Enabling LTE RACH Collision Multiplicity Detection via Machine Learning

Davide Magrin[†], Chiara Pielli[†], Čedomir Stefanović[*], and Michele Zorzi[†]
[†]Department of Information Engineering, University of Padova, Padova, Italy. Email: {name.lastname}@dei.unipd.it
[*]Department of Electronic Systems, Aalborg University, Aalborg, Denmark. Email: cs@es.aau.dk

***Abstract*—The collision resolution mechanism in the Random Access Channel (RACH) procedure of the Long-Term Evolution (LTE) standard is known to represent a serious bottleneck in case of Machine-Type Communication (MTC). Its main drawbacks are seen in the facts that Base Stations (eNBs) typically cannot infer the number of collided User Equipments (UEs) and that collided UEs learn about the collision only implicitly, through the lack of the feedback in the later stage of the RACH procedure. The collided UEs then restart the procedure, increasing the RACH load and making the system more prone to collisions. In this paper, we leverage machine learning techniques to design a system that, besides outperforming the state-of-the-art schemes in preamble detection for the LTE RACH procedure, is able to estimate the collision multiplicity and thus gather information about how many devices chose the same preamble. This data can be used by the eNB to resolve collisions, increase the supported system load and reduce transmission latency. Besides LTE, the presented approach is applicable to novel 3GPP standards that target massive Internet of Things (IoT), e.g., LTE-M and NB-IoT, as well as 5G, since their RACH procedures are based on the same principles.**

I. Introduction

The RACH procedure in LTE serves as a synchronization mechanism between the eNB and the UEs: the UEs compete for resources by randomly choosing a preamble and getting resources assigned by the eNB, provided that they do not collide with each other in the preamble-based contention phase. The current system was designed to handle human-type communication, but the rapid spread of MTC in the cellular access has led to a dramatic traffic increase, causing channel congestion problems. For example, in case of synchronized alarms that trigger a feedback from multiple devices, simultaneous accesses to the channel may yield up to a tenfold increase with respect to normal traffic, almost guaranteeing that all UEs will collide [1], [2].

The RACH overload can significantly degrade the network performance, and in the last years many approaches to mitigate collisions were proposed [3]. Such studies generally focus either on improving the preamble detection (i.e., identifying whether a preamble has been chosen by any UE) or on mechanisms to efficiently use the random access-related resources.

In this work, we propose a novel strategy to deal with the increased LTE traffic: estimate the collision multiplicity during the RACH procedure to infer the current traffic load. In particular, we propose a machine learning based scheme that requires no modification of the current protocol stack, and that can be fully implemented at the eNB. We first verify the validity of applying machine learning techniques to collision detection, showing that they can compete with the state-of-the-art threshold-based detection algorithms described in [4], [5]. We then show the effectiveness of machine learning techniques in estimating the number of UEs that pick the same preamble, providing information about the collision, which is instead not available with the current techniques. Such information could be used to tune successive stages of the contention resolution phase, thereby handling even higher traffic loads. The development of such mechanisms is, however, outside of the scope of this paper. We want to remark that the contribution of this work is not the development of new machine learning techniques, but rather the novel application of known machine learning algorithms to infer previously unseen information in the RACH procedure, information that can be exploited to mitigate the increasing overload issue.

The rest of the paper is organized as follows. Sec. II describes the LTE RACH procedure and the algorithms currently used for preamble detection, and briefly discusses how the knowledge of the collision multiplicity could alleviate the RACH overload issue. Sec. III explains the machine learning approaches and the datasets they used. Sec. IV shows the improvements brought by the new schemes compared to current approaches, and explores new features that can be enabled by collision multiplicity detection. Finally, Sec. VI concludes the paper.

II. LTE RACH

The random access procedure in LTE is performed in the Physical Random Access Channel (PRACH), a dedicated physical channel with an overall bandwidth of 1.08 MHz and duration between 1 and 4 LTE subframes. We first give an overview of how PRACH preambles are generated and how they are used in the multiple channel access scheme, and then discuss the potential benefits of knowing the collision multiplicity.

A. Preamble generation

LTE uses Zadoff-Chu (ZC) sequences, complex-valued sequences that satisfy the Constant Amplitude Zero Autocorrelation (CAZAC) property, as a basis to create RACH preambles. A ZC sequence of odd length N_{ZC} is defined as:

$$z_r(n) = \exp\left\{-j2\pi r \frac{n(n+1)}{N_{\text{ZC}}}\right\}, \quad n = 0, 1, \ldots, N_{\text{ZC}}-1 \tag{1}$$

where $r \in \{1,\ldots,N_{\text{ZC}}-1\}$ is the sequence root index; the LTE standard uses $N_{\text{ZC}} = 839$[1]. Given a root r, it is possible to generate multiple versions of the base sequence z_r through a circular shift, thereby obtaining orthogonal sequences that at the receiver exhibit a zero correlation with one another. The cyclic correlation of a ZC sequence with its root is a delta function with a peak corresponding to the shift that was applied to the root sequence. Consequently, circular correlation of a root against a superposition of different sequences obtained from that root results in multiple peaks corresponding to the individual shifts.

