Anomaly Detection in IoT Networks: A Performance Comparison of Transformer, 1D-CNN, and GrowNet Models on the Bot-IoT Dataset

Aurelia Kusumastuti[1], Denis Rangelov[1][a], Philipp Lämmel[1][b], Michell Boerger[1][c], Andrei Aleksandrov[1][d] and Nikolay Tcholtchev[1,2][e]

[1]*Fraunhofer Institute for Open Communication Systems (FOKUS), Berlin, Germany*

[2]*RheinMain University of Applied Sciences, Wiesbaden, Germany*

fi

Keywords: Transformer, Random Forest, Deep Neural Networks, CNN, Anomaly Detection, Intrusion Detection, IoT.

Abstract: This paper presents an exploratory analysis of deep learning techniques for intrusion detection in IoT networks. Specifically, we investigate three innovative intrusion detection systems based on transformer, 1D-CNN and GrowNet architectures, comparing their performance against random forest and three-layer perceptron models as baselines. For each model, we study the multiclass classification performance using the publicly available IoT network traffic dataset Bot-IoT. We use the most important performance indicators, namely, accuracy, F1-score, and ROC, but also training and inference time to gauge the utility and efficacy of the models. In contrast to earlier studies where random forests were the dominant method for ML-based intrusion detection, our findings indicate that the transformer architecture outperforms all other methods in our approach.

1 INTRODUCTION

Internet of Things (IoT) refers to a group of interconnected devices which exchange and collect data without human intervention (Hounsell et al., 2009), e.g. over the cloud or a blockchain infrastructure (Kullig et al., 2020). As more IoT devices connect to the internet on a daily basis, the potential for the technology to alter businesses and sectors grows rapidly. Among the benefits of using so-called smart devices are valuable data insights, cost savings due to task and controlling automation, and the ability to connect various type of devices and data. In IoT's strengths, however, also lie its weaknesses. For example, data collected by smart devices in the health and insurance industry could be leaked, exposing large volumes of sensitive medical and financial data (Shahid et al., 2022) (Chatterjee and Ahmed, 2022). The ability to carry out tasks remotely means possibly exposing equipments to bad actors that might try to tamper with IoT devices, be it on a software or a hardware level (Stellios et al., 2018). In addition, the vast amount of devices introduces heterogenous software and hardware stacks, which means more attack surface (Stoyanova et al., 2020). One way to detect possible vulnerabilities or attacks is to use anomaly detection methods for incoming traffic at various levels, from the IoT network to the data center. Anomaly detection is a data analysis process that looks for irregular patterns, unexpected behavior, or deviations from the standard mode of operation of a given system.

As mentioned previously, IoT devices are widely used for collecting and transporting sensitive data in a variety of sectors, including medical, manufacturing, and finance. Therefore, anomaly detection in IoT networks are crucial to obtain critical actionable information for e.g. fraud detection (Min et al., 2021) or condition monitoring (Li et al., 2020).

Most IoT anomaly detection approaches require extensive human involvement and optimizations, despite initiatives for autonomic and automated network resilience (Chaparadza et al., 2013) (Tcholtchev and Chaparadza, 2010). Establishing an automated model in an IoT setting presents various challenges. It is difficult and not always possible to appropriately characterize and categorize various types of anomalous data, especially when labeled training data is not accessible. The concept of normal behavior is continually changing and evolving in various domains (Ryu et al., 2021). Furthermore, noise is always present

[a] https://orcid.org/0000-0002-2006-4218
[b] https://orcid.org/0000-0002-4411-0557
[c] https://orcid.org/0000-0002-5741-9043
[d] https://orcid.org/0000-0002-4717-4206
[e] https://orcid.org/0000-0001-6821-4417

Kusumastuti, A., Rangelov, D., Lämmel, P., Boerger, M., Aleksandrov, A., Tcholtchev and N.
Anomaly Detection in IoT Networks: A Performance Comparison of Transformer, 1D-CNN, and GrowNet Models on the Bot-IoT Dataset.
DOI: 10.5220/0013637600003967
In *Proceedings of the 14th International Conference on Data Science, Technology and Applications (DATA 2025)*, pages 633-643
ISBN: 978-989-758-758-0; ISSN: 2184-285X

in observed data, and when the signal-to-noise ratio is low, the size of noise resembles actual anomalies. The quantity of interconnected systems and data types further enhance the complexity (Chatterjee and Ahmed, 2022).

To address some of these challenges, researchers often rely on AI-based mechanisms due to AI's ability not just to process and analyze large amounts of data in real time, but also adapt to new incoming data. Deep learning (DL) algorithms have evolved into the most extensively used and viable intrusion detection technology in networks. Deep learning is in general widely employed in cybersecurity because it can detect previously unknown patterns in raw data (Khan et al., 2022). Due to the large body of work in the DL domain, it would be of interest to compare the different approaches implementing the concept of DL using different architectures.

1.1 Objective and Scope of the Paper

With this study we aim to provide a performance comparison of different DL architectures for detecting IoT-traffic anomalies. To this end, we present the results of experiments utilizing three novel DL architectures based on transformer, 1D-CNN, and GrowNet models, as well as a simple random forest classifier and a multilayer perceptron for baseline benchmarking. The experiments are run on Bot-IoT - a publicly available dataset containing malign as well as benign network traffic observations from IoT-devices. We then present the empirical results of the individual performances based on previously defined metrics. While previous research identified random forests as the top-performing method for ML-based intrusion detection, we will reveal that the transformer architecture exceeds the performance of all other techniques in our approach.

1.2 Structure of the Paper

Having explained the background and scope of this paper, in the following sections we will proceed with reviewing the related work in anomaly detection, the role of DL algorithms for intrusion detection, and publicly available datasets for network intrusion detection. Then we describe in brief the different DL architectures up for comparison, as well as the relevant methodology we use to compare them, including the metrics used to evaluate the models' performances and the dataset on which we run our experiments. Using the aforementioned metrics we will then discuss the experiment results and examine its implications. Finally, we conclude with a summary of the experiments and discuss possible future opportunities based on our empirical findings.

