2024 11th International Conference on Future Internet of Things and Cloud (FiCloud) | 979-8-3315-2719-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/FiCloud62933.2024.00055

Shallow vs. Deep Learning: Prioritizing Efficiency in Next Generation Networks

Rafael Teixeira*†, Leonardo Almeida†, Pedro Rodrigues†, Mário Antunes*, Diogo Gomes* and Rui L. Aguiar*
*Instituto de Telecomunicações, Universidade de Aveiro
Email: {rafaelgteixeira,mario.antunes,dgomes,ruilaa}@av.it.pt
†DETI, Universidade de Aveiro
Email: {rafaelgteixeira,leonardoalmeida,pedrofrodrigues}@ua.pt

***Abstract*—With the exponential growth of mobile network traffic and diverse application demands, traditional network management methods struggle to meet the stringent requirements of 5th Generation (5G) and Beyond-5G (B5G) networks. Artificial Intelligence (AI), particularly Deep Learning (DL), has emerged as a promising solution. However, large and complex DL models can be computationally expensive for real-time applications. This paper investigates the potential of shallow Machine Learning (ML) models for 5G/B5G tasks. We compare the performance and training/inference time of shallow ML models with state-of-the-art DL models in two key tasks: network slicing attribution and Intrusion Detection Systems (IDS). The results demonstrate that shallow models achieve comparable performance with significantly faster training and prediction, leading to an acceleration of over 90% in most cases.**

***Index Terms*—Shallow Learning, Deep Learning, Intrusion Detection System, Network Slicing, 5G**

I. Introduction

In recent years, mobile network traffic has grown exponentially. Similarly, the number of applications with time-sensitive or high throughput characteristics has also increased. To answer the divergent requirements of the various consumers, 3rd Generation Partnership Project (3GPP) Long Term Evolution (LTE)'s rigid framework needed to change [1]. So, the Fifth-Generation Network (5G) was developed as a highly configurable and virtualized system to provide a cost-effective approach to on-demand services.

Besides the increased flexibility, 5G requires support up to 1000 times higher data volumes, low latency (1ms), and massive device connectivity sustaining 10-100 times more devices. Furthermore, these characteristics must be achieved with ultra-high reliability (99.999%) [2]. The combination of these stringent requirements and the increased network complexity makes traditional methods for network design, deployment, and optimization no longer feasible.

Artificial Intelligence (AI) is seen as the means to reach the stringent requirements of 5G while improving energy efficiency and security. More specifically, Deep Learning (DL), a subfield of AI, is a popular approach [3]. Its popularity in telecommunications derives from the results obtained in other tasks such as Natural Language Processing (NLP) or Computer Vision and the availability of large amounts of heterogeneous data, which makes some problems too complex for Shallow Machine Learning (ML) models [3].

In Beyond Fifth-Generation Networks (B5G), the role of AI is even more critical as it will be a key enabler of the even stricter performance requirements and a crucial tool to improve user experience. That said, the correct implementation of AI/ML models in 5G is of enormous importance as it will significantly impact how AI/ML will be applied in future networks.

Although bigger Deep Neural Networks (DNNs) can be built and applied in NLP or Computer Vision, the time the model has to train and infer in some critical telecommunication tasks does not allow DNNs to be applied. Furthermore, some telecommunications networks will be too vast for a single model to access the information from every node, limiting the available information and, consequently, the practical application of deeper DNNs. Furthermore, some examples exist where shallow ML models achieve the same results as DNNs [4].

Given how DL models might not always meet the stringent 5G/B5G requirements, this paper compares the current DL proposals with shallow ML models in two typical 5G tasks. In the analysis, besides the performance, the training and prediction times are also considered. The results show that most shallow models achieve the same performance using a fraction of the training and prediction time, leading to a training and inference acceleration of more than 90% for most shallow models.

The main contributions of this work are the following: 1) An extensive analysis of shallow ML models for network slicing and Intrusion Detection System (IDS); 2) Comparison between the state-of-the-art deep ML models and the classical shallow ML models in the mentioned tasks; 3) Discussion of the trade-offs between deep and shallow models; 4) Discussion of the models' applicability in a real-world scenario.

The remainder of the paper is organized as follows. Section II briefly introduces network slicing, IDSs, and the shallow models used in the experiments. Section III describes the experiments performed, the datasets used, the collected metrics, and the DL models. In section IV, the results from the various experiments are presented. Section V discusses the obtained results and highlights the problems in ML development and future research directions. Finally, section VI concludes the work by giving the essential findings and possible extensions of this work.

979-8-3315-2719-8/24/$31.00 ©2024 IEEE
DOI 10.1109/FiCloud62933.2024.00055

II. Background

This section provides the necessary background regarding the tasks considered (Network Slice Attribution and IDSs) and ML, highlighting the shallow ML models used.

