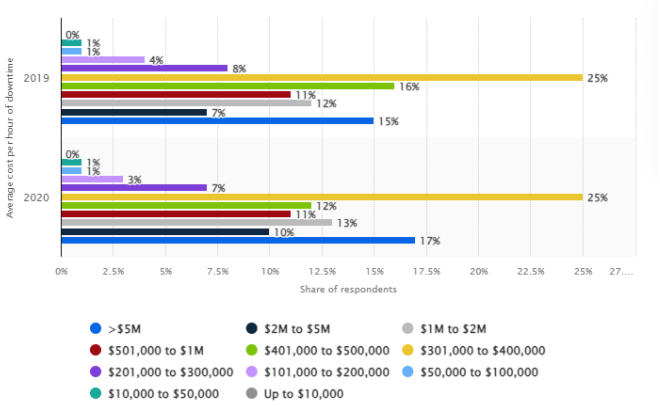

Figure 2: Graph displaying cost against server downtime[3].

Performing a denial-of-service attack with malicious intentions has legal ramifications such as a prison sentence, a fine or both [4]. They are relatively easy to perform, allowing someone with average skill-set with the help of toolkits to perform these attacks. A distributed denial-of-service attack needs a previous infection of devices to be part of the botnet, trojan insertions, as well as social engineering techniques that result in a download of a bot code turning your device into a zombie part of a botnet.

For this research we have set side by side the different methods and compared their performance on a dataset in our local machine. Therefore, no DDoS attack has been conducted, however, in the field of cybersecurity, it is essential to understand the attacks to be able to mitigate them. This poses a major moral implication in terms of how such research is conducted and the potential for misuse.

When working in cybersecurity, there is a thin line between developing necessary defenses and the risk of providing attackers with information that could enhance their malicious capabilities[5]. Researchers must ensure that their work benefits security efforts without inadvertently aiding malicious actors.

4. RESEARCH

Numerous techniques exist to prevent, detect, and mitigate DDoS attacks. There are two primary approaches: signature-based and anomaly-based Intrusion Detection Systems (IDS). While Signature-based IDS has proved to be effective to known threats, anomaly-based IDS (usually a component of Machine Learning models) has proved to be efficient even against an unknown attack.[6]

In recent years we have seen how ML and DL have impacted most technological aspects of society and it has been has been defined as obtaining a computed model of complex non-linear relationships or complex patterns within data [7]. Deep learning can be seen as a more specific application for usage in closed environments. However, there are several challenges in implementing a successful DL and ML model in real world-settings since the choice of suitable parameters for the algorithm can be difficult, nevertheless, it has yielded outstanding results when properly implemented [8].

Hackers are specialized in turning normal protocols and methods into a form of malicious exploits or attacks. Cyber-security can be seen as a race between attackers and defenders. With the emerging new technologies and methods being implemented to find new exploits or forms of attacks ML and DL can bring new ways to maintain a safe space.

DDoS attacks have a long-documented history being present since the beginning of networking. There are various ways to mitigate these attacks and depending on the scale and method they can be easily detected. With the incorporation of AI, the attacks can learn from normal traffic in a network and proceed without deviating from legitimate user behavior making conventional cyber-security tools inefficient in detecting the attack.

4.1. MALICIOUS BOTNETS

A malicious botnet is a network of computers called “Bots” that have been compromised under the control of a “Botmaster” [9]. Botnets pose a significant threat against cybersecurity because of their ability to launch coordinated attacks in large scales. The detection of botnets has been a major research topic in recent years due to the growth in technology advances.

Once a device is compromised and turned into a bot it is difficult to mitigate the infestation, they can target all IOT devices connected to a network. There are botnets with millions of different bots waiting for a command without alerting the users that are infected [10]. Botnets can be used for a variety of malicious purposes, including DDoS attacks, spamming, phishing, and cryptocurrency mining.

Due to their complexity and distributed nature, they pose several challenges in detecting them, some of the major problems are the sophisticated communication channels between the botmaster and the bots, using techniques such as domain generation algorithms and encrypted communications, cybercriminals cannot be easily detected by conventional IDS.

When facing a botnet, advanced detection algorithms are essential, using ML and DL can help in identifying unusual patterns of behavior, these technologies can adapt and evolve depending on the network traffic, making them able to classify malicious traffic previously unseen.

For instance, one of the most famous network companies, Cloudflare, offers the possibility of having your own web running on their server securely. It has a service that allows your network traffic to be analyzed to detect the presence of botnets disrupting the behavior of the server. It has been numerous cases that companies have not been able to mitigate DDoS attacks until they purchase this tool for detecting botnets. For security reasons they do not disclose much information of the architecture, but as seen in their web

page [11], the main bullet points of their model are: Behavioral analysis, detecting anomalies in your own web, Fingerprinting, relying on millions of internet properties and Machine learning. Obviating the ML point, the other two points are the pillars for a robust model. This demonstrated how one of the leading companies in networking utilizes techniques of ML and DL to achieve outstanding results in protecting the users.

4.2. **MODERN DDoS ATTACKS**

Trinoo is considered the first widely distributed DDoS attack tool, it used to attack using constant size UDP packets that targeted random ports on the victim's machine [12]. Since then, there have been numerous toolkits to perform these attacks, most of them are just modified versions of the previous one making most basic DDoS attacks to be easily mitigated with signature-based defense mechanisms. They have been around for more than twenty years, and modern IDS can detect them easily, however, now a days toolkits are advanced enough to utilize ML and DL themselves to disguise the attack in between benign traffic. We are going to analyze a common DDoS attack to demonstrate why ML and DL are crucial for the mitigation of attacks.

There are hundreds of toolkits available to download from the internet. One common toolkit is Saphyra, it utilizes a list with more than 3000 different headers and referrers [Figure 1] to make sure every time it sends a http request to flood the server it has different user information [Figure 2, line 3465], making it more difficult to detect the patterns. Finally, the Execution logic initializes 700 instances of the HTTPTcall method, resulting in an overload of the server.

```
37  def useragent_list():
38          global headers_useragents
39          headers_useragents.append('Mozilla/5.0 (Windows; U; Windows NT 5.1; en-US) AppleWebKit/532.1 (KHTML, like Gecko) Chrome/4.0.219.6 Safari/532.1')
40          headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.1; WOW64; Trident/4.0; SLCC2; .NET CLR 2.0.50727; InfoPath.2)')
41          headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 6.0; Trident/4.0; SLCC1; .NET CLR 2.0.50727; .NET CLR 1.1.4322; .NET CLR 3.5.30729; .NET CLR 3.0.30729)')
42          headers_useragents.append('Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.2; Win64; x64; Trident/4.0)')
```

```
3456  def httpcall(url):
3457            useragent_list()
3458            referer_list()
3459            code=0
3460            if url.count("?")>0:
3461                    param_joiner="&"
3462            else:
3463                    param_joiner="?"
3464            request = urllib2.Request(url + param_joiner + buildblock(random.randint(3,10)) + '=' + buildblock(random.randint(3,10)))
3465            request.add_header('User-Agent', random.choice(headers_useragents))
3466            request.add_header('Cache-Control', 'no-cache')
3467            request.add_header('Accept-Charset', 'ISO-8859-1,utf-8;q=0.7,*;q=0.7')
3468            request.add_header('Referer', random.choice(headers_referers) + buildblock(random.randint(50,100)))
3469            request.add_header('Keep-Alive', random.randint(110,160))
3470            request.add_header('Connection', 'keep-alive')
3471            request.add_header('Host',host)
```

We find this toolkit of DDoS attack to be particularly intuitive and easy to run. This poses a major problem for security, since with basic human engineering and natural language engineering knowledge anyone can take the attack further by personalizing the requests for a specific server.

The most widely used toolkit nowadays is MHDDoS, an attack script that uses python3 with 56 different methods, making it extremely versatile. It also integrates various bypass techniques against different web protection services like OVH, CloudFlare, and Google Project Shield.

The attacker usually owns or rents a large botnet to succeed with the attacks. Once a device is infected and turned into a zombie part of the botnet, it is difficult to revert it. Since a worm virus can infect all the IoT devices connected to a server. These botnets can be composed with millions of devices around the world, making it possible for the attacker to perform various attacks in parallel and flood with different models at the same time.

Major companies are targets of large attacks with thousands of bots working simultaneously to overwhelm the server daily. Usually, these attacks do not last long, and big companies can afford to protect themselves with better IDS or increasing the server's bandwidth until the attack stops.

At the beginning of the attack if the botnet is not big and the requests come mainly from a certain country or are directed to a server vulnerability in specific, even if it is a sophisticated attack the connections can be blocked temporally for specific locations or requests, in some cases this is enough to make the attacker give up. If the attack continues and has a higher scale, we will need to analyze the traffic and detect the patterns of behavior on the requests. Here is where ML and DL are extremely helpful.

4.3. ML AND DL FOR DDoS

Signature-based IDS involves a lot of hours of testing and developing before seeing an implementation. They remain only effective against previously known attacks. ML and DL methods have been shown to be more effective in most cases since they benefit from a framework which learns from the data allowing it to make accurate predictions [13], another advantage is the ability to detect patterns without the need to know the pattern itself.

In this section we have an overview of different ML and DL models studied in various research papers where we compare their accuracy in detecting DDoS attacks and properly classifying them without interfering with genuine users.

4.3.1. MACHINE LEARNING:

Machine Learning relies on the automatization of processes and the analysis of data. It can be supervised, unsupervised or hybrid. K-NN, Decision Trees (DT), Support Vector Machine (SVM) and Naïve Bayes (NB) seem to be some of the most studied and reliable methods.

