Comparative Analysis of ML Algorithms for Network Traffic Classification

Dimitar Dimitrov
Advanced Technology
Faculty Science & Technology

April 2024

As we are approaching Industry 4.0, the use of IoT devices has increased dramatically. With it, managing network traffic becomes increasingly challenging due to the higher volume and complexity of data exchanges. Traffic classification is essential for identifying applications and their Quality of Service (QoS) requirements to subsequently meet them. Traditional methods like Deep Packet Inspection (DPI) and port-based techniques are inadequate due to their computational intensity and port variability, respectively. This study creates a comparative analysis of different Machine Learning Algorithms for network traffic classification, focusing on their suitability for implementation on edge devices with limited computational resources. Specifically, we compare Decision Tree (DT), Random Forest (RF), and Neural Network (NN) models in terms of accuracy, memory usage, and computational requirements. Using a dataset from UNSW Sydney, which includes traffic data from various IoT and non-IoT devices, the models are trained and evaluated based on flow statistical features derived from packet captures. The results indicate that RF and NN models achieve high accuracy and F1 scores, particularly with a 60-second time interval, making them the most effective for real-time network traffic classification in IoT environments. Despite the NN model's higher memory usage and slower prediction time, it remains robust in capturing complex data relationships. Conversely, DT models, though fast and memory-efficient, exhibit lower accuracy. This comparative analysis underscores the importance of balancing model performance and computational efficiency, with RF models emerging as the most versatile choice for practical applications.

I. Introduction

There has been an increase in Internet of Things(IoT) devices [1]. IoT devices [2] connect and exchange data over the Internet or other types of communication networks. Such devices can include actuators, sensors, computers, etc.

As we are approaching the Fourth Industrial Revolution [3], also referred to as Industry 4.0, the integration of new technologies is transforming industries and societies at never-seen pace. At the forefront of it, IoT devices are the main contributors to this shift. With the ability of the devices to communicate with each other through a network, they facilitate seamless data exchange and real-time monitoring, enabling a more interconnected and efficient industrial environment. IoT devices empower industries to optimize their operations by providing actionable insights from the vast amounts of data they generate. This interconnectivity allows for predictive maintenance, reducing downtime and increasing the lifespan of machinery. In manufacturing, IoT-enabled smart factories can automate processes, monitor production in real time, and ensure quality control.

In addition to manufacturing, IoT devices are part of other sectors such as healthcare, agriculture, and logistics [4]. In healthcare, they enable remote patient monitoring

and telemedicine, improving patient outcomes and reducing the burden on healthcare systems. In agriculture, IoT devices provide precise data on soil conditions, weather patterns, and crop health, facilitating smarter farming practices and increasing yields. In logistics, IoT devices enhance supply chain visibility and efficiency, reducing delays and optimizing inventory management.

IoT getaways are crucial components in IoT network systems. It connects various IoT devices and sensors to the cloud and each other. It acts as a bridge, which allows devices with different communication protocols to exchange data, whether they use WiFi, Bluetooth, Zigbee, or another connectivity technology [5]. Other key functions include data aggregations, security, device management, and edge computing. Edge computing consists of local data processing and analytics, thus reducing latency and the need for constant connectivity to a central server. This is especially useful for applications requiring real-time data processing. A typical edge gateway can be comprised of multiple different Raspberry Pi's [6].

Quality of Service (QoS) is crucial in network management, especially as the number of IoT devices continues to grow. QoS ensures that the network can meet the specific requirements of each device, such as latency, bandwidth, and reliability, which are critical for the smooth operation of different applications. Failing to meet these QoS requirements can lead to various issues such as increased latency, packet loss, and overall network congestion. These issues can severely impact the performance and reliability of IoT applications and potentially lead to critical failures in systems. Traffic classification plays a vital role in this context by identifying the specific QoS needs of each type of traffic. This identification allows the network to prioritize and allocate resources effectively, ensuring that high-priority traffic receives the necessary bandwidth and low latency while preventing network overload. By accurately classifying traffic, network access technologies can dynamically adjust to meet QoS requirements, maintaining optimal network performance and reliability.

Traditional methods such as Deep Packet Inspection (DPI) [7] and port-based techniques [8] have been widely used. DPI excels in accurately classifying traffic by examining the payload of packets, making it highly effective for identifying QoS requirements. However, DPI is computationally intensive and can become a bottleneck in high-traffic networks. Port-based techniques, on the other hand, rely on predefined port numbers to classify traffic. While this method is less computationally demanding, it is increasingly ineffective as modern applications often use dynamic or non-standard ports.