2 RELATED WORK & TECHNOLOGICAL CONTEXT

2.1 Anomaly Detection in IoT Networks

At present, most techniques for identifying anomalies in IoT networks use a considerable amount of human intervention to set up the systems and analyze the data produced. As the number of interconnected devices grows, effectively analyzing and interpreting data becomes increasingly complex, even for experts. In the following paragraphs, existing automated methods that enable experts to focus only on the most significant observed events are briefly described. However, a detailed comparison of related approaches with our presented approach and results is discussed in Section 4.3.

Statistical and probabilistic methods use previously recorded data to model expected network behavior. New observations are then compared against the statistical model. If an observation fails to fit the model, it is classified as an anomaly (Markou and Singh, 2003). Examples for probabilistic methods are Hidden Markov Models (Görnitz et al., 2015) and Bayesian Networks (Hill et al., 2007).

Using long-term traffic trends, it is also possible to create regression models that predict expected network traffic behavior. New observations are then compared to the previously generated expected behavior. If the observed and expected values vary greatly, the observed incident is marked as anomalous (Giannoni et al., 2018). The complexity of the prediction model depends on the choice of architecture. Simple Support Vector Machines (SVMs) (Shahid et al., 2015) could already yield useful results, but more complex models using Deep Neural Networks (DNNs) and Long-Short-Term Memory (LSTM) are also viable, if not more widely used nowadays (Malhotra et al., 2015).

2.2 Deep Learning Algorithms for Anomaly Detection

A large number of deep anomaly detection methods have been introduced, demonstrating significantly better performance than conventional anomaly detection on addressing network intrusion. Major challenges in anomaly detection, which deep learning tackles, include class imbalances (anomalies are by

definition much rarer than the standard case) and sensitivity to noise (Pang et al., 2020). For example, Meidan et al. train deep autoencoders to learn normal behavior in IoT network traffic and try to reconstruct new, unseen traffic (Meidan et al., 2018). If the autoencoder fails to reconstruct the input data accurately, then it is a strong indication that the observed behavior is anomalous. Benefits of using autoencoders to detect anomalous behavior are heterogenity tolerance and the ability to flag a previously unseen behavior.

Another deep learning architecture recently used for anomaly detection is Generative Adversarial Network (GAN). The model is composed of two neural networks, a generator and a discriminator, which are trained in adversarial manner. Iliyasu et al. (Iliyasu and Deng, 2022) use GAN in a semi-supervised manner, in that the discriminator trains on a few malicious examples in addition to the normal ones to learn adequate representations. The generator then reconstructs from the latent features in the network traffic feature space to compute the anomaly score.

Architectures based on Long short-term memory (LSTM) are also commonly used for anomaly detection, especially to detect anomalous behaviors over time. This is because LSTM models use so-called gates which choose what to keep or discard in the memory as well as incorporate changes over time. This gives LSTM the ability to capture long-term dependencies thereby leading to LSTM being used as a basis algorithm for anomaly detection. For example, Imrana et al. extend the LSTM architecture by building a bidirectional LSTM network (Imrana et al., 2021). The solution the authors propose is making use of two LSTM networks. The first LSTM trains using the normal training data and the second uses a reversed version of the data, thus providing more time-related context to the algorithm and solving the vanishing gradient problem. When compared with other machine learning methods like SVM, Multilayer Perceptron, and Recurrent Neural Network, LSTM performed better by 6-20 percent (Imrana et al., 2021).

2.3 Publicly Available IoT Datasets for Anomaly Detection

Several datasets for anomaly detection in IoT networks have been made publicly available for developing machine learning algorithms to detect and prevent IoT malware infections.

IoT-23 (Garcia et al., 2020) belongs to the most used publicly available dataset for network intrusion detection. It has twenty malware as well as three normal IoT-traffic captures containing more than 50 million records ranging from 2018 to 2019. The malware traffic are generated by executing the respective malware on a Raspberry Pi. Among the executed attacks are Mirai and Okiru botnets as well as DDoS and C&C. The benign traffic records are recorded from three different IoT devices.

Another widely used dataset is Bot-IoT (Koroniotis et al., 2019), which was collected in a virtual environment and also contains normal and malware IoT traffic. The attack launched for generating traffic data include data exfiltration, keylogging, and DDoS. The entire dataset consists of more than 72 million records. We will provide a more detailed description including a feature and class analysis of the BoT-IoT dataset in Section 3.2.

The N_BaIoT dataset (Meidan et al., 2018) gathered malware traffic data by injecting commercial IoT devices with Mirai and BASHLITE botnets. The botnets carry in sum ten different attacks, including network flooding, scanning for vulnerable connection, and sending spam data. N_BaIoT also contains normal IoT traffic data. More than seven million records comprise the dataset in total.

3 METHODOLOGY

This section explains the details of the approach we use in this study, from the choice of DL algorithms to the datasets and metrics we chose to evaluate said algorithms.

3.1 Selection of DL Algorithms

The algorithm selection process employed during our research efforts was guided by two main criteria: innovation and performance. More specifically, we selected relatively recent and novel deep learning architectures, examined their practical applications in the field of intrusion detection and compared them to more traditional methods. By focusing on these novel algorithms/architectures, we aim at contributing to the current body of research and achieving possible improvements over the existing anomaly detection methods. Furthermore, performance is a critical factor in assessing the efficiency and reliability of deep learning algorithms, so the selected algorithms were evaluated based on their accuracy in network intrusion detection. In our evaluation, we compare the novel DL algorithms with baseline algorithms, namely random forest classifiers (RFC) and deep neural networks (DNN).