AUTHOR	ML MODEL	ACCURACY	METRICS	DATASET	KEY FINDINGS
Kimmi Kumari & M.Mrunalini [1]	NB	99 to 98%	Not provided	CAIDA 2007	Hiher bias but smaller variance compared with LR.
Kimmi Kumari & M.Mrunalini [1]	Logistic Regression	99 to 100%	Not provided	CAIDA 2007	Probability prediction.
M.J.Kotpalliwar and Wajgi [14]	SVM	89.85%	Classification accuracy: 99.9%	"Mixed" and "10 percent KDDCUP'99"	Took long time to process the data

Muhammad S. Pervez and Md. Farid [15]	SVM	91% with 3 features. 99% with 36 input features.	F1-score of 0.99 in training but 0.77 in training	NSL-KDD	Only works properly with known threats.
H. Shapoorifard and P.Shamsinejad [16]	KNN	99.6%	Detection rate: 98% False alarm: 4%	NSL-KDD	k-FN and KNN was used to classify the data.
Ilham Ramadhan [17]	KNN	98.94%	Precision: 99.23% Recall: 99.12% FNR: 0.88% FPR: 1.4% TNR: 98.4% F-score: 99.18%	CICIDS2017	Simpler calculations
Ilham Ramadhan [17]	DT	99.91%	Precision: 99.93% Recall: 99.91% FNR: 0.09% FPR: 1.4% TNR: 99.88% F-score: 99.92%	CICIDS2017	Smaller running time and better accuracy compared with KNN
Chandrashekhar Azad and Vijay Kumar Jha [18]	DT	99.89%	FAR: 0.11%	KDD Cup 99	It combined the model with genetic algorithms.
Ramin Fadaei Fouladi [19]	NB	95.93%	Precision: 96.79% Recall: 96.14% FNR: 3.82% FPR: 4.43% TNR: 94.68% F-score: 96.46%	DFT+DWT	It examines analyzing the frequency attributes of traffic in stead of the payload of packages.
Rizvi, F. [20]	GA and K-NN	100%	Not provided	APA-DDoS	25% improvement after adding GA for data classification
Rizvi, F. [20]	GA and Adaboost	97.6%	Not provided	APA-DDoS	Improvememt after adding GA

Figure 3: Table comparison of different machine learning methods.

4.3.2. DEEP LEARNING:

DL is suitable for detecting DDoS attacks since it can perform feature extraction and classify unlabeled data, although the labelled data of legitimate users is easily available, data of malicious traffic is infrequent. DL approaches can extract information from incomplete datasets and still identify DDoS attacks [21].

Methods which see frequent usage are Deep Neural Networks (DNN), Recurrent Neural Networks (RNN), Deep Belief Networks (DBN) and Long Short-Term Memory (LSTM) Networks. They have been shown to be extremely helpful in the detection of low-rate attacks and detecting long-term sequences [22].

AUTHOR	DL MODEL	ACCURACY	METRICS	DATASETS	KEYFINDINGS
G. Zhao, C. Zhan, L.Zheng. [23]	DBN	99.14%	Precision; 93.25% FAR: 0.76%	KDD Cup 99	Use of DBN and PNN to classify
C.Yin et al. [24]	RNN	99.81% in the training set. 83.28% in the testing set	Recall: 71.44% FPR: 3.07%	NSL-KDD	Higher accuracy with multiclass classification compared with conventional IDS.
R.C. Staudemeyer [25]	LSTM	93.85%	Average FAR 1.62%	KDD Cup 99	Advantage in detecting unique time series.
R.B. Krishnan and N.R. Raajan[26]	RNN	97.4%	Recall: 97.05% Precision: 99.9%	KDD Cup 99	Detects DoS attacks better than Probe, U2R and R2L.
Kumar, P. [27]	DNN	99.39%	Not provided	CICDDOS2019	Less computational time, more accurate
Kim, J., Kim, J., Thu, H.L.T. and Kim, H. [28]	LSTM	96.93%	Precision: 98.8% FAR: 10%	KDD Cup 99	Higher false alarm rate while higher detection rate.
Yu, Y., Long, J. and Cai, Z. [29]	dilated convolutional autoencoders (DCAEs) and RNN	98.44%	Recall: 98.40% F-score: 98.41%	CTU-UNB and Contagio-CTU-UNB	Certain that these feature representations are effective in detecting attacks and lowering the FAR.
W.Wang et al. [30]	1D-CNN	non-VPN: 100% VPN: 99%	Not provided	ISCX VPN-nonVPN	Accuracy decreases when new classes are added

Figure 4: Table comparison of different deep learning methods.

4.3.3. SUMMARY

In summary, both ML and DL offer robust techniques for data analysis and prediction. These methods have proven to be extremely helpful in detecting attacks, outperforming conventional IDS in every case. On the other hand, most of the studied models are simple, they lack a combination of techniques to achieve better results, as we have seen in some other research papers, like Rizvi, F, et al. [20], that achieved a significant improvement in their models after applying another ML model for the data processing.

It is clear the advantage of using AI for protecting ourselves from cybercriminals, as technology advances so do the techniques with malicious intentions, making it essential for defense mechanisms to evolve at the same ratio. ML and DL provide the efficiency required to maintain a secure environment.

5. ML AND DL MODELS

We are comparing one of the most used models of ML, K-Nearest Neighbor and one of the least used in DL, Convolutional Neural Networks. KNN is widely used due to its simplicity, training time and ability to work with multiclass datasets [31]. CNN has been mostly used in computer vision, but it has proven to work particularly well in data in the form of multiple arrays or data that have strong local correlations [30].

To take it further, we are comparing the models applying random forest classifier to decide the best 10 features that determine if the request is from an attacker. We find this to be helpful to enhancing the model's performance, since we are working with two large datasets and 79 and 42 columns with traffic information and most of them will not give any new insights that can help the model classify the attacks.

5.1. DATASETS

We are using two different datasets to contrast our results, this decision has been made as we consider one dataset to be insufficient in determining the model's accuracy.

- NSL-KDD dataset: Developed by the Canadian Institute for Cybersecurity (CIC) in 2009 to solve the issues of the KDD cup 99 dataset, which included a large number of redundant records, resulting in a bias towards certain classes when applied to models.

  The dataset consists of a set of features extracted from network connections recorded over a period of time, with each connection represented by a record. We believe it will be helpful for the models to learn from normal traffic with different intrusion methods allowing for a more robust model. It has also been shown that the models that rely on this dataset tend to perform highly.

- CICDDoS2019 dataset: Also developed by the CIC, it was part of a bigger effort to provide a more up to date dataset with newer DDoS methods. This dataset is a part of a series of datasets provided by CIC, improving the earlier datasets like NSL-KDD, designed to represent modern network traffic and attack techniques more accurately.

  It contains features extracted from the network traffic and includes both raw traffic data and derived feature sets, which can be used directly for machine learning purposes. The traffic was recorded over a week with a different attack for every day. This results in various files that can be used for more specific tests.

  For our research we needed to group all the datasets into one to have a bigger amount of values to train our models. At first, this posed a problem since it required a lot of computational resources, making us unable to process all our data in our local machine. Therefore, we decided to download a version that was released later on that grouped all the datasets together.

5.2. LIBRARIES

First we utilized pandas to read our datasets and be able to manipulate and implement them to our models.

There are two main libraries that we have used for the development of our ML and DL models, scikit-learn and keras. We decided to use them due to their intuitive architecture and our familiarization with them. Both provide robust frameworks for ML and DL.

- Scikit-learn offers a wide range of tools for data processing, it is known for its good results with traditional machine learning models such as, SVM, KNN, RF, among others.

  We made use of *GridSearchCV* and *KNeighborsClassifier* for classifying our data since they fit perfectly for our task. Then we imported different methods like *confusion_matrix* and *classification_report* to analyze our results. Also, for processing the CICDDoS dataset, since all the data was grouped in one file we utilized the *train_test_split* method to split our datasets and be able to fit it accordingly to our models

- Keras is a high-level neural networks API. We made use of the different methods it provides to develop the layers of the CNN. *Convolution1D, Dense, Dropout, Flatten, MaxPooling1D, Sequential* and *to_categorical* have been the most important methods for creating a robust CNN while maintaining an intuitive and easy to navigate environment.

Lastly, we utilized *matplotlib* and *seaborn* to visualize our results. To process the confusion matrix, we utilized keras *confusion_matrix* method.

We believe that by selecting these libraries we will build a robust model while being easy to read and intuitive to manipulate, resulting in a better working environment and allowing us to focus more on the model's architecture.

5.3. DEVELOPMENT

We have conducted two experiments with four different models each. First, we analyzed the datasets with a basic implementation of the models, then we analyzed the models' performance when combined with Random Forest (RF) classifier to detect the most important features to differentiate legitimate from malicious traffic. In addition, we compared the performance of the models when working with NSL-KDD but changing the labels to attack or benign, making it a binary classification instead of multi-class.

We believe RF to be helpful due to its ability to decrease computational process by decreasing the data spread and only taking in count the important values. Research papers that yielded higher accuracy, where mostly when the combination of different processes is applied to work in designated sections. RF will reduce the possibility of overfitting and will improve our working pace.

5.4. DATA PREPROCESSING

When working with a large database it is important to process and sanitize our data before applying it to our models to achieve higher efficiency. NSL-KDD is known for its high imbalance between classes and can bias the models towards specific classes. Therefore, the data sanitation and processing, becomes fundamental for the development of a robust model.

First, we need to get rid of the categorical text values and give it numeric values to be able to fit it into the model. The only categorical columns are the protocol_type, service and flag. We apply a loop that iterates

over the categorical values and use .cat.codes to assign a value. For instance, Protocol_type has three categories *['tcp', 'udp', 'icmp']* which are turned into *[0, 1, 2]*

Data normalization is essential to ensure that all values contribute equally to the training. This way, when the data is being handled it takes less computational resources. To normalize the values, we selected the minimum *(min(x))* and maximum *(max(x))* values of each column, then we iterated over each value *(x)* subtracting *min(x)* and dividing it by the difference between the *min(x)* and *max(x)* (figure 5).

$$x_{\text{norm}} = \frac{x - \min(x)}{\max(x) - \min(x)}$$

Figure 5: Formula for data normalization.

6. KNN

Rizvi, F. et al[20] analyzed the most common ML and DL models used in IDS. In over 20 papers studied, they found that over 15 models utilized KNN. We utilized our own research, Rizvi, F. et al[20] research and our familiarization with the model as benchmarks to select KNN.

KNN is an ML algorithm for classifying data based on the majority of nearest neighbor objects [32]. It has been effective in detecting several types of network attacks due to its ability to interpret complex data patterns.

Selecting the right parameters for the model are detrimental for achieving high accuracy. We decided to perform a grid search to check the model's performance and return the best parameters to analyze our datasets. First we check the model's performance with the values [3, 5, 7, 10] for *k* as our number of neighbors to select a class.

We need to specify a weight function that will address the importance of each point in the neighborhood, *uniform* which weights all points equally or *distance* weights points by the inverse of their distance. Closer neighbors will have a greater influence than neighbors that are further away.

To calculate proximity between neighbors, Euclidean distance can be used, it calculates the distance between two points in a straight line and is the most often used in processes of ML. It is the result of the square root of the difference between two vectors, it is written as follows:

$$d(\mathbf{p}, \mathbf{q}) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Figure 6: Euclidean distance

Where *n* represents the number of values, *i* the sequence, *qi* represents the training data value assigned to *i*, and *pi* the test data value attributed to *i.* After checking the proximity values, they are sorted in ascending order. Finally it specifies the attribute based on the value of *k.*

Manhattan distance, also known as "city block distance" or L1 norm is the sum of the distances from all the attributes [33], it is written as follows:

$$distance = \sum_{1}^{n} |p_i - q_i|$$

Figure 7: Manhattan distance

Where *n* represents the number of values, *i=1* the sequence. Here, */pi−qi/* is the absolute difference between the ith coordinates of point. For our model we are going to check the results using both since it can affect performance significantly.

