Department of Computer Science

Submitted in part fulfilment for the degree of BEng.

Distributed Denial of Service attack detection using Recurrent and Convolutional Neural Networks

Jake Fawcett

Supervisor: Kofi Appiah

Acknowledgements

Thank you to my supervisor Kofi, for all the support throughout the project.

Contents

List of Figures

List of Tables

Executive Summary

The frequency and size of Distributed Denial of Service (DDoS) attacks has been steadily growing for several years, becoming one of the biggest threats to online services. In 2020, cybersecurity firm Nustar reported a 151% increase in DDoS attacks [1] and Amazon Web Services experienced a record-breaking attack, at 2.3 Terabytes per second it was 44% larger than any previously detected event [2].

The key challenge in DDoS prevention is the continued development of new attacks, with new weaknesses and attack vectors constantly being developed. Traditional methods, such as signature detection, are quickly made obsolete by zero-day attacks that have not been seen before. Therefore, new methods of detecting DDoS attacks are required, focusing on detecting anomalies in network traffic rather than specific signatures, allowing automatic attack detection. Machine learning is an effective technique for classifying data, particularly Deep Neural Networks (DNNs) such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), and could be used for DDoS detection.

Convolutional Neural Networks (CNNs) have been proven to be excellent at establishing patterns from complex data, such as; classifying heartbeats from noisy electrocardiogram data[3], accurately modelling the responses of mammalian retina [4] and classification of lung abnormalities [5]. This makes them an excellent candidate for DDoS detection due to the complexity of the wide range of attack vectors and features. Recurrent Neural networks (RNNs) perform well on time-series data due to the feedback (memory) component, with Long-Short Term Memory (LSTMs) having been shown to perform well on complex time series data [6] and have been proven effective at classifying DDoS attacks by Yuan et al. [7].

While there has been a lot of research into NNs ability against known DDoS attacks, there has been little work into the detection of unknown, zero-day attacks. Therefore, as they have been shown to be two of the highest performing architectures, a CNN and LSTM were chosen. To test these networks, it was essential to use a new dataset containing the latest attacks, that also contained a wide range of attack allowing known and unknown attacks to be tested against.

CIC-DDoS2019 [8] has been chosen as it contains "the most up-to-date common DDoS attacks" and contains a large number of samples, ideal for training and testing a neural networks. This dataset is ideal as it has been generated to represent real traffic and therefore contains no user data, so no ethical issues will arise from handling data. The Syn, LDAP and NetBIOS attacks were chosen for training, providing a good variety of attack types, while UDPLag was chosen to evaluate the model, as it is a different type of attack and would mimic a zero-day attack.

To effectively test the neural networks, they were benchmarked against common machine-learning-based techniques. Sarker [9], demonstrated K-Nearest Neighbors (KNN), Support Vector Machines (SVMs) and Decision Trees (DT) to be effective benchmarks, so exploratory testing was performed to ensure good parameters were chosen, and they were used to compare neural network performance.

To measure and compare the performance of the models, 4 metrics were measured; the number of true positives, true negatives, false positives and false negatives. This allowed for accuracy, precision, recall and f-score to be calculated to assess performance. The metric recall was prioritised, to ensure that attack packets are detected correctly so disruption to users can be prevented. Secondary to this shall be the precision, to ensure that benign traffic isn't misclasified and genuine users blocked.

Both models are very effective, achieving high precision, accuracy and recall, outperforming other machine-learning-based techniques in most instances. Both were shown to perform exceptionally against known attacks, outperforming KNN, SVM and DT models, achieving an accuracy of over 99.98%. Expectedly, performance was worse against unknown attacks, however, an accuracy of over 99.94% was achieved for the CNN and over 99.96% for the LSTM. This demonstrates that they were both able to effectively meet the goal of correctly classifying attack packets, having a high true positive rate (as high as 100%) to ensure DDoS attack prevention, and low false-positive rate (as low as 0.001%) which ensures genuine users are not disrupted. Therefore, both CNNs and LSTMs have both been proven to be a suitable method of detecting existing and zero-day attacks.

There are no legal, social, ethical, professional or commercial issues within this paper or the experiment carried out. No data contained any private users information and the dataset has been used and cited under its licence. All software, languages and packages used are open source or free to use.

1. Introduction

1.1 Background

Denial of Service (DoS) attacks attempt to prevent genuine users from accessing a resource, typically a website, by disrupting the machine hosting it. This is often accomplished by flooding the machine with a huge amount of requests in as attempt to use all of the networks capacity or overload the system. This is similar to when a 'Flash Crowd' of genuine users might rush online to book tickets to a popular concert, but due to the sudden increase in traffic, the server cannot handle the sheer number of users, which can cause it to slow, deny access or crash. However, DoS attacks are carried out with malicious intent, typically for activism or blackmail.

A Distributed Denial of Service (DDoS) attack is a coordinated attack using a distributed set of resources to carry out a DoS attack, allowing larger and more severe attacks. There are a variety of different methods of DDoS attack, but all types focus on consuming all of the victims' network or computing resources, thus preventing legitimate users from accessing the resource. This is often carried out by compromising a range of systems, from servers to personal computers. These 'Secondary Victims' are infected using other attacks, such as; worms, trojan horses, backdoors, etc [10]. These attacks allow an attacker to execute code, which can be very difficult to detect, from compromised machines to form an attack against the chosen victim.

The use of Secondary Victims allows DDoS attacks to be much more difficult to detect as they allow the attacker to remain anonymous while attacking from a wide range of sources that must be independently inspected and classified. There are also many other difficulties, such as the need to filter a mix of genuine and malicious traffic, high loads being caused by a 'Flash Crowd' of public attention or difficult to detect low rate attacks that exploit weaknesses while using a small amount of traffic.

DDoS attacks are very widespread and prevalent, affecting 11% of large UK firms in 2019 [11]. This is reportedly only getting worse, with security company Nustar reporting a 151% increase in DDoS attacks in 2020 compared to 2019 [1] and Amazon Web Services reporting a record attack at

2.3 TB per second, approximately 44% larger than any previously detected event [2]. This demonstes how large of a problem DDoS attacks can be, and will continue to be in future, especially with the acceleration of online services due to the COVID-19 pandemic.

While these attacks are a major threat to business, they also affect services people use everyday: in 2016 a Domain Name Service (DNS) provider Dyn faced a huge attack [12] which brought down over 80 of its customers including; Twitter, Reddit, Spotify, Netflix and Amazon. This was carried out by a large group of 100,000 compromised smart devices consisting of radios, TVs, printers which were programmed to send a large amount of traffic to Dyn, bringing down services and prevent users from accessing them.

A potential defense against DDoS attacks is through storing a database of all existing records of attacks so that signatures of attacks can be detected. However, this has one key flaw: it is unable to prevent new attacks which aren't already known to the database. These attacks, which often exploit newfound vulnerabilities, are known as zero-day attacks and can be incredibly difficult to defend against using traditional methods.

However, with the advancement of Machine Learning (ML) and Neural Networks (NNs), it is becoming possible to prevent some of these attacks. NNs are ideal for this task, as they are modelled on the behaviour of the brain, constructed from layers of 'neurons' allowing the network to learn relationships from data. This makes them ideal for anomaly detection, where they are trained to pick out anomalies in features of network traffic data that indicate a packet is malicious, allowing them to be identified and stopped.

More recently, Deep Neural Networks (DNNs), which comprise multiple hidden layers of neurons, have been essential to the modern detection of DDoS attacks due as they are capable of navigating complex network traffic, while distinguishing packets and attacks. Networks such as Convolutional Neural Networks (CNNs), can be very effective in extracting the key features [13] from data, which could then be used to detect anomalies. While Recurrent Neural Networks (RNNs) have been proven to be very effective at predicting time series data, even in complex scenarios [6], using a form of memory to learn relationships over time.

These networks are very effective, with Yuan [7] able to achieve an accuracy of up to 98.4% with RNNs on the training dataset. However, there has been little research and testing of NNs against zero-day attacks.