In this work, we consider the preamble format 0, which is typically used in cells with a radius up to 14 km, and consists in a 1 ms random access burst with preamble sequences of duration 800 μs [4]. The full PRACH preamble is obtained by prepending the Cyclic Prefix (CP) to the ZC sequence chosen by the UE, to counteract the multi-path reflection delay spread. The CP also introduces an additional shift in the correlation peak if the preamble signal is delayed in time, enabling the eNB to estimate the channel delay experienced by a device. In order to correctly identify the chosen preamble despite the propagation delay, UEs are allowed to pick only sequences whose shift is a multiple of a base quantity N_{CS}, which is chosen so as to guarantee that the eNB can identify different preambles by applying a correlator and a peak detector to the received signal.

The LTE standard uses $N_{\text{prb}} = 64$ preambles in each cell. In this work, we only focus on the case $N_{\text{CS}} = 13$, which is the smallest possible cyclic shift that allows to obtain exactly N_{prb} preambles, so that a single root is used to generate all the LTE preambles of the cell. This configuration can be used in all cells covering a radius up to 0.79 km [4], and thus suits well densely deployed urban scenarios. From now on, we will denote as *bin* each portion of the correlation signal that corresponds to a preamble (we thus have 64 different bins).

B. RACH Procedure

The RACH procedure consists of four phases.

1) *Random Access Preamble*: the UEs that intend to transmit data randomly choose one among N_{prb} available preambles and send it to the eNB.
2) *Random Access Response (RAR)*: the eNB processes the received signal, consisting in the superposition of all the transmitted preambles. For each detected preamble, it sends a RAR message to assign the uplink resources to the corresponding UE.
3) *L2/L3 message*: the UEs use the newly assigned data channel resources to communicate their connection request and a unique identifier.
4) *Contention Resolution Message*: the eNB responds to the UEs using the identifiers they communicated in their L2/L3 message, granting the requested resources.

Typically, UEs access the channel in a contention-based mode,[2] where collisions happen when multiple UEs pick the same preamble. If the colliding preambles are received with a high enough Signal-to-Noise Ratio (SNR) and are sufficiently spaced apart in time, the eNB can detect all of them and recognize the collision, avoiding to send the RAR message to the involved UEs. Otherwise, the eNB erroneously detects a single preamble and all the colliding UEs will receive the RAR message and try to access the same resource simultaneously, colliding in the L2/L3 message phase. The UEs are implicitly informed about the collision because they do not receive the contention resolution message. In both cases, the collided UEs can then try again in a new PRACH phase, potentially increasing the RACH load. Moreover, in the latter case, channel resources are allocated fruitlessly.

C. Interest in multiplicity detection

The conventional approach to preamble detection in LTE RACH is represented by *threshold-based* detectors [4], which compare the circular correlation of the received signal with its base sequence against a previously set threshold. A preamble is detected when the corresponding bin in the correlation signal contains values above the threshold. Such threshold is typically a function of the estimated noise level: this provides direct control over the false alarm probability, which can be made arbitrarily low (at a price in missed detection performance).

Several works in the literature extend and adapt the threshold-based mechanism, e.g., to improve the computational performance at the expense of a larger missed detection probability [5], to consider multiple root sequences [6], or to counteract the problem of performance degradation due to time dispersion of the channel [7].

The emergence of the IoT paradigm and the resulting increase of MTC in many applications has however exacerbated the overload issue in the RACH procedure, requiring an extensive research effort to counteract the performance degradation due to RACH congestion. Most of the available solutions are based on initial proposals compiled by the 3GPP, including separation of random-access resources, Access Class Barring (ACB) schemes, pull-based random access and parameter optimization in the medium access control layer [8]. Other approaches assume reengineering of the LTE RACH procedure, for example by grouping LTE preambles in codewords that could convey information to the eNB [9] or by using advanced collision resolution algorithms RACH [10].

Multiplicity detection could be beneficial for many of the existing mechanisms that mitigate the RACH overload, helping a proper dimensioning of the resources dedicated to the PRACH. It may in fact allow to infer the current load of the LTE RACH, as well as trends regarding its changes. Several

[1]Except for preamble format 4, which is not treated in this paper.

[2]The standard provides also a contention-free mode for handover and other special delay-sensitive cases.

approaches leverage on (possibly dynamic) estimates of the traffic load, implying that knowing the collision multiplicity could yield enhanced performance. For example, [11] proposes a learning automaton based algorithm that dynamically adapts an ACB scheme based on the estimated arrival rate and asymptotically converges to the optimal case of full information available at the eNB: knowing the collision multiplicity the algorithm could optimally adapt the ACB parameters from the beginning. Similarly, [12] proposes a strategy to automatically configure the random access parameters based on the number of collided preambles and the estimated traffic rate, which is dynamically updated using a moving average filter. Such estimate could be replaced or integrated with the information about the collision multiplicity, yielding a more realistic value of the traffic demand and improved collision resolution performance.

As a final remark, we highlight that the focus of the paper is on the multiplicity detection, while its coupling with advanced LTE RACH algorithms is left for further work.