A different approach that the industry is opting for, is to use machine learning (ML) algorithms for traffic classification [9]. Unlike DPI and port-based, these algorithms are can be smart and flexible, requirements for classifying networks that have different means of transferring data. Machine learning (ML), a branch of artificial intelligence, involves the scientific study of algorithms and statistical models that enable computer systems to perform specific tasks without explicit instructions, instead relying on patterns and inference. There have been other works that focused on implementing ML models to predict corresponding applications using flow-based features, but they have not focused on creating models that can be implemented on devices that have limited computational power and memory space.

In this study, different ML algorithms will be used to predict the type of device connected to the network, and from the type, the QoS of the devices can be determined. The performance of the models will be compared to see how well each one of them is performing for the given task. The evaluation metrics are accuracy, memory, and computational requirements for implementing the model. The goal is to develop a framework tailored for networks consisting of IoT devices. This framework will be implemented on edge devices with limited computational power. This is mainly a comparative study of ML Algorithms for Network Traffic Classification, to determine which are suitable for lightweight devices.

The lightweight Edge Gateway-as-a-Service (GaaS) [10] exemplifies a typical edge device designed for IoT applications. Performance evaluations reveal that it effectively handles CPU, memory, disk I/O, and network tasks with minimal overhead. The device, particularly the Raspberry Pi 3 model can efficiently operate with up to 1 GB of RAM and storage capacities extending to 16 GB. These metrics will serve as the criteria for whether an ML model is feasible for the classification tasks.

Due to time constraints, training and testing of the algorithms have been conducted using a dataset sourced from researchers external to the University of Twente, which is permissible for further utilization as per the dataset creators' permissions. The dataset will be further explained in the methodology section III.

The rest of the paper is organized as follows: Section II provides an overview of related work. The methodology will be presented in Section III. The evaluation process and the results are in Section IV and finally, the conclusion is in Section V.

II. Related work

Over the last decade, using ML learning as a way of handling the QoS has been studied extensively. Overall different types of Machine Learning algorithms were considered, supervised, unsupervised and even semi-supervised. Supervised learning is a type of machine learning where the model is trained on a labeled dataset. This means that each training example is paired with an output label. The goal is for the model to learn a mapping from inputs to outputs that can be used to predict the label for new, unseen data. Contrarily, Unsupervised learning is a type of machine learning where the model is trained on data that does not have labeled outputs. Finally, Semi-supervised learning falls between supervised and unsupervised learning. It uses a small amount of labeled data along with a large amount of unlabeled data for training. This approach can significantly improve learning accuracy when acquiring a fully labeled dataset is expensive or time-consuming. Semi-supervised learning is particularly useful when the cost of labeling data is high but there is an abundance of unlabeled data. All of the models used in this paper are Supervised.

Owusu and Nayak (2020) [11] present a machine learning-based approach for traffic classification in Software-Defined Networking (SDN) integrated with IoT (SDN-IoT) networks. Software-Defined Networking (SDN) decouples the network control plane from the data plane, centralizing network management in a software-based controller that enables dynamic and programmable network configurations. The paper proposes the use of three machine learning algorithms—Random Forest, Decision Tree, and K-Nearest Neighbors—to classify network traffic in SDN-IoT environments. The authors also compare two feature selection methods, Sequential Feature Selection (SFS) and Shapley Additive Explanations (SHAP), to identify the most impactful features for classification. Their experiments reveal that the Random Forest classifier combined with SFS achieves the highest accuracy (0.833) using just six features. Classifiers were applied on a ToR dataset for classification [12]. The ToR dataset is encrypted and contains 8 types of traffic: browsing, chat, audio-streaming, videostreaming, mail, VOIP, P2P, and file transfer. The results were not particularly high and they have not considered whether these models can be implemented in edge gateway devices.

AlZoman and Alenazi [13] conducted a comprehensive study on traffic classification techniques for smart city networks, focusing on the performance and efficiency of different methods in managing network traffic. Smart city networks, characterized by diverse applications with specific Quality of Service (QoS) requirements, present unique challenges for network management. The authors evaluate four supervised machine learning algorithms—Support Vector Machine (SVM), Random Forest (RF), K-Nearest Neighbors (KNN), and Decision Tree (DT)—for their effectiveness in traffic classification. The study also compares these machine learning approaches to the traditional port-based method. The evaluation results indicate that the Decision Tree algorithm achieves the highest average accuracy of 99.18%, followed by Random Forest, while K-Nearest Neighbors performs the least effectively with an accuracy of 97.16%. The ML algorithms, particularly DT and RF, significantly outperform the port-based method in terms of accuracy and efficiency. The authors also considered real-time applications, which is why training and execution time was considered for each of the ML models. The result are quite promising, however, the authors have not mentioned how big the models are and how much memory is required.