1.2 Structure

In section 2, a literature review shall explore and analyse previous work relating to DDoS attacks and Neural Networks. This will be followed by the methodology, in section 3, which will explain the approach taken to the problem of DDoS detection, before going onto section 4, where the implementation of neural networks shall be discussed. Finally, the results of the tests on these networks will be reviewed and evaulated in section 5, determining the success of the project. Concluding remarks and a discussion of future work can be found in section 6.

2. Literature Review

2.1 Introduction

In this section, a range of literature covering Neural Networks and DDoS attacks shall be presented and discussed, to provide a good background for the project going forward. Potential architectures will be investigated, and takeaways from previous literature, that can be built on, will be discussed.

2.2 Related Work

2.2.1 Neural Networks

Deep Neural Networks are a powerful tool in the prevention of DDoS attacks, as they are formed from stacking multiple layers of neurons to find complex relationships. They are particularly suitable to the detection of DDoS attacks, which have very varied signatures. Provided with a large set of data with many dimensions, a DNN can learn to detect anomalies in network traffic, and so detect both known and unknown attacks.

Convolutional Neural Networks (CNNs) are a variation of multilayer perceptrons designed to need minimal preprocessing. CNNs learn traditionally hand-engineered filters, which extract features from the data, removing the need for prior knowledge. Typically, CNNs are applied to images, such as in Figure 1.1, but have been shown to work on other problems such as DDoS [14], providing features are reshaped into a 2D 'image like' array.

CNNs are built of 3 key layers; Convolution layers, Pooling layers and a Fully-Connected layer. The key component is the Convolution filters, which define a small matrix h by w smaller than the input (a convolution kernel). This is then overlayed on top of the input matrix in all possible locations, recording the sum of elementwise products between the input and the kernel. Convolution layers compute the convolution of the output of a previous layer using several filters, or kernels, to extract features. The layer is specified by a set of kernels in the same way a Multi-Layer

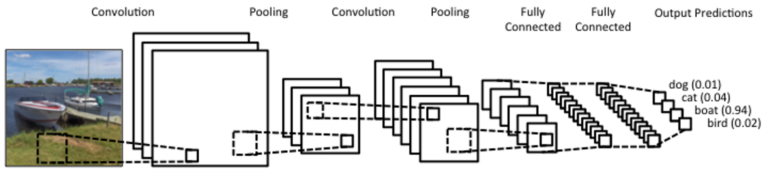

Figure 1.1: The application of CNN to an Image.

Perceptron (MLP) is specified by the weights for each neuron. The output is then, typically, fed into a pooling layer which effectively downsamples the output by taking values e.g. 2x2 chunks of the input matrix and aggregating them into a single value. Multiple different pooling operations can be used depending on what is required: max pooling is most commonly used which takes the maximum value from the chunk but there is also; average pooling, weighted average, L2 norm, etc. Finally, after a series of convolution and pooling layers have been passed through, the output is flattened and fed into a fully connected network, which commonly uses the softmax function to produce a vector of probabilities.

Recurrent Neural Networks (RNNs) are a variation of NN with a feedback connection, which might be; from the output back to inputs, from the hidden neuron output back to hidden neuron inputs or any variation of these. The basic model is similar to an MLP, but with the addition of feedback between neurons at each timestep, this changes the properties of the network by extending the model to use information from previous time steps. This is particularly useful as DDoS attacks can be viewed as a series of packets, therefore previous data is useful in detecting if a DDoS attack is currently occurring, hopefully allowing for a lower false-positive rate during non-attack periods and a higher true positive rate during attacks.

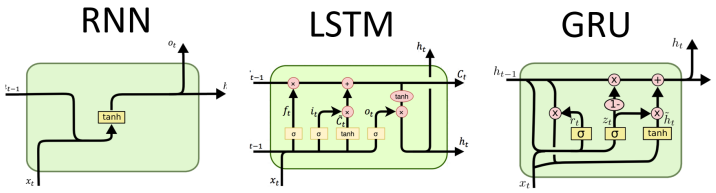

Figure 1.2: RNN Architectures [15].

Three different RNN architectures may be considered: Simple RNN, Long-Short Term Memory (LSTM) and Gated Recurrent Unit (GRU). The architecture of these networks can be seen in Figure 1.2. Simple RNNs can be powerful, but the simple structure means that information fades

away quickly over time which may reduce the effect of the advantages an RNN provides, unlike GRU and LSTM can retain information over a longer period due to the use of a series of gates. Both architectures take different approaches: GRU uses a simple gate structure allowing it to perform faster, while LSTMs use a more complex structure that may perform better. This means that rather than information fading away over time, it is retained, so long-term patterns in data can be learnt from.

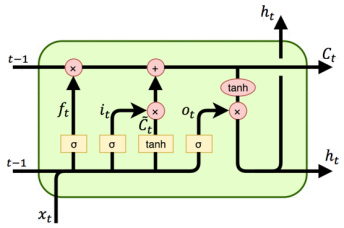

Figure 1.3: Example LSTM Architecture [15].

The key to the LSTM architecture is the cell state [15], which the LSTM adds and removes information from using a forget gate and an update cell state gate. There is also an input gate and output gate. The forget gate is a sigmoid layer, taking the input x_t and the previous output h_{h-1} and outputting a number between 0 and 1 indicating how much information to keep. The output is multiplied by the previous cell state, which allows the LSTM to forget some information.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f$$

The next is the input gate: this is another sigmoid layer that decides the set of values to update, i_t. This is combined with a tanh layer, which creates the set of new values C_t. After the forget operation has been applied, the output of the input gate and tanh layer are multiplied and then added to the cell state to produce the new cell state.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i)$$
$$C_t = tanh(W_c.[h_{t-1}, x_t] + b_C)$$

Finally, we have the output gate. The current input and previous output are put through a sigmoid layer and multiplied by the tanh of the current cell sate to produce the output, h_t.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o)$$
$$h_t = o_t * tanh(C_t)$$

The above architecture is a common variation of LSTM, but there are different variations, such as Gers' [16] that allows the gate forget, input and output sigmoid gate layers to use cell state.

2.2.2 DDoS Detection

One approach to detecting attacks is to monitor resource utilization on the victim's system, rather than network traffic. This allows ineffective attacks and normal operation to be ignored, therefore resulting in no false positives during this period. However, it can add impurities to the training data and inherently misses low rate attacks. Seufert [17] tested this out, extracting features from multiple levels: IP packet headers (OSI Level 3), TCP headers (OSI Level 4) and application layer data i.e. HTTP requests (OSI Level 7). During normal operation, samples of network traffic are taken and added to the baseline profile. When an attack is detected the attack packets are sampled and used with the baseline profile to train a three-layer feed-forward Artificial Neural Network (ANN) with a sigmoid transfer function. An ANN is used as its learning method is robust to errors in the training data, such as impurities due to ineffective attacks. The classifier is then used to directly filter traffic or generate rules for an existing filtering framework to block traffic from malicious sources. The results were very succesful, performing well on both sample data and using a network simulation. It was suggested that the solution could be extended to use multiple algorithms.

Sarker et al. [9] took a different approach to Seufert, producing IntruDTree, a machine learning based detection model. First carrying out feature encoding, feature scaling and then feature selection through the calculation of an importance score. This importance score is based on the relationship between the feature and the model output and uses the "Gini Index", an importance score. The top n features are taken, and a tree model is constructed, with each node being the feature with the next highest importance or a terminal node (either anomaly or normal). This allowed IntruDTree to outperform Naive Bayes, Logistic Regression, K-Nearest Neighbour and Support Vector Machines, although this is only for one dataset, from which IntruDTree's features were picked. Therefore, it may perform poorly against new attacks or attacks outside the dataset where a different set of features may be of greater importance. This highlights the need to test against a range of unknown attacks, to determine its performance.