III. Machine learning approaches

In this work, we apply Machine Learning (ML) techniques to preamble detection, i.e., determining whether a certain preamble was sent or not, and preamble multiplicity detection, i.e., determining how many devices sent the same preamble, in LTE. The proposal of using ML to solve the problem of preamble detection is motivated by the fact that the peaks in the correlation signal (see Sec. II-A) contain structured information based on the behavior of the channel. This fact can be seen in Fig. 1, which compares the peaks obtained in the AWGN and Rayleigh channels, and highlights how the peak takes the shape of the channel's power delay profile in the Rayleigh case. While basic approaches can still obtain satisfactory results by simply comparing the height of the peaks to a threshold, ML approaches can access more information by considering the whole correlation signal; potentially, they can learn to rely on the shape, position and height of the peaks to make their classification decision and correctly distinguish real peaks from those caused by noise. In this work, we consider two different ML techniques, namely Logistic Regression (LR) and Neural Network (NN), which strike a good balance between complexity and performance.

In the remainder of this section, we first describe the dataset we used, and then discuss the employed ML techniques.

A. Learning from data

The necessary premise for every ML algorithm is to have data available, which can be used to *train* these systems and make them "learn." Our goal is to detect the number of UEs that chose a certain preamble, given the signal received at the eNB. This can be modeled as a classification problem since our desired output is a category. Classification problems are a type of supervised learning, where the model is fed with input data X and the corresponding desired output data $Y = f(X)$. These labeled data are denoted as *training data* and are used as examples to extract a general rule to map an input to the

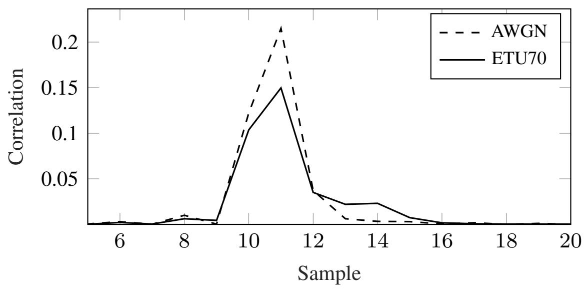

Figure 1: Correlation generated by different channel models

correct output: the goal is to approximate the mapping function $f(\cdot)$ so as to correctly predict the output data corresponding to a previously unseen input. This is achieved by iteratively optimizing an objective function based on a cost function, which measures the distance between the desired output and the prediction and thus must be minimized.

The performance of a classification algorithm is typically evaluated using the *accuracy* metric, which represents the percentage of predictions that are correct. Once the algorithm has been trained, its performance is evaluated over a previously unseen *test dataset*.

Other important metrics in binary classification are the false positive and true positive rates. The false positive rate, also referred to as *false alarm probability*, is the probability of falsely identifying a sample as belonging to a class. The true positive rate, also known as *detection probability*, is the probability of correctly identifying a sample.

B. Dataset generation

Research about multiplicity collision in the RACH procedure is quite modest and, to the best of our knowledge, there is no publicly available dataset that relates the signal received at the eNB with the information of the preambles chosen by each UE. Hence, we generated our own dataset using the MATLAB LTE System Toolbox^{TM},[3] which provides standard-compliant functions for the design, simulation, and verification of the LTE communication system.

In this section, we explain how we generate our dataset, i.e., which data we extracted and the scenarios we considered.

1) Data labeling: to run ML algorithms, we need a map between (i) the signal received at the eNB at the end of Phase 1) of the RACH procedure (see Sec. II-B), and (ii) the number of UEs that selected each preamble. These will represent the input and output data of the algorithms, respectively. Accordingly, each simulation consists of simultaneous random access attempts from a certain number of UEs, each choosing one of the N_{prb} available preambles. The MATLAB LTE module allows to extrapolate the corresponding correlation signal at the eNB, i.e., the output of the cyclic correlation of the received signal with respect to the root index (see Sec. II-A). Each preamble is uniquely mapped to a bin, i.e., a

[3]https://www.mathworks.com/products/lte-system.html

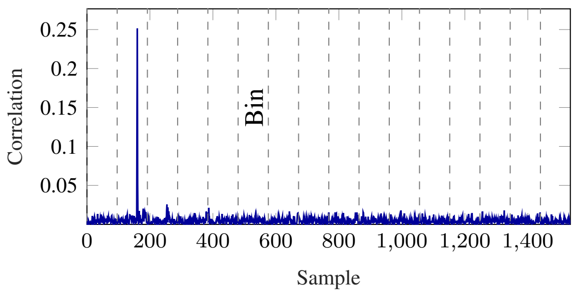

Figure 2: Example of correlation signal with 16 bins.

correlation window of predefined size, according to the value of N_{CS}. Since we are interested in how many devices chose each preamble, we decided to consider each bin separately, rather than the correlation signal as a whole. Fig. 2 shows an example output of the MATLAB LTE module.

2) Applications: we investigated two distinct applications, preamble and multiplicity detection. The former aims at comparing the performance of the proposed ML systems to the state of the art, so the goal is to identify *whether* and *which* preamble was sent. The latter application, instead, targets the estimate of the number of UEs that chose the same preamble.