6.1. KNN with NSL-KDD

Since we are working with a large dataset, we apply 5 folds of cross validation, making 5 smaller sets and the model is trained and validated 5 times. This reduces the possibility of overfitting and makes the model generalization better for new, unseen data.

As we see in the following code (figure 8), we set the parameters for our grid search and apply them using *GridSearchCV,* a method from scikit-learn to optimize our parameters:

```
model = KNeighborsClassifier()
param_grid = {
    'n_neighbors': [3, 5, 7, 10],
    'weights': ['uniform', 'distance'],
    'metric': ['euclidean', 'manhattan']
}

# Setup the grid search
grid = GridSearchCV(model, param_grid, cv=5, verbose=1, scoring='accuracy')
grid.fit(trainX, y_train)
```

Figure 8: Grid-search for KNN

After running the model for the first time, the results showed that the best parameters to analyze the data are: 3 as the number of neighbors, *distance* as our weight function and Manhattan equation to address the distance. Achieving an accuracy of 99.87%, as shown in figure 9:

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Best score: 0.9986822602703438
```

Figure 9: KNN with NSL-KDD results

If we analyze the confusion matrix (figure 11) and its corresponding metric scores (figure 10), it is clear the high accuracy in classifying the attacks, although having some problems in labeling the exact type of attack.

We can consider the model a success. However, in real world applications having a model with less than 99.9% accuracy makes it not accountable and therefore not suitable for IDS and having a precision of less than 80% in some of the attacks is a big deal, cybercriminals could take advantage of these attacks that go through undetected and modify the rest to adjust more accurately the rest of the packets.

	precision	recall	f1-score	support
Normal	0.74	0.98	0.84	9889
DoS	0.92	0.85	0.89	7460
Probe	0.79	0.69	0.73	2421
R2L	0.95	0.12	0.21	2707
U2R	0.36	0.45	0.40	67
accuracy			0.80	22544
macro avg	0.75	0.62	0.61	22544
weighted avg	0.83	0.80	0.77	22544

Figure 10: KNN NSL-KDD metrics scores

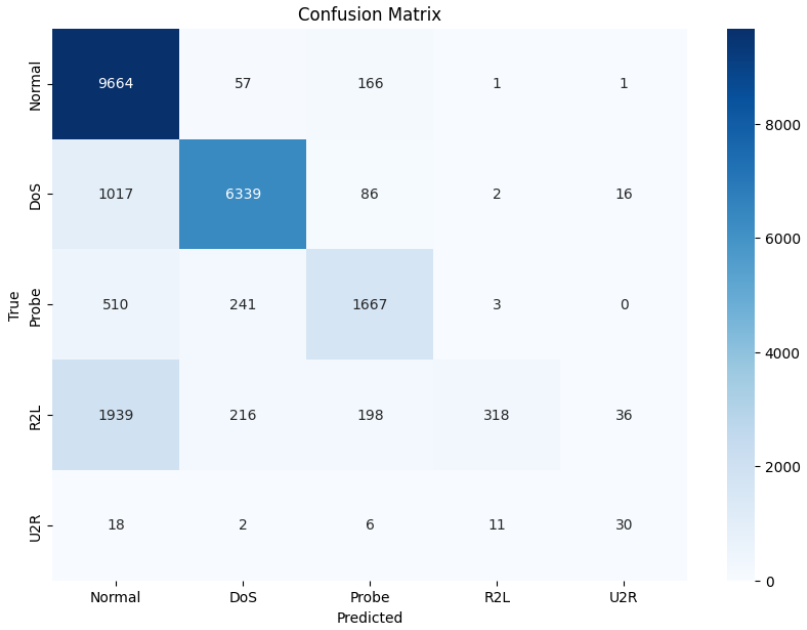

Figure 11: KNN NSL-KDD condusion matrix

The model achieving a high accuracy suggests that it is robust in terms of generalizing data, and it proves to be a valuable tool for cybersecurity. In addition, if we compare the results of the different models across the grid-search, it shows satisfactory results in all the models with small variations, being more accurate towards smaller groups of neighbors.

On the other hand, since we are working with a large dataset, it is common to have a lot of invariances between classes making it more difficult to classify groups. Also, the computational requirements to process the data are extremely high, with an average of two hours to perform the grid-search.

6.2. **KNN with CICDDoS2019**

To compare the performance with CICDDoS2019, we used the exact same model. Although the model requires a larger computational time, the dataset includes 79 columns with values instead of 42 like NSL-KDD. The results did not exceed expectations. Having 99.50% accuracy, with this dataset the best parameters were: 3 for the number of neighbors, *distance* for the weights and *Manhattan* equation, as seen in figure 12.

