Internet of Things 34 (2025) 101786

Contents lists available at ScienceDirect

Internet of Things

journal homepage: www.elsevier.com/locate/iot

Research article

A deep learning-based Adaptive Data Rate algorithm for LoRaWAN networks

Luca Leonardi [a,*], Giancarlo Iannizzotto [b], Mattia Pirri [a], Gaetano Patti [a], Alessio Pirri [a], Lucia Lo Bello [a]

[a] Department of Electrical, Electronic and Computer Engineering, University of Catania, Catania, Italy
[b] Department of Cognitive Sciences, Psychology, Education and Cultural Studies, University of Messina, Messina, Italy

ARTICLE INFO

Keywords:
LoRaWAN
Adaptive Data Rate
Deep learning
Long short term memory
Internet of Things

ABSTRACT

LoRaWAN is emerging as a key protocol for several Internet of Things (IoT) applications, as it enables long-range communication with low power consumption between a large number of end-devices. LoRaWAN end-devices are characterized by a number of configurable transmission parameters, whose values need to be carefully selected, as they significantly influence the network performance. The Adaptive Data Rate (ADR) algorithm recommended by Semtech dynamically adjusts the transmission parameters of LoRaWAN end-devices to improve the network reliability while keeping energy consumption low. However, ADR is a rule-based algorithm not suitable for dynamic IoT scenarios in which the network conditions can be highly variable and the end-devices move around in the sensing area. In contrast, deep learning techniques appear a promising solution to set the transmission parameters of LoRaWAN end-devices in such dynamic IoT environments, thanks to their ability to learn from data a non-linear, state-dependent model. For this reason, this paper proposes a deep learning-based mechanism, called Rel-ADR, that dynamically tunes the transmission parameters of LoRaWAN end-devices to improve the transmission reliability, while maintaining low power consumption in dynamic and dense networks. The paper presents the design of Rel-ADR and the results of an extensive comparative performance evaluation between Rel-ADR and existing approaches in the literature, obtained through OMNeT++ simulations in realistic scenarios.

1. Introduction

Low Power Wide Area Networks (LPWANs) support long-range communication with low energy consumption between a large number of devices. For this reason, LPWANs are replacing or complementing traditional cellular and short-range wireless technologies to meet the requirements of Internet of Things (IoT) applications [1,2]. Among the LPWAN protocols, LoRaWAN [3] is gaining a significant interest from both the academia and industry for IoT and Industrial IoT (IIoT) applications [4–6]. LoRaWAN is based on LoRa (Long Range), a radio modulation scheme that ensures communication over long distances while maintaining low power consumption.

In a LoRa-based network, several communication parameters must be set before starting a transmission, and a large number of configurations can be obtained [7]. The transmission parameters configuration has an impact on the network performance metrics, such as the packet delivery ratio (i.e., the ratio between the number of messages delivered and the number of messages sent) and

* Corresponding author.
E-mail addresses: luca.leonardi@unict.it (L. Leonardi), giancarlo.iannizzotto@unime.it (G. Iannizzotto), mattia.pirri@phd.unict.it (M. Pirri), gaetano.patti@unict.it (G. Patti), alessio.pirri@phd.unict.it (A. Pirri), lucia.lobello@unict.it (L. Lo Bello).

https://doi.org/10.1016/j.iot.2025.101786
Received 29 July 2025; Received in revised form 14 September 2025; Accepted 2 October 2025
Available online 9 October 2025

the energy consumption. Moreover, as the wireless channel changes over time, the network configuration has to be tuned at runtime to maintain the required performance.