For preamble detection, we used three distinct datasets: (i) a noise-only set of correlations, used to evaluate false alarm probabilities according to the eNB testing specification [13]; (ii) a set of correlation signals obtained when only a single UE is transmitting, used to test missed detection; (iii) a dataset containing both noise-only and single UE correlations, used for the training of the ML systems. For multiplicity detection, instead, we generated a dataset with bins containing from 0 to 5 preambles; 70% of the bins were used for training and 30% for testing. All datasets were generated considering multiple scenarios, that differ in two features:

- *Noise level*. The signal received at the eNB is the superposition of all the transmitted preambles, corrupted by noise, expressed in terms of SNR. Based on the performance of the ML systems we tested, we decided to focus on an SNR range between -20 dB and -12 dB.
- *Channel model*. We considered an Additive White Gaussian Noise (AWGN) channel and an ETU70, i.e., an Extended Typical Urban channel with 70 Hz Doppler affected by Rayleigh fading.

For each scenario, we ran over 10^5 independent simulations, obtaining the correlation signal in each bin and the number of UEs that selected the corresponding preamble.

3) Traffic intensity: while the preamble detection problem requires the transmission of at most one preamble per RACH attempt, the number of competing devices is a critical parameter for the multiplicity detection. The 3GPP proposes a model for highly correlated traffic arrivals [14], where the number of UEs in each random access opportunity in the considered time frame follows a Beta distribution. The study defines a possible massive access scenario with up to 30000 devices accessing the channel "synchronously," i.e., over a time interval of about 10 s. Considering RACH opportunities to happen every 20 ms, this corresponds to a maximum expected number of devices that participate in the same RACH opportunity of about 120. We therefore generated our dataset with 120 UEs randomly choosing among the N_{prb} available preambles: due to the binomial distribution nature of the preamble selection mechanism, 99% of the bins generated in this way have at most 5 devices choosing that preamble.

C. Logistic regression

LR is a statistical method that predicts the probability that an input data belongs or not to a certain class. The rationale behind LR is that the input space can be separated into two complementary regions, one for each class, by a linear boundary. It differs from linear regression because the dependent, output variable is binary rather than continuous, and this method is in fact intended for classification problems. The probability that the output belongs to one or the other class is expressed by means of a sigmoid $\sigma(\cdot)$, also called logistic function, from which LR takes its name. Function $\sigma(\cdot)$ depends on some weights, which are tuned during the training phase of the LR in such a way to minimize a predefined cost function (see Sec. III-A).

Given an SNR value, we trained different LR models for each type of channel considered, and for both the investigated problems (preamble detection and collision multiplicity), using the open source scikit-learn library in Python[4]. We used the dataset described in Sec. III-B, so that the logistic regressor is fed with the correlation signal obtained at the receiver in a bin, while the output data is the number of UEs that picked the preamble corresponding to the considered bin. For preamble detection, there are two possible classes, i.e., 0 or at least one UE, while the number of possible classes in the collision multiplicity problem is $N_{\max}+1$, where $N_{\max}$ is the maximum number of colliding UEs we are interested in estimating; we set $N_{\max}=5$. In our case, choosing among $N_{\max}+1$ alternatives can be modeled as a set of $N_{\max}$ independent binary choices (a pivot alternative is compared to the remaining $N_{\max}$ ones). This method, commonly denoted as "one vs. all," makes the training complexity linear in $N_{\max}$.

D. Neural network

Artificial NNs [15] are a class of mathematical models that are considered universal approximators [16], as they are able to represent, up to any accuracy, any non linear function. The basic units of NNs are called *neurons*, and are organized in multiple layers. Any NN has an input layer fed with the data to process, an output layer that represents the corresponding output determined by the network, and possibly one or more hidden layers. Neurons represent non-linear multi-input single-output functions: such functions are characterized by some weights and biases, which represent the interconnections among the neurons and are progressively

[4]https://scikit-learn.org

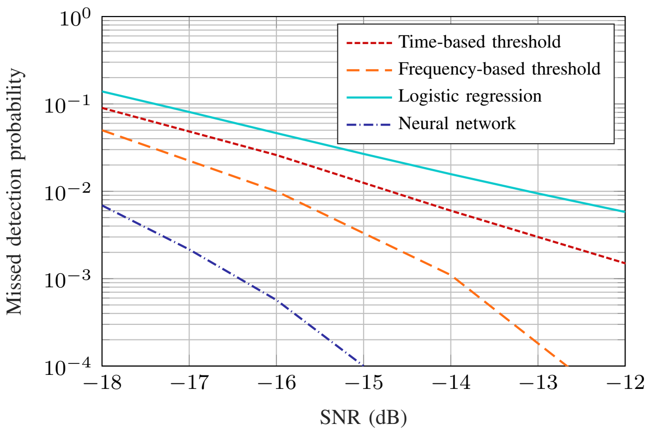

Figure 3: Detection performance comparison in an AWGN channel.