Shon [18] also carries out extensive Feature selection by using a Genetic algorithm with three operators: selection, crossover, and mutation. Selection is based on a probabilistic survival of the fittest mechanism based on a problem-specific evaluation. However, random mutations can occur, and we must choose which individual matches our desired conditions manually. Shon's paper is also interesting as it uses Support Vector Machines to carry out supervised and unsupervised learning, which can be trained (albeit at a lower accuracy) to work for non-labelled datasets, which is a source other

methods may be missing out on.

Alternative ML methods have also been found to be effective. Feng [19], focusing on Level 7 DDoS attacks by assuming that L7 Attacks are stateful, as TCP Connections and HTTP requests are stateful. This assumption allows reinforcement learning to be used, a method of training by returning a reward based on which of 6 mitigation actions are taken (a defensive action signals a malicious packet, a passive action a benign packet). This implementation is useful as it allows a focus on minimising the rate of false positives when the occupation is low and maximise true positives when the occupation is high and was proved to be very effective during testing. However, this could be developed further as 6 mitigation actions are reduced to Defensive or Normal, removing the direction indication of the effectiveness of each e.g. a False Positive is particularly harmful if we 'Drop all Similar' compared to just calling 'Drop Message'.

Rather than using machine learning, Bhuyan [20] uses the empirical evaluation of information metrics by constructing a model based on entropy metrics and various probability distributions for legitimate network traffic. David [21] takes this a step further by carrying out flow aggregation and calculating fast entropy to greatly improve attack detection. David's also uses an adaptive threshold value based on the entropy, to lower the false positive rate. However, it works on the assumption that entropy drops dramatically as one flow is dominating, which would not be the case for a botnet as there are multiple flows. Similarly, to these, Chen [22] takes a statistical approach, considering distributions of SYN and ACK packets, and carrying out T-tests. Both of these methods have low overheads which can be very important for large datasets and highlight the importance of some specific features, however, I also think they could be limited in their detection as they only take into account 3 and 2 features respectively and only consider 1 dataset.

An early paper by Specht [23] covers the taxonomies, tools, and countermeasures for DDoS attacks, discussing how to prevent secondary victims (those whos' devices were hijacked and turned into 'zombies') and deflecting attacks through honeypots, which consist of data that appears to be legitimate but contains no value to attackers. Similar to Specht, Zargar [10] may be particularly useful as it reviews how a huge range of more modern defence and detection mechanisms work as well as their flaws, which I may be able to learn from. It also brings up some key points, such as; the trade-off of wasted resources in detecting an attack at the victim (greater resource usage but higher accuracy) and lack of consistent metrics to evaluate defence mechanisms. It could be particularly useful to find consistent metrics so that I can evaluate the effectiveness of my solution, and potentially different models in my solution, against others accurately.

Guo [24] considers DoS attacks on critical internet infrastructure in the Domain Name System (DNS). This involves either overloading a DNS server, causing requests to be blocked, or saturating a victim's bandwidth via an amplified DNS response. This paper highlights the effectiveness of DoS attacks to disrupt key pieces of internet infrastructure (not just a singular victim) and the effectiveness of amplification attacks in improving the effectiveness of DoS and DDoS attacks with limited resources.

Rather than preventing DDoS at the victims side, they can also be prevented at the source, as network resources are saved by stopping DDoS attacks earlier in the network. Mirkovics' [25] implementation of DDoS prevention at the source, installed a system at a source router which monitors the behaviour of peers, rather than investigating packets, and compares this to model traffic. This allows communication difficulties to be spotted, such as a reduction in response packets or longer inter-arrival times, so this selection of traffic can be rate limited. However, this could not be tested in a real attack scenario due to the memory and processing limitations of the kernel and software at the time. Despite this, this is an interesting look at a different approach of preventing DDoS, away from the victim.

Xu [26] introduces the problem of bottlenecks in a centralised system, caused by to having to download and process the traffic from one machine. Most DDoS traffic IP addresses are new to the victim whereas, in a flash crowd they are normally known. An IP database is dynamically updated after an IP is observed m times or has packet numbers greater than n. Similarly, Berra [27] uses multiple nodes spread across the network which detect changes in the amount of traffic towards a given node, using the CUSUM algorithm. It also improves on Xu [26] by using a Naive Bayes classifier to adapt its thresholds for detection and trust of information from other nodes. However, by spreading nodes across a network, distributed learning increases the complexity of training and adds a communication and trust burden. Xu only analyse one feature, potentially weakening their solution to many DDoS methods, such as slow Loris attacks (due to reliance on large changes in traffic) or amplification attacks, which I must ensure my solution can detect to ensure its robustness.

Given new datasets are essential, as suggested by Yuan [7], Elsayeds' [28] use of a new dataset 'CICDDoS2019' may be useful in investigating the features of new DDoS attacks. This starts with an Autoencoder which is fed the features and outputs a feature vector approximately equal to the original input data but significantly improves anomaly detection. This is built with multiple simple RNN layers and is first trained on unlabelled data, allowing it to learn hierarchical features, then is fine-tuned with labelled samples to optimize the network. The authors suggest this could be expanded by classifying attacks separately rather than in a binary classification model, which may be a useful topic for this paper.

RNNs have been shown to be effective by Yuan [7], by using a series of time windows, up to 100 packets, rather than looking at packets in sequence. This allows the network to analyse recent historical data which could be important as DDoS attacks take place over time. Yuan also compared a range of models; LSTM, CNNLSTM, GRU and 3LSTM and tested these against conventional ML methods, such as random forest, which they greatly outperformed. Yuan also suggested a newer dataset is required to account for the features of new DDoS attacks, which is essential to this paper.

Shaaban [14] focuses on using Convolutional Neural Networks, which the authors believe to be the first use of CNN separately in Classification and Detection. CNNs are applied by extracting parameters and converting them into n*n matrices to be considered as a 2D image. During testing against other classifiers, such as SVM, KNN, Decision Tree and NN, there appeared to be a relatively significant improvement in the accuracy of classification. Due to this possibly being one of the first tests of using independent CNNs on DDoS, it is essential to test this further on different data, and compare results against other proposed deep learning methods.

2.3 Aims and Objectives

This report aims to investigate the application of DNNs to detect DDoS attacks. Other methods such as reinforcement learning and empirical evaluation have been shown to perform well, but given the strength of neural networks in pattern recognition and modelling complex relationships they are ideal, especially with the evidence for their performance on DDoS data [7] [14].

As previous literature has shown, carrying out signature detection by applying trained neural networks is a good way to measure the performance of known attacks. However, little testing has been carried out on the effectiveness of models against zero-day attacks, therefore to build on previous work models shall be tested against unknown attacks that they haven't been trained on, simulating a zero-day attack.

This will allow the effectiveness of DNNs for anomaly detection to be shown, rather than just signature detection. For this, a robust and up to date dataset will be required, with a wide range of attacks so that the network can be effectively trained and evaluated. It will be essential to use some simple models to test the constructed networks against, to benchmark performance and measure effectiveness.

3. Methodology

3.1 Introduction

This chapter shall cover the approach taken within the paper, touching on the choice of tools, dataset, neural networks and any ethical implications. Focusing on the decisions made with regards to the direction of the report, based on the literature reviewed.

3.2 Review of Tools

To construct the neural networks, Python was noted as a suitable language due to the wide range of free-to-use and open-source data science packages. MATLAB was also considered, but Python was ultimately chosen due to the ease of use and higher performance. Within Python, the Pandas, Numpy, Sci-kit Learn, Keras and Matplotlib packages were used. The choice of language, libraries and tools was made to ensure there were no ethical issues and everything was easily reproducible, therefore all are open-source or free to use.

All code was written within Visual Studio Code on Windows or the open-source variant Code OSS on ubuntu. Google Colaboratory was considered due to the ability to leverage Googles' cloud hardware, however, RAM limitations meant a large dataset couldn't be used, so it wasn't suitable.

Due to the ongoing effects of COVID-19, computer laboratories were unavailable, requiring some restrictions to be imposed due to hardware limitations. The size of the dataset had to be limited and the parameters reduced to stay within hardware limitations and ensure reasonable runtimes.