Sivanathan et al. (2019) [14] present a robust framework for classifying IoT devices in smart environments based on network traffic characteristics. The study addresses the challenge faced by operators in identifying and monitoring the myriad of IoT devices

connected to their networks, which is critical for ensuring device functionality and cybersecurity. The authors instrument a smart environment with 28 different IoT devices, including cameras, lights, plugs, motion sensors, appliances, and health monitors, and collect traffic traces over six months. The dataset created by the authors, was also used in this paper. The research involves a multi-stage machine learning-based classification algorithm that leverages statistical attributes of network traffic, such as activity cycles, port numbers, signaling patterns, and cipher suites. This algorithm successfully identifies specific IoT devices with around 99% accuracy. The paper also discusses the trade-offs between cost, speed, and performance when deploying this classification framework in real-time, emphasizing the practicality of using network traffic characteristics for device visibility without requiring specialized hardware or protocols. Even though, the executions time is not mentioned or similar, neither are the memory requirements.

III. Methodology

Dataset for Evaluation

As mentioned in I, the dataset used in this work is created by UNSW Sydney researchers [14]. The data was collected in the lab, where different IoT and non-IoT devices were connected to the gateway. Information about the traffic was collected at the network access point and saved in both PCAP and CSV files. In the case of the CSV file, each row consisted of a single packet recorded. The network traffic was monitored for 20 days, while devices were working.

IoT devices consisted of Nest Dropcam, Samsung SmartCam, iHome, TP-Link Smart Plug, NEST Protect smoke alarm, Netatmo, and others. Since there are various devices, each one of them can be classified as one of these six types: cameras, health-monitor, controllers/hubs, energy management devices, appliances, and non-IoT. The type of device served as the label used for the models to make predictions, as it allows for easily setting the QoS requirements for that specific device. Table [1] shows the different types of device categories and their respective quantity.

As it was convenient to use, the CSV files were the ones used for training and testing the models. However, one issue with the CSV files was that they did not indicate the originating devices or their types. Fortunately, the dataset creators provided a list of devices along with their corresponding MAC addresses. This allows for mapping the MAC addresses to the device types, effectively resolving the issue of missing labels. The other features of the dataset are Packet ID, TIME (timestamp of packet capture), Size (packet size in bytes), eth.src (source device MAC address), eth.dst (destination device MAC address), IP.src (source device IP address), IP.dst (destination device IP address), IP.proto (protocol used by the packet, e.g., TCP, UDP), port.src (source port number), and port.dst (destination port number). Further, for each day on average 800 thousand packets were captured. In this work only the data from September 23rd to October 10th was used as it was enough to train and test the different models.

Table 1: Overview of types of devices in the dataset

Device Category	**Count**	**Percentage (%)**
Cameras	2,527,215	28.6
Non-IoT	2,311,906	26.2
Health-Monitor	565,079	6.4
Hubs/Controllers	522,818	5.9
Energy Management	318,764	3.6
Appliances	162,238	1.8

Machine Learning Models

The ML models that their performance will be tested on are: Decision Tree (DT), Random Forest (RF), and Neural Network (NN). As the task at hand is classification, these models perform are the most suitable. Support Vector Machine (SVM) is also an appropriate candidate model, however, due to its complexity that is between $\mathcal{O}(n^2)$ and $\mathcal{O}(n^3)$, training SVM model would take too long to train and would likely have same or lower performance compared to the other models.

Decision tree is a popular ML model used for both classification and regression tasks. It works by recursively partitioning the data space into subsets based on the value of input features, ultimately forming a tree-like structure where each internal node represents a decision based on an attribute, each branch represents the outcome of the decision, and each leaf node represents a final output or class label. The process begins with the entire dataset at the root, and the algorithm selects the attribute that best splits the data into distinct classes or values. This selection is typically based on criteria such as Information Gain for classification tasks, calculated using entropy, or the Gini Index. The splitting continues until the stopping criteria are met, which can include reaching a maximum tree depth, having a minimum number of samples per leaf, or achieving a sufficiently homogeneous node.

Random Forest is an ensemble learning method that constructs multiple decision trees and combines their predictions to achieve greater accuracy and stability. This approach leverages the power of multiple decision trees to improve performance and reduce overfitting, a common issue with single decision trees. The Random Forest algorithm operates by creating a large number of decision trees during training and combining their outputs to make a final prediction, which is the mode of the classes for classification tasks.