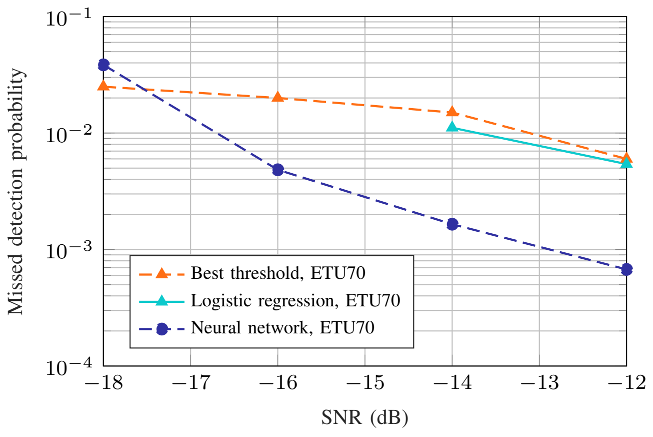

Figure 4: NN, LR and best threshold system detection performance comparison in an ETU70 channel.

tuned during the training phase to produce the desired output. NNs are an extremely powerful tool because they can infer even very complex functions, that analytical models or simpler approaches such as LR fail to describe, and thus provide an adequate way to detect and make use of the information that is inherently available in the correlation signal.

We used a simple feedforward NN, i.e., a fully-connected network where any neuron in layer i is connected to every neuron in the next layer $i+1$. For each SNR value, channel type and problem to solve (preamble detection or collision multiplicity), we trained a different NN using the Keras[5] library. The input data is the correlation signal obtained at the eNB for one bin, requiring 24 neurons in the input layer. The output layer has $N_{\max}+1$ nodes, where $N_{\max}$ is the maximum number of colliding UEs we are interested in estimating ($N_{\max} = 1$ for preamble detection, $N_{\max} = 5$ for collision multiplicity). We used 2 hidden layers.

In both scenarios, for all layers but the last one, the activation function (i.e., the function that dictates how a neuron's weighted input is mapped into its output) is the rectifier function, whereas for the output layer we decided to use a softmax function, as usually done in classification problems. This implies that the NN outputs a separate probability for each of the possible classes and chooses the most probable class. Additionally, we inserted dropout layers with the aim of helping the NN generalize and speed up the learning process. Dropout refers to ignoring neurons during the training phase: more specifically, at each training stage, individual neurons are either dropped out of the network with probability $1-p$ or kept with probability p, so that a reduced network is left; incoming and outgoing edges to a dropped-out neuron are also removed. This technique is used to prevent over-fitting of training data, caused by the co-dependencies that neurons develop amongst each other, which curb the individual power of each neuron.

[5]https://keras.io

IV. Results

We first compared the performance of the LR and NN methods in preamble detection to those obtained with state-of-the-art detectors, and then evaluate the performance that the ML approaches achieve in collision multiplicity estimation.

A. Preamble detection performance

To guarantee a fair comparison with state-of-the-art threshold-based detection, we conducted our simulations according to the LTE specifications on eNB conformance testing [13]. We thus measured the *missed detection probability*, i.e., the probability that a transmitted preamble remains undetected at the eNB, and the *false alarm probability*, i.e., the probability that the eNB wrongly detects a transmission in an unused bin (see Sec. III-A).

The test dataset used to evaluate the detection performance contains bins belonging to correlation signals in which exactly one device was transmitting. The dataset used for training of the ML systems, instead, consists of bins coming from 10^5 correlation signals containing either 0 or 1 preambles.

Fig. 3 shows the missed detection probability for both the LR and the NN schemes in the case of an AWGN channel, together with the performance of the algorithms proposed in [5]. The scheme based on LR yields worse performance than those leveraging thresholds, because of its extreme simplicity; the mediocre performance obtained with LR suggests that the correlation signal at the eNB and the transmission of defined preambles are data that are not completely separable, but are rather related in a more complex way. This is supported by the outstanding performance of the NN, which provides a gain in the range of 2 to 3 dB with respect to the best threshold-based detector for all values of SNR, yielding a performance which conforms to the standard requirements. We remark that, rigorously, missed detection probability should also include the cases of a wrong UE delay estimation. However, in the case of ML based systems, which are not capable of estimating this parameter, delay detection cannot be taken into account.

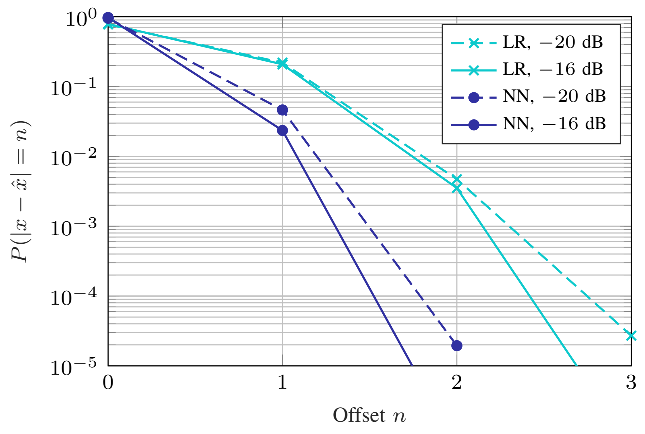

Figure 5: Probability that the estimate on the multiplicity is wrong by different offsets in an AWGN channel.