3.3 Dataset

In 2017, Yuan [7] stated that it is "urgent to create a new dataset to meet the new challenges in DDoS identification". This is due to new features found in more recent DDoS attacks, which may prove essential to allowing trained models to detect the latest attacks.

Another concern is the size of the dataset, as the intention to test networks against multiple known and known attacks, which combined with the complexity of this problem, means that a large dataset with a wide range of attacks will be required so that any models can be trained, tested and evaluated effectively.

Without a suitable existing dataset, this would bring a new set of challenges, as a new dataset would have to be created. This would cause a wide range of ethical and professional issues as user traffic would have to be monitored and an attack carried out. This would violate users privacy, as it may not be possible to allow users to opt-out and traffic may contain large amounts of private information, such as; sites visited or IP addresses. There would also be issues with carrying out an attack on a public network, which would not be professional or ethical due to the impact it would have on other users.

An alternative to this would be to use an older dataset such as ISCX-IDS2012 [29] as used by Yuan [7]. This is a high-quality dataset, that has thoroughly considered ethical issues and has a large amount of data covering malicious and benign traffic. However, this dataset only consists of 4 attacks and it was constructed in 2012, so does not include any recent attacks and therefore is unlikely to have a wide enough range or the latest features for newer attacks to be detected effectively.

For this reason, the CIC-DDoS2019 [8] dataset would be an ideal candidate, as it contains "the most up-to-date common DDoS attacks", see Figure 1.4, which have been modelled to resemble real-world data. This is much newer than similar datasets, such as the 2012 TUIDS DDoS dataset [30] or the 2007 CAIDA DDoS Attack dataset [31]. The dataset is free to use, redistribute, republish, and mirror, as long as cited. CIC-DDoS also benefits from the fact 13 different attacks were recorded, which would allow the networks to be trained, tested and evaluated on a range of attacks and subsequently evaluated against some attacks not used in training or testing. It is also a good candidate, as the dataset has been generated to represent real traffic, meaning it contains no real users data, therefore, no ethical issues are arising from the collection or handling of user data.

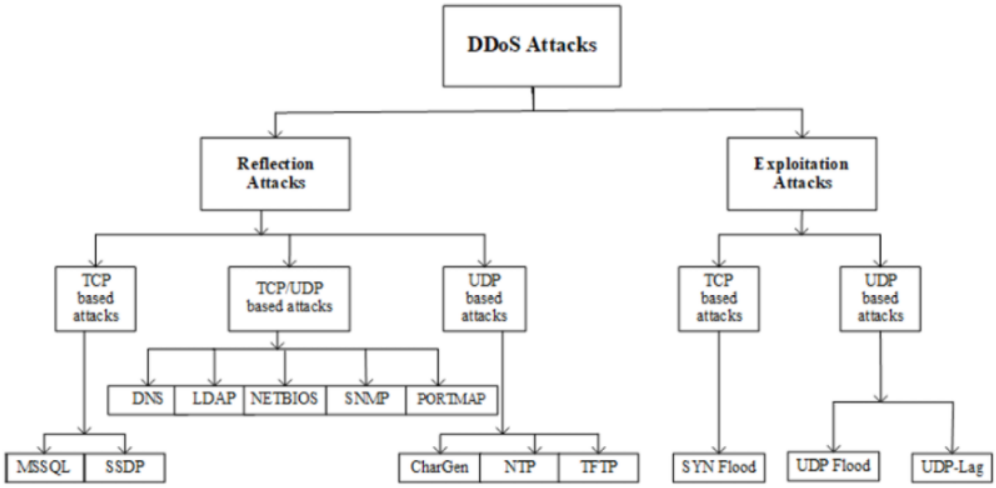

Figure 1.4: Attacks covered in the CIC-DDoS2019 dataset [8].

Due to resource limitations, it was decided that a reduced selection of CIC-DDoS2019 attacks would be used to stay within memory and performance boundaries. Therefore, Syn, LDAP and NetBIOS attacks were chosen for training as they provide variety in attack types, and UDPLag for testing, so to mimic an unknown zero-day attack. Figure 1.5 demonstrates how the training data has similar magnitudes of each attack type, which will help prevent overfitting to one dataset.

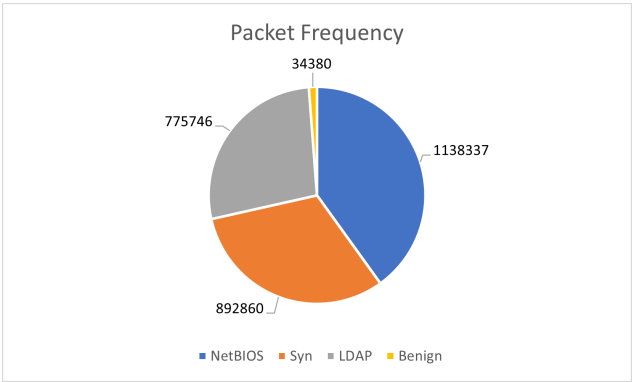

Figure 1.5: Testing data packet type frequency.

3.4 Choice of Neural Networks

Patterns in network traffic can be used to detect malicious packets through anomaly detection, however, they are typically very complex. To cope with this complexity, DNNs, which are ANNs with multiple hidden layers, are required. The simplest DNN is a Multilayer Perceptron (MLP) which is a series of 3 or more layers of neurons, but there are also more specialised and complex constructs like Recurrent Neural Networks (RNNs) and Convolutional Neural Networks (CNNs).

There has been little research into the use of CNNs to detect DDoS attacks, which are usually used to classify images, but they are very effective at establishing patterns within complex data, such as; classifying heartbeats from noisy electrocardiogram data[3], accurately modelling the responses of mammalian retina [4] and classification of lung abnormalities [5]. This makes CNNs a great candidate for DDoS detection, especially due to their ability to extract features without the need for manual feature engineering, which is difficult due to the variety of DDoS attacks and scale of packet fields required to detect them.

Shaaban [14] found CNNs to be extremely good at detecting DDoS attacks, with an accuracy of 99% across 2 network traffic datasets. This was a marked improvement on the other classifiers tested; K-Nearest Neighbors, Support Vector Machines and Decision Trees. Therefore, it is important to build on this research and test the performance of CNNs against the robust CIC-DDoS2019 dataset and against known and unknown attacks. However, as CNN's are prone to overfitting, it is essential to include measures like dropout layers to prevent this, or performance against unknown data will be inhibited.

Since packet data is sequential, an RNN may also be a suitable architecture due to the high performance on time-series data. RNNs have been shown to perform well on complex time series data [6] and are effective at classifying DDoS attacks by Yuan et al. [7]. There are multiple types of RNN to consider, but primarily Gated Recurrent Units (GRU) and Long-Short Term Memory (LSTM) networks are used, as the use of gates means they can have a long term memory. Yuan [7] found that LSTM architectures performed better than GRU, with an accuracy of around 98%, and LSTMs have been shown to increase both performance and speed [32] compared with other RNN architectures.

This demonstrates the high-performance of CNNs and RNNs, particularly LSTMs, against both complex datasets and DDoS traffic. Therefore, they will both be suitable candidates for testing, to determine performance against the latest attacks and also against a simulated zero-day attack. This will hopefully demonstrate their performance against unknown attacks and add to evidence of their suitability in detecting DDoS attacks.

3.5 Summary

In this section, Python has been chosen for neural network construction, along with open-source or free-to-use data science libraries,such as Keras and SciKit Learn. The CICDDoS-2019 dataset will be used as it consists of a wide range of simulated attacks so no ethical issues arise from handling data. This dataset will be used to train, test and evaluate both a CNN and RNN against known and unknown attacks, due to their previously proven performance to test their effectiveness against new and zero-day attacks.

4. Design and Implementation

4.1 Introduction

This chapter shall contain the steps taken in the design and implementation of an RNN and CNN, alongside any data-processing carried out, and any information gained from exploratory testing.