A. Network slicing

First introduced by the Next Generation Mobile Network Alliance [5], network slicing leverages network function virtualization and software-defined networks to create multiple independent virtual networks over the same physical one [6]. These networks are called slices and can be configured to offer specific network capabilities and characteristics, such as bandwidth, latency, and reliability. This is necessary to ensure a cost-effective way to meet the different consumers' heterogeneous requirements, as developing a network capable of achieving every requirement would be unfeasible [7].

For 5G, 3GPP defined three main service categories for network slicing: the enhanced Mobile Broadband (eMBB), the Ultra Reliable Low Latency Communications (URLLC), and the massive Machine Type Communications (mMTC). The eMBB enables high throughput applications, such as high-definition video streaming or mobile TV, by creating a network with considerable bandwidth. The trade-off in this profile is that the applications are expected to be less latency-sensitive. In cases where a massive number of devices need to be connected (e.g. Internet of Things (IoT)), the profile to use is mMTC. Here, the devices are not expected to require high bandwidth or low latency, just the connection between them. Lastly, when consistent low latency and high reliability are mandatory, the URLLC profile should be used [8].

In network slicing, ML models are used for two primary use cases: slice management, where the models manage the resources attributed to each slice and how many slices of each type are needed, or end-user slice attribution, where the models decide to which slice the end-user requests go to.

In this work, the latter task is addressed, where given a set of slice requirements, the ML model attributes the request to one of the existing slices.

B. Intrusion Detection Systems

Despite many IoT devices having limited security capabilities due to resource constraints, IoT networks are becoming increasingly complex. This creates a significant challenge for security experts as they become harder to monitor and protect.

One promising approach to securing IoT networks involves using IDS powered by ML algorithms. These ML-based IDS systems can analyze network traffic and classify it as normal or malicious, and in case of malicious traffic, identify the type of attack.

Nonetheless, considering the constant changes in IoT networks related to removing/adding new devices, achieving a reliable IDS system can be difficult, as usually the model suffers from time drift. To overcome this issue, the models must be constantly retrained to keep up with the new attack types and benign flows and avoid false positives/negatives.

C. Machine learning models

Machine learning models can be divided into two major categories: shallow and deep learning. Shallow models are most models proposed before 2006, including Artificial Neural Networks (ANNs) with just one hidden layer [9]. They differ from DL models in their architecture as the latter models are just ANNs with more than one hidden layer called DNNs.

Although shallow models have difficulty solving complex tasks without feature engineering, their most significant trade-off is that when using appropriate features, they can train and predict faster. During the experiments, we used eight shallow models, Logistic Regression (LR), Support Vector Machine (SVM), Gaussian Naive Bayes (GaussianNB), K-Nearest Neighbors (k-NN), Decision Tree (DT), Random Forest (RF), AdaBoost, and Gradient Boosting. These models can be divided into individual (first five models) and ensemble models.

The simplest model is the LR. It estimates the probability of an event happening given features. The output is between zero and one, indicating certainty of the event not happening and happening, respectively [10]. After optimizing the weights, predictions are considered zero if the output is below 0.5 and one otherwise.

The SVM is similar to LR but maximizes margins between the decision boundary and examples [11]. This makes SVM a better classifier than LR, with improved resilience to outliers and better-fitted curves. It can handle non-linear separation using kernels, such as the Radial Basis Function (RBF) kernel.

The k-NN algorithm classifies examples based on the closest k terms using a voting system [11]. The class with the most votes becomes the label for the new data point. The performance depends on the number of neighbors and their influence on the decision.

The DT formulates class prediction as a tree structure where nodes verify features [10]. It's easily explainable and versatile, with options to limit depth for a trade-off between accuracy and prediction rate. Factors like the splitting criterion and considered features influence the tree's quality.

The final individual classifier, the GaussianNB, applies the Bayes theorem considering a "naive" assumption of independence between features [10]. In practice, for each example, the model outputs the probability of it belonging to the class given the features.

Ensemble classifiers aim to create robust and well-rounded classifiers by combining models. In the experiments, one bagging classifier and two boosting classifiers were used, the RF, Gradient Boosting, and AdaBoost,

The RF creates multiple DTs trained on different subsets of the data to reduce overfitting [11]. The DTs' results are combined through a weighted vote based on probability estimates, using the class with the highest mean probability as the final prediction.

Gradient Boosting is very similar to RF. However, instead of training the DTs independently of each other, each DT is trained sequentially. The output of the previously combined

trees is used to compute the pseudo-residuals, which are the target values for the next tree [12].

Lastly, the AdaBoost, similar to Gradient Boosting, also fits multiple models sequentially. The difference here is how the subsequent models are created. Instead of generating new target values in each iteration, it trains models specialized on the previous model's mistakes [12].

III. Experiments

Given that most papers regarding network slicing attribution and IDS use DL models and that most studies do not account for the training/inference cost, the discussion presented in the state-of-the-art lacks an important analysis. To mitigate this gap in the research, the experiments performed assessed the applicability of shallow models in these tasks and their cost. This section describes the used datasets, the hyperparameter optimization, the evaluation metrics considered, and the state-of-the-art DL models used for comparison.