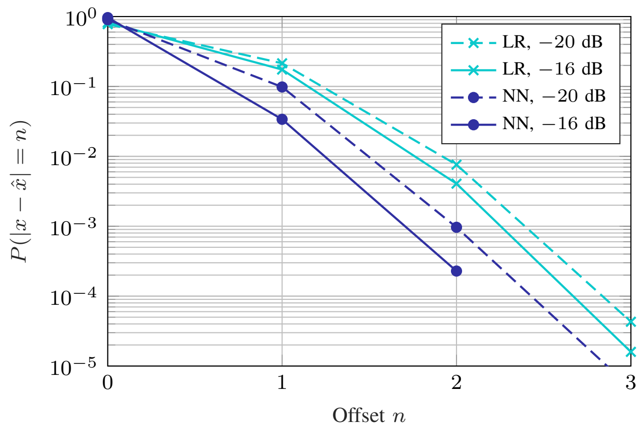

Figure 6: Probability that the estimate on the multiplicity is wrong by different offsets in an ETU70 channel.

Fig. 4 shows a comparison between the detection performance of the NN and LR in the case of an ETU70 channel, and compares them with the performance of the best threshold-based detection scheme. Also in this case, the NN achieves a significant improvement in detection performance, upwards of 4 dB with respect to the threshold-based schemes described in [5]. LR performance is only shown at SNR values in which the false alarm probability requirement is satisfied.

As concerns the false alarm probability, the standard sets the minimum requirement for this metric at 0.1% [13]. ML approaches do not provide the same direct control over the false alarm probability that threshold-based schemes offer. With the LR scheme, the false alarm probability requirement is respected for SNR larger than -16 dB. On the other hand, the false alarm probability obtained with the NN was consistently under the required 0.1% threshold for all the analyzed SNR values, for both AWGN and ETU70 channels.

B. *Multiplicity detection*

The main goal of this work is to show the validity of ML schemes in multiplicity detection. To do so, we decided to measure the frequency of errors in the estimation. Let x represent the actual number of preambles in a bin, i.e., the number of UEs that chose the same preamble, and $\hat{x}$ the corresponding estimate made by the ML scheme. Fig. 5 shows, for both the LR and NN approaches, the probability that the absolute difference between the estimated and the actual number of preambles in a bin is an *offset* n, namely $P(|x-\hat{x}|=n)$. An offset of 0 represents the probability of guessing exactly the number of preambles in the bin, an offset of 1 represents the probability that the difference between the estimated and real number of transmitted preambles is ± 1, and so on. Analogously, Fig. 6 shows the same results in the case of an ETU70 channel for the same system configurations.

As expected, lower SNR values give higher error probabilities. The NN clearly outperforms the LR approach, thanks to its higher complexity, which allows to infer even very complicated relations between its input and output data. Although LR could not compete with the NN in preamble detection (see Fig. 3), a linear approach is still able to extract the information needed to identify the number of transmitters in each bin from the signal received at the eNB with a probability of around 0.8 in the AWGN case, and gets the multiplicity wrong by at most 1 with a probability of 0.99. The NN scheme, instead, in the AWGN case guarantees with a probability of about 0.999 to have an estimate that is either correct or only off by one, even for very low SNR values, and yields exact guesses with a probability of 0.9. The performance of the ML schemes, and especially of NN, gets worse for worse channel conditions, as Fig. 6 shows.

Tab. I contains the normalized confusion matrices for the NN and LR systems, in the case of SNR $= -16$ dB: the rows represent the actual multiplicity value x, while the columns represent the value estimated by the ML scheme $\hat{x}$. Hence, the confusion matrix for an ideal system would have ones along the diagonal, signifying that for each input

Table I: Confusion matrices; SNR $= -16$ dB, AWGN channel.

Neural Network						
$\hat{x}$ \ x	0	1	2	3	4	5
0	**0.996**	0.003	0	0	0	0
1	0	**0.972**	0.027	0	0	0
2	0	0.005	**0.987**	0.007	0	0
3	0	0	0.013	**0.978**	0.007	0
4	0	0	0	0.018	**0.971**	0.009
5	0	0	0	0	0.033	**0.966**
Logistic regression						
$\hat{x}$ \ x	0	1	2	3	4	5
0	**1**	0	0	0	0	0
1	0.018	**0.735**	0.239	0.007	0	0
2	0	0.048	**0.604**	0.334	0.011	0
3	0	0	0.062	**0.659**	0.278	0
4	0	0	0	0.086	**0.813**	0.099
5	0	0	0	0	0.088	**0.912**