4.2 Data Processing

The first stage of data processing is cleaning up the data. Initially, any rows with Null and NaN values are removed, but can also be replaced with an average value but this was deemed unnecessary due to the large size of the dataset and the small number of null values. Following this, three columns were removed from the data; 'Unnamed: 0' was removed as it is just an identifier, 'Flow ID' as it is a combination of other fields so is redundant, and Timestamp as the data is provided in sequence, so this is not required. These were chosen to be removed, as they were all deemed unnecessary and may add complexity and reduce performance.

An advantage of using a CNN is that, due to pooling and convolution, they carry out feature extraction, therefore this doesn't need to be done manually. However, this is not the case for RNNs, as typically feature extraction may be carried out manually, however, a feature extraction layer shall be added to the RNN to keep the dataset consistent for a fairer test. This ensures that no human bias is added or performance penalty due to incorrect feature extraction, although this will influence the performance of the RNN so needs to be thoroughly tested. The final list of fields from the CIC-DDoS2019 dataset that will be used can be seen in table 1.1.

Field	Field Type	Field	Field Type
Source IP	object	Max Packet Length	float64
Source Port	int64	Packet Length Mean	float64
Destination IP	object	Packet Length Std	float64
Destination Port	int64	Packet Length Variance	float64
Protocol	int64	FIN Flag Count	int64
Flow Duration	int64	SYN Flag Count	int64
Total Fwd Packets	int64	RST Flag Count	int64
Total Backward Packets	int64	PSH Flag Count	int64
Total Length of Fwd Packets	float64	ACK Flag Count	int64
Total Length of Bwd Packets	float64	URG Flag Count	int64
Fwd Packet Length Max	float64	CWE Flag Count	int64
Fwd Packet Length Min	float64	ECE Flag Count	int64
Fwd Packet Length Mean	float64	Down/Up Ratio	float64
Fwd Packet Length Std	float64	Average Packet Size	float64
Bwd Packet Length Max	float64	Avg Fwd Segment Size	float64
Bwd Packet Length Min	float64	Avg Bwd Segment Size	float64
Bwd Packet Length Mean	float64	Fwd Header Length.1	int64
Bwd Packet Length Std	float64	Fwd Avg Bytes/Bulk	int64
Flow Bytes/s	float64	Fwd Avg Packets/Bulk	int64
Flow Packets/s	float64	Fwd Avg Bulk Rate	int64
Flow IAT Mean	float64	Bwd Avg Bytes/Bulk	int64
Flow IAT Std	float64	Bwd Avg Packets/Bulk	int64
Flow IAT Max	float64	Bwd Avg Bulk Rate	int64
Flow IAT Min	float64	Subflow Fwd Packets	int64
Fwd IAT Total	float64	Subflow Fwd Bytes	int64
Fwd IAT Mean	float64	Subflow Bwd Packets	int64
Fwd IAT Std	float64	Subflow Bwd Bytes	int64
Fwd IAT Max	float64	Init_Win_bytes_forward	int64
Fwd IAT Min	float64	Init_Win_bytes_backward	int64
Bwd IAT Total	float64	act_data_pkt_fwd	int64
Bwd IAT Mean	float64	min_seg_size_forward	int64
Bwd IAT Std	float64	Active Mean	float64
Bwd IAT Max	float64	Active Std	float64
Bwd IAT Min	float64	Active Max	float64
Fwd PSH Flags	int64	Active Min	float64
Bwd PSH Flags	int64	Idle Mean	float64
Fwd URG Flags	int64	Idle Std	float64
Bwd URG Flags	int64	Idle Max	float64
Fwd Header Length	int64	Idle Min	float64
Bwd Header Length	int64	SimillarHTTP	object
Fwd Packets/s	float64	Inbound	int64
Bwd Packets/s	float64	Label	object
Min Packet Length	float64		

Table 1.1: Fields used from the CIC-DDoS2019.

Any categorical labels in the data also had to be encoded, therefore, all benign labels were replaced with 0s and any attack labels were replaced with 1s, as this is a binary classification problem and therefore the models can predict a 0 or 1 to indicate packet type. Encoding the labels so that the type of attack could be predicted was considered, but as the focus was on detecting if an attack was occurring and the fact zero-day attacks would not be able to be correctly classified, alongside the additional complexity, this was deemed out of scope. In addition, the SimilarHTTP field was converted to a binary 1 if a row contained a non-zero value, as non-zero values were always benign, or kept as a 0 if not.

The data shall also be scaled, using the Scikit learn standard scaler. This is due to the large differences in ranges of fields in the data, i.e. 'Idle Min' ranges between 0 and 10^8, while 'Total Fwd Packets' ranges from 1 to 4000. Without scaling the data, bias would be introduced as variables with larger ranges would outweigh variables with lower ranges, so this is essential for good performance.

Next, the data is split into testing and training data sets, with 72% used for training, 20% for testing and 8% for validation: this is a commonly used split and during initial testing was found to perform well so was not modified. Preprocessing for the unknown UDPLag data is carried out in the same way, but this data was not split as it shall all be used for testing and evaluation.

One issue with the dataset can be seen in Figure 1.5, as there is a very small amount of benign data in comparison to attack data, therefore the data will need to be sampled to obtain a more balanced dataset or there is a risk of poor performance detecting benign data. As undersampling would reduce the size of the dataset too much and oversampling could cause overfitting, it is sensible to perform a combination of both. Therefore the attack data shall be undersampled by a factor of 10, and the benign data shall be oversampled to match the frequency of attack data. This can be seen in Figure 1.6 which shows the new balanced dataset, with an equal number of attack and benign packets, making it suitable for training. It is important to note that the testing data was split before sampling, as this should remain as realistic to attack data as possible.

Finally, the data needs to be reshaped to suit each network; for the CNN, the 84 input fields need to be shaped into a 3D array of 'images' (samples, x-features, y-features), so willl be reshaped into a 12 by 7 array. While the RNN data needs to be shaped into a 3D array of time series (timesteps, samples, features), so will be reshaped to a 1 by 84 array.

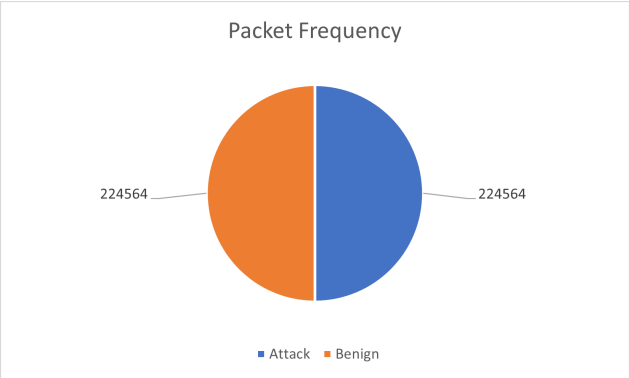

Figure 1.6: Sampled training data packet type frequency.

4.3 Construction of Neural Networks

As this is a binary classification problem, the loss function for all networks shall be binary cross-entropy, with the final layer of both networks using a sigmoid activation function to predict if the packet is malicious (1) or benign (0). The Adam optimizer has been chosen, which is a combination of RMSprop and Stochastic Gradient Descent (SGD) with momentum, due to its speed, accuracy and high performance [33]. Exploratory testing was carried out to test the Adam optimizer, and it was shown to perform better than SGD, RMSProp, Adamax and Adagrad.

Each network was set to run for 100 epochs, however, a callback was implemented so that training would stop if either loss or accuracy did not improve within 3 epochs to try to ensure the networks did not overfit. Alongside this, a batch size of 20 was chosen, while other parameters were tweaked on a network by network basis.

The initial design of the RNN shall be; a 60 unit relu dense layer (for feature extraction), 40 unit simple RNN, LSTM or GRU layer with recurrent dropout, a dense relu layer with 20 units, a dropout layer, a dense relu layer with 10 units and finally a sigmoid output layer.