Training a Random Forest involves several key steps. First, multiple samples are generated from the original dataset through random sampling with replacement. Each of these samples is then used to train an individual decision tree. During the construction of each tree, a random subset of features is selected at each split, rather than considering all features. This process ensures that each tree is unique and captures different patterns in the data, enhancing the diversity of the model. Mathematically, let $\{T_1(x), T_2(x), \ldots, T_m(x)\}$ be the set of decision trees in the forest. For a classification problem, the Random Forest prediction $\hat{y}$ for an input x is given by the majority vote:

$$\hat{y} = \text{mode}\{T_1(x), T_2(x), \ldots, T_m(x)\}$$

where m is the number of trees in the forest. By aggregating the predictions of multiple trees, Random Forests reduce the variance of the model and improve generalization. Additionally, they provide measures of feature importance, which can be valuable for understanding the underlying data. The robustness, ease of use, and versatility of Random Forests make them an effective tool for various machine learning applications, especially in handling large datasets with complex interactions among features.

Neural Networks (NNs) are a class of machine learning models inspired by the human brain's structure and function. They consist of layers of interconnected nodes, or neurons, where each connection has an associated weight. NNs are highly versatile and can be used for a variety of tasks, including classification, regression, and pattern recognition. A basic NN architecture includes an input layer, one or more hidden layers, and an output layer. Each neuron in a layer receives input from neurons in the previous layer, processes it using a weighted sum and an activation function, and passes the result to the next layer. Mathematically, the output of a neuron j in layer l can be expressed as:

$$h_j^{(l)} = \sigma\left(\sum_i w_{ij}^{(l)} h_i^{(l-1)} + b_j^{(l)}\right)$$

where $h_j^{(l)}$ is the activation of neuron j in layer l, $w_{ij}^{(l)}$ is the weight from neuron i in layer $l-1$ to neuron j, $b_j^{(l)}$ is the bias term, and σ is the activation function (e.g., sigmoid, ReLU).

Training a NN involves adjusting the weights and biases to minimize a loss function, which measures the difference between the predicted and actual outputs. This

is typically done using an optimization algorithm such as stochastic gradient descent (SGD) and the backpropagation algorithm, which calculates the gradient of the loss function with respect to each weight by applying the chain rule. The weights are then updated in the direction that reduces the loss. NNs are capable of capturing complex relationships in data due to their deep architectures and non-linear activation functions, making them powerful tools for modern machine learning applications. Their ability to learn from data and improve over time makes them particularly suitable for tasks where traditional models like decision trees may struggle to achieve high performance.

IV. Evaluation

Data Preprocessing

As mentioned in III, the original dataset consists of 10 features. However, some features like packet ID and time do not carry significant information. Therefore, we need to employ feature engineering. A common approach is to use flow statistical features instead of individual packet attributes.

In network systems, packets are defined as a unit of data that is transmitted across a network, whereas flows represent a sequence of packets sent from a particular source to a particular destination over a network. Flows are used to manage and analyze the data traffic in networks, providing insights into the behavior of the network and the applications running over it. This is why working with flows is preferable, as they carry more useful information, which the model can later use to accurately make predictions.

The flow attributes that are used as features in the models are: the mean packet size of a specific flow, the maximum size of packets transported, total packets in a flow, total bytes, byte rate, packet rate, and mean and maximum inter-arrival time. The packet rate is defined as the total packets divided by the flow duration, similar to the byte rate. Inter-arrival time refers to the duration between the arrival of consecutive packets in a given flow. These flow statistics are chosen because they are easily calculated and suitable for real-time computations, which is essential for the models' use.

Finally, the combined features that the models are trained and tested on are: Source IP address, Destination IP address, Source Ethernet address, total packets in a flow, total bytes in a flow, mean packet size, maximum packet size, packet rate, byte rate, mean inter-arrival time, and maximum inter-arrival time.

Flow intervals refer to the periods during which flows are observed and analyzed. As this is an important parameter, the impact of flow intervals is studied in this work. The chosen intervals are 15 seconds, 30 seconds, 60 seconds, and 300 seconds (5 minutes). For each model, the training and testing data is transformed into four variations. For training the models, datasets from September 16th to September 29th and October 4th and October 10th are used. To test the performance of each model, datasets from September 30th to October 5th, excluding October 4th, are utilized.

Evaluation metrics

The performance metrics are: Accuracy, F1 score, model size, and model prediction time.
Accuracy is calculated as:

$$\text{Accuracy} = \frac{\text{Number of correct classifications}}{\text{Number of all tested samples}} \tag{1}$$

The F1 score is a metric that combines precision and recall into a single value by taking their harmonic mean. It provides a balance between precision and recall.

$$\text{F1 Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2}$$

Recall measures the ability of a model to identify all relevant instances in the dataset. It is defined as the ratio of true positive predictions to the total number of actual positive instances. While precision is a metric that measures the accuracy of positive

predictions made by a model. It is defined as the ratio of true positive predictions to the total number of positive predictions. Precision and recall are crucial metrics as they provide detailed insights into the model's performance. Precision measures the accuracy of positive predictions. Recall measures the model's ability to identify all relevant instances.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{3}$$

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{4}$$

Prediction time is the time it takes a model to make a prediction for one example. In this setup, the total time taken to predict the testing set will be measured, then it will be divided by the total samples in the set to calculate the average time it takes to make a prediction. The model size is just the size of the architecture of the model in MBs.