The Adaptive Data Rate (ADR) mechanism provided by LoRaWAN manages two transmission parameters, i.e., the spreading factor (SF) and the transmission power (TP). The spreading factor represents the number of bits encoded by a LoRa symbol. The transmission power is configurable in steps of 1 dBm with values between 0 and 20 dBm [8]. According to the LoRaWAN standard, the allowed spreading factor values are $\{7, 8, 9, 10, 11, 12\}$. For each SF unit increment, the data rate is approximately halved, therefore the Time on Air (ToA), i.e., the duration of a message transmission (calculated as in [9]) and, consequently, the energy consumption are roughly doubled. ADR aims to set the fastest data rate and maintain the energy consumption as low as possible. In particular, for each end-device, the network server analyzes the time series of the Signal-to-Noise Ratio (SNR) values measured during the most recent receptions by the end-device to calculate the new transmission parameters and send them to the end-device through an ADR command.

ADR is a simple rule-based mechanism that calculates the transmission parameters through a set of rules that define a relationship between input (i.e., SNR) and output (i.e., SF and TP). However, as discussed in [10], ADR works for static end-devices, as it assumes to run under stable radio conditions, while it is unsuitable for dynamic scenarios with mobile end-devices moving around the sensing area under highly variable network conditions. To solve this problem, Semtech recently introduced the Blind ADR (or Application-Driven ADR) [11] technique in which a given application defines multiple data rates to be used over time. Blind ADR operates at the application layer by cycling through a fixed set of spreading factors. Although the technique adopted by Blind ADR simplifies the network management and configuration, it is not clear to what extent it could achieve defined targets on the packet delivery ratio and the energy consumption, as it assigns the spreading factor blindly, i.e., regardless of the actual network conditions.

In this context, by leveraging large datasets generated through simulation tools, deep neural networks (DNNs) can learn with fine granularity the non-linearities of the wireless channel. As a consequence, DNN models can be trained to predict the most suitable transmission parameters, according to the constraints on the packet delivery ratio and energy consumption imposed by the IoT application. For this reason, this paper proposes Rel-ADR, a deep learning-based algorithm specifically designed to dynamically tune the spreading factor and transmission power of LoRaWAN end-devices. Rel-ADR is designed to operate effectively in both static and dynamic scenarios with mobile end-devices and high channel variability. By leveraging Long Short-Term Memory (LSTM) networks, Rel-ADR overcomes the limitations of the ADR mechanism in dynamic scenarios. Moreover, Rel-ADR enables reliable communication, while minimizing the energy consumption of the end-device. The Rel-ADR algorithm captures long-term dependencies in the time series data on the end-device transmissions signal quality that are readily available at the network server, i.e., the SNR and Received Signal Strength Indicator (RSSI) values, without the need for using the end-devices position. Rel-ADR, therefore, runs on the network server, does not require any hardware or software modification to the LoRaWAN end-devices and can be easily adopted in existing LoRaWAN networks.

The main contributions of this work are the design of Rel-ADR, which includes the DNN architecture and training procedure, the relevant input data aggregation and preprocessing, and the evaluation of the Rel-ADR performance in realistic scenarios. In particular, the results of comparative assessments with relevant approaches in the literature obtained through OMNeT++ simulations are presented.

The paper is organized as follows. Section 2 outlines the LoRaWAN standard and the ADR algorithm. Section 3 deals with related work. Section 4 presents the DNN model design, training and evaluation, while Section 5 addresses the Rel-ADR architecture and operation. Section 6 presents the methodology adopted to assess Rel-ADR. Section 7 presents the results of preliminary assessments aimed at tuning the Rel-ADR parameters. Section 8 discusses the results of comparative assessments between Rel-ADR and other relevant solutions in the literature. Finally, Section 9 concludes the paper and outlines future research directions.

2. LoRaWAN background

LoRaWAN uses a star-of-stars topology, where a large number of end-devices communicate with one or more gateways. The gateways act as bridges, forwarding uplink and downlink messages between the end-devices (EDs) and the network server (NS), and vice versa, via standard IP connections. The term uplink is used to indicate a message sent by an end-device to the network server via one or multiple gateways, while the term downlink indicates a message sent by the network server to a single end-device through one gateway. LoRaWAN adopts the LoRa modulation developed by Semtech as physical layer. LoRa operates in Sub-GHz bands and exhibits remarkable propagation characteristics that enable long-range communication while maintaining high energy efficiency. Each LoRa device may configure multiple transmission parameters. The values allowed for the transmission parameters depend on the region in which the LoRaWAN devices are deployed [12].