Exploratory testing was carried out to ensure the design was suitable for the problem. First, the type of RNN was tested, and it was found that using an LSTM layer outperformed both GRU and Simple RNN layers, therefore this was used in the final architecture. However, the initial performance was slightly worse than expected. This was thought to be due to poor feature extraction so different numbers of layers and units were tested before the LSTM. A few different architectures were tested, however, it was found that a single 84 unit dense relu layer followed by a dropout layer improved performance. Alongside this, reducing the number of units in each of the

	LSTM
Dense Input units	84
Dropout 1	0.05
LSTM units	20
Recurrent Dropout	0.1
Dense Hidden 1 units	10
Dropout 2	0.05
Dense Hidden 2 units	5
Dense Output units	1

Table 1.2: LSTM Parameters

	CNN
Convolution Depth	20
Kernel Size	3
Pooling Size	2
Dropout 1	0.05
Dense Hidden 1 units	10
Dropout 2	0.1
Dense Hidden 2 units	5
Dense Output units	1

Table 1.3: CNN Parameters

layers was found to increase performance, so the number of units in the later layers were halved; the RNN was reduced to 20 units, the first hidden layer to 10 units and the second hidden layer down to 5 units. It is likely that the network was overfitting before reducing the number of units, and therefore reducing the complexity improved performance. The final LSTM architecture can be seen in Table 1.2.

The design of the CNN shall consist of 2 groups of layers, each consisting of 2 2D convolution layers, a max-pooling layer and a dropout layer. This would then be flattened, before being passed into a 20 unit dense relu layer and finally a sigmoid output layer.

However, during initial testing, it was found that the model was overfitting and underperforming. To solve this one of the groups of initial layers were removed, resulting in the CNN architecture only consisting of two 2D convolutional layers, a max-pooling layer and a dropout layer. In addition to this, it was found that just a single hidden layer was too simple, and therefore an extra hidden relu dense layer was added between the initial hidden layer and output layer. Just like with the LSTM architecture, it was found that reducing the number of units was beneficial, so the first hidden layer was reduced to 10 units and the new second hidden layer was set to 5 units. Finally, another dropout layer was added between the two hidden layers, with the combined changes solving the overfitting problem and improving performance greatly. The final CNN architecture can be seen in Table 1.3.

4.4 Summary

In this section, the preprocessing of the dataset was carried out to ensure the data was suitable for training both models. The models were then constructed with an initial architecture before some exploratory testing was carried out to maximise performance. The final LSTM and CNN architectures can be seen in Tables 1.2 and 1.3.

5. Results and Evaluation

5.1 Introduction

This chapter shall cover the methods used for evaluating the models created, with the expectations and results from these experiments being discussed and analysed.

5.2 Evaluation Metrics

A common approach is to test the performance of neural network against other machine-learning-based techniques, as used by Sarker [9]. Therefore, to evaluate the models created, several popular baseline models have been chosen, based on those available within the sci-kit learn library, including; K-Nearest Neighbors (KNN), Support Vector Machines (SVMs) and Decision Tree (DT). To ensure the test was fair, initial testing was carried out to best fit the parameters of these models, however, some had to be restricted due to hardware limitations. Therefore, KNN used 3 neighbours and euclidean distance, SVM used a linear kernel and DT with a depth of 3. For each of these models, the same dataset shall be used with the same data preprocessing stages carried out to ensure an equal test.

To effectively compare the constructed networks to these classification algorithms, the same metrics as Sarker [9] shall be used, due to the effective comparisons it allows. This measures the number of true positive (TP), false positive (FP), true negative (TN) and false-negative (FN) categorisations. These can be easily compared, to see how well the models correctly and incorrectly classify packets, but can also be used to calculate the Precision, Recall, F-score and Accuracy, as defined by Han [34]:

Precision - the percentage of correctly classified attack packets of all packets predicted to be malicious.

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

Recall - the percentage of correctly classified attack packets of all packets that were malicious.

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

Fscore - a measure of Accuracy, based on Precision and Recall.

$$Fscore = 2 * \frac{Precision * Recall}{Precision + Recall}$$

Accuracy - the percentage correctly classified packets.

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

The metric recall was prioritised, to ensure that attack packets are detected correctly so disruption to users can be prevented. Secondary to this shall be the precision, to ensure that benign traffic isn't misclasified and genuine users blocked.

These measures and metrics shall be measured and calculated across all 5 models for the three attacks they are trained on, as well as the unknown attack. This will allow a suitable comparison of performance against known and unknown attacks.

5.3 Known Attacks

It was expected that during testing against known attacks, all models would likely perform well, as both CNN and LSTMs have been shown to perform well against DDoS attacks previously. The use of 3 types of attacks may have one of two effects; it may improve the network's ability to generalise and improve performance or potentially harm the networks ability to extract relationships and lower performance. However, in practice, it was believed that it would most likely help counteract overfitting.

Performance was found to be good across the board, with all models achieving an accuracy above 98.5%. All models achieved a high precision, with both the CNN and DT achieving a perfect score of 1, misclassifying 0 benign packets (of 568,265), while SVM was the worst with a score of 0.9991, misclassifying 157 benign packets. The LSTM also performed well, only misclassifying a single benign packet. Therefore, both networks have been shown to perform well with regards to precision, both outperforming KNN and SVM while remaining similar to DT.

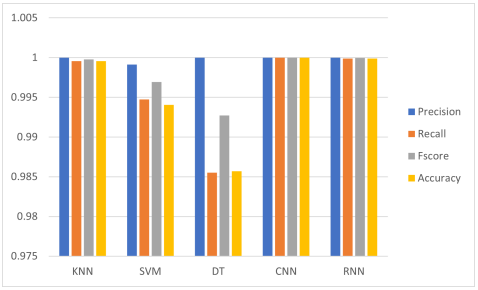

Figure 1.7: Performance metrics for known attacks.

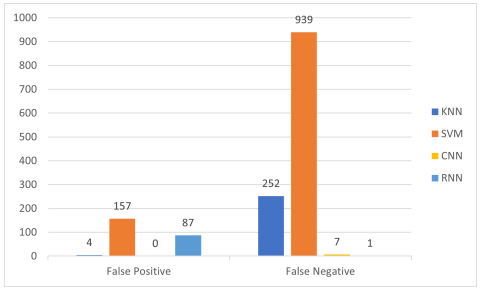

Figure 1.8: Incorrect predictions for known attacks.

Performance against the other metrics was found to be much lower, with the DT performing badly in comparison, achieving a recall and accuracy of around 0.985. The SVM was also shown to underperform the other 3 models, achieving a recall and accuracy of around 0.994. These models performed well, but underperformed relative to the other 3, and therefore have been removed for Figure 1.9 so a better comparison can be seen with the best performing models.

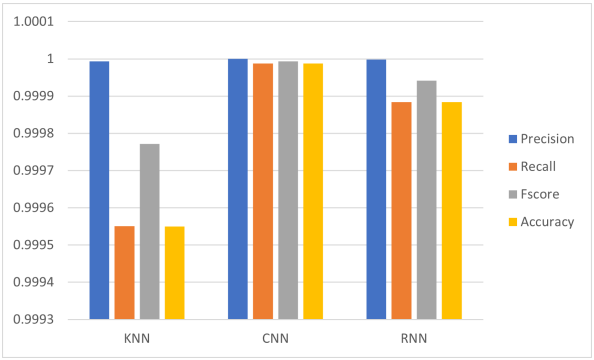

Figure 1.9: Performance metrics for known attacks, reduced.

Figure 1.9 shows the high performance of the CNN, outperforming all of the other 4 models across all 4 metrics, demonstrating how the CNN can perform well against known data. Importantly, the recall was very high, only misclassifying 7 packets, which can be seen in Figure 1.8 to be much lower than all other models. However, this may be an indication that the CNN has been overfit to the dataset, and therefore may perform comparatively worse against unknown data.

While the LSTM performed worse than the CNN, it still achieved a high recall, misclassifying 88 packets (of 568,265). While the number of misclassified packets was much hgiher, this is still an accuracy of over 99.9% and demonstrates the LSTMs high performance against known data. It is possible that this may be improved by implementing a more complex

feature extraction layer, either a different structure or through the use of an autoencoder, rather than the singular dense layer.