The code used to run the experiments is available on GitHub[1]. All the experiments were performed on a virtual machine with 24 VCPUs, 64 GBs of RAM, and one RTX 2080. The DL and the shallow models were implemented in Python using the Keras API from Tensorflow and the Scikit-Learn library, respectively.

A. Datasets

To ensure generality in the results presented, three datasets were selected for slice attribution and two for IDS. This section briefly describes each dataset and where to find them.

1) Network Slice Attribution: The first dataset used, Slicing5G, is publicly available at IEEEDataPort[2] and was used in various publications [1], [6], [13]. The raw dataset is composed of eight input features and one output. The inputs are device and network Key Performance Indicators (KPIs), and the output is one of three classes (eMBB, URLLC, or mMTC). These classes represent three network slices, one for each Quality of Service (QoS) profile mentioned in subsection II-A.

Seven of the input features are categorical, so as a pre-processing step, they were transformed into one-hot-encoding features. The only exception is the 'time' feature, which remained unchanged. After the pre-processing, the dataset comprised 60 features and 466739 examples. The examples were then shuffled and divided into two data sets with ratios of 80/20 for training and testing. The final dataset comprises 373391 examples for training and 93348 for testing. A more detailed view of the dataset is presented in the original paper [1].

The second dataset, NetworkSlicing5G, although not publicly available, was provided after a request to the authors of [14], which proposed it. The raw dataset is similar to the first one being composed of eight input features and one output. However, some of the input features considered are different. The expected output is the same as the first dataset. In this dataset, five of the input features are categorical, so the same preprocessing was applied to these features. After that, the dataset was also shuffled and divided using the same ratios. The final dataset comprises 796 training examples and 199 testing examples.

The last dataset, NetSlice5G, is publicly available at Kaggle[3] and is already divided into training and testing. Nonetheless, the testing dataset does not have the expected labels for the input, so it was discarded, as it was not possible to extract accuracy metrics from it. This dataset has similar features to the other two, differing in the fact that this dataset was published with the categorical features already preprocessed as one-hot-encoding. The only step performed in this dataset was the data division using the same ratio as before, which resulted in a dataset with 25266 training examples and 6317 testing ones.

2) Intrusion Detection Systems: There are several well-known datasets that could be used to train and test IDS models, for example, the KDD98 [15], KDDCUP99 [16], or NSLKDD [17]. However, as highlighted in [18], even though these and other similar datasets are sufficiently large and trusted by the research community, they are outdated and do not inclusively reflect today's network traffic, which seriously limits their applicability within contemporary studies of network security.

Taking this into account, we decided to use two recent and widely used state-of-the-art datasets, namely UNSW-NB15 [19] and IoT-DNL[4], which contain network traffic data from IoT devices.

The UNSW-NB15 dataset is a synthetically generated dataset created by the Australian Centre for Cyber Security (ACCS). This dataset has a hybrid of the real modern normal and the contemporary synthesized attack activities of the network traffic. Comprising over 2 million records with 49 features, the UNSW-NB15 dataset is prepared to train both binary and multi-class classification models since each malicious activity is additionally labeled with one of the 9 attack categories, namely DoS, Analysis, Backdoor, Exploits, Fuzzers, Generic, Reconnaissance, Shellcode, and Worms.

To avoid complex preprocessing procedures, in this study, we used the NF-UNSW-NB15[5] preprocessed dataset published in [20] by the University of Queensland.

The IoT-Device-Network-Logs (IoT-DNL) is a dataset tailored for network-based Intrusion Detection Systems. This open-source dataset, initially introduced by Sahil Dixit and sourced from Kaggle, is specifically designed for evaluating anomaly-based IDS in wireless environments. The dataset is flow-based and labeled, focusing on IoT devices, and has undergone preprocessing to suit the requirements of network-based IDS.

This dataset's network logs were gathered by monitoring the network using ultrasonic sensors, specifically the Arduino and NodeMCU with the ESP8266 Wi-Fi module. These devices

[1]https://github.com/rgtzths/shallow_vs_deep_learning
[2]https://ieee-dataport.org/open-access/crawdad-umkcnetworkslicing5g
[3]https://www.kaggle.com/datasets/amohankumar/network-slicing-in-5g
[4]https://www.kaggle.com/datasets/speedwall10/iot-device-network-logs
[5]https://espace.library.uq.edu.au/view/UQ:ffbb0c1

transmit data to the server via Wi-Fi. In total, the dataset comprises 477,426 samples, each with 14 features.

Since the datasets were already preprocessed, the only steps performed were feature scaling and dataset division. The dataset division followed the same ratio as the network slicing attributions datasets, ending with 1912220/381940 training examples and 478055/95486 testing examples for the UNSW and IoT-DNL, respectively.