```
Fitting 5 folds for each of 16 candidates, totalling 80 fits
Best parameters: {'metric': 'manhattan', 'n_neighbors': 3, 'weights': 'distance'}
Best score: 0.9949825726783041
```

Figure 12: KNN CICDDoS2019 grid-search results.

After analyzing the confusion matrix (figure 13), we can determine the success of the model and proves to be strong enough to detect most network traffic attacks. Having only 53 misclassifications in total shows the scope of the model. It demonstrates versatility and the ability to adapt to new unseen data.

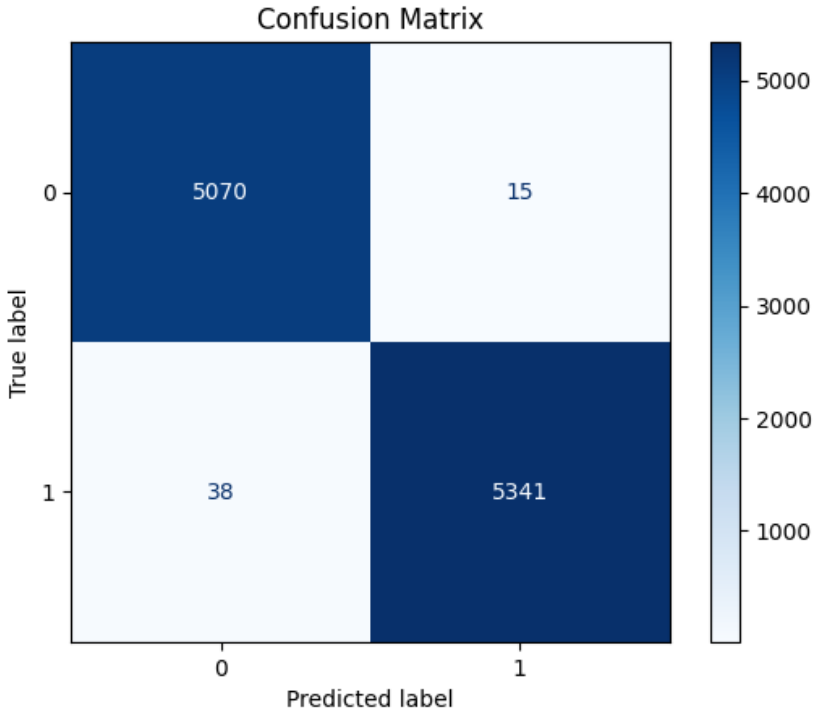

Figure 13: KNN CICDDoS2019 confusion matrix

7. CNN

W.Wang et al. [30] used a 1D-CNN to classify end-to-end encrypted network traffic. He achieved in one of his experiments a 100% accuracy. This is unexpected since CNN are mainly used in the field of computer vision, making them not the best model for this classification task.

On the other hand, Gu, J. et al [34] in their research showed new successful applications of 1D-CNN in Natural language processing tasks due to their ability to read sequential data. 2D-CNN are better for 2-dimensional arrays data and so on. We used these researches as our benchmark to start exploring the classification of DDoS attacks using CNN.

The architecture of the CNN is inspired by visual perception, and it varies depending on the task. However, their basic components consist of three types of layers, convolutional, pooling and fully-connected. The convolutional layer aims to learn feature representations of the inputs, composed of several convolution

kernels in charge of extracting the features maps. Specifically, each neuron of a feature map is connected to a region of neighboring neurons in the previous layer. The previous neighborhood, which is called the receptive field of the neuron in the prior layer. The new features map is generated by convolution on the input data with a learned kernel, then, applying nonlinear activation function to the convolved results that acts as a threshold to determine the classification. Pooling is implemented since it lowers the computational burden by reducing the number of connections while still maintaining valuable information.

Lecun et al. [35] studied in their paper the applications of 1D-CNN in fields such as health, aerospace structures, high power circuity, etc. The results demonstrated the superior performance on those applications which have limited labeled data and high signals variations. As we can see in *figure 14*, the CNN takes one dimensional data, then it processes the data in each convolutional layer, with their corresponding activation function and pooling. Then the fully-connected (dense) layers that are identical to the layers of a typical Multi-layer Perceptron (MLP) and therefore called "MLP-layers".

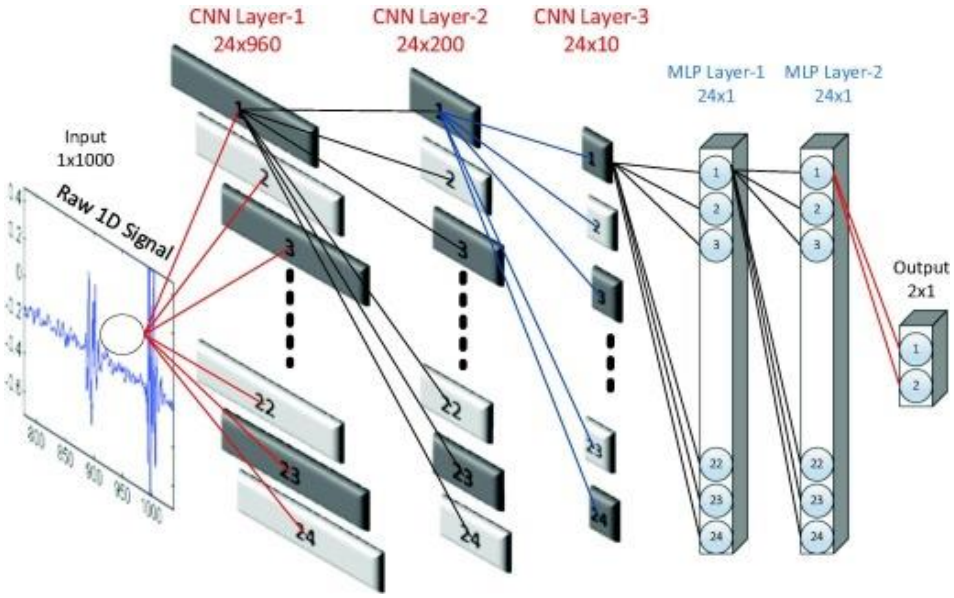

Figure 14: Architecture of 1D-CNN

7.1. **CNN with NSL-KDD**

Our CNN is composed of seven different layers. First, we used a sequential CNN that takes one convolution layer with 32 kernels with the size of 3, we are using the *relu* activation function. Layer two is *maxpooling* to reduce the dimension of the data while still maintaining the valuable information. Layer three and layer four use the same processes as before to further analyze the data. Layer five with the *Flatten()* method ensures that the data is 1 dimensional. Layer six, there are 64 fully connected neurons that processes the features extracted. Layer seven randomly selects 50% of the values and sets them to zero to avoid overfitting and ensure all values are treated equally. Finally, the output layer consists of a single neuron with a sigmoid activation function to output a probability indicating the model's accuracy to classify classes

When we fit the data into the model, initially we specified fifty epochs, then we would leave it overnight training with five hundred epochs. At first the results were extremely poor, achieving an average of 20.47%

accuracy in detecting the attacks. To achieve higher accuracy, we increased the number of kernerls and layers, making slight changes to slowly increase the model's performance. In general, when the model is set to training, it finds the highest accuracy on the first ten epochs, from that point it will make slight variations every epoch, but it does not give the impression of having a proper learning curve.

From this point, we started to reduce the number of epochs to ten since it sped up our working time and we could check new results every couple minutes. We can assume that this is due to overfitting, or the model is not complex enough.

Since the experiment is conducted using specifically the data in this format, we continued investigating how to improve the model. After numerous tests, the CNN showed results like no other before. Adding 3 new convolutional layers, with a higher kernel count and a higher dense layer, the model was able to achieve 92.05% accuracy on the training set and 80.56% on the validation set.

Although we increased accuracy, the model's performance is poor, demonstrating why it is the model least studied and applied for IDS. Unsatisfied with the results we decided to process the data differently, we decided to give more importance to the weight between classes, since we are working with a large dataset, it is common to have invariance which can bias the model towards classes that are more populated than others.

We made use of the *compute_class_weight* function from Scikit-learn's utilities. It computes weights that can be used to make the model pay more attention to underrepresented classes during training. Boosting the weights of classes that are less frequent than others increases the loss function's penalty for misclassifying samples of these classes, forcing the model to ensure higher accuracy of minority class classification. This results in a model that is more balanced and performs better on both balanced datasets and those which consist of highly imbalanced classes.

The model's improvement was notable since the first test. We proceeded working on our model with seven different layers since we had already discovered that it was the most accurate. The accuracy on the training set increased to 97.07% and 77.46% on the validation set.

If we analyze the confusion matrix (Ffigure 16) and its metric scores (figure 15), they suggest that the model is accurate enough to detect most of the attacks. On the other hand, having many undetected attacks can help the attacker learn from those requests, apply small changes and redirect them to those vulnerabilities.

	precision	recall	f1-score	support
Normal	0.79	0.90	0.84	9889
DoS	0.85	0.84	0.85	7460
Probe	0.60	0.63	0.61	2421
R2L	0.81	0.33	0.47	2707
U2R	0.11	0.40	0.17	67
accuracy			0.78	22544
macro avg	0.63	0.62	0.59	22544
weighted avg	0.79	0.78	0.77	22544

Figure 15: CNN NSL-KDD metrics scores.

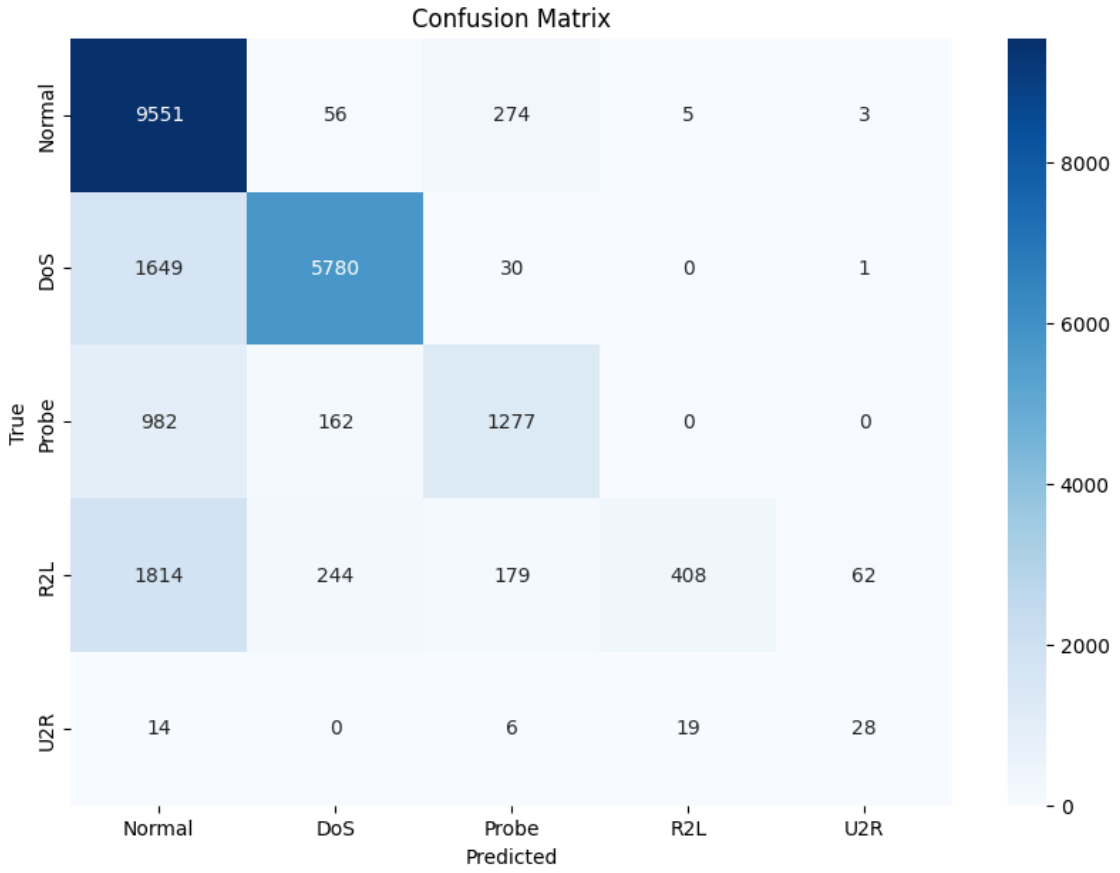

Figure 16: CNN NSL-KDD confusion matrix

7.2. **CNN with CICDDoS2019**