5.4 Unknown Attacks

Performance against unknown attacks was expected to be reasonably worse, as the models haven't encountered the attack before and therefore may not be able to reliably detect attack packets. This was true and all models saw a decrease in performance, however, they proved to be very resilient, with only a small decrease in performance. The results in Figure 1.10 demonstrates that while accuracy decreased for all models, it remained at a high level, with a minimum of over 96.5%.

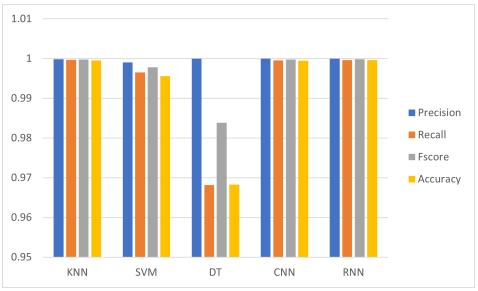

Figure 1.10: Performance metrics for known attacks.

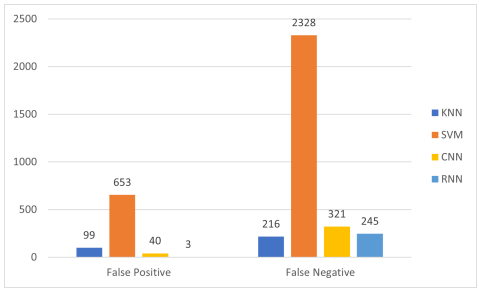

Figure 1.11: Incorrect predictions for unknown attack.

It is also clear that, once again, all models achieved a high level of precision. Notably, the LSTM performed the best this time, rather than the CNN. Figure 1.11 shows how the LSTM achieved a much lower false-positive rate than all other models, misclassifying only 3 benign packets, compared to the CNNs misclassification of 40 benign packets (of 674,341), although these increases will be partially due to the slightly larger dataset. While the CNN performed worse, it still had a precision of over 0.999 and outperformed the KNN and SVM, which misclassified 99 and 653 benign packets respectively. This high precision is likely due to the large amount of benign data the network was trained on, which has been seen before and can be classified correctly.

Performance against the other metrics was found to be lower than against known attacks in most cases, with the DT performing the worst, achieving a recall and accuracy of around 0.968. Unexpectedly, the SVM was shown to perform better against the unknown dataset rather than the known datasets, but underperformed comparatively achieving a recall and accuracy of around 0.995. As these two models underperformed relative to the other 3, they have been removed for Figure 1.12 so a better comparison can be

seen with the best performing models.

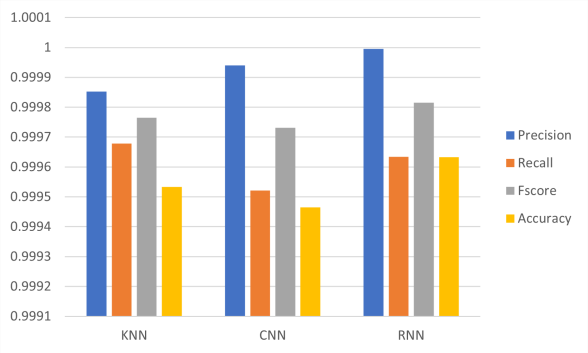

Figure 1.12: Performance metrics for known attacks, reduced.

Figure 1.12 shows how the LSTM, CNN and KNN performed well and were effective against a zero-day DDoS attack, however, all performed worse than previously. The LSTM performed best in Precision, F-score and Accuracy, but was beaten by the KNN in Recall. The high recall of the KNN is significant as it means it correctly classified the most attack packets, however, all models achieved a recall over 0.9995. The worst performing for recall was the CNN, misclassifying 321 attack packets, however, this was only 0.47% and would make little difference in a practical context. Surprisingly, the CNN was outperformed by the KNN in all metrics except precision, which may be due to overfitting to the known data.

As expected, Figure 1.11 shows an increase in incorrect predictions for all models, with the most significant increase being in the false-negative rate, likely due to never having seen the given attack packets previously. As shown above, by the KNNs high recall rate, it performed particularly well against attack packets, only misclassifying 216 compared to 245 for the LSTM, and 321 for the CNN. However, the LSTM performed best overall only misclassifying a total of 248 compared to 315 for KNN and 361 for the CNN.

The LSTM performed very well and has proven to be a suitable method of detecting DDoS attacks. There may also be potential to improve this further, as mentioned above, through the addition of better feature extraction to improve performance further. The high performance of the LSTM shows that RNNs may be well suited to detecting unknown attacks, and therefore further testing of a GRU architecture may also be useful, due to GRUs increased computing performance it may prove very suitable in a time-sensitive environment. However, as exploratory testing showed, performance may be worse.

While the CNN performed the best against known data, the performance took the largest hit against unknown data, compared to the LSTM and KNN. This dip in performance may be due to the CNN overfitting, however, this is unlikely as accuracy only fell roughly 0.03%. This may demonstrate that the chosen fully-connected architecture was less able to correctly classify the unknown attacks, and therefore it would be suitable to test further against a wider range of known and unknown attacks. However, the CNN was still shown to perform well and be suitable for DDoS detection.

5.5 Summary

In this section, the evaluation methods for the NNs were laid out. The NNs were tested against SVM, DT and KNN models to compare performance with the true positive, true negative, false positive and false negative rates monitored and then used to calculate a range of metrics for this comparison. The results of the experiment have shown that both the CNN and LSTM designed are suitable for detecting both known and unknown DDoS attacks. They achieved a minimum accuracy of over 99.94%, and were able to accurately classify attack packets so that an attack can be prevented, while also rarely misclassifying a benign packet.

Conclusion

Conclusion

In this paper, the CICDDoS-2019 dataset [8] has been used to evaluate the effectiveness of LSTM and CNN architectures at classifying known and unknown attacks. The CICDDoS-2019 dataset has proven to be a suitable modern dataset, due to the wide range of the latest attacks, meaning it was ideal for training and testing models against known attacks and unknown DDoS attacks. It also has a large sample size for each attack, although this varies, making it a great candidate for training future neural networks.

Both models are very effective, achieving high precision, accuracy and recall, outperforming other machine-learning-based techniques in most instances. Both were shown to perform exceptionally against known attacks, outperforming KNN, SVM and DT models, achieving an accuracy of over 99.98%. Expectedly, performance was worse against unknown attacks, however, an accuracy of over 99.94% was achieved for the CNN and over 99.96% for the LSTM. This demonstrates how they were both able to effectively meet the goal of correctly classifying attack packets, having a high true positive rate, as high as 100%, to ensure DDoS attack prevention, and low false-positive rate, as low as 0.001%, ensures genuine users are not disrupted. Therefore, they have both been proven to be a suitable method of detecting existing and zero-day attacks.

Future Work

Future work would ideally test CNNs and RNNs further against a wider range of attacks, as performance limitations meant the number of known and unknown attacks had to be limited. This could test if more known attacks harms performance or how resilient the proposed architectures are against a wider range of unknown attacks. This would also effectively test the CICDDoS-2019 dataset further, as it proved to be very useful in this paper, but only 4 of the datasets were tested.

Several avenues could be explored with regards to the architecture. The success of the LSTM suggests that RNNs may be a suitable architecture for detecting DDoS attacks and so other RNN architectures like GRU should be tested. GRU may be suitable as its simpler gate structure means it often performs quicker at the cost of performance, which may be ideal given the time-sensitive environment and excellent performance of the LSTM. Alternatively, other improvements could be tested, such as a more complex feature extraction stage before the LSTM, or a different fully connected architecture which could improve performance further.

Alternatively, it would be useful to test both models in a practical environment, to test if they perform fast enough and accurately in real-time to detect known and unknown attacks.

Bibliography

[1] N. Inc., 'Ddos attacks increase by 151% in first half of 2020,' [Online]. Available: https://www.home.neustar/about-us/news-room/press-releases/2020/ddos-attacks-increase-by-151-in-first-half-of-2020.