B. Shallow models optimization

Since some of the shallow models were not implemented for these datasets, they needed to be optimized to enable a fair comparison between the deep and shallow models. The optimization was done through a simple grid search where each model was cross-validated five times. The data used during this procedure was only the training set, and the optimization performed was considered only the most important hyperparameters, and a reduced amount of values were analyzed for them. The reduced amount of hyperparameters evaluated is intentional as the paper aims to compare shallow with deep models, not analyze model optimization. The full list of hyperparameters evaluated and values tested is available on the paper's GitHub[1].

The criteria for choosing the best hyperparameters were the same as used for model comparison: meaning, performance, and training/prediction times. The hyperparameters that were not evaluated kept their default values (check the Scikit-learn website for more information[6]).

C. Deep learning models

Although ideally, the DNNs models used were models proposed for the considered datasets, two of the slice attribution datasets did not have any models proposed. In this subsection, we overview the models considered for every dataset and explain how the models were obtained for the datasets that did not have one.

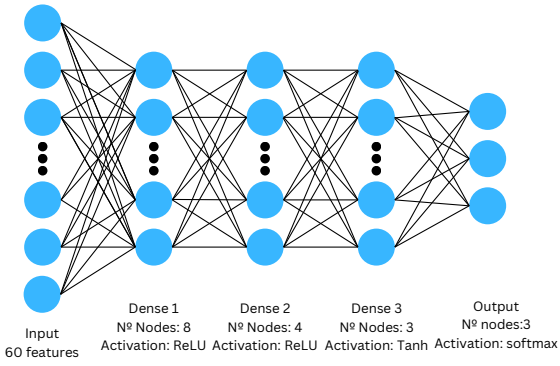

Fig. 1. Architecture of the DEEPSLICE model proposed in [1] and implemeted for the Slicing5G dataset.

1) Network Slice Attribution: In the slice attribution problem, most ML models implemented are DL ones [1], [6], [13]. However, there are also some cases where shallow learning models are also applied [6]. The deep learning model implemented for the Slicing5G dataset was specifically designed for it [1] and is a simple DNN with three dense layers with few nodes each, as depicted in Figure 1.

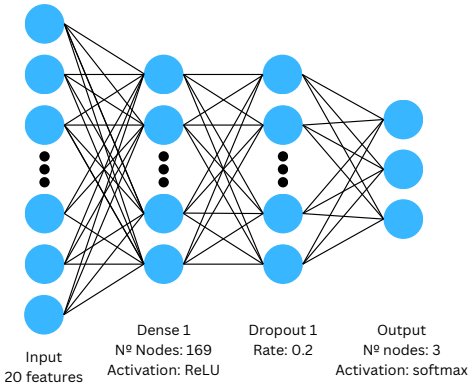

Fig. 2. Architecture of the model implemented for the NetworkSlicing5G dataset.

For the other two datasets, the same approach could not be applied as it was not possible to obtain valid DNNs for them. To mitigate this issue, a quick random search was performed to find suitable candidates to solve the datasets. Similarly to the shallow model optimization, the details regarding the considered hyperparameters and the results are available on the paper's GitHub[1]. The final model for the NetworkSlicing5G is presented in Figure 2, and the model for the NetSlice5G is presented in Figure 3.

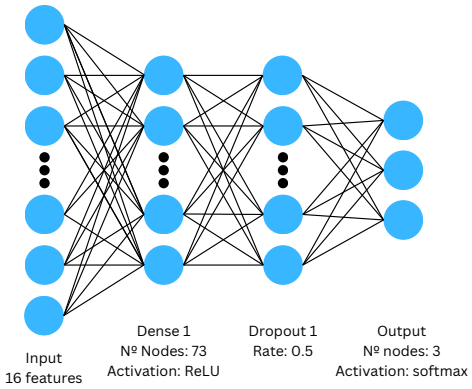

Fig. 3. Architecture of the model implemented for the NetSlice5G dataset.

Analyzing both models, it is clear that simple shallow neural networks were sufficient to achieve the best results as both models only have a single hidden dense layer followed by a dropout one. Nonetheless, they differ in the number of nodes in the hidden layer and the dropout value used.

2) Intrusion Detection Systems: Starting with the UNSW-NB15 dataset, in [21], the authors use 3 different DNNs, including Multi-layer Perceptron (MLP), Convolutional Neural

[6]https://scikit-learn.org/stable/

Network (CNN), and Recurrent Neural Network (RNN). All models performed well and achieved similar results, so we decided to use the MLP model, which is the simplest and fastest model to train. The model, presented in Figure 4, consists of 3 hidden layers with 128, 96, and 64 neurons, respectively, and a dropout layer with a dropout rate of 0.25 after the hidden layers. The activation function used in the hidden layers is the Rectified Linear Unit (ReLU) function, and the optimizer used is the Adam optimizer.