Fig. 1 shows the architecture of a LoRaWAN network, highlighting the roles of the end-device, gateway, and network server. The end-device implements the LoRaWAN protocol stack that runs over the LoRa physical layer. The end-device software stack includes the ADR module, called ADR Backoff, that is in charge of managing transmission parameters.

The gateway acts as a transparent relay, forwarding LoRa packets from the end-devices to the network server, and vice versa, via IP using a packet forwarder. In particular, each gateway adds metadata, such as SNR and RSSI, to the message when forwards it to the network server. This way, the network server is able to analyze the signal quality history and send ADR commands accordingly.

The network server runs the LoRaWAN stack and includes the ADR module, here called ADR-NS, which processes signal quality metrics (e.g., RSSI, SNR) to recommend transmission parameters to the end-devices.

The ADR is designed for static end-devices, while for mobile deployments Semtech proposed Blind ADR [11]. Both mechanisms are described in the following.

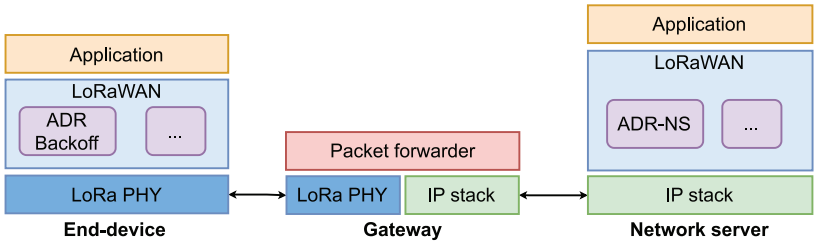

Fig. 1. LoRaWAN architecture.

2.1. Adaptive Data Rate (ADR)

The ADR mechanism asynchronously operates on the end-devices and the network server, with most of the complexity handled by the network server to keep the end-devices simple. The ADR mechanism run by the end-devices, called ADR Backoff, and the configuration commands exchanged between the end-devices and the network server are defined in [3]. The algorithm run by the network server, here called ADR-NS, is not standardized, but Semtech provided some recommendations in [10]. Both the end-devices and the network server can enable or disable the ADR by setting the ADR bit in the LoRaWAN frame. If the ADR is enabled, the network server sends the recommended values of spreading factor and transmission power (through a LinkADRReq command).

The ADR Backoff and ADR-NS algorithms are presented in the following.

2.1.1. ADR Backoff

When the ADR is enabled and a given end-device no longer uses the default transmission parameters, i.e., the maximum spreading factor and transmission power allowed in the region where the LoRaWAN network is deployed [12], the end-device starts running the ADR Backoff algorithm defined in the LoRaWAN specification [3]. According to said algorithm, if no acknowledgment from the network server is received, a given end-device will set the ADR acknowledgment request bit (ADRACKReq) in uplink transmissions to ask the network server to send new transmission parameters. If the network server does not reply within a given time interval, the end-device progressively changes the spreading factor and transmission power values to increase the coverage range. If no response is received, the end-device keeps on changing these values until the default parameters are reached.

2.1.2. ADR-NS

According to [10], the network server stores the data (including RSSI and SNR) of the most recent twenty transmissions for each end-device within the network. The ADR-NS algorithm aims to ensure that the received power is greater than the sensitivity of the demodulator and that the SNR is sufficient for demodulation. The ADR mechanism uses simple rules to achieve this. If the SNR is higher than a minimum predefined SNR value, then the spreading factor is progressively reduced, thus increasing the data rate, until either the minimum spreading factor value allowed by the LoRaWAN standard or the minimum SNR value needed for demodulation is reached. At this point, to reduce the power consumption, the transmission power is also reduced. If the SNR is lower than the minimum required for demodulation, the network server progressively increases the end-device transmission power up to the maximum allowed value, while the control of the spreading factor is left to the end-device, which adjusts it using the ADR Backoff algorithm (Section 2.1.1). The SNR value used by the ADR-NS is calculated as the maximum SNR value among the last twenty values, as recommended in [10].