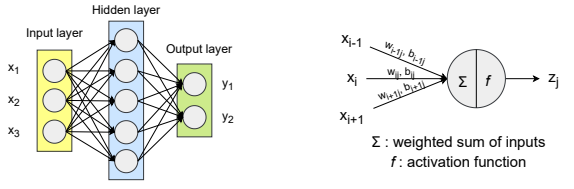
RFC are a type of machine learning algorithm that predict the outcome of a certain event by combining

multiple decision trees (Breiman, 2001). A decision tree in a classification context is a graph that decides which class a sample belongs to. The algorithm starts from a root node and traverses through the next nodes conditionally, evaluating a specific criterion at each node. It then arrives at a leaf node representing a class. An RFC is made up of many decision trees, each trained on a different subset of the data and using a random selection of features. This makes the model less prone to overfitting than a single decision tree. To make a prediction, the random forest classifier combines the output of all the decision trees to come up with a final decision.

In contrast, DNNs are made up of multiple layers of connected nodes or artificial neurons (see Fig. 1). Each layer takes input from the previous layer and performs calculations, which are then passed on to the next layer. The idea is that each layer extracts more abstract information from the input, allowing the network to perform complex tasks.

As a baseline model, we opt for a simple multilayer perceptron (MLP). MLPs can be seen as a specific type of DNN with a simpler architecture; our MLP has only one hidden layer (see Figure 1a). The first layer of the neural network receives the input data. Each node in this layer represents a feature or attribute of the input data. The values of each neuron are then multiplied by their respective weights and added up to compute the weighted sum of neurons. Each connection between nodes in the neural network has an associated weight that controls the strength of the connection. The network learns to adjust these weights during training in order to minimize the error between the predicted output and the actual output.

The weighted sum of neurons is then forwarded to the activation function that decides whether the neuron should be activated or not. An activation function is a non-linear function applied to the output of each node in a layer. The activation function introduces non-linearities into the model and enables it to capture complex relationships between the input and output data. The output of the activation function is then forwarded to the hidden layer. After computing the weighted sum of neurons and passing it through



(a) Example of a simple neural network

(b) Inner working of a neuron

Figure 1: Structure of a neural network.

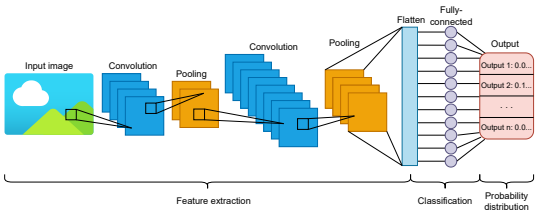


Figure 2: Example structure of a CNN.

an activation function, the result is then forwarded to the final output layer.

Since our problem is that of a multiclass classification, we use softmax as the activation function for the output layer. The softmax function outputs the probability of the input belonging to a certain class (Bridle, 1989). To assess the model's accuracy, we compute the loss function, which calculates the difference between the predicted and the original class or label of the input data. To decrease prediction error, the gradient of the cost function with respect to the network weights is computed using the chain rule. This process is commonly known as backpropagation. We then use the gradient to update the network's parameters by moving in the direction of steepest descent, which is why this process is called gradient descent. With θ_i denoting network parameters at iteration i, η denoting the learning rate, and $\nabla J(\theta_i)$ denoting the gradient of the cost function regarding θ_i, the gradient descent is defined as follows:

$$\theta_{i+1} = \theta_i - \eta \nabla J(\theta_i)$$

Moving towards the direction of steepest descent minimizes the cost function, and the rate of the movement is denoted by the aforementioned learning rate η. If the learning rate for a deep neural network is set too high, the training process of the model might become unstable and fail to converge to an optimal solution. This might cause strong oscillations or spikes in the loss function, preventing the network from learning successfully. The optimization method may overshoot the ideal solution, causing the loss function to increase rather than decrease. This phenomenon is often referred to as "exploding gradients". Furthermore, the model will have low accuracy with unseen data, resulting in high generalization error.

Most of the DL algorithms we are comparing in this paper are based on **Convolutional Neural Network (CNN)**. CNNs were introduced to solve image recognition problems and are similar to regular DNNs in that they are made up of neurons that optimize themselves through learning (LeCun et al., 1995). The basic components of a CNN include convolution layers, pooling layers, and fully connected layers. The convolution layer is the core component of a CNN, where the input image is convolved with

a set of filters, each of which captures different features of the image. The filters move across the image and perform dot products at each position, generating a feature map that represents the presence or absence of various features in the input image. The filters are learned during training, meaning the network will learn to automatically identify features that are useful for the classification task.

Pooling layers are used to reduce the spatial dimensions of the feature maps generated by the convolution layers. Pooling typically involves summarizing a group of neighboring pixels by taking their maximum or average value. This helps to reduce the dimensions of the feature maps while preserving important information.

Finally, fully connected layers are used to classify the features learned by the convolution layers into specific classes or categories. These final layers take in the output of the convolution and pooling layers, and apply traditional deep learning concepts of weights and biases to classify objects.

Standard DNNs were not effective in recognizing patterns in images because they did not take into account the spatial relationships between the pixels in an image. CNNs solve this problem by using convolutional layers to filter the image and extract features at different scales. This allows the network to identify patterns and features in the image regardless of its location in the image. Additionally, pooling layers are used to downsample the image and reduce the dimensionality of the convolved feature map, which helps to reduce overfitting.