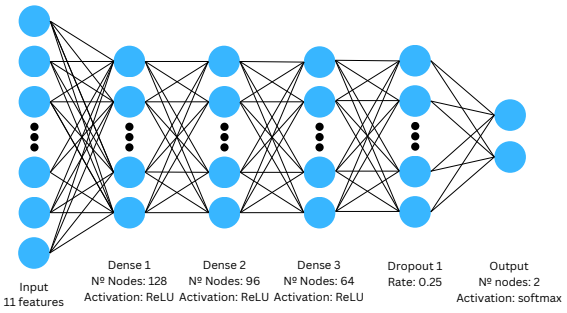

Fig. 4. Architecture of the model proposed in [21] and implemeted for the UNSW-NB15 dataset.

In [22], the authors explore the IoT-DNL dataset with a MLP model but do not specify the number of neurons in each layer. To replicate their model, we evaluated 18 different configurations of hidden layers and neurons and selected the best-performing one. The final model, presented in Figure 5, consists of 4 hidden layers with 64 neurons each and a dropout layer with a dropout rate of 0.1 between each hidden layer. The activation function used in the hidden layers is the ReLU function, and the model was trained using the Adam optimizer. Our evaluation of the model was made taking into account training and validation accuracy, and the complexity of the model.

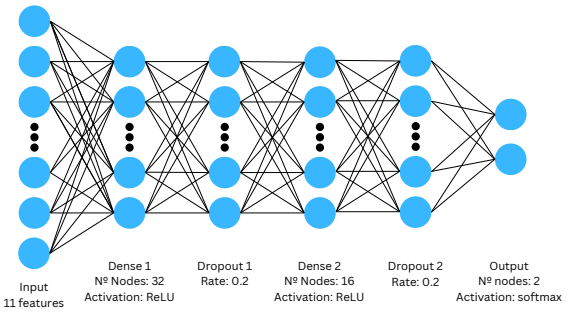

Fig. 5. Architecture of the model proposed in [22] and implemeted for the IoT-DNL dataset.

D. Model evaluation

Evaluating a model only considering its performance can lead to a false sense of success. Given the strict latency requirements of B5G, models need to predict accurately and fast, as otherwise, the network will not be able to handle the request efficiently. Furthermore, B5G networks will be significantly more dynamic, creating the need for models that can adapt quickly. Consequently, besides performance, two key metrics that should be considered are the training and prediction times.

To measure the training and prediction time correctly, we ran both training and prediction 30 times and averaged the results. Since we set a seed of 42 for every random generator, the models' initial state and training were always the same, meaning that any variation in training/prediction was a consequence of the machine.

Based on the training time from the DNNs the acceleration of the shallow ML models was calculated following Equation 1, where DNN_{time} was the time the DNN took to train/infer and the $model_{time}$ is the time a shallow model took.

$$Acceleration = (1 - \frac{model_{time}}{DNN_{time}}) * 100 \quad (1)$$

The performance was evaluated using Mathews Correlation Coefficient (MCC). Since there is some class imbalance in the datasets, we opted for a metric that penalizes wrong classifications harder than the typical accuracy.

IV. Results

Due to page limitations, this section will focus on comparing shallow ML models with the DL ones. The results of hyperparameter optimization of shallow and DL models are available on GitHub[1] and can be easily reproduced by executing the code.

A. Model Accuracy

Starting with the analysis of performance Tables I and II, present the results of every model for every dataset. The results for the first model, the DNN, represent the results obtained for the DNN implemented specifically for each dataset and presented in the previous section.

TABLE I
ACCURACY RESULTS OF THE VARIOUS MODELS FOR SLICE ATTRIBUTION DATASETS USING MCC AS METRIC.

Model	Slicing5G	NetworkSlicing5G	NetSlice5G
DNN	1.00	0.83	1.00
LR	1.00	0.83	1.00
k-NN	1.00	0.82	1.00
SVM	1.00	0.83	1.00
GaussianNB	1.00	0.78	1.00
DT	0.85	0.82	1.00
RF	1.00	0.83	1.00
AdaBoost	1.00	0.81	1.00
Gradient Boosting	1.00	0.83	1.00

Analyzing the tables, it is clear that most shallow models were able to achieve the same performance as the DNNs. The only exceptions were the LR, k-NN, GaussianNB, DT, and AdaBoost, which dropped performance in some of the datasets, although none of them performed worst in every

dataset and the difference in accuracy is not significant in some cases.

TABLE II
ACCURACY RESULTS OF THE VARIOUS MODELS FOR IDS DATASETS USING MCC AS METRIC.

Model	UNSW	IoT-DNL
DNN	0.96	1.00
LR	0.94	0.87
k-NN	0.96	1.00
SVM	0.95	1.00
GaussianNB	0.82	1.00
DT	0.95	1.00
RF	0.95	1.00
AdaBoost	0.95	0.80
Gradient Boosting	0.97	1.00

Considering the performance results, it is also noticeable that the NetworkSlicing5G dataset is the most complex dataset, followed by the UNSW, as these were the only datasets where the models were not capable of achieving perfect accuracy.