[2] A. W. S. (Shield), 'Threat landscape report q1 2020,' [Online]. Available: https://aws-shield-tlr.s3.amazonaws.com/2020-Q1_AWS_Shield_TLR.pdf.

[3] U. R. Acharya *et al.*, 'A deep convolutional neural network model to classify heartbeats,' *Computers in Biology and Medicine*, vol. 89, pp. 389–396, 2017. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0010482517302810.

[4] D. Bálya, B. Roska, T. Roska and F. S. Werblin, 'A cnn framework for modeling parallel processing in a mammalian retina,' *International Journal of Circuit Theory and Applications*, vol. 30, no. 2-3, pp. 363–393, 2002. [Online]. Available: https://www.onlinelibrary.wiley.com/doi/abs/10.1002/cta.204.

[5] S. Kido, Y. Hirano and N. Hashimoto, 'Detection and classification of lung abnormalities by use of convolutional neural network (cnn) and regions with cnn features (r-cnn),' in *2018 International Workshop on Advanced Image Technology (IWAIT)*, 2018, pp. 1–4.

[6] J. Zhang and K. Man, 'Time series prediction using rnn in multi-dimension embedding phase space,' in *SMC'98 Conference Proceedings. 1998 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.98CH36218)*, vol. 2, 1998, 1868–1873 vol.2.

[7] X. Yuan, C. Li and X. Li, 'Deepdefense: Identifying ddos attack via deep learning,' in *2017 IEEE International Conference on Smart Computing (SMARTCOMP)*, 2017, pp. 1–8. DOI: 10.1109/SMARTCOMP.2017.7946998.

[8] C. I. for Cybersecurity, 'Ddos evaluation dataset (cic-ddos2019),' [Online]. Available: https://www.unb.ca/cic/datasets/ddos-2019.html.

[9] I. H. just, Y. B. Abushark, F. Alsolami and A. I. Khan, 'Intrudtree: A machine learning based cyber security intrusion detection model,' *Symmetry*, vol. 12, no. 5, 2020, ISSN: 2073-8994. DOI: 10.3390/sym12050754. [Online]. Available: https://www.mdpi.com/2073-8994/12/5/754.

[10] S. T. Zargar, J. Joshi and D. Tipper, 'A survey of defense mechanisms against distributed denial of service (ddos) flooding attacks,' *IEEE Communications Surveys Tutorials*, vol. 15, no. 4, pp. 2046–2069, 2013. DOI: 10.1109/SURV.2013.031413.00127.

[11] M. UK Department for Digital Culture and Sport, 'Cyber security breaches survey 2019,' [Online]. Available: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/950063/Cyber_Security_Breaches_Survey_2019_-_Main_Report_-_revised_V2.pdf.

[12] G. STRAWBRIDGE, '10 biggest ddos attacks and how your organisation can learn from them,' [Online]. Available: https://www.metacompliance.com/blog/10-biggest-ddos-attacks-and-how-your-organisation-can-learn-from-them/.

[13] A. S. Razavian, H. Azizpour, J. Sullivan and S. Carlsson, 'CNN features off-the-shelf: An astounding baseline for recognition,' *CoRR*, vol. abs/1403.6382, 2014. eprint: 1403.6382. [Online]. Available: http://arxiv.org/abs/1403.6382.

[14] A. R. Shaaban, E. Abd-Elwanis and M. Hussein, 'Ddos attack detection and classification via convolutional neural network (cnn),' in *2019 Ninth International Conference on Intelligent Computing and Information Systems (ICICIS)*, 2019, pp. 233–238. DOI: 10.1109/ICICIS46948.2019.9014826.

[15] C. Olah, 'Understanding lstm networks,' [Online]. Available: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[16] F. Gers and J. Schmidhuber, 'Recurrent nets that time and count,' in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3, 2000, 189–194 vol.3.

[17] S. Seufert and D. O'Brien, 'Machine learning for automatic defence against distributed denial of service attacks,' in *2007 IEEE International Conference on Communications*, 2007, pp. 1217–1222. DOI: 10.1109/ICC.2007.206.

[18] Taeshik Shon, Yongdae Kim, Cheolwon Lee and Jongsub Moon, 'A machine learning framework for network anomaly detection using svm and ga,' in *Proceedings from the Sixth Annual IEEE SMC Information Assurance Workshop*, 2005, pp. 176–183. DOI: 10.1109/IAW.2005.1495950.

[19] Y. Feng, J. Li and T. Nguyen, 'Application-layer ddos defense with reinforcement learning,' in *2020 IEEE/ACM 28th International Symposium on Quality of Service (IWQoS)*, 2020, pp. 1–10. DOI: 10.1109/IWQoS49365.2020.9213026.

[20] M. Bhuyan, D. K. Bhattacharyya and J. Kalita, 'An empirical evaluation of information metrics for low-rate and high-rate ddos attack detection,' *Pattern Recognition Letters*, vol. Early Access, Jan. 2014. DOI: 10.1016/j.patrec.2014.07.019.

[21] J. David and C. Thomas, 'Ddos attack detection using fast entropy approach on flow based network traffic,' *Procedia Computer Science*, vol. 50, pp. 30–36, 2015, Big Data, Cloud and Computing Challenges, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2015.04.007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050915005086.

[22] C.-L. Chen, 'A new detection method for distributed denial-of-service attack traffic based on statistical test,' *J. UCS*, vol. 15, pp. 488–504, Jan. 2009.

[23] S. Specht and R. Lee, 'Distributed denial of service: Taxonomies of attacks, tools, and countermeasures.,' Jan. 2004, pp. 543–550.

[24] Fanglu Guo, Jiawu Chen and Tzi-cker Chiueh, 'Spoof detection for preventing dos attacks against dns servers,' in *26th IEEE International Conference on Distributed Computing Systems (ICDCS'06)*, 2006, pp. 37–37. DOI: 10.1109/ICDCS.2006.78.

[25] J. Mirkovic, G. Prier and P. Reiher, 'Attacking ddos at the source,' Dec. 2002, pp. 312–321, ISBN: 0-7695-1856-7. DOI: 10.1109/ICNP.2002.1181418.

[26] X. Xu, Y. Sun and Z. Huang, 'Defending ddos attacks using hidden markov models and cooperative reinforcement learning,' Apr. 2007, pp. 196–207, ISBN: 978-3-540-71548-1. DOI: 10.1007/978-3-540-71549-8_17.

[27] J. Berral, N. Poggi, J. Alonso Lopez, R. Gavaldà, J. Torres and M. Parashar, 'Adaptive distributed mechanism against flooding network attacks based on machine learning,' Jan. 2008, pp. 43–50. DOI: 10.1145/1456377.1456389.

[28] M. Elsayed, N.-A. Le-Khac, S. Dev and A. Jurcut, 'Ddosnet: A deep-learning model for detecting network attacks,' Jun. 2020. DOI: 10.1109/WoWMoM49955.2020.00072.

[29] C. I. for Cybersecurity, 'Intrusion detection evaluation dataset (iscxids2012),' [Online]. Available: https://www.unb.ca/cic/datasets/ids.html.

[30] C. for Applied Internet Data Analysis, 'Ddos attack 2007,' [Online]. Available: https://www.caida.org/data/passive/ddos_dataset_request.xml.

[31] P. G. M. H. B. D. Bhattacharyya and J. K. Kalita, 'Packet and flow based network intrusion dataset,' 2012, pp. 322–334.

[32] B. Research, 'Deepbench,' [Online]. Available: https://github.com/baidu-research/DeepBench.

[33] Y. Wang, J. Liu, J. Mišić, V. B. Mišić, S. Lv and X. Chang, 'Assessing optimizer impact on dnn model sensitivity to adversarial examples,' *IEEE Access*, vol. 7, pp. 152 766–152 776, 2019.

[34] J. Han, M. Kamber and J. Pei, 'Data mining concepts and techniques third edition,' *The Morgan Kaufmann Series in Data Management Systems*, vol. 5, no. 4, pp. 83–124, 2011.