With this in mind, the DL architectures examined throughout our work are defined as follows:

- **SoftOrdering1DCNN** uses one-dimensional CNNs to classify tabular data. CNNs are a widely used NN architecture for solving computer vision problem. That is because convolutional kernels extract both the local connectivity and the spatial locality of the input image. Because tabular features are often not spatially connected, we cannot input a tabular dataset directly to a convolutional layer. SoftOrdering1DCNN uses a fully connected layer to expand the size of the input (tabular data), apply non-linear combination, and sort the original features as to create spatial correlation. Following the reshaping, features are retrieved in numerous 1-dimensional convolutional layers connected by a skip-like network. After flattening, the collected features are used to predict targets via a fully linked layer. This algorithm won second place in the Mechanisms of Action (MoA) Prediction Research Code Competition to develop accurate and efficient computational models for predicting the mechanism of action of new drugs (noa,).
- The **FT-Transformer** (Feature Tokenizer + Transformer) algorithm (Gorishniy et al., 2021) modifies the transformer architecture (Vaswani et al., 2017) for the use on tabular data. FT-Transformer first tokenizes the input features into embeddings, which are then processed by the transformer. In comparison to eight other deep learning methods for tabular data classification over eleven datasets, FT-Transformer outperforms the other methods in six cases, including real estate (Pace and Barry, 1997), income prediction (Kohavi et al., 1996), and simulated physical particles (Baldi et al., 2014).
- **GrowNet** is a novel neural network model that combines the strengths of neural networks and boosted trees (Badirli et al., 2020). The model trains using boosted trees that have a shared feature map. In a boosted tree model, individual decision trees are trained sequentially on subsets of the data, with each subsequent tree attempting to correct the errors made by the previous trees. Each decision tree is built using a splitting criterion that optimizes the reduction in the error of the current model. The final prediction is made by combining the predictions of all the individual trees, weighted by their accuracy (Hastie et al., 2009). The trees are built by simultaneously predicting the gradient of the loss function and the output at each iteration. The authors of the GrowNet paper compare it with other boosting methods, and GrowNet achieves better performance in regression, classification and learning to rank on multiple datasets.

3.2 Datasets

We choose the Bot-IoT dataset (Koroniotis et al., 2019) as the benchmark dataset to evaluate the performance of the deep learning algorithms for network intrusion detection. Bot-IoT is a publicly available dataset emulating the real-world network traffic of IoT devices. The dataset has been used in several previous research studies to assess the performance of various intrusion detection algorithms (Bovenzi et al.,) (Saba et al., 2022) (Ibitoye et al., 2019), providing a sound basis for comparison.

3.2.1 Description of the Dataset

Koroniotis et. al (Koroniotis et al., 2019) constructed the BoT-IoT dataset by emulating a normal and botnet network traffic scenario. The dataset includes DDoS,

DoS, OS and Service Scan, Keylogging and Data exfiltration attacks. They then sampled 5% of the original dataset. The sample consists of about three million records. Using statistical measures of correlation coefficient (Sedgwick, 2012) and entropy (Lesne, 2014), the authors selected the most important features of the data, which are described in Table 1.
The labels in the dataset describe different attack scenarios:

- **Denial of Service (DoS):** attacks aim to flood a network, server, or website with traffic and requests, resulting in system failure or slowdown. The attacker's purpose is to disrupt the target system's regular operation, denying legitimate users access to its services or information. The dataset contains DoS as well as Distributed DoS (DDoS), each on TCP, UDP, and HTTP based networks (Koroniotis et al., 2019). While DoS attacks are carried out by a single device, DDoS involves many devices, usually through networks of controlled computers that have been compromised with malwares, also known as botnets (Kolias et al., 2017).
- **Information Theft:** refers to a class of attacks in which an adversary attempts to steal sensitive data. Two types of information theft attacks are included in the dataset, namely data exfiltration and keylogging. As the name suggests, data exfiltration attacks target a remote machine and attempt to obtain unauthorized access to data (Sabir et al., 2021). Keylogging, on the other hand, attempts to compromise a remote machine in order to capture a user's keystrokes and potentially steal user credentials (Singh et al., 2021).
- **Scanning:** or *fingerprinting* attacks are malicious operations that search remote machines for user information based on the specifications of said machines. Based on the type of information scanned, there are two key subcategories scanning attacks, namely OS and service scanning (Hoque et al., 2014). In a OS scanning attack, an adversary acquires information on the remote operating system by comparing its replies to pre-existing ones or based on OS differences in TCP/IP stack implementations. In a service scan attack, an adversary sends request packets to identify the services that run behind the system ports (0-65535) (Hoque et al., 2014).

3.2.2 Data Preprocessing

Before training and evaluating the investigated ML models using the Bot-IoT dataset, we first preprocess the data by imputing missing values, standardizing the numerical features, and one-hot-encoding string-formatted features. We also combined the labels category and "subcategory", resulting in ten possible classes network flows fall into. Table 2 depicts the distribution of labels. As we can see, the dataset exhibits a strong class imbalance, which can cause the ML models to overfit on the higher represented classes and perform poorly in recognising the underrepresented classes. To avoid this, we oversample the data using the SMOTE (Synthetic Minority Over-sampling Technique) algorithm (Chawla et al., 2002). SMOTE detects minority class instances that have majority class nearest neighbors. After discovering the minority-class cases, the algorithm chooses one of them and determines its k nearest neighbors. Then, at random, it selects one of the nearest neighbors and constructs a new synthetic instance between them based on the difference between the features of the two selected instances. This process is repeated until a specified number of synthetic instances are produced. This approach reduces the class imbalance by balancing the number of instances in the minority and majority classes.

Table 1: Description of the features in Bot-IoT dataset.

Feature	Description
proto	Transaction protocol
sport	Source port number
dport	Destination port number
seq	Argus sequence number
stddev	Standard deviation of aggregated records
N_IN_Conn_P_SrcIP	Num. of inbound conn. per source IP
min	Minimum duration of aggregated records
state	Feature state
mean	Average duration of aggregated records
N_IN_Conn_P_DstIP	Num. of inbound conn. per dest IP
drate	Destination-to-source packets per second
srate	Source-to-destination packets per second
category	Traffic category
subcategory	Traffic subcategory

Table 2: Distribution of labels in the Bot-IoT dataset, from most to least amount of records.