2.2. Blind ADR

Given the limitations of ADR in dynamic scenarios, such as mobile deployments where the channel conditions rapidly fluctuate, Semtech proposed an alternative to ADR, known as Blind ADR, to dynamically assign the spreading factor. Blind ADR is typically implemented at the application layer as a mechanism that blindly cycles through a predefined set of spreading factors without taking any input from the network. A commonly adopted configuration, shown in Fig. 2, assigns three different spreading factors, i.e., SF7, SF10, and SF12, with a fixed usage frequency per hour. In particular, SF7 is used three times, SF10 is used twice, and SF12 is used once per hour. This pattern continuously repeats regardless of the actual network conditions (e.g., SNR, RSSI), thus making the implementation simpler, but potentially suboptimal in dynamic environments.

3. Related work

In recent years, several improvements to the ADR mechanism have been introduced to overcome the limitations in environments characterized by variable channel conditions. As discussed in [13], the approach adopted by ADR is ideal when there is no variability in the channel quality, but it does not respond properly when the physical channel conditions are varying, e.g., due to moving obstacles between two communicating devices. For this reason, other approaches in the literature use a different value of SNR in the ADR-NS algorithm to tune the transmission parameters. For example, ADR+ [13] is a modified version of the LoRaWAN

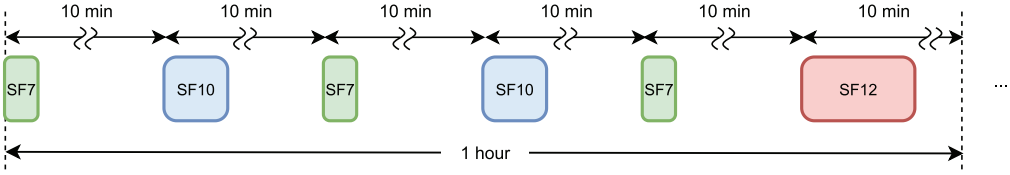

Fig. 2. Typical Blind ADR spreading factor selection.

ADR mechanism that, to cope with the high variability of the wireless channel in fast-fading conditions, instead of estimating the link quality through the maximum SNR among the last twenty values, adopts the average SNR value. The Rel-ADR approach here proposed shares with ADR+ the goal of improving the packet delivery ratio and energy consumption in dynamic IoT scenarios where the network conditions can be highly variable and the end-devices move around the sensing area. However, Rel-ADR estimates the link quality using the median of the last twenty values to smooth the measured SNR. In fact, as discussed in [14], picking the median value avoids the effect of extreme values that might skew the data. The Blind ADR mechanism proposed by Semtech dynamically assigns the spreading factor without considering the actual radio conditions. Conversely, Rel-ADR dynamically adjusts the transmission parameters according to the application needs based on learned patterns derived from the inputs, i.e., the RSSI and the SNR.

Several works have recently focused on the use of Machine Learning (ML) techniques for the adaptive tuning of the transmission parameters in LoRaWAN networks to overcome the ADR limitations and improve resource allocation [15]. The deep learning-based method proposed in [16] uses a Gated Recurrent Unit (GRU) to predict the appropriate spreading factor for LoRaWAN end-devices. The GRU model is trained offline using a dataset that includes end-device location (i.e., the GPS coordinates), SNR, and RSSI. The method in [16] outperforms the traditional ADR mechanism, but the proposed GRU model has high complexity on the end-device, as it consists of almost one million parameters, and requires significant computational time, i.e., about 250 Mega multiply–add cumulation floating-point operations. To solve this problem, in [17] the same authors propose AI-ERA, a Deep Neural Network (DNN) model that reduces the complexity by using the data collected from the last six transmissions, instead of twenty (as done in [10,16]). However, both the approaches in [16,17] require the implementation of a neural network in the LoRaWAN end-devices, thus entailing significant hardware costs and energy consumption. In contrast, the Rel-ADR algorithm here proposed runs on the LoRaWAN network server, thus keeping the complexity and energy consumption of the LoRaWAN end-devices as low as possible.