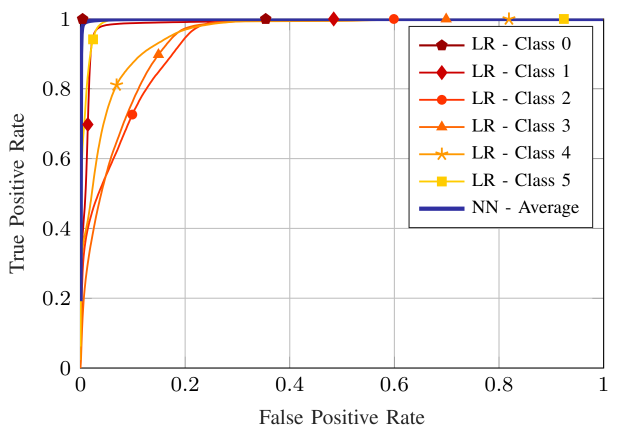

Figure 7: Logistic Regression ROC for SNR = -16 dB

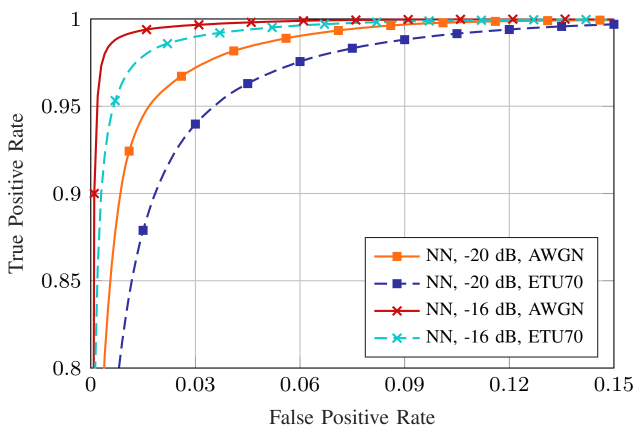

Figure 8: Neural Network average ROC for various combinations of SNR and channel type.

the labeling is correct with probability 1. It can be seen that the NN consistently outperforms LR, except for the case of no preambles being sent, where LR achieves a perfect score. Despite this perfect performance in the correct detection of no preambles in the bin, the LR scheme is heavily affected by a wrong detection rate in case of 1 preamble being sent, thus motivating the bad overall detection performance seen in Fig. 3. Furthermore, the confusion matrices also show that the LR scheme tends to overestimate the number of preambles in a bin, while the NN yields a more symmetrical error.

The behavior highlighted with the confusion matrices can be also seen in the Receiver Operating Characteristic (ROC) curves, which show the true positive rate against the false positive rate, giving a measure of the diagnostic ability of a binary classifier system as its discrimination threshold is varied. The dataset used for the evaluation of the ROC curves contains balanced amounts of each class, so as not to skew the curves. Since in our case we are dealing with multiple classes, we used the “one vs. all” method to perform binary classification and obtain a ROC curve, thus using a binary classifier for each possible class, as previously done for LR (see Sec. III-C). Fig. 7 shows the ROC curve obtained with LR for each class (i.e., number of users that chose the same bin) for the same configuration of the confusion matrices, i.e., AWGN channel and SNR $= -16$ dB. The curves confirm the perfect detection performance for Class 0, and illustrate how Classes 2 to 4 are the most difficult ones to correctly detect, because of the tendency of the LR to overestimate the number of preambles in a bin. When this overestimation is not possible, as in the case of Class 5, the performance is significantly better and very close to that of Class 1. In the case of the NN instead, all the classes basically yield the same performance, as per Table I. We thus only represented the average ROC, obtained as the average of the ROC curves of each class, which confirms the validity of the NN in multiplicity detection.

The NN yields an outstanding ROC curve even for worse channel conditions and more challenging channels. This can be seen in Fig. 8, which shows a portion of the average ROC curve obtained with the NN for the two considered channel types (AWGN and ETU70) and for two values of the SNR, namely -20 and -16 dB. We only reported the upper left corner of the whole curve, to highlight the differences of the system configurations. Clearly, the performance improves with higher SNRs and with a simpler channel.

V. Machine Learning Limitations

Although they have been shown to provide good results in preamble detection and collision multiplicity estimation, ML approaches entail some disadvantages, which we describe next.

A. *Complexity*

Even though the training phase of ML based detection is performed offline, such approaches are more complex than threshold-based detection, and inevitably require a larger computational effort. In fact, while state-of-the-art schemes only need to compare each correlation sample with a predefined threshold, a detector based on LR needs to perform a multiplication for each sample, and then sum all the obtained results to get to a decision. To perform multiplicity detection, such operations need to be performed for each classifier as described in Sec. III-C. Neural networks also need to perform a number of operations that is proportional to the network complexity, and could be larger than the operations required by a LR scheme in the case of detection. When compared to a LR scheme in the case of multiplicity estimation, however, a NN needs to perform roughly the same number of operations as in the detection scheme, since it already outputs a classification. In fairness to ML, we remark that the complexity burden of the proposed algorithms is on the eNB, whose hardware and software capabilities are constantly improving and which typically have very little computational and energy restrictions, such that the higher complexity demanded by ML schemes may not be a limiting factor.

B. Dataset collection

Both in the case of LR and NNs, a dataset of sufficient size must be available to perform effective training. In this work, the ML based approaches were trained on computer generated signals using a state-of-the-art simulator in order to have a representation as close as possible to what the eNB will actually see in a real deployment. This allowed us to exactly know the number of UEs choosing each preamble. For real deployments, this dataset could be integrated with some samples taken by the eNB. In particular, a bin associated to a certain preamble can be labeled according to the outcome of the RACH procedure: label as 0 if no device sends a *msg3*, as 1 if exactly 1 device answers with a *msg3*, and as 2 if a collision happens during *msg3*. Unfortunately, this does not allow for a precise estimation of the number of devices sending a specific preamble. Such an estimation may instead be possible if, once the eNB detects a collision (i.e., if it labels the considered bin as 2), another ad hoc contention resolution phase were to take place iteratively among the colliding nodes, until resolution of all collisions.