Label	Count
DoS (UDP)	1032975
DDoS (TCP)	977380
DDoS (UDP)	948255
DoS (TCP)	615800
Service Scan	73168
OS Scan	17914
DDoS (HTTP)	1485
DoS (HTTP)	989
Normal	477
Keylogging	73
Data exfiltration	6

3.3 Evaluation Metrics

To measure the performance of the above-mentioned algorithms, we evaluate not only the correctness of the result, but also the efficacy. The correctness metrics (accuracy, F1-score, and AUC) are based on the components of the confusion matrix. The confusion matrix compares the actual and predicted classes and partition the data as follows:

- **True Positive** (TP)**:** is the number of samples correctly classified as belonging to a certain class.
- **False Positive** (FP)**:** is the number of samples incorrectly classified as belonging to a certain class.
- **False Negative** (FN)**:** is the number of samples incorrectly classified as not belonging to a certain class.
- **True Negative** (TN)**:** is the number of samples correctly classified as not belonging to a certain class.

Following are the metrics we use in evaluating the DL algorithms.

- **Accuracy:** is one of the most widely used metrics for multi-class classification problems. It is the ratio of correctly classified samples to the total amount of samples.

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

- To calculate the **F1-score**, we need to first calculate precision and recall, which are defined as follows:

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

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

The F1-score is consequently given as the harmonic mean of precision and recall.

$$F1\text{-}score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

In our analysis, we employed micro-averaging for multiclass classification to compute precision and recall, ensuring that all instances across classes are considered equally, which helps provide a more comprehensive measure of model performance across imbalanced datasets.

- A receiver operating characteristic curve (**ROC curve**) is a graph that depicts the performance of a classification model over all classification levels by comparing True Positive Rate (TPR) and False Positive Rate (FPR). The area under the ROC curve, mostly just called Area Under Curve (**AUC**), indicates how well the model can discriminate between classes. AUC may be interpreted as the likelihood that the model rates a random positive case higher than a random negative example.
- **Training time** measures the time it takes to train a DL model. In its training phase, a DL model learns to recognize relationships and patterns by being exposed to a large amount of data to be able to precisely classify similar data in the future. The training process can be time-consuming and intensive in terms of computing resources.
- **Inference time**, on the other hand, measures the time it takes for a previously trained model to perform classification on previously unseen data. Because the model can depend on pre-learned weights and doesn't need to perform as many calculations as during the training phase, inference time is typically much quicker than training time.

4 EXPERIMENTS AND RESULTS

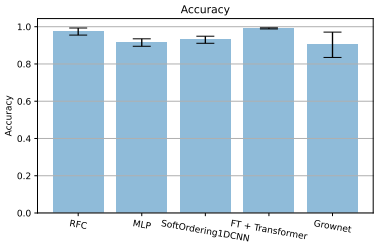
4.1 Experimental Setup

We use the complete Bot-IoT dataset (Koroniotis et al., 2019) with over three million traffic records to evaluate the performance of the deep learning algorithms. We repeat the experiments using the cross validation method. We perform the experiments on a JupyterHub server running CUDA 11.8 with 4x 12GB-NVIDIA GPUs.

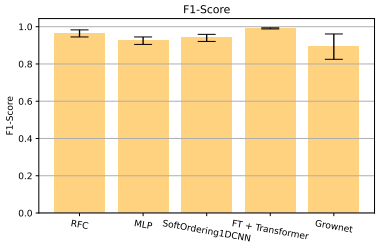
4.2 Results and Discussion

The experiment results are illustrated in Figures 3a, 3b, 3c, 3d, and the ROC-AUC is plotted in Figure 3e. As seen in Figure 3a, for the performed multiclass classification task, FT-Transformer achieved the highest accuracy and F1-Score of 0.991, followed by RFC with 0.974, SoftOrdering1DCNN, MLP, and lastly GrowNet with 0.904. In Figure 3c, FT-Transformer takes the least amount of time to train at 226 seconds, followed by MLP with 256 seconds. GrowNet took the longest time to train at 3088 seconds. RFC has the shortest inference time of 2.663 seconds, followed by MLP with 7.64 seconds. GrowNet has the longest inference time of 15 seconds. In Figure 3e, we see FT-Transformer with the highest AUC of 0.99, followed by MLP and SoftOrdering1DCNN with 0.95, RFC with 0.90, and GrowNet with 0.88.

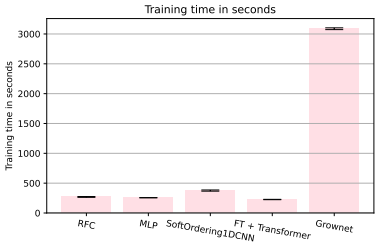
In the previous subsection, particularly in Figures 3a and 3b, we see very similar values of accuracy and F1-Score. This is because we oversampled the dataset



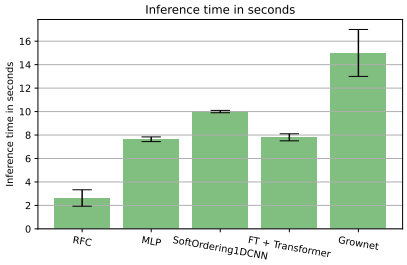
(a) Accuracy of the algorithms



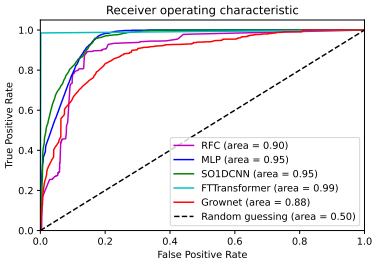
(b) F1-Score of the algorithms



(c) Training time of the algorithms in seconds



(d) Inference time of the algorithms in seconds



(e) Area Under Curve of the algorithms

Figure 3: Experiment Results.

to become balanced across all classes. Had we not done so, the model would over-predict the majority class, resulting in an F1-score that was significantly lower than the accuracy.