A similar challenge is addressed in [18] with the Contextual Aware Enhanced LoRaWAN Adaptive Data Rate (CA-ADR) scheme, which adopts a hybrid Convolutional Neural Network-LSTM (CNN-LSTM) model trained offline using an ACK-based dataset refined through a contextual rule-based learning (CRL). CRL enhances training robustness by addressing unreliable ACKs in congested networks. For instance, if ACKs are received for the messages transmitted with SF7 and SF9, but not for those sent with SF8, CRL infers that the missing ACKs for SF8 are likely due to temporary network limitations rather than a failed transmission, and updates the dataset accordingly. The model uses as an input the data of the latest three transmissions, i.e., RSSI, SNR, the X and Y coordinates of the end-device, and the distance from the gateway. Although CA-ADR leverages TinyML [19] for deployment on low-power devices, it still requires computation on the end-device. Conversely, Rel-ADR runs entirely on the network server without device-side intelligence. Moreover, the approaches in [16–18] require the availability of a GPS at the end-device. This assumption may be limiting, especially for dynamic networks, as LoRaWAN devices are typically designed to be low-power and cost-effective.

To reduce the energy consumption on the end-device, in [20] the implementation of the neural model is moved to the network server. However, the approach in [20] still involves the use of the end-device position (e.g., GPS coordinates), while Rel-ADR only leverages data that are readily available at the network server, i.e., the SNR and the RSSI.

Besides the spreading factor allocation, the transmission power selection also significantly impacts on the energy consumption and reliability [7]. For this reason, Rel-ADR assigns both the spreading factor and transmission power. A hybrid approach proposed in [21] combines deep learning performed centrally on the network server with TinyML models optimized for deployment on resource-constrained end devices. Specifically, a CNN-LSTM model is trained and executed on the network server, while an optimized TinyML version of the same model is also deployed on the end-devices, where it replaces the ADR Backoff mechanism. This local model is activated only when the end-device does not receive feedback from the network server for more than three consecutive uplinks. Unlike Rel-ADR, this approach still relies on GPS-based location data.

In [22], two deep learning models are proposed to jointly address collision detection and spreading factor allocation in LoRaWAN. The authors explore both fully connected neural networks and convolutional neural networks, trained offline on simulated data using the end-device coordinates and spreading factor as input features. The models output a classification label indicating whether a transmission would be successful, interfered, or under sensitivity. However, the approach requires GPS data and does not directly infer the spreading factor. Instead, for each transmission, the system starts from the lowest spreading factor and incrementally tests higher spreading factors until a successful transmission is predicted. This iterative process increases both the computational complexity and latency. In contrast, Rel-ADR directly performs spreading factor and transmission power assignment at runtime using features, such as RSSI and SNR, which are readily available without requiring GPS or additional end-device-side intelligence. In fact, while the approach in [22] focuses on predicting collisions to guide an iterative selection of the spreading factor, Rel-ADR directly learns to predict the transmission parameters through supervised learning, thus avoiding intermediate classification steps and repeated inference.