Experimental design

The code is implemented in Jupiter Notebook. Standard machine learning libraries were utilized in this study. Specifically, Scikit-learn was employed for the DT and RF models, while TensorFlow was used for the NN models. To find the most suitable hyper-parameters, fine-tuning was done on the models. The table below shows the corresponding parameters.

Additionally, the ADAM or Adaptive Moment Estimation optimizer was used instead of the standard Stochastic Gradient Descent optimizer, as it leads to faster convergence and is less likely to get stuck in local minima. The ADAM implementation in TensorFlow dynamically adjusts the learning rate, so it does not need to be specified manually. A standard scaler was included as part of the model to normalize the input data. Dropout layers were added between each layer to prevent overfitting. Early stopping was employed to avoid wasted computational time. The batch size and epochs were set to 64 and 50, respectively.

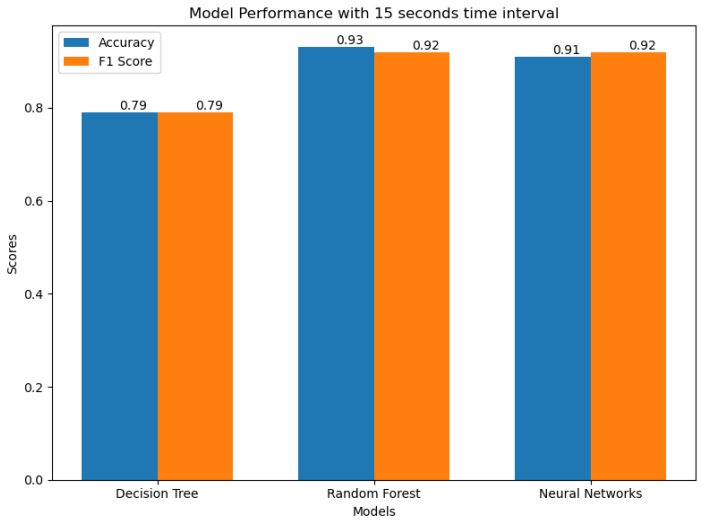
In all implementations, the random state and seed were set to 100 to ensure that the results were reproducible. All of the models were trained on a local machine, ASUS Vivobook 15 Pro with 16 GB RAM, 11th Gen Intel R core i9 2.5 GHz processor. The operating system was OS Windows 11. The system type can affect the time used to train and test the algorithms.

Table 2: Hyperparameters of the different ML models

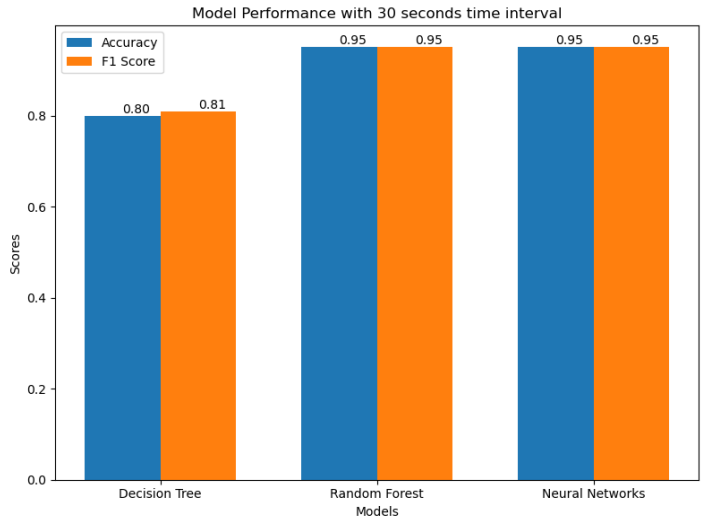
Model	**Hyperparameter**	**15 seconds**	**30 seconds**	**60 seconds**	**300 seconds**
Decision Tree	Max Depth	7	7	7	7
	Min Samples Leaf	1	1	1	1
	Min Samples Split	2	2	2	2
Random Forest	Max Depth	6	6	6	6
	Min Samples Leaf	1	1	1	1
	Min Samples Split	2	2	5	2
	Number of Estimators	80	200	100	200
Neural Network	Hidden Layers	6	6	6	6
	Activation function	ReLu	ReLu	ReLu	ReLu