In our experiments, FT-Transformer shows the best training time. This comes down to the self-attention mechanism and parallelization in FT-Transformer. Transformers' self-attention mechanism allows the model to learn dependencies between distinct segments in the input sequence. During training, this attention mechanism allows the model to focus on key sections of the input, which can lead to faster convergence and improved performance (Vaswani et al., 2017). Moreover, transformers process input sequences in parallel. This parallelization is accomplished by self-attention mechanisms, in which each token in the input sequence may concurrently attend to all other tokens. By lowering the sequential computation required in classic recurrent neural networks, this parallel processing capabilities speeds up training (Schlag et al., 2021). However, RFC shows the fastest inference time by far, followed by MLP. This is because RFC and MLP have the simplest model structures in comparison to the other algorithms we use.

Additionally, the RFC is a collection of decision trees that can be processed in parallel with the proper setup in terms of software and hardware. This also speeds up inference. The RFC all-in-all achieves very good results, even besting MLP, SoftOrdering1DCNN, and GrowNet in terms of accuracy and F1-score, all of which are more powerful neural-network-based models. An explanation for this could be its robustness against noise in the training data (Ishii and Ljunggren, 2021) (Xu et al., 2023).

The MLP shows high AUC with somewhat lower accuracy. This could be explained when looking at the confusion matrix: There is an entire class which is misclassified, namely data exfiltration. The same case can be made with the SoftOrdering1DCNN. Most likely the extracted data exfiltration patterns aren't sufficient during the training process. The MLP is a simple three-layer-perceptron with one hidden layer, while SoftOrdering1DCNN has multiple fully-connected and convolutional layers, which possibly explain SoftOrdering1DCNN's higher accuracy. This would also explain the shorter training and inference time taken by MLP.

GrowNet shows the poorest performance in all aspects in our experiments. The slow training time, for one, can be explained by the architecture. The model is first initialized with a single neural network, then it adds new neural network to the ensemble iteratively. Each iteration goes through forward propagation, loss calculation, backward propagation, and the addition of the new neural network, with virtually no parallelization (Badirli et al., 2020). Moreover, with each iteration there are more neural networks already in the ensemble, meaning each iteration takes longer than the last. This training strategy might also explain the suboptimal accuracy, F1-Score, and AUC: GrowNet relies on its weak neural networks, and its sequential

training technique may be less successful in capturing complex dependencies in the data than other deep learning algorithms' end-to-end training approach.

In contrast, we see FT-Transformer showing the best performance across the board. This comes down largely to its attention mechanism, as explained previously. Transformers' self-attention mechanism enables the model to properly represent complex relationships between features in the dataset. By paying attention to key features and learning how they interact, the Transformer can get a greater understanding of the connections between distinct features, resulting in higher classification accuracy (Vaswani et al., 2017).

4.3 Comparison with State of the Art

We compare our experiment results with other works which also use the Bot-IoT as their baseline dataset. Ferrag et al. (Ferrag et al., 2020) compared among others the performance of CNN, Recurrent Neural Network (RNN), and DNN. The authors showed that a CNN achieves the highest accuracy of 98.371%, followed by RNN (98.331%), then DNN (98.221%), though DNN has the fastest training time of 991.6 seconds. This is consistent with our results (Figure 3a), which show that a CNN (SoftOrdering1DCNN) achieves higher accuracy (93%) than a DNN/MLP (91%). We also show that the MLP has faster training time than the SoftOrdering1DCNN (256 and 376 seconds respectively). The difference in accuracy as well as training time in our results in comparison to Ferrag et. al can be explained by the difference in model complexity: Our MLP has only one hidden layer while Ferrag's et. al's DNN has at least two, and our SoftOrdering1DCNN has one-dimensional convolutional layers, while Ferrag et. al use two-dimensional convolutional layers.

A comparison between a simple neural network and a random forest classifier was done by Churcher et. al (Churcher et al., 2021), which shows higher accuracy and F1-score of the neural network (97%) than the random forest classifier (95%). This differs from our results, which shows higher accuracy and F1-score for RFC in Figures 3a and 3b in comparison to MLP. This might be again be explained by the low complexity of our MLP, which only has one hidden layer. Our RFC scores higher than Churcher's, because we use a balanced version of RFC, which in addition to oversampling the dataset adds more stability to the model.

In another work by Özer et. al (Özer et al., 2021) RFC has slightly higher accuracy and F1-Score (99.9%) than a neural network (99.7%), which in general matches our results of our experiments. A similar result is also achieved by Alkadi et al (Alkadi et al., 2023), who use ANOVA F-Score for feature selection, with RFC scoring 99.9% and MLP 98% accuracy. ANOVA F-Score calculates variation between sample means or variation within the samples, which might explain the performance improvement in comparison to our results. Usoh et al. (Usoh et al., 2023) also show RFC and neural network achieving very similar accuracy of 99.8%, but RFC has F1-Score of 99.9% and neural network 96.33%, possibly because of remaining class imbalance due to lack of resampling of the dataset.

5 CONCLUSION AND FUTURE WORK

In this study, we evaluated various deep learning algorithms for intrusion detection using the Bot-IoT dataset, employing a comprehensive experimental setup to assess their multiclass classification performance. Precisely, we analyzed three intrusion detection systems based on deep learning methods, namely SoftOrdering1DCNN, FT-Transformer, and GrowNet, while benchmarking their results against random forest and MLP baselines.