A machine learning-based transmission power control scheme for LoRaWAN is introduced in [23]. This approach predicts the SNR expected for a transmission using environmental variables, such as barometric pressure and particulate matter. The predicted SNR is used as input to an algorithm similar to the ADR proposed by Semtech [10], thus allowing the algorithm to be run on the end-device. The approach in [23] is trained on the dataset presented in [24], which includes both radio and environmental measurements collected in an urban scenario. Unlike Rel-ADR, which directly predicts the spreading factor and transmission power from RSSI and SNR, this method relies on an intermediate SNR prediction step and requires additional sensing capabilities that may increase both the energy consumption and the hardware cost of the end-device.

The work in [25] addresses the twofold aim of minimizing the energy consumption and maximizing the packet delivery ratio through two distinct strategies. Energy consumption is reduced using a centralized supervised machine learning approach for transmission power allocation. In contrast, the spreading factor assignment is treated as a contextual multi-armed bandit problem and tackled in a decentralized manner to improve the packet delivery ratio. However, the approach requires the knowledge of the end-devices location to calculate the distance to the nearest gateway.

The centralized carrier frequency assignment mechanism proposed in [26] requires the gateway to implement an independent neural network for each end-device connected to it, thus limiting the scalability when the number of end-devices increases. Conversely, Rel-ADR requires a single neural network for handling all the end-devices. The work in [27] proposes a deep reinforcement learning algorithm, called LoRaDRL, that uses a single neural network in the gateway to dynamically adapt the spreading factor and transmission power. LoRaDRL improves the packet delivery ratio and responsiveness to dynamic conditions, but it introduces additional overhead, as each end-device, after each transmission, waits for the gateway acknowledgment. The overhead increases the energy consumption of the end-devices and may be a limitation in the regions in which LoRaWAN devices must comply with duty cycle constraints according to [12].

An approach based on Multi-Agent Reinforcement Learning (MARL) is presented in [28], where the energy efficiency of LoRaWAN-based Wireless Underground Sensor Networks (WUSNs) is improved by dynamically adjusting the spreading factor and transmission power. The proposed algorithm runs entirely at the gateway using a centralized training with decentralized execution (CTDE) paradigm, and assigns a dedicated agent to each node to learn optimal transmission policies. The reward function considers link quality, energy consumption, and packet collisions, thus enabling the system to adapt to dynamic underground conditions, such as varying soil moisture and burial depth. Unlike Rel-ADR, this approach requires a more complex MARL framework and is tailored to WUSNs rather than general LoRaWAN scenarios.

The work in [29] introduces a novel reinforcement learning algorithm for configuring the transmission parameters of end-devices that works in a centralized mode on the network server and aims to reduce the energy consumption and enhance the packet delivery ratio. In particular, a Low-Power Multi-Armed Bandit (LP-MAB) mechanism is proposed that combines non-stationary adversarial and stochastic methods employing the exponential weights for the exploration and exploitation and the successive elimination techniques. The network server learns how to configure the spreading factor, coding rate (CR), transmission power, and carrier frequency (CF) of all the end-devices, based on a reward system that depends on the reception of an acknowledgment message. However, this approach, unlike Rel-ADR, requires that LoRaWAN works in confirmed mode. This choice entails additional traffic, and therefore a network overhead that may affect the packet delivery ratio.

In contrast to LP-MAB, Rel-ADR adopts a machine learning model that runs on the network server and uses SNR and RSSI as input features without introducing additional traffic overhead.

A different approach is proposed in [30], where the problem of resource allocation in LoRaWAN networks is addressed through network slicing and Multi-Armed Bandit (MAB) algorithms. The work introduces three alternative lightweight solutions that dynamically allocate resource blocks (i.e., a set of channels for a time slot) to virtual network slices based on service-level priorities. Unlike most ML-based methods that require device-side intelligence or GPS data, this solution runs entirely on the network server and aims to improve the packet delivery ratio by reducing the number of collisions, while meeting the Service Level Agreements (SLAs) of each slice. Unlike Rel-ADR, the approach in [30] does not directly address the adaptive tuning of transmission parameters, such as spreading factor or transmission power. However, the work demonstrates the effectiveness of MAB-based strategies for centralized, low-complexity resource management in LoRaWAN environments.