Model Performance

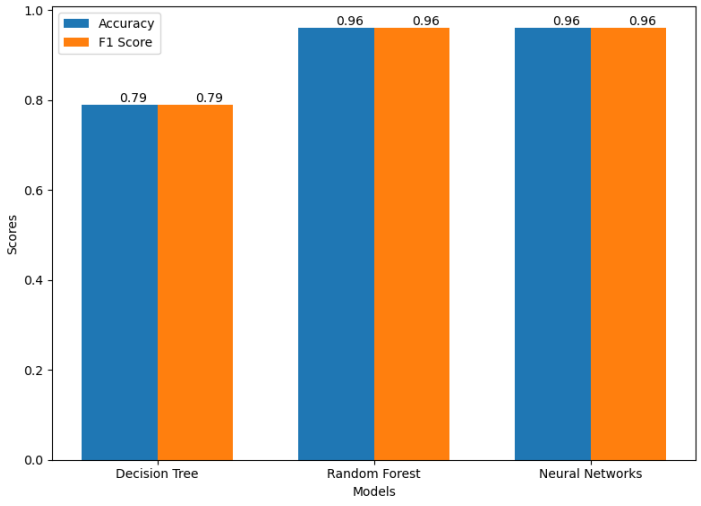
Below are the graphs showcasing the performance of the models. Overall, all the models correctly guessed more than 70% of the instances As it can be seen the worst-performing model is DT on 300-second intervals, the accuracy and f1 score are both 0.72. In comparison, the best performing are RF and NN on 60-second interval with 0.96 accuracy and f1 score. In all of the different time intervals, RF and NN outperform DT. All Neural



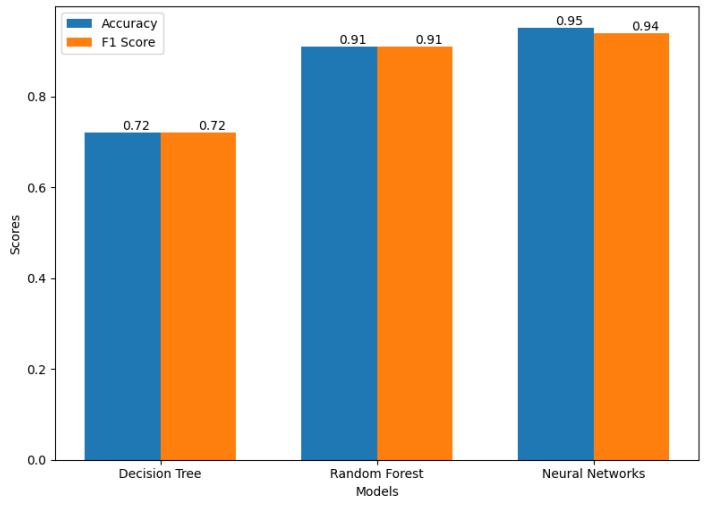
(a) Model performance at 15s



(b) Model performance at 30s



(c) Model performance at 60s



(d) Model performance at 300s

Figure 1: Performance of the models

Network (NN) and Random Forest (RF) models achieve accuracy and F1 scores above 0.90.

F1 score and accuracy are similar for each model, indicating that the models perform consistently across different classes, even though the datasets are not completely evenly distributed between various types of devices [1]. This is a desirable outcome, indicating balanced performance. As mentioned in III, the Decision Tree (DT) model can easily overfit and struggle with unseen data. An improved version of DT is the Random Forest (RF) model, which combines multiple decision trees to improve generalization and performance. Neural Networks (NN) are also complex models with significant nonlinearity, allowing them to capture complex relationships within the data.

The datasets, differentiated by 15-second, 30-second, 60-second, and 300-second intervals, are essentially the same dataset evaluated over different time frames. As the interval increases, the flow features become more accurate, capturing more comprehensive statistics. However, since the dataset remains the same, longer intervals result in fewer training examples because more packet instances are aggregated into a single training example. This reduction in training data can lead to worse performance. Therefore, the preferred interval lies in the intermediate range, balancing the accuracy of flow features with a sufficient number of training examples. This balance is reflected in the performance of the models, where intermediate intervals tend to achieve optimal results.

Table 3 shows the average time it takes for each model to predict one flow. The fastest one is DT with 15 seconds interval with 3 µs. The slowest is Neural Networks, 69 µs. The best-performing models in terms of accuracy, RF 60sec and NN 60sec, have a prediction time of 10 µs and 45 µs respectively.

It can be observed that overall, DT models are the fastest, RF models are slightly slower, and NN models are the slowest by a significant margin. NN models have a

complex architecture with numerous parameters and computations, including multiple layers of neurons and intricate weight adjustments, which contribute to their higher prediction time. In contrast, DTs are quite fast due to their simpler structure, and RFs, being an ensemble of multiple Decision Trees, exhibit slightly slower performance but offer improved accuracy and robustness.