As before, we are going to apply the same model that we used on the NSL-KDD dataset. Since CICDDoS2019 only has two classes, *Normal* or *Attack*, we modified the model to work with binary cross entropy instead of categorical. We believe this will help the model to classify the traffic by focusing on optimizing predictions for two distinct outputs. This can lead to more fine-tuning of model parameters and thresholds, which could further increase the accuracy and predictive confidence of the model.

After running the model on this dataset, the results are almost instantaneous and exceptional. Achieving 100% accuracy on the training and validation sets, the model has successfully classified every attack. When we analyze the metrics (figure 17) and the confusion matrix (figure 18), it is obvious the success of the model, having 100% score in all the metrics and zero misclassified values.

	precision	recall	f1-score	support
Dos	1.00	1.00	1.00	66709
Normal	1.00	1.00	1.00	19566
accuracy			1.00	86275
macro avg	1.00	1.00	1.00	86275
weighted avg	1.00	1.00	1.00	86275

Figure 17: CNN CICDDoS2019 metrics scores.

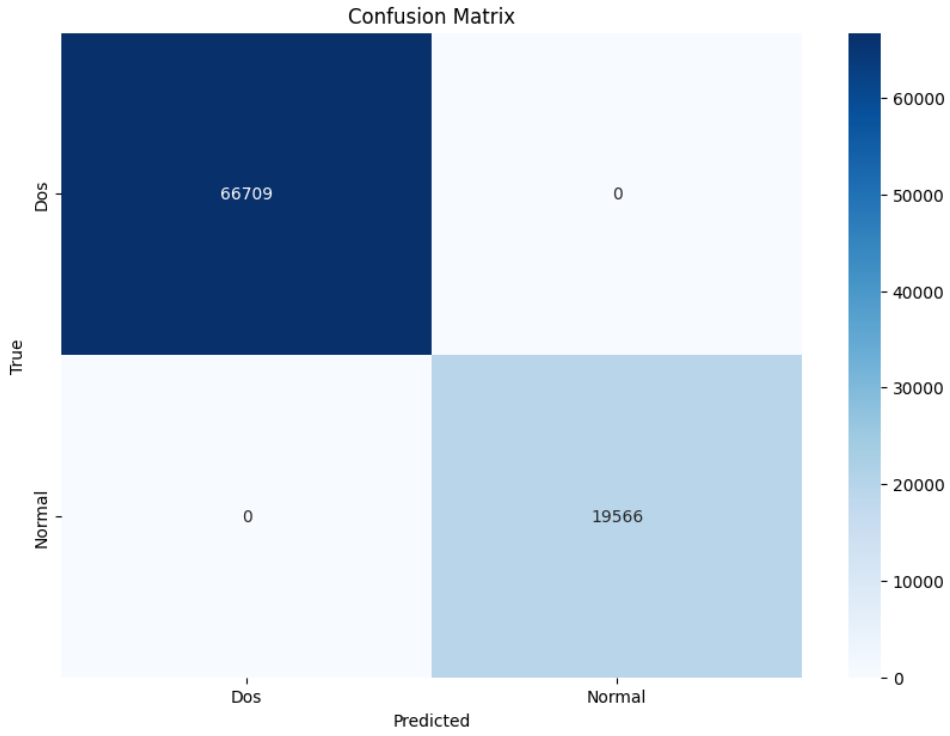

Figure 18: CNN CICDDoS2019 confusion matrix

We can assume that these results correlate with the dataset nature, having more features, focused on DoS behavior and a binary classification. Using a 1D Convolutional Neural Network (1D-CNN) for binary classification, especially in tasks like traffic analysis for intrusion detection, offers several advantages due to the unique properties of convolutional layers and the nature of sequence data in network traffic. Here's why employing a 1D-CNN can be particularly effective for this application. We believe this experiment to be a good benchmark for studying the applications of CNN in fields that were previously overlooked.

By employing 1D-CNNs for the binary classification of network traffic, we are not only exploring its efficacy in cybersecurity, a field where CNNs are not traditionally used, but also setting a new application for other non-conventional applications of CNN technology.

8. RANDOM FORES CLASSIFIER

RF is a further refinement of classification and regression tree (CART) models. Its versatility makes it suitable in either classification or regression [36]. It works by creating decision trees with random features from the dataset, bootstrapping, then the values are classified on each decision tree to address the columns that give most information about the values to classify it correctly. It is believed that RF helps the data processing of large datasets more efficiently because decision trees are heavily influenceable by the data invariance and can affect the results, however, by assigning random columns we make sure all classes are equally processed.

To address the impurity of the classification of each decision trees, and make the model learn from the data, we make use of the Gini Formula (figure 19). Given a dataset *D* that contains *k* classes. The probability of a sample belonging to class *i* at a given node can be noted as *pi.*

$$Gini\ Index = 1 - \sum_{i=1}^{n} \left(P_i\right)^2$$

$$= 1 - [(P_+)^2 + (P_-)^2]$$

Figure 19: Gini index

After we found the best classes, we created a subset with only the best ten. Our hope is to make the model rely more on the important classes and disregard the columns that do not give any new insights in classifying the attacks. Also, by decreasing the size of the dataset we will reduce the computational requirements, hence reducing the time for training and allowing us to perform more experiments and focus more on the refinement of the models.

- **Top ten features for NSL-KDD**:

  iff_srv_rate: 0.11410175949346983

  same_srv_rate: 0.10267243802032483

  flag: 0.10040431417124152

  dst_host_same_srv_rate: 0.06043220700609204

  outcome: 0.05200002821060657

  dst_host_diff_srv_rate: 0.05099309559556243

  count: 0.04859095603925552

  dst_host_srv_serror_rate: 0.046941547767606784

  protocol_type: 0.040618330052214706

  dst_host_serror_rate: 0.0398199528475093

- **Top ten features for CICDDoS2019:**

Label: 0.19649052085269106

Fwd Packet Length Min: 0.0767958106072246

Bwd Packet Length Mean: 0.06490708750834484

Bwd Packet Length Max: 0.052186228840741034

Packet Length Min: 0.04497955916168707

Down/Up Ratio: 0.044714129938091735

Subflow Fwd Bytes: 0.035827490449619505

Avg Fwd Segment Size: 0.03186342331813333

URG Flag Count: 0.03126298369451662

Subflow Bwd Bytes: 0.029604286477011584

While the NSL-KDD dataset concentrates on features related to the network and connection, the CICDDoS2019 dataset is more focused on packet size and certain behaviors on how packets were transmitted. CICDDoS2019 was built to explain features of DoS attacks.

For NSL-KDD features like *iff_srv_rate* and *same_srv_rate* suggest that the rate of connections to the same service and the error rate are significant indicators of malicious traffic. High rates might indicate automated actions typical of botnets . In CICDDoS2019 features like *Fwd Packet Length Min* and *Bwd Packet Length Mean* highlight the size and direction of packets. Larger or smaller than usual packet sizes can be an indicator of malicious traffic. Also, in this dataset the *Label* becomes the most important feature, it shows how the protocol type is essential in differentiating the attacks, different types of protocols may have unique vulnerabilities or may be used differently in attack scenarios. For instance, UDP packets might be commonly used in flood attacks.

8.1. **KNN and RF with NSL-KDD**

We have applied the same model of KNN to our subset with the most important values. Not only with a much smaller computational time, but also achieving 99.42% accuracy. Although it has less accuracy than the KNN model with the CICDDoS2019, it is a huge milestone to have similar results in a fraction of the time.

The metrics (figure 20) and the confusion matrix (figure 21) demonstrate a high accuracy in classifying attacks but there is still a big number of attacks that go through undetected, giving a lot of space for improvement. We believe that these results are due to the invariance between classes which makes it difficult for the model to detect the different types of attacks.

	precision	recall	f1-score	support
Normal	0.82	0.95	0.88	9889
Dos	0.90	0.87	0.88	7460
Probe	0.72	0.77	0.74	2421
R2L	0.88	0.40	0.55	2707
U2R	0.26	0.16	0.20	67
accuracy			0.84	22544
macro avg	0.72	0.63	0.65	22544
weighted avg	0.84	0.84	0.82	22544

Figure 20: KNN RF NSL-KDD metric scores

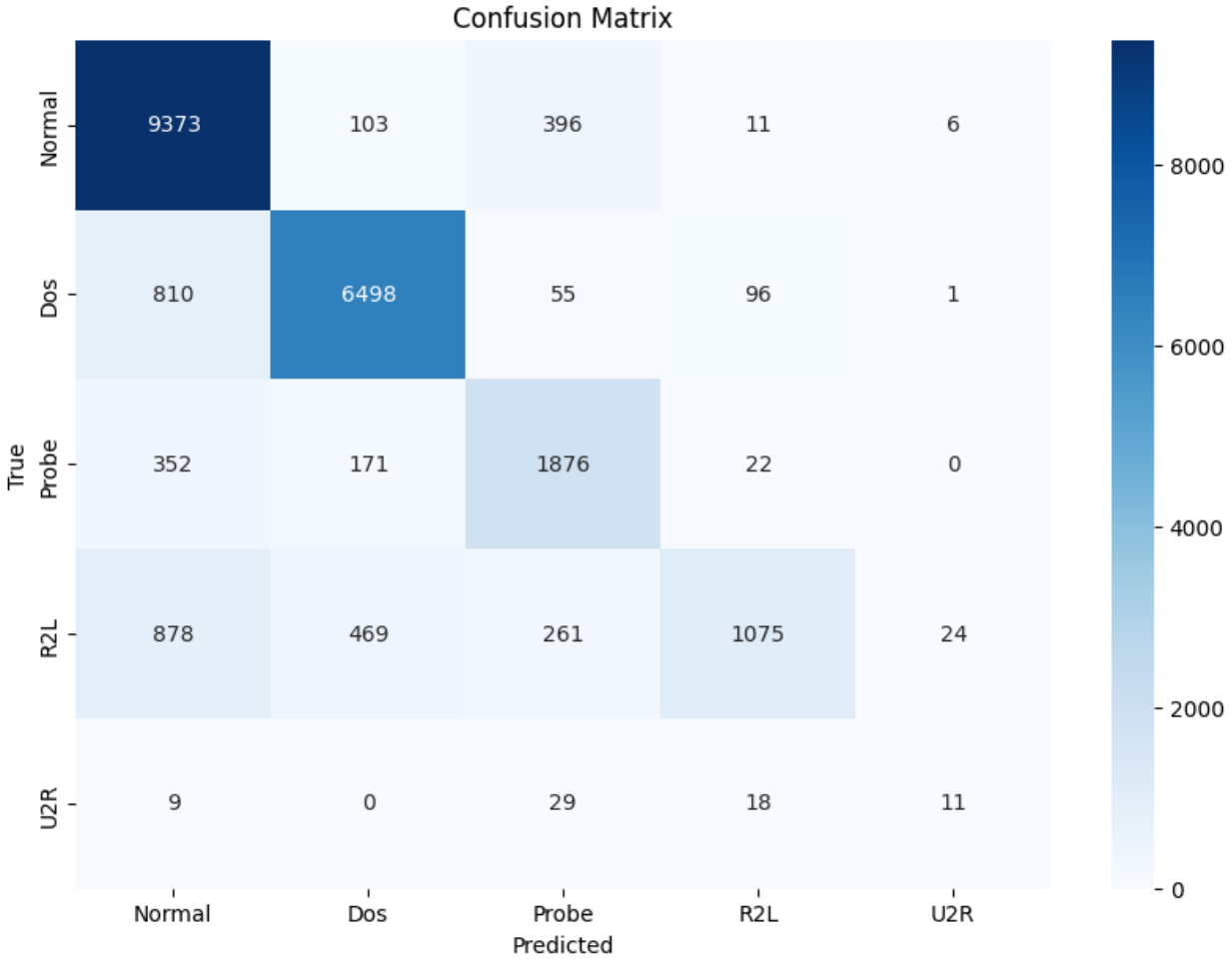

Figure 21: KNN RF NSL-KDD confusion matrix

After this last experiment with the NSL-KDD dataset, we can determine that the model is not complex enough to classify the four different types of attacks. We assume that by changing the labels of the attacks to *Normal* and *Attack* like in the CICDDoS2019 dataset, the performance of the models will improve by

making a binary classification instead of a categorical. Hence, unsatisfied with the results, we decided to add a new experiment to our research and compare this same model but doing a binary classification, like what we are doing with the CICDDoS2019 dataset. We hope that by simplifying the classes of attacks the performance of the model will increase, yielding better results.