B. Training/predicting results

Before presenting the training/inference results, it is essential to highlight that the DL models were trained until convergence using an early stopping rule with a patience of ten. This means that once the model accuracy does not improve, the training stops after 10 epochs. This is the usual value applied for DNNs, as the optimization has lots of local minima, and the model might need to reduce its accuracy before improving it further. Using zero patience would provide the best possible scenario for the DL models regarding training time, as it does not do any unnecessary epochs, although it would be unrealistic to set it that way.

1) Network Slice Attribution: Considering that the accuracy between models is generally negligible, the main difference between them will be the training and inference times. Looking at Tables III and IV, it is clear that the shallow models are much faster in both the model training and inference, achieving an acceleration of above 90% in most cases.

TABLE III
TRAINING TIME RESULTS OF THE VARIOUS MODELS FOR SLICE ATTRIBUTION DATASETS. THE RESULTS SHOW THE TIME AND IN PARENTHESIS THE ACCELERATION RELATIVE TO THE DNN MODEL

Model	Slicing5G	NetworkSlicing5G	NetSlice5G
DNN	10.8696(-)	1.084(-)	2.9085(-)
LR	1.7102(84%)	0.0051(100%)	0.0506(98%)
k-NN	0.0295(100%)	0.0006(100%)	0.0029(100%)
SVM	2.4933(77%)	0.2138(80%)	2.1721(25%)
GaussianNB	0.2707(98%)	0.0012(100%)	0.0041(100%)
DT	0.3477(97%)	0.0013(100%)	0.0043(100%)
RF	0.3773(97%)	0.0692(94%)	0.1283(96%)
AdaBoost	2.2681(79%)	0.0115(99%)	0.0466(98%)
Gradient Boosting	3.6893(66%)	0.1221(89%)	0.1116(96%)

Nonetheless, there are some relevant exceptions to this rule. Analyzing the training time, it is noticeable that the SVM acceleration is significant in the Slicing5G and NetworkSlicing5G, while in the NetSlice5G, it is significantly closer. The LR, AdaBoost, and Gradient Boosting also present accelerations below 90% for the Slicing5G dataset, although all these models are still considerably faster (above 50% acceleration).

TABLE IV
INFERENCE TIME RESULTS OF THE VARIOUS MODELS FOR SLICE ATTRIBUTION DATASETS. THE RESULTS SHOW THE TIME AND IN PARENTHESIS THE ACCELERATION RELATIVE TO THE DNN MODEL

Model	Slicing5G	NetworkSlicing5G	NetSlice5G
DNN	0.1602(-)	0.0058(-)	0.0081(-)
LR	0.0073(95%)	0.0001(98%)	0.0005(94%)
k-NN	25.71(-15948%)	0.0527(-809%)	0.0837(-933%)
SVM	0.1906(-19%)	0.0011(81%)	0.0234(-189%)
GaussianNB	0.0851(47%)	0.0003(95%)	0.0011(86%)
DT	0.0089(94%)	0.0001(98%)	0.0002(98%)
RF	0.0173(89%)	0.0178(-207%)	0.013(-60%)
AdaBoost	0.0635(60%)	0.0012(79%)	0.0028(65%)
Gradient Boosting	0.0526(67%)	0.0006(90%)	0.0024(70%)

Regarding the prediction times, the k-NN, SVM, and RF presented slower prediction times than the DNN in some datasets, with k-NN being much slower in all datasets. The results for the SVM and RF are dataset-dependent and are tied with the hyperparameters used for the datasets.

The fastest overall model in network slicing was the DT, achieving fast training and inference in every dataset.

2) Intrusion Detection System: In the IDS datasets, the trends experienced are considerably different. While in network slice attribution, all shallow models were faster at training, and only a few were slower at predicting, in IDS, some models are considerably slower at training and predicting. Tables V and VI present the training and inference acceleration results, respectively.

TABLE V
TRAINING TIME RESULTS OF THE VARIOUS MODELS FOR IDS DATASETS. THE RESULTS SHOW THE TIME AND IN PARENTHESIS THE ACCELERATION RELATIVE TO THE DNN MODEL

Model	UNSW	IoT-DNL
DNN	463.8929(-)	54.5591(-)
LR	732.6907(-58%)	1542.5571(-2727%)
k-NN	0.2367(100%)	0.0206(100%)
SVM	7774.3669(-1576%)	1165.8551(-2037%)
GaussianNB	0.771(100%)	0.0711(100%)
DT	1.5604(100%)	0.109(100%)
RF	3.4698(99%)	1.1961(98%)
AdaBoost	487.8032(-5%)	1.9359(96%)
Gradient Boosting	289.3045(38%)	6.7282(88%)

Starting with the training time analysis, it is clear that LR and SVM are much slower than the DNNs and the remaining shallow models. Furthermore, the AdaBoost is slightly slower when applied to the UNSW dataset. The remaining models present considerably faster training times.

TABLE VI
INFERENCE TIME RESULTS OF THE VARIOUS MODELS FOR IDS DATASETS. THE RESULTS SHOW THE TIME AND IN PARENTHESIS THE ACCELERATION RELATIVE TO THE DNN MODEL.