Table 4 displays the size of each machine learning model in megabytes (MB) and their peak memory usage during prediction. It is observed that the Decision Tree (DT) model maintains a consistent size of 0.01 MB across all time intervals, which is the smallest among the models. This consistency is expected as all DT models share the same architecture, as indicated in Table 2. The same applies to Neural Network (NN) models, which also retain a uniform size of 0.58 MB due to their fixed architecture.

In contrast, the Random Forest (RF) models exhibit significant variations in size. This variation is attributed to the differing number of estimators used for each time interval. Specifically, the RF model for the 15-second interval uses 80 estimators, while the 30-second and 300-second intervals utilize 200 estimators. Consequently, the RF models for the 30-second and 300-second intervals are the largest, each with a size of approximately 1.9 MB.

Regarding peak memory usage, DT and RF models exhibit relatively stable memory requirements, averaging around 250 MB. However, NN models demonstrate a markedly higher peak memory usage, especially for the shorter 15-second interval, where the memory usage peaks at 2461.05 MB. This high memory demand decreases as the time interval increases, with the lowest memory usage recorded at 185.06 MB for the 300-second interval. This might be due to the fact that the testing examples are decreasing in number as the time interval increases.

Finally, the best performing models are RF and NN with 60 second time interval. However, RF has higher model size than NN but requires less memory to work. RF and NN with 30 seconds interval have slightly worse values than the 60sec counterpart. DT with 300 sec is undesirable in any situation as it is inaccurate. NN with 300 seconds has also satisfactory performance and low memory usage and doest take much memory space. The fastest and smallest is DT but the accuracy is 0.80.

Finally, the best performing models are the Random Forest (RF) and Neural Network (NN) models with a 60-second time interval. While the RF model has a larger model size compared to the NN model, it requires less peak memory during operation. Both models offer a balance between performance and memory efficiency.The RF and NN models with a 30-second interval show slightly lower performance metrics compared to their 60-second counterparts. I n contrast, the Decision Tree (DT) model with a 300-second interval is undesirable due to its poor accuracy, making it unsuitable for scenarios where precise predictions are critical. The NN model with a 300-second interval, however, offers satisfactory performance and relatively low memory usage. The DT model, while being the fastest and having the smallest size, achieves an accuracy of only 0.80.

Table 3: Prediction Time Per Flow for Different ML Models

Model	**15s (µs/flow)**	**30s (µs/flow)**	**60s (µs/flow)**	**300s (µs/flow)**
Decision Tree	3	5	5	25
Random Forest	8	18	10	21
Neural Networks	34	50	45	69

Table 4: Model Size and Peak Memory Usage during Prediction

Interval	**Model**	**Size (MB)**	**Peak Memory Usage (MB)**
15 sec	DT	0.010	223.28
	RF	0.757	290.90
	NN	0.580	2461.05
30 sec	DT	0.010	233.00
	RF	1.859	282.23
	NN	0.580	2121.80
60 sec	DT	0.010	225.12
	RF	1.002	270.44
	NN	0.580	1288.58
300 sec	DT	0.010	236.39
	RF	1.910	249.45
	NN	0.580	185.06

V. Conclusion

In this study, a comparative analysis was conducted on network traffic classification using machine learning algorithms. The algorithms compared were Decision Tree (DT), Random Forest (RF), and Neural Networks (NN). These models were applied to a dataset featuring both IoT and non-IoT devices, utilizing primarily flow statistical features. The classification labels were the types of devices, which is crucial for determining Quality of Service (QoS).

The evaluation metrics included F1 score, accuracy, model size, peak memory usage, and prediction time. Additionally, different time intervals were examined to understand their impact on these metrics.

The findings indicate that DT models, while being the fastest and most memory-efficient with a consistent size of 0.01 MB, achieve lower accuracy, making them less suitable for scenarios requiring high precision. RF and NN models with a 60-second time interval both exhibited the highest accuracy and F1 score, each achieving 0.96. Despite having a larger model size, the RF model required less peak memory during operation compared to the NN model.

NN models with a 300-second time interval also performed well, with both accuracy and F1 score at 0.95. This configuration had the lowest peak memory usage, making it memory-efficient, although it was also the slowest in terms of prediction time.

Overall, RF models emerged as the best performers, offering the highest accuracy while maintaining computational efficiency. They achieved a good balance between model size and memory usage, making them suitable for various applications. The study highlights the importance of selecting appropriate time intervals to optimize the balance between the number of training examples and the accuracy of flow features. All the machine learning models discussed in this study can be implemented on an edge device, similar to the one discussed in Section I as the hardware characteristics allow it. Depending on the specific task and constraints, certain models may be more suitable than others.