8.2. KNN and RF with NSL-KDD with binary classification:

After performing the grid-search, as expected, the model's accuracy has increased to 99.82%. The main difference in the best parameters for this task is the decreased number of neighbors, three neighbors. Manhattan equation and weighting the values based on the distance keeps being the best way to analyze the data.

	precision	recall	f1-score	support
Normal	0.73	0.96	0.83	9889
Attack	0.96	0.72	0.83	12655
accuracy			0.83	22544
macro avg	0.84	0.84	0.83	22544
weighted avg	0.86	0.83	0.83	22544

Figure 22: KNN RF NSL-KDD BINARY metric scores

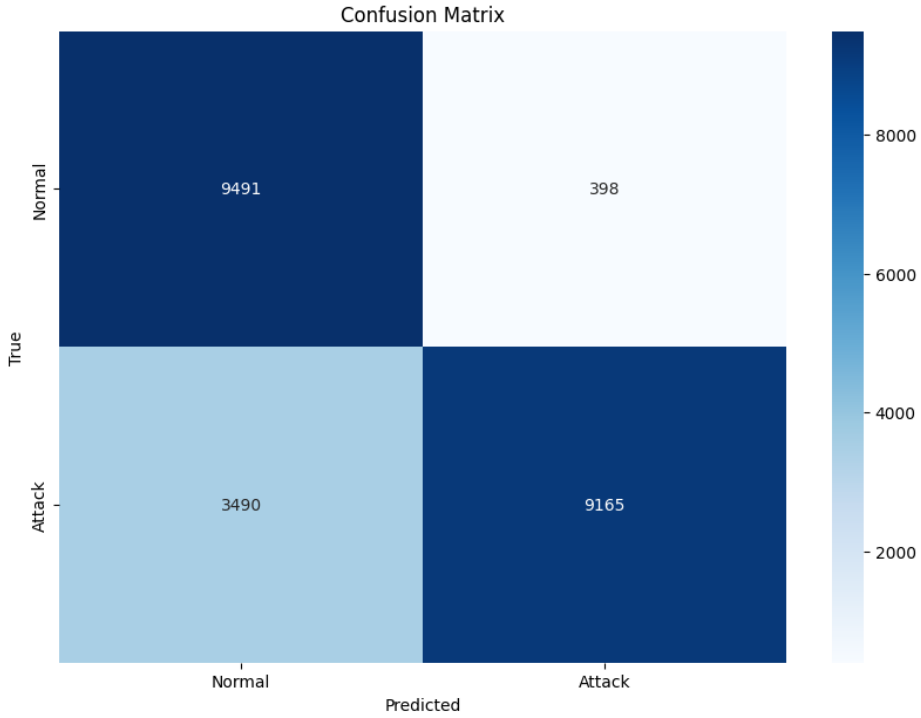

Figure 23: KNN RF NSL-KDD BINARY metric scores

After analyzing the confusion matrix (figure 23) and its respective metric scores (figure 22), we can determine a minor improvement in classification. However, the number of overlooked values is large, suggesting that there is still a lot of room for improvement.

8.3. **KNN and RF with CICDDoS2019**

We took the same model as the ones previously studied to assess its performance. After running the grid-search on CICDDoS2019, the results presented a high accuracy in detecting attacks. It further ensures our belief of using binary classification for IDS.

Achieving 99.96% accuracy, the parameters that presented the best results were: five as the number of neighbors, *distance* as our weight function and Manhattan equation to address the distance. It correlates with the decrease in the dataset features, since our goal by using RF was to make the data easier to group to classify the attacks. When most of the data is widely spread and with a lot of impurities it will make the model rely on smaller groups of neighbors to classify more accurately.

	precision	recall	f1-score	support
Dos	1.00	1.00	1.00	66709
Normal	1.00	1.00	1.00	19566
accuracy			1.00	86275
macro avg	1.00	1.00	1.00	86275
weighted avg	1.00	1.00	1.00	86275

Figure 24: KNN RF CICDDoS2019 metric scores

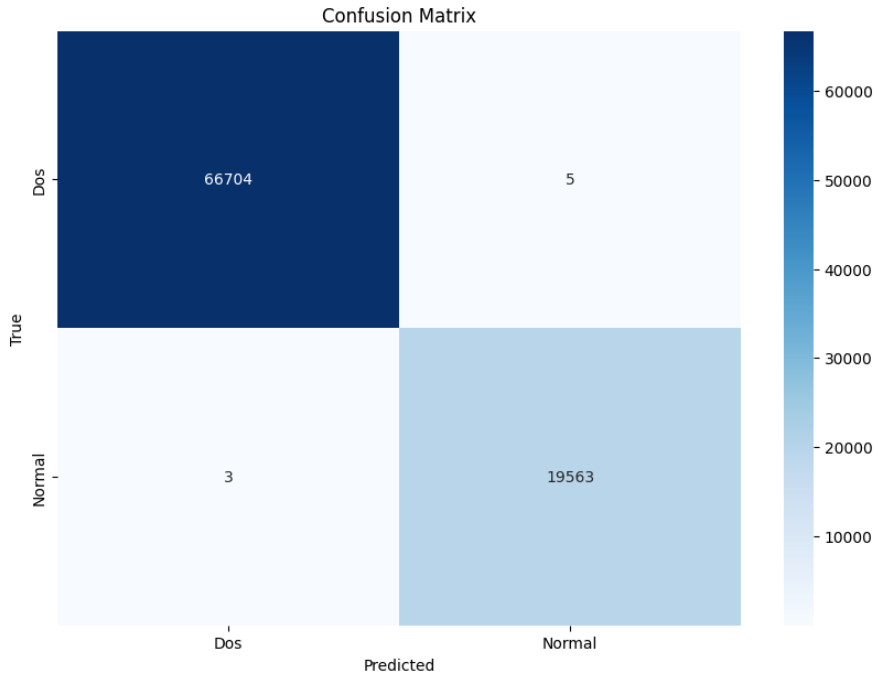

Figure 25: KNN RF CICDDoS2019 confusion matrix

After analyzing the confusion matrix (figure 25) and the classification report (figure 24) with its corresponding metrics, it is clear the success of the model on this dataset. With only 8 misclassified values,

this model poses a big challenge for the attacks to go through and overrun the server. Improving its original results after applying the RF demonstrates the importance of feature selection for optimal performance.

8.4. CNN and RF with NSL-KDD

When the CNN is applied to the subset of best values of NSL-KDD, the improvement is clear. Starting with already more accuracy on the first epoch, the model achieved an accuracy of 99.14% and 81.52% on the training and validation set respectively.

	precision	recall	f1-score	support
Normal	0.82	0.95	0.88	9889
DoS	0.91	0.84	0.88	7460
Probe	0.61	0.69	0.65	2421
R2L	0.67	0.37	0.48	2707
U2R	0.10	0.10	0.10	67
accuracy			0.81	22544
macro avg	0.63	0.59	0.60	22544
weighted avg	0.81	0.81	0.80	22544

Figure 26: CNN RF NSL-KDD metric scores.

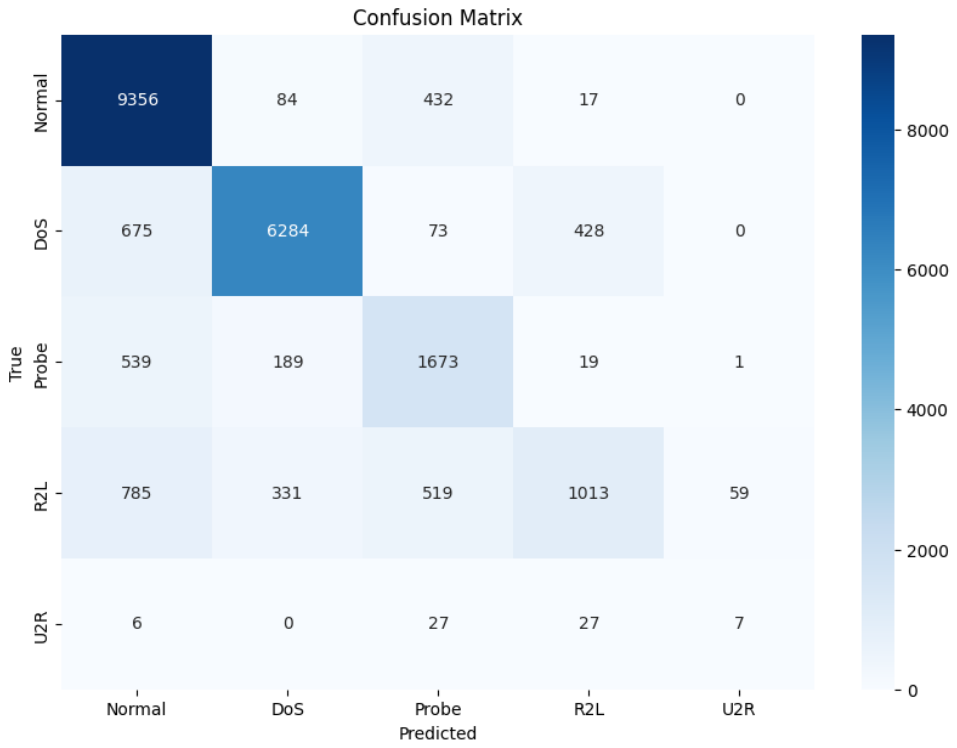

Figure 27: CNN RF NSL-KDD confusion matrix

After analyzing the confusion matrix (figure 27) and its respective metric scores (figure 26), it is clear the inconsistency of the model in classifying the attacks. With a large amount of misclassifications, the model has specific problems in differentiating between the different types of attacks, excluding DoS which is the best class to classify.

Like before, unsatisfied with the results we are going to compare the model's performance by changing the attack classes to *Normal* and *Attack.* To compare its behavior with binary classification in hopes of yielding a higher accuracy.

8.5. CNN and RF with NSL-KDD using binary classification:

As expected, the accuracy of the model increased, 99.57% and 84.60% in the training and validation sets respectively. After analyzing the confusion matrix (figure 29) and their respective metrics scores (figure 28), it is clear the improvement of the model.

	precision	recall	f1-score	support
Normal	0.75	0.96	0.85	9889
Attack	0.96	0.76	0.85	12655
accuracy			0.85	22544
macro avg	0.86	0.86	0.85	22544
weighted avg	0.87	0.85	0.85	22544

Figure 28: CNN RF NSL-KDD BINARY metric scores

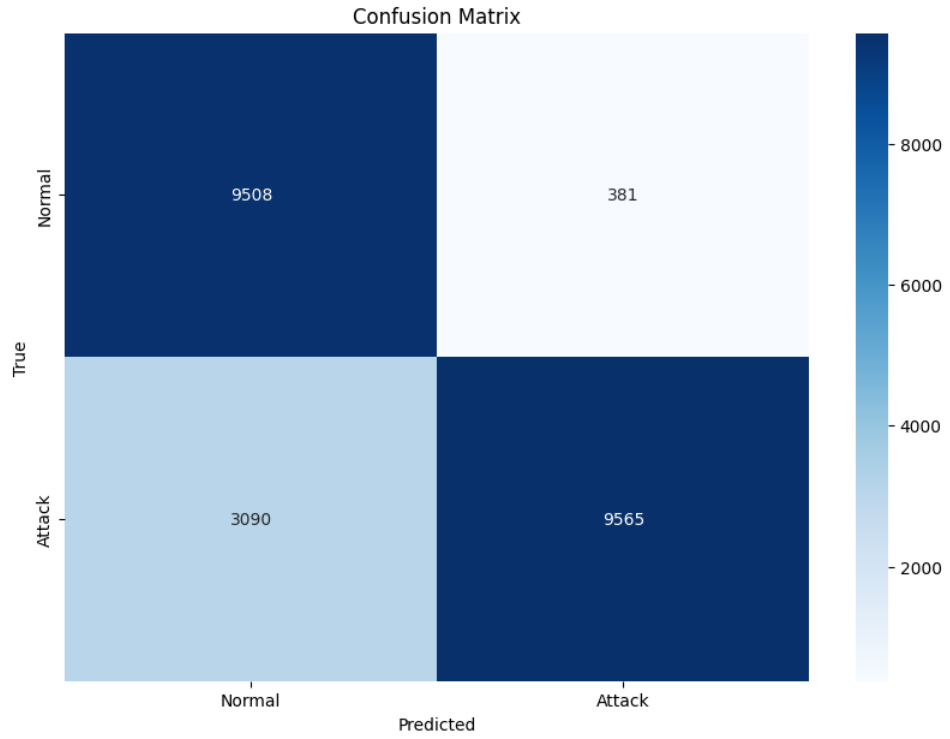

Figure 29: CNN RF NSL-KDD BINARY confusion matrix

On the other hand, the large amount of undetected traffic suggests that the model is not complex enough to detect every type of attack, hence not strong enough to be considered a suitable IDS even after the improvement.

8.6. **CNN and RF with CICDDoS2019**

After already achieving the best results we can hope for. Reducing the dimensions of the dataset will help to decrease the computational time while achieving the same results, hence a more optimized model.

As expected, the results have been outstanding, achieving 100% accuracy on both training and validation sets without a single misclassification (figure 30 and 31) in a fraction of the time. This model further demonstrates its ability to process data and fight against cyberattacks.

	precision	recall	f1-score	support
Dos	1.00	1.00	1.00	66709
Normal	1.00	1.00	1.00	19566
accuracy			1.00	86275
macro avg	1.00	1.00	1.00	86275
weighted avg	1.00	1.00	1.00	86275

Figure 30: CNN RF CICDDoS2019

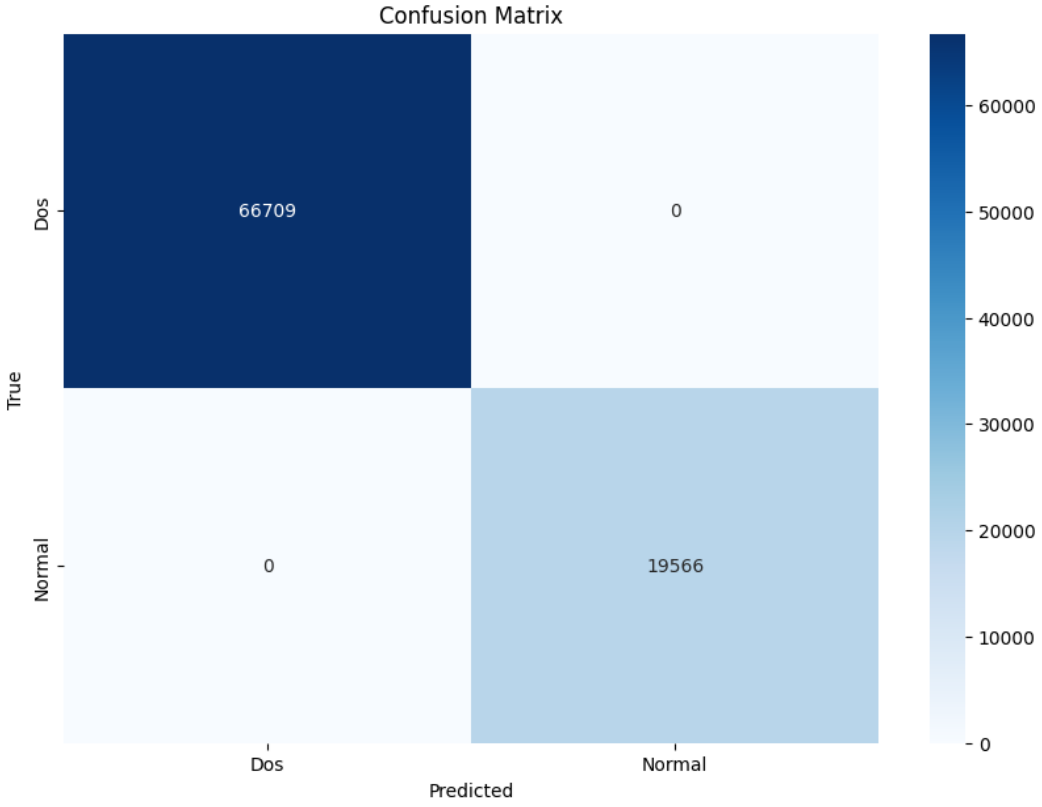

Figure 31: CNN RF CICDDoS2019 confusion matrix

We can conclude our experiment with CNN after achieving extremely good results across the training and validation sets. The reduction of the computational time while still classifying accurately the attacks is a big step towards a secure and robust IDS, making it the best model studied.

9. RESULTS

Overall, the results are overwhelmingly positive, hence, the potential of ML and DL in cybersecurity is clear. Their ability to analyze data patterns has proven to work efficiently on unseen attacks just from learning what is the average behavior of benign users.

Model	Dataset	Accuracy
KNN	NSL-KDD	Best score: 99.86%
KNN	CICDDoS2019	Best score: 99.50%
1D-CNN	NSL-KDD	97.07%
1D-CNN	CICDDoS2019	100%
KNN with RF	NSL-KDD	99.42%
KNN with RF with Binary classification	NSL-KDD	99.82%
KNN with RF	CICDDoS2019	99.96%
1D-CNN with RF	NSL-KDD	99.14%
1D-CNN with RF with Binary classification	NSL-KDD	99.57%
1D-CNN with RF	CICDDoS2019	100 %

Figure 32: Results on experiments conducted.

After analyzing the models side by side, we can determine certain common bullet points:

- All the models tested show high accuracy, with many surpassing 99% indicating that the selected features and the model architectures are highly effective for this type of data and task. The 1D-CNN models, particularly for the CICDDoS2019 dataset, achieve perfect accuracy, highlighting the effectiveness of using convolutional networks for sequence data in network traffic classification.