Differing from recent studies, our findings indicate that the FT-Transformer outperforms all other methods, achieving an accuracy and F1-Score of 0.991, while also demonstrating the fastest inference time. This superior performance is likely due to the transformer's self-attention mechanism and parallelization. In contrast, GrowNet shows the weakest performance across all metrics, with the lowest accuracy, F1-Score, and AUC, as well as the longest training and inference times, likely due to its reliance on weak neural networks and an iterative training process. The RFC performs better than the MLP and SoftOrdering1DCNN in accuracy and F1-Score, although it scores lower in AUC. RFC and MLP benefit from simpler architectures, resulting in faster training times, with RFC achieving the lowest inference time overall. In summary, our research positions the FT-Transformer as a promising alternative for ML-based intrusion detection, highlighting a paradigm shift in the effectiveness of deep learning approaches.

However, our study presents opportunities for further exploration, particularly regarding the number of models and datasets utilized. While our findings offer valuable insights from evaluating deep learning algorithms on the Bot-IoT dataset, incorporating a broader range of datasets and models in future studies could enhance the generalizability and depth of our find-

ings. Hence, our future research will address this by incorporating additional IoT anomaly detection datasets, such as IoT-23 and N-BaIoT. By validating our models across multiple datasets, we hope to enhance the robustness and applicability of our conclusions, thereby providing a more comprehensive understanding of the performance of deep learning techniques in diverse IoT environments.

REFERENCES

Mechanisms of Action (MoA) Prediction.

Alkadi, S., Al-Ahmadi, S., and Ben Ismail, M. M. (2023). Toward improved machine learning-based intrusion detection for internet of things traffic. *Computers*, 12(8).

Badirli, S., Liu, X., Xing, Z., Bhowmik, A., Doan, K., and Keerthi, S. S. (2020). Gradient boosting neural networks: Grownet. *arXiv preprint arXiv:2002.07971*.

Baldi, P., Sadowski, P., and Whiteson, D. (2014). Searching for exotic particles in high-energy physics with deep learning. *Nature communications*, 5(1):4308.

Bovenzi, G., Aceto, G., Ciuonzo, D., Persico, V., and Pescapé, A. A Hierarchical Hybrid Intrusion Detection Approach in IoT Scenarios. In *GLOBECOM 2020 - 2020 IEEE Global Communications Conference*, pages 1–7.

Breiman, L. (2001). Random forests. *Machine learning*, 45:5–32.

Bridle, J. (1989). Training stochastic model recognition algorithms as networks can lead to maximum mutual information estimation of parameters. *Advances in neural information processing systems*, 2.

Chaparadza, R., Wodczak, M., Ben Meriem, T., De Lutiis, P., Tcholtchev, N., and Ciavaglia, L. (2013). Standardization of resilience & survivability, and autonomic fault-management, in evolving and future networks: An ongoing initiative recently launched in etsi. In *2013 9th International Conference on the Design of Reliable Communication Networks (DRCN)*, pages 331–341.

Chatterjee, A. and Ahmed, B. S. (2022). IoT anomaly detection methods and applications: A survey. *Internet of Things*, 19.

Chawla, N. V., Bowyer, K. W., Hall, L. O., and Kegelmeyer, W. P. (2002). Smote: synthetic minority oversampling technique. *Journal of artificial intelligence research*, 16:321–357.

Churcher, A., Ullah, R., Ahmad, J., ur Rehman, S., Masood, F., Gogate, M., Alqahtani, F., Nour, B., and Buchanan, W. J. (2021). An experimental analysis of attack classification using machine learning in iot networks. *Sensors*, 21(2).

Ferrag, M. A., Maglaras, L., Moschoyiannis, S., and Janicke, H. (2020). Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study. *Journal of Information Security and Applications*, 50:102419.

Garcia, S., Parmisano, A., and Erquiaga, M. J. (2020). IoT-23: A labeled dataset with malicious and benign IoT network traffic. Type: dataset.

Giannoni, F., Mancini, M., and Marinelli, F. (2018). Anomaly detection models for iot time series data. *arXiv preprint arXiv:1812.00890*.

Gorishniy, Y., Rubachev, I., Khrulkov, V., and Babenko, A. (2021). Revisiting deep learning models for tabular data. *Advances in Neural Information Processing Systems*, 34:18932–18943.

Görnitz, N., Braun, M., and Kloft, M. (2015). Hidden markov anomaly detection. In *International conference on machine learning*, pages 1833–1842. PMLR.

Hastie, T., Tibshirani, R., Friedman, J., Hastie, T., Tibshirani, R., and Friedman, J. (2009). Boosting and additive trees. *The elements of statistical learning: data mining, inference, and prediction*, pages 337–387.

Hill, D. J., Minsker, B. S., and Amir, E. (2007). Real-time bayesian anomaly detection for environmental sensor data. In *Proceedings of the Congress-International Association for Hydraulic Research*, volume 32, page 503.

Hoque, N., Bhuyan, M. H., Baishya, R. C., Bhattacharyya, D. K., and Kalita, J. K. (2014). Network attacks: Taxonomy, tools and systems. *Journal of Network and Computer Applications*, 40:307–324.

Hounsell, N., Shrestha, B., Piao, J., and McDonald, M. (2009). Review of urban traffic management and the impacts of new vehicle technologies. *IET intelligent transport systems*, 3(4):419–428.

Ibitoye, O., Shafiq, O., and Matrawy, A. (2019). Analyzing adversarial attacks against deep learning for intrusion detection in iot networks. In *2019 IEEE global communications conference (GLOBECOM)*, pages 1–6. IEEE.

Iliyasu, A. S. and Deng, H. (2022). N-gan: a novel anomaly-based network intrusion detection with generative adversarial networks. *International Journal of Information Technology*, 14(7):3365–3375.

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

Ishii, S. and Ljunggren, D. (2021). A comparative analysis of robustness to noise in machine learning classifiers.

Khan, A. R., Kashif, M., Jhaveri, R. H., Raut, R., Saba, T., and Bahaj, S. A. (2022). Deep learning for intrusion detection and security of internet of things (iot): Current analysis, challenges, and possible solutions. *Security and Communication Networks*, 2022.