Model	UNSW	IoT-DNL
DNN	0.773(-)	0.1568(-)
LR	0.0157(98%)	0.0061(96%)
k-NN	545.8093(-70509%)	15.5203(-9798%)
SVM	567.1413(-73269%)	21.9185(-13879%)
GaussianNB	0.1711(78%)	0.0219(86%)
DT	0.0343(96%)	0.0058(96%)
RF	0.1104(86%)	0.077(51%)
AdaBoost	5.1721(-569%)	0.1039(34%)
Gradient Boosting	1.082(-40%)	0.0889(43%)

Regarding prediction, the k-NN and SVM are significantly slower models, followed by the AdaBoost and Gradient Boosting, which, although faster than the other two, are still slower than the DNN. In this scenario, the DT is once again the fastest overall model that maintains close MCC values when compared to the DNN.

V. DISCUSSION

Looking at the results table, some results were unexpected and require further discussion. This section addresses these results by providing explanations that support the presented results.

Starting with the k-NN model, it might be unexpected that a model with such low training times presents a tremendously more expensive inference. However, since k-NN is a lazy learner, the only thing it does during training is load the examples in the training dataset to be part of the model. When the inference needs to be performed is when the model does most calculations, comparing the new data sample with all the existing ones to understand which are closest. The higher the number of features and samples in the training set, the longer the model will take to infer, justifying why the model takes the longest in the UNSW, followed by the Slicing 5G and IoT-DNL datasets.

Another model with an unexpected behavior is the SVM, which has faster training for the Slicing5G and NetworkSlicing5G but is considerably slower in the IDS datasets. This can easily be explained by considering the kernel the SVM uses. If it uses the 'linear' kernel, then training and inference will be fast. On the other hand, if the kernel used is the 'rbf', the training is considerably slower, as each training sample is mapped to a new plan. In the IDS models, the opted kernel is the 'rbf,' which, when combined with a large number of training examples, makes the SVM classifier the slowest model. This also justifies the lower acceleration seen in the NetSlice5G, although the lower number of training examples helps the model stay faster than the DNN.

In the IDS datasets, there is also one interesting result regarding the ensemble models, more specifically the AdaBoost and Gradient Boosting, that, although were not the fastest in the network slicing dataset, present considerably less acceleration and even negative values. This is most likely due to the increase in the number of estimators and depths used, as the UNSW, being a complex dataset, required more classifiers, which slowed the training. Furthermore, the impact the classifiers had in predicting is even more noticeable as the AdaBoost and Gradient Boosting are considerably slower than the DNN. Compared to the RF, it is clear that the latter is a more resilient classifier as the results are not significantly different from the ones obtained in the simpler models.

Considering the overall results, it is clear that some shallow models are much better alternatives than DNNs, as they achieve similar performance with a fraction of the training and inference time. Furthermore, these shallow models did not require specialized hardware to achieve these results, as they all ran on the CPU, while the DNNs used the GPU. Nonetheless, it is also clear that as the dataset complexity and amount of data increases, the DNNs appears as a more reasonable approach since they are more resilient and can even be faster.

That said, the final question to be answered is: When to use DNNs? Although it might seem unsatisfactory, the answer depends on the use case. In the datasets analyzed DNNs, although viable, are not good approaches. Considering network slice attribution, which should take as little as possible, as the latency needs to be at most 1 ms, opting for the most lightweight approach is mandatory. Similarly, in an IoT network, devices capable of training a DNN will most likely not be available, as the IoT devices will have very weak GPUs or none at all.

The same can be said for various other tabular datasets, where the results achieved by the shallow models have been proven to be on par with the ones from DNNs at a fraction of the cost. Despite that, if we consider object detection, image classification, natural language processing, and many other tasks, the performance achieved by the shallow models has been proven to be insufficient, and DNNs is the only approach that achieves reasonable results. In these scenarios, although DNNs have much higher costs, they are the only solution.

VI. CONCLUSION

In conclusion, this paper compares the performance and training and inference time of DNNs with shallow ML models on two important tasks for 5G/B5G networks. The DNNs used were proposed in the state-of-the-art, and the shallow models were built based on the best-performing hyperparameters of a GridSearch, ensuring a fair comparison between them.

The results obtained show that most shallow ML model obtained similar performance to the DNNs at a fraction of the training and inference time. The overall best-performing shallow model was the DT as it maintained consistent results in every dataset tested, except Slicing5G.

Nonetheless, it is also observable that when considering more complex datasets, DNNs are a viable option and can sometimes be faster than the shallow models. Highlighting its usefulness when leading with complex patterns.

In future work, we would like to expand this analysis by considering a broader set of 5G/B5G tasks, an increased number of datasets, and expanding the GridSearch performed on the shallow ML models.

ACKNOWLEDGMENT

This study was funded by the PRR – Plano de Recuperação e Resiliência and by the NextGenerationEU funds at University of Aveiro, through the scope of the Agenda for Business Innovation "NEXUS: Pacto de Inovação – Transição Verde e Digital para Transportes, Logística e Mobilidade" (Project nº 53 with the application C645112083-00000059).