- The combination of models with RF results in higher accuracy. For example, when KNN is used with RF for NSL-KDD, there is an improvement from 99.86% to 99.82%, and even more notably, combining 1D-CNN with RF improves the accuracy from 99.14% to 99.57% on the NSL-KDD dataset. Suggesting that methods like RF can effectively improve the robustness and predictive power of the base classifiers.
- The application of models specifically designed for binary classification show a very high effectiveness, particularly with the CICDDoS2019 dataset where accuracies are near perfect. This indicates that adjusting model configurations to the specific nature of the dataset can significantly enhance performance.
- The differences in performance across the two datasets may indicate the models' adaptability to different types of network traffic captured in each dataset. The CICDDoS2019 dataset shows to be more suitable to perfect classification with the 1D-CNN model.

On the other hand, we can determine that the NSL-KDD dataset poses a challenge in achieving perfect accuracy. After revising the classes in this dataset, it is clear the imbalance between them. This leads to overfitting and inaccurate results. With only a few values for the U2R, R2L and Probe attacks compared to the other classes, it is understandable to achieve a lower accuracy across the models that rely on this dataset.

CICDDoS2019 however, has proved to be well balanced and with more features that can provide the models with new insights on how to classify the attacks, as seen in the following figure. Also having more up to date attacks it should be more difficult for the model to detect them, but the architecture of this dataset allows the model to generalize better, resulting in outstanding results.

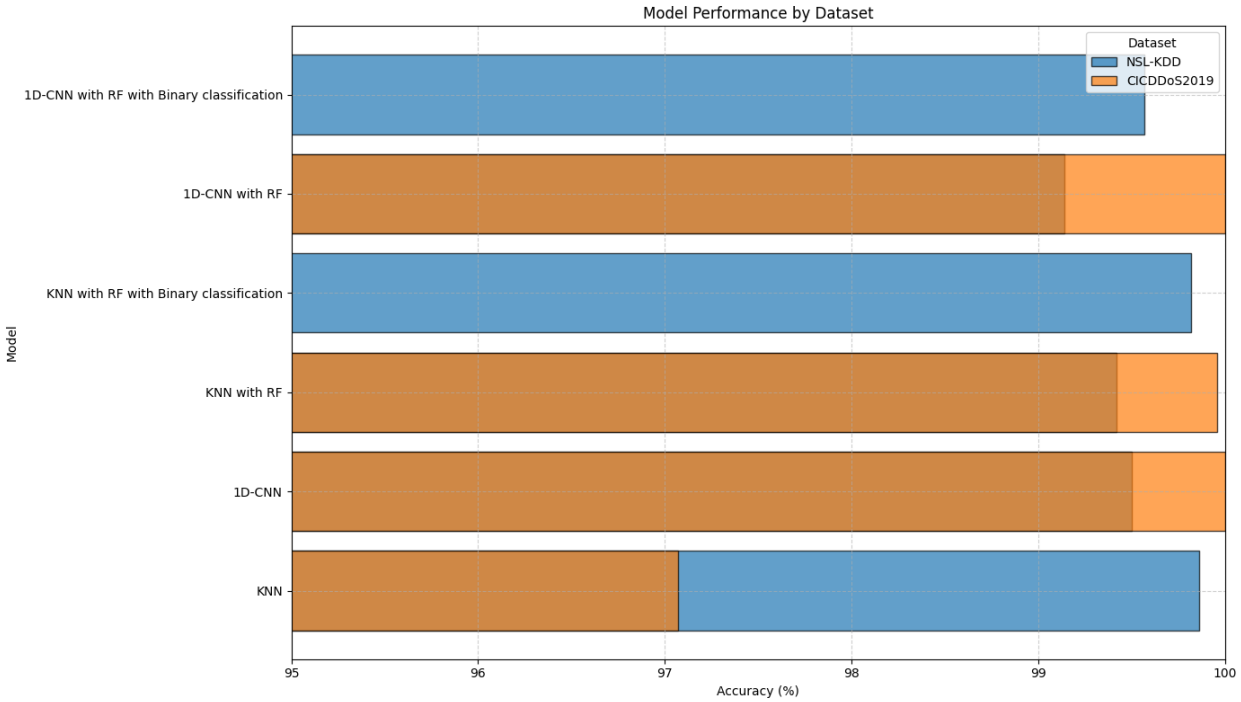

Figure 33: Different models' accuracy on the datasets

10. CONCLUSIONS

The field of cybersecurity is constantly evolving, making detrimental the use of new techniques to analyze data. After conducting this report, it is clear the potential of ML and DL for detecting cyberattacks, disregarding their potential would be a significant oversight for any cybersecurity professional. They not only improve the accuracy of the existing models but also enhancing their performance. Furthermore, the versatility of these models makes them easily appliable for different types of attacks. By integrating these technologies, organizations can anticipate and counteract sophisticated cyberattacks with unprecedented precision.

On the other hand, when working with large datasets, at first the revision for each model would take up to several hours, making it difficult to advance with the results and having a slow working pace. The data processing and sanitation has been detrimental for the performance of the model. At first there was a lot of overfitting, and the models did not have a proper learning curve. After refining the labels, normalizing the data and extracting the important features, we have achieved outstanding results up to the standards of real IDS.

When we applied the RF classification, our working environment improved drastically, allowing us to compare new results in a fraction of the time, then when a good model was developed, we would leave it training overnight with the complete dataset to compare the results later on and address if there is any advancement. This demonstrates our initial hypothesis that the combination of ML and DL methods will enhance the models' performance.

After analyzing more than fifty research papers for this experiment, we determined that models that utilized KNN were often highly effective, therefore a good fit for this research to compare the results. The CNN has shown to be more effective than one of the most implemented methods for classification in cybersecurity, with perfect metric scores and a low computational time we consider the model a success.

We can conclude that the potential of 1D-CNN is highly underestimated and often overlooked. With that in mind, the achievement of 100% accuracy is exceptional, we have successfully applied the concept that Wang et al. [30] studied for classifying end-to-end encrypted traffic and achieved similar results. Therefore, proving our initial hypotheses We believe that after this research, new insights and applications for cybersecurity using 1D-CNN will arise.

11. REFERENCES

[1] Kumari, K., Mrunalini, M. Detecting Denial of Service attacks using machine learning algorithms. J Big Data 9, 56 (2022). https://doi.org/10.1186/s40537-022-00616-0

[2] Xin, Y., Kong, L., Liu, Z., Chen, Y., Li, Y., Zhu, H., Gao, M., Hou, H. and Wang, C., 2018. Machine learning and deep learning methods for cybersecurity. *Ieee access*, *6*, pp.35365-35381.

[3] ITIC, IBM. (2020). Average cost per hour of enterprise server downtime worldwide in 2019. Statista. Statista Inc.. Accessed: November 13, 2023. https://www.statista.com/statistics/753938/worldwide-enterprise-server-hourly-downtime-cost/

[4] https://www.nationalcrimeagency.gov.uk/?view=article&id=243:ddos-attacks-are-illegal&catid=2#:~:text=If%20you%20conduct%20a%20DDoS,sentence%2C%20a%20fine%20or%20both.

[5] Shou, D., 2012. Ethical considerations of sharing data for cybersecurity research. In Financial Cryptography and Data Security: FC 2011 Workshops, RLCPS and WECSR 2011, Rodney Bay, St. Lucia, February 28-March 4, 2011, Revised Selected Papers 15 (pp. 169-177). Springer Berlin Heidelberg.

[6] Otoum, Y., Nayak, A. AS-IDS: Anomaly and Signature Based IDS for the Internet of Things. J Netw Syst Manage 29, 23 (2021). https://doi.org/10.1007/s10922-021-09589-6

[7] imiduk, D.M., Holm, E.A. & Niezgoda, S.R. Perspectives on the Impact of Machine Learning, Deep Learning, and Artificial Intelligence on Materials, Processes, and Structures Engineering. Integr Mater Manuf Innov 7, 157–172 (2018).

[8] Janiesch, C., Zschech, P. & Heinrich, K. Machine learning and deep learning. Electron Markets 31, 685–695 (2021). https://doi.org/10.1007/s12525-021-00475-2

[9] Feily, M., Shahrestani, A. and Ramadass, S., 2009, June. A survey of botnet and botnet detection. In 2009 Third International Conference on Emerging Security Information, Systems and Technologies (pp. 268-273). IEEE.

[10] Abu Rajab, M., Zarfoss, J., Monrose, F. and Terzis, A., 2006, October. A multifaceted approach to understanding the botnet phenomenon. In Proceedings of the 6th ACM SIGCOMM conference on Internet measurement (pp. 41-52).

[11]https://www.cloudflare.com/en-gb/application-services/products/bot-management/

[12] Douligeris, C. and Mitrokotsa, A., 2004. DDoS attacks and defense mechanisms: classification and state-of-the-art. Computer networks, 44(5), pp.643-666.

[13] Arshi, M., Nasreen, M.D. and Madhavi, K., 2020. A survey of DDoS attacks using machine learning techniques. In E3S Web of Conferences (Vol. 184, p. 01052). EDP Sciences.

[14] M. V. Kotpalliwar and R. Wajgi, "Classification of Attacks Using Support Vector Machine (SVM) on KDDCUP'99 IDS Database," 2015 Fifth International Conference on Communication Systems and Network Technologies, Gwalior, India, 2015, pp. 987-990, doi: 10.1109/CSNT.2015.185.