Kohavi, R. et al. (1996). Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid. In *Kdd*, volume 96, pages 202–207.

Kolias, C., Kambourakis, G., Stavrou, A., and Voas, J. (2017). Ddos in the iot: Mirai and other botnets. *Computer*, 50(7):80–84.

Koroniotis, N., Moustafa, N., Sitnikova, E., and Turnbull, B. (2019). Towards the development of realistic botnet dataset in the Internet of Things for network forensic

analytics: Bot-IoT dataset. *Future Generation Computer Systems*, 100:779–796.

Kullig, N., Lämmel, P., and Tcholtchev, N. (2020). Prototype implementation and evaluation of a blockchain component on iot devices. *Procedia Computer Science*, 175:379–386. The 17th International Conference on Mobile Systems and Pervasive Computing (MobiSPC),The 15th International Conference on Future Networks and Communications (FNC),The 10th International Conference on Sustainable Energy Information Technology.

LeCun, Y., Bengio, Y., et al. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

Lesne, A. (2014). Shannon entropy: a rigorous notion at the crossroads between probability, information theory, dynamical systems and statistical physics. *Mathematical Structures in Computer Science*, 24(3):e240311.

Li, C., Mo, L., Tang, H., and Yan, R. (2020). Lifelong condition monitoring based on nb-iot for anomaly detection of machinery equipment. *Procedia Manufacturing*, 49:144–149. Proceedings of the 8th International Conference on Through-Life Engineering Services – TESConf 2019.

Malhotra, P., Vig, L., Shroff, G., Agarwal, P., et al. (2015). Long short term memory networks for anomaly detection in time series. In *ESANN*, volume 2015, page 89.

Markou, M. and Singh, S. (2003). Novelty detection: a review—part 1: statistical approaches. *Signal processing*, 83(12):2481–2497.

Meidan, Y., Bohadana, M., Mathov, Y., Mirsky, Y., Breitenbacher, D., Shabtai, A., and Elovici, Y. (2018). N-BaIoT: Network-based Detection of IoT Botnet Attacks Using Deep Autoencoders. *IEEE Pervasive Computing*, 17(3):12–22. arXiv:1805.03409 [cs].

Min, M., Lee, J. J., Park, H., and Lee, K. (2021). Detecting anomalous transactions via an iot based application: A machine learning approach for horse racing betting. *Sensors*, 21(6).

Pace, R. K. and Barry, R. (1997). Sparse spatial autoregressions. *Statistics & Probability Letters*, 33(3):291–297.

Pang, G., Shen, C., Cao, L., and van den Hengel, A. (2020). Deep learning for anomaly detection: A review. *CoRR*, abs/2007.02500.

Ryu, J.-Y., Kim, D.-W., and Kim, M.-K. (2021). Household differentiation and residential electricity demand in korea. *Energy Economics*, 95.

Saba, T., Rehman, A., Sadad, T., Kolivand, H., and Bahaj, S. A. (2022). Anomaly-based intrusion detection system for iot networks through deep learning model. *Computers and Electrical Engineering*, 99:107810.

Sabir, B., Ullah, F., Babar, M. A., and Gaire, R. (2021). Machine learning for detecting data exfiltration: a review. *ACM Computing Surveys (CSUR)*, 54(3):1–47.

Schlag, I., Irie, K., and Schmidhuber, J. (2021). Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, pages 9355–9366. PMLR.

Sedgwick, P. (2012). Pearson's correlation coefficient. *Bmj*, 345.

Shahid, J., Ahmad, R., Kiani, A. K., Ahmad, T., Saeed, S., and Almuhaideb, A. M. (2022). Data protection and privacy of the internet of healthcare things (iohts). *Applied Sciences*, 12(4):1927.

Shahid, N., Naqvi, I. H., and Qaisar, S. B. (2015). One-class support vector machines: analysis of outlier detection for wireless sensor networks in harsh environments. *Artificial Intelligence Review*, 43:515–563.

Singh, A., Choudhary, P., et al. (2021). Keylogger detection and prevention. In *Journal of Physics: Conference Series*, volume 2007, page 012005. IOP Publishing.

Stellios, I., Kotzanikolaou, P., Psarakis, M., Alcaraz, C., and Lopez, J. (2018). A survey of iot-enabled cyberattacks: Assessing attack paths to critical infrastructures and services. *IEEE Communications Surveys & Tutorials*, 20(4):3453–3495.

Stoyanova, M., Nikoloudakis, Y., Panagiotakis, S., Pallis, E., and Markakis, E. K. (2020). A survey on the internet of things (iot) forensics: challenges, approaches, and open issues. *IEEE Communications Surveys & Tutorials*, 22(2):1191–1221.

Tcholtchev, N. and Chaparadza, R. (2010). Autonomic fault-management and resilience from the perspective of the network operation personnel. In *2010 IEEE Globecom Workshops*, pages 469–474.

Usoh, M., Asuquo, P., Ozuomba, S., Stephen, B., and Inyang, U. (2023). A hybrid machine learning model for detecting cybersecurity threats in iot applications. *International Journal of Information Technology*, pages 1–12.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

Xu, J., Xu, J., Tong, Z., Yu, S., Liu, B., Mu, X., Du, B., Gao, C., Wang, J., Liu, Z., et al. (2023). Impact of different classification schemes on discrimination of proteins with noise-contaminated spectra using laboratory-measured fluorescence data. *Spectrochimica Acta Part A: Molecular and Biomolecular Spectroscopy*, 296:122646.

Özer, E., İskefiyeli, M., and Azimjonov, J. (2021). Toward lightweight intrusion detection systems using the optimal and efficient feature pairs of the bot-iot 2018 dataset. *International Journal of Distributed Sensor Networks*, 17(10):15501477211052202.