In summary, the choice of model and time interval should be guided by the specific requirements and constraints of the application. While DT models offer speed and memory efficiency, they fall short in accuracy. NN models, although memory-intensive, provide robust performance and the ability to capture complex data relationships. RF models stand out for their optimal balance of accuracy, memory usage, and computational speed, making them a versatile choice for network traffic analysis.

References

[1] James Manyika, Michael Chui, Peter Bisson, Lola Woetzel, Richard Dobbs, Jacques Bughin, and Dan Aharon. *Unlocking the potential of the Internet of Things*. 2015.

[2] Sachin Kumar, Prayag Tiwari, and Mikhail Zymbler. "Internet of Things is a revolutionary approach for future technology enhancement: a review". en. In: *Journal of big data* 6.1 (2019). ISSN: 2196-1115. DOI: 10.1186/s40537-019-0268-2. URL: http://dx.doi.org/10.1186/s40537-019-0268-2.

[3] Heiner Lasi, Peter Fettke, Hans-Georg Kemper, Thomas Feld, and Michael Hoffmann. "Industry 4.0". In: *Business amp; Information Systems Engineering* 6.4 (June 2014), pp. 239–242. DOI: 10.1007/s12599-014-0334-4.

[4] Li Da Xu, Wu He, and Shancang Li. "Internet of Things in Industries: A Survey". In: *IEEE Transactions on Industrial Informatics* 10.4 (2014), pp. 2233–2243. DOI: 10.1109/TII.2014.2300753.

[5] Hao Chen, Xueqin Jia, and Heng Li. "A brief introduction to IoT gateway". In: *IET International Conference on Communication Technology and Application (ICCTA 2011)*. 2011, pp. 610–613. DOI: 10.1049/cp.2011.0740.

[6] Gunjan Beniwal and Anita Singhrova. ""A systematic literature review on IoT gateways"". In: *Journal of King Saud University - Computer and Information Sciences* 34.10, Part B (2022), pp. 9541–9563. ISSN: 1319-1578. DOI: https://doi.org/10.1016/j.jksuci.2021.11.007. URL: https://www.sciencedirect.com/science/article/pii/S1319157821003219.

[7] Reham Taher El-Maghraby, Nada Mostafa Abd Elazim, and Ayman M. Bahaa-Eldin. "A survey on deep packet inspection". In: *2017 12th International Conference on Computer Engineering and Systems (ICCES)*. 2017, pp. 188–197. DOI: 10.1109/ICCES.2017.8275301.

[8] Andrew W. Moore and Konstantina Papagiannaki. "Toward the Accurate Identification of Network Applications". In: *Passive and Active Network Measurement*. Ed. by Constantinos Dovrolis. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 41–54. ISBN: 978-3-540-31966-5.

[9] Fannia Pacheco, Ernesto Exposito, Mathieu Gineste, Cedric Baudoin, and Jose Aguilar. "Towards the Deployment of Machine Learning Solutions in Network Traffic Classification: A Systematic Survey". In: *IEEE Communications Surveys Tutorials* 21.2 (2019), pp. 1988–2014. DOI: 10.1109/COMST.2018.2883147.

[10] Roberto Morabito, Riccardo Petrolo, Valeria Loscrí, and Nathalie Mitton. "Enabling a lightweight Edge Gateway-as-a-Service for the Internet of Things". In: *2016 7th International Conference on the Network of the Future (NOF)*. 2016, pp. 1–5. DOI: 10.1109/NOF.2016.7810110.

[11] Ampratwum Isaac Owusu and Amiya Nayak. "An Intelligent Traffic Classification in SDN-IoT: A Machine Learning Approach". In: *2020 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom)*. 2020, pp. 1–6. DOI: 10.1109/BlackSeaCom48709.2020.9235019.

[12] Arash Habibi Lashkari, Gerard Draper Gil, Mohammad Mamun, and Ali Ghorbani. "Characterization of Tor Traffic using Time based Features". In: Jan. 2017, pp. 253–262. DOI: 10.5220/0006105602530262.

[13] Razan M. AlZoman and Mohammed J. F. Alenazi. "A comparative study of traffic classification techniques for smart city networks". en. In: *Sensors (Basel, Switzerland)* 21.14 (2021), p. 4677. ISSN: 1424-8220. DOI: 10.3390/s21144677. URL: https://www.mdpi.com/1424-8220/21/14/4677.

[14] Arunan Sivanathan, Hassan Habibi Gharakheili, Franco Loi, Adam Radford, Chamith Wijenayake, Arun Vishwanath, and Vijay Sivaraman. "Classifying IoT Devices in Smart Environments Using Network Traffic Characteristics". In: *IEEE Transactions on Mobile Computing* 18.8 (2019), pp. 1745–1759. DOI: 10.1109/TMC.2018.2866249.