[15] M. S. Pervez and D. M. Farid, "Feature selection and intrusion classification in NSL-KDD cup 99 dataset employing SVMs," The 8th International Conference on Software, Knowledge, Information

Management and Applications (SKIMA 2014), Dhaka, Bangladesh, 2014, pp. 1-6, doi: 10.1109/SKIMA.2014.7083539.

[16] Shapoorifard, H. and Shamsinejad, P., 2017. Intrusion detection using a novel hybrid method incorporating an improved KNN. Int. J. Comput. Appl, 173(1), pp.5-9.

[17] I. Ramadhan, P. Sukarno and M. A. Nugroho, "Comparative Analysis of K-Nearest Neighbor and Decision Tree in Detecting Distributed Denial of Service," 2020 8th International Conference on Information and Communication Technology (ICoICT), Yogyakarta, Indonesia, 2020, pp. 1-4, doi: 10.1109/ICoICT49345.2020.9166380.

[18] Azad, C. and Jha, V.K., 2015. Genetic algorithm to solve the problem of small disjunct in the decision tree based intrusion detection system. International Journal of Computer Network and Information Security, 7(8), pp.56-71.

[19] Fouladi, R.F., Kayatas, C.E. and Anarim, E., 2016, June. Frequency based DDoS attack detection approach using naive Bayes classification. In 2016 39th International Conference on Telecommunications and Signal Processing (TSP) (pp. 104-107). IEEE.

[20] Rizvi, F., Sharma, R., Sharma, N., Rakhra, M., Aledaily, A.N., Viriyasitavat, W., Yadav, K., Dhiman, G. and Kaur, A., 2024. An evolutionary KNN model for DDoS assault detection using genetic algorithm based optimization. Multimedia Tools and Applications, pp.1-24.

[21] Van, N.T., Thinh, T.N., & Sach, L.T. (2017). An anomaly-based network intrusion detection system using Deep learning. 2017 International Conference on System Science and Engineering (ICSSE), 210-214.

[22] X. Yuan, C. Li and X. Li, "DeepDefense: Identifying DDoS Attack via Deep Learning," 2017 IEEE International Conference on Smart Computing (SMARTCOMP), Hong Kong, China, 2017, pp. 1-8, doi: 10.1109/SMARTCOMP.2017.7946998.

[23] Zhao, G., Zhang, C. and Zheng, L., 2017, July. Intrusion detection using deep belief network and probabilistic neural network. In 2017 IEEE international conference on computational science and engineering (CSE) and IEEE international conference on embedded and ubiquitous computing (EUC) (Vol. 1, pp. 639-642). IEEE.

[24] Yin, C., Zhu, Y., Fei, J. and He, X., 2017. A deep learning approach for intrusion detection using recurrent neural networks. Ieee Access, 5, pp.21954-21961.

[25] Staudemeyer, R.C., 2015. Applying long short-term memory recurrent neural networks to intrusion detection. South African Computer Journal, 56(1), pp.136-154.

[26] Krishnan, R.B. and Raajan, N.R., 2016. An intellectual intrusion detection system model for attacks classification using RNN. Int. J. Pharm. Technol, 8(4), pp.23157-23164.

[27] Kumar, P., Kushawaha, C., Yadav, D. and Kota, S., 2024, March. Exploring the Potential of Artificial Intelligence Model to Detect Distributed Denial of Service Attacks. In Proceedings of the 1st International Conference on Artificial Intelligence, Communication, IoT, Data Engineering and Security, IACIDS 2023, 23-25 November 2023, Lavasa, Pune, India.

[28] Kim, J., Kim, J., Thu, H.L.T. and Kim, H., 2016, February. Long short term memory recurrent neural network classifier for intrusion detection. In 2016 international conference on platform technology and service (PlatCon) (pp. 1-5). IEEE.

[29] Yu, Y., Long, J. and Cai, Z., 2017. Network intrusion detection through stacking dilated convolutional autoencoders. Security and Communication Networks, 2017.

[30] Wang, W., Zhu, M., Wang, J., Zeng, X. and Yang, Z., 2017, July. End-to-end encrypted traffic classification with one-dimensional convolution neural networks. In 2017 IEEE international conference on intelligence and security informatics (ISI) (pp. 43-48). IEEE.

[30] Jacq, A.D., Orsini, M., Dulac-Arnold, G., Pietquin, O., Geist, M. and Bachem, O., 2023, July. On the importance of data collection for training general goal-reaching policies. In Sixteenth European Workshop on Reinforcement Learning.

[31] Wazirali, R. An Improved Intrusion Detection System Based on KNN Hyperparameter Tuning and Cross-Validation. Arab J Sci Eng 45, 10859–10873 (2020). https://doi.org/10.1007/s13369-020-04907-7

[32] M. Malik and Y. Singh, "International Journal of Computer Science and Mobile Computing A Review: DoS and DDoS Attacks", International Journal of Computer Science and Mobile Computing, vol. 4, pp. 260-265, 2015

[33] Suwanda, R., Syahputra, Z. and Zamzami, E.M., 2020, June. Analysis of euclidean distance and manhattan distance in the K-means algorithm for variations number of centroid K. In Journal of Physics: Conference Series (Vol. 1566, No. 1, p. 012058). IOP Publishing.

[34] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., Liu, T., Wang, X., Wang, G., Cai, J. and Chen, T., 2018. Recent advances in convolutional neural networks. Pattern recognition, 77, pp.354-377.

[35] LeCun, Y., Bengio, Y. & Hinton, G. Deep learning. Nature 521, 436–444 (2015). https://doi.org/10.1038/nature14539

[36] Rigatti, S.J., 2017. Random forest. Journal of Insurance Medicine, 47(1), pp.31-39.

12. APPENDIX

REQUIREMENTS

To run these models, the only requirements are to have the datasets downloaded, which have been provided in the zip file, in our machine and redirect the path to the current directory. In this case since the notebooks have been developed using Google Colab.

df = pd.read_csv("/content/drive/MyDrive/ddos/CICDDoS2019/cicddos2019_dataset.csv")

The previous line should be modified to the directory where we are storing the dataset.

All the libraries have been implemented at the beginning of each notebook making it easy to run the models.

MEETINGS WITH SUPPERVISOR

Here are the notes of the meeting with Dr Imran Khan throughout the development of this research.

1st meeting: We discussed the topic, structure and main points of the project. We decided to focus on the different ML and DL models in DDoS attacks detection, Imran suggested to start making an initial research on how are these models implemented for IDS and start comparing the models and making a table to have a easier way to read the data.

2nd meeting: After acquiring more knowledge on the topic and understanding how the attacks worked. We discussed the outline for the structure of the report and the results achieved after comparing numerous research paper side by side. Then, Imran pointed me into the direction of comparing the least studied method with one that is widely implemented.

3rd meeting: After continuing with the research analyzing the models developed by other scientists, we concluded that the most suitable models for this task were KNN and 1D-CNN. Here we discussed how we will perform the models. At this point we decided to use the NSL-KDD and CICDDoS2019 datasets.

4th meeting: At the end of our period to complete the project, we had another meeting to discuss the results on the models up to that point and address some issues with the report writing.