REFERENCES

[1] A. Thantharate, R. Paropkari, V. Walunj, and C. Beard, "Deepslice: A deep learning approach towards an efficient and reliable network slicing in 5g networks," in *2019 IEEE 10th Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON)*, pp. 0762–0767, 2019.

[2] M. Agiwal, A. Roy, and N. Saxena, "Next generation 5g wireless networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 3, pp. 1617–1655, 2016.

[3] C. Zhang, P. Patras, and H. Haddadi, "Deep learning in mobile and wireless networking: A survey," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 3, pp. 2224–2287, 2019.

[4] L. Grinsztajn, E. Oyallon, and G. Varoquaux, "Why do tree-based models still outperform deep learning on typical tabular data?," in *Advances in Neural Information Processing Systems* (S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, eds.), vol. 35, pp. 507–520, Curran Associates, Inc., 2022.

[5] N. Alliance, "Ngmn 5g white paper," 2015. https://ngmn.org/wp-content/uploads/NGMN_5G_White_Paper_V1_0.pdf Accessed: 2023-09-15.

[6] X. Zhu, J. Wang, Q. Lai, X. Luo, R. Ren, and H. Lu, "Research on 5g network slicing type prediction based on random forest and deep neural network," in *2023 IEEE 8th International Conference on Big Data Analytics (ICBDA)*, pp. 154–158, 2023.

[7] S. Wijethilaka and M. Liyanage, "Survey on network slicing for internet of things realization in 5g networks," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 2, pp. 957–994, 2021.

[8] H. Babbar, S. Rani, A. A. AlZubi, A. Singh, N. Nasser, and A. Ali, "Role of network slicing in software defined networking for 5g: Use cases and future directions," *IEEE Wireless Communications*, vol. 29, no. 1, pp. 112–118, 2022.

[9] Y. Xu, Y. Zhou, P. Sekula, and L. Ding, "Machine learning in construction: From shallow to deep learning," *Developments in the Built Environment*, vol. 6, p. 100045, 2021.

[10] S. Ray, "A quick review of machine learning algorithms," in *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 35–39, 2019.

[11] A. Singh, N. Thakur, and A. Sharma, "A review of supervised machine learning algorithms," in *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, pp. 1310–1315, 2016.

[12] X. Dong, Z. Yu, W. Cao, Y. Shi, and Q. Ma, "A survey on ensemble learning," *Frontiers of Computer Science*, vol. 14, pp. 241–258, Apr 2020.

[13] D. Chowdhury, R. Das, R. Rana, A. D. Dwivedi, P. Chatterjee, and R. R. Mukkamala, "Autodeepslice: A data driven network slicing technique of 5g network using automatic deep learning," in *2022 IEEE Globecom Workshops (GC Wkshps)*, pp. 450–454, 2022.

[14] M. A. Islam Arif, S. Kabir, M. F. Hussain Khan, S. Kumar Dey, and M. M. Rahman, "Machine learning and deep learning based network slicing models for 5g network," in *2022 25th International Conference on Computer and Information Technology (ICCIT)*, pp. 96–101, 2022.

[15] I. Parsa, "KDD Cup 1998 Data." UCI Machine Learning Repository, 1998. DOI: https://doi.org/10.24432/C5401H.

[16] S. Stolfo, W. Fan, W. Lee, A. Prodromidis, and P. Chan, "KDD Cup 1999 Data." UCI Machine Learning Repository, 1999. DOI: https://doi.org/10.24432/C51C7N.

[17] M. Tavallaee, E. Bagheri, W. Lu, and A. A. Ghorbani, "A detailed analysis of the kdd cup 99 data set," in *2009 IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp. 1–6, 2009.

[18] K. Kostas, M. Just, and M. A. Lones, "Iotgem: Generalizable models for behaviour-based iot attack detection," *arXiv preprint arXiv:2401.01343*, 2023.

[19] N. Moustafa and J. Slay, "Unsw-nb15: a comprehensive data set for network intrusion detection systems (unsw-nb15 network data set)," in *2015 Military Communications and Information Systems Conference (MilCIS)*, pp. 1–6, 2015.

[20] M. Sarhan, S. Layeghy, and M. Portmann, "Towards a standard feature set for network intrusion detection system datasets," *Mobile Networks and Applications*, vol. 27, p. 357–370, Nov. 2021.

[21] R. Abou Khamis and A. Matrawy, "Evaluation of adversarial training on different types of neural networks in deep learning-based idss," in *2020 international symposium on networks, computers and communications (ISNCC)*, pp. 1–6, IEEE, 2020.

[22] D. K. K. Reddy, J. Nayak, B. Naik, and G. S. Pratyusha, "11 deep neural network–based security model for iot device network," *Deep Learning for Internet of Things Infrastructure*, p. 223, 2021.