VI. Conclusions

With RACH overload becoming an increasingly serious issue in massive access LTE scenarios, it is necessary to develop techniques to better manage the contention of resources by multiple users. In this work, we have investigated the use of machine learning techniques to infer the number of users that pick a selected preamble and estimate the collision multiplicity. We also showed that ML approaches, and in particular neural networks, can improve the performance of state-of-the-art threshold-based schemes.

The results described in this paper may serve as a first step to improve the current RACH procedure. They are also applicable to novel 3GPP standards like LTE-M, NB-IoT, and 5G [17], as their RACH procedures are based on the same principles. In particular, the possibility to immediately identify the collision multiplicity allows more efficient collision management than the current approach, where the RACH procedure is repeated until all UEs uniquely pick a preamble. This would have a positive impact on the latency, the device power consumption, and the supported system load. Moreover, collision multiplicity detection may also allow to promptly identify a switch in the data reporting regime, e.g., in case of an alarm that triggers synchronous transmissions from multiple devices, and act accordingly.

In the future, we plan to evaluate additional ML techniques, assess the robustness of the proposed ML classifiers to difficult signals, test their behavior using further time-varying channel models and implement a link model based on the performance of the scheme proposed in this paper as part of a system-level simulator, in order to quantify the impact of the additional information on multiplicity on collision resolution and RACH management under overload scenarios.

Furthermore, the insights gained using ML techniques may be helpful to develop an analytical scheme to infer the preamble collision multiplicity, possibly based on the CAZAC properties of the root sequences.

Acknowledgement

The work of Čedomir Stefanović was in part supported by the "Visiting Scientis" program at the University of Padova and in part by the ERC Consolidator Grant nr. 648382 WILLOW. This work has been also partially supported within the program "Supporting TAlent in ReSearch@University of Padua - STARS Grants," through the project "Cognition-BAsed NETworkS: building the next generation of wireless communications systems using learning and distributed intelligence."

References

[1] M. Y. Cheng, G. Y. Lin, H. Y. Wei, and A. C. C. Hsu, "Overload control for machine-type-communications in LTE-Advanced system," *IEEE Communications Magazine*, vol. 50, no. 6, pp. 38–45, June 2012.

[2] A. Laya, L. Alonso, and J. Alonso-Zarate, "Is the Random Access Channel of LTE and LTE-A Suitable for M2M Communications? A Survey of Alternatives," *IEEE Communications Surveys Tutorials*, vol. 16, no. 1, pp. 4–16, First Quarter 2014.

[3] M. S. Ali, E. Hossain, and D. I. Kim, "LTE/LTE-A random access for massive machine-type communications in smart cities," *IEEE Communications Magazine*, vol. 55, no. 1, pp. 76–83, Jan. 2017.

[4] S. Sesia, M. Baker, and I. Toufik, *LTE - The UMTS Long Term Evolution: from theory to practice*. John Wiley & Sons, 2011.

[5] F. J. López-Martínez, E. del Castillo-Sánchez, E. Martos-Naya, and J. T. Entrambasaguas, "Performance evaluation of preamble detectors for 3GPP-LTE physical random access channel," *Digital Signal Processing*, vol. 22, no. 3, pp. 526–534, May 2012.

[6] T. Kim, I. Bang, and D. K. Sung, "An enhanced PRACH preamble detector for cellular IoT communications," *IEEE Communications Letters*, vol. 21, no. 12, pp. 2678–2681, Dec. 2017.

[7] X. Yang and A. O. Fapojuwo, "Enhanced preamble detection for PRACH in LTE," in *IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2013, pp. 3306–3311.

[8] 3GPP TR 37.868 v11.0.0, "Study on ran improvements for machine-type communications," 2011.

[9] N. K. Pratas, Č. Stefanović, G. C. Madueño, and P. Popovski, "Random Access for Machine-Type Communication Based on Bloom Filtering," in *IEEE Global Communications Conference*, Dec 2016, pp. 1–7.

[10] G. C. Madueño, Č. Stefanović, and P. Popovski, "Efficient LTE access with collision resolution for massive M2M communications," in *IEEE Globecom Workshops (GC Wkshps)*, Dec 2014, pp. 1433–1438.

[11] F. Morvari and A. Ghasemi, "Learning automaton based adaptive access barring for M2M communications in LTE network," in *24th Iranian Conference on Electrical Engineering (ICEE)*. IEEE, May 2016, pp. 1466–1470.

[12] S. Choi, W. Lee, D. Kim, K.-J. Park, S. Choi, and K.-Y. Han, "Automatic configuration of random access channel parameters in LTE systems," in *Wireless Days (WD), 2011 IFIP*. IEEE, 2011, pp. 1–6.

[13] 3GPP, "LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Base Station (BS) conformance testing," TS 36.141 V.11.3.0.

[14] ——, "Study on RAN improvements for machine-type communications," TR 37.868 V.11.0.0.

[15] N. K. Bose and P. Liang, *Neural Network Fundamentals with Graphs, Algorithms, and Applications*. New York, NY, USA: McGraw-Hill USA, 1996.

[16] K. Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural networks*, vol. 4, no. 2, pp. 251–257, 1991.

[17] 3GPP TS 38.300 v15.3.1, "NR; overall description; stage-2," 2018.