Among the techniques proposed in the literature for the adaptive tuning of the transmission parameters in LoRaWAN networks, in this work the most relevant approaches were considered for a comparative simulative assessment with Rel-ADR. In particular, ADR [10], ADR+ [13], and Blind ADR [11] (only for mobile scenarios) were considered as baseline approaches. Moreover, three other approaches in the literature were investigated for the sake of comparison, i.e., LP-MAB [29], AI-ERA [17], and CA-ADR [18]. LP-MAB is a novel reinforcement learning algorithm that, like Rel-ADR, works in a centralized mode on the network server and aims at improving the packet delivery ratio and energy consumption. Conversely, AI-ERA and CA-ADR are two supervised learning approaches like Rel-ADR, but they run on the end-devices rather than on the network server. Table 1 summarizes the comparison of the approaches here considered.

4. Model design and training

Rel-ADR is a centralized approach, alternative to the ADR software module that runs on the network server, which dynamically configures the transmission parameters of the end-devices in a LoRaWAN network to achieve reliable communication while maintaining low power consumption. Rel-ADR exploits a pre-trained neural network model that generates, for each end-device, the values of the spreading factor and transmission power to be used for message transmission. The model training requires as an input a dataset consisting in a set of tuples, i.e., the SNR, the RSSI and the transmission power used during the last successful

Table 1
Comparative summary of the considered approaches.

Approach	Tuned parameters	Type	Execution	Performance metrics	Strengths	Limitations
ADR [10]	SF, TP	Rule-based	Network server	Packet delivery ratio, energy consumption	Simple, low overhead	Poor adaptability to dynamic channel conditions and mobility scenarios, requires tuned parameters transmission by the network server
ADR+ [13]	SF, TP	Rule-based	Network server	Packet delivery ratio, energy consumption, requires tuned parameters transmission by the network server	Ability to smooth out the SNR fluctuations	Poor adaptability to dynamic channel conditions and mobility scenarios
Blind ADR [11]	SF	Rule-based	End-device	Packet delivery ratio, energy consumption	Simple, no need for channel estimation	Ignores actual radio conditions
LP-MAB [29]	SF, TP, CF, CR	Reinforcement Learning	Network server	Packet delivery ratio, energy consumption	No device-side computation	Requires confirmed mode, requires tuned parameters transmission by the network server
AI-ERA [17]	SF	Supervised Learning	End-device	Packet delivery ratio, energy consumption, convergence period	No need for tuned parameters transmission by the network server	Requires GPS, device-side computation
CA-ADR [18]	SF	Supervised Learning	End-device	Packet delivery ratio, energy consumption, inference time	No need for tuned parameters transmission by the network server	Requires GPS, device-side computation
Rel-ADR	SF, TP	Supervised Learning	Network server	Packet delivery ratio, energy consumption, inference time	No need for GPS or device-side computation, LoRaWAN standard compliance	Requires tuned parameters transmission by the network server

transmission. Any interfering signals originating from other technologies (e.g., other Sub-GHz technologies) or from overlapping LoRa transmissions are treated as noise in the computation of the SNR. Specifically, the noise term in the SNR calculation includes both the thermal noise and any undesired interfering signal, essentially, everything that is not part of the intended useful signal. As a result, any form of interference contributes to lowering the SNR value, thus affecting the perception of the link quality of the Rel-ADR model. The SNR used in the dataset is computed as:

$$\mathrm{SNR}_{\mathrm{dB}} = 10\log_{10}\left(\frac{P_{\mathrm{signal}}}{P_{\mathrm{noise}}}\right) \tag{1}$$

where P_{signal} and P_{noise} are, respectively, the received signal power and the total noise power, which includes both the thermal noise and all interfering signals.

Rel-ADR does not require as an input the distance between a given end-device and the closest gateway, as the model is able to assign the transmission parameters based on the presence of obstacles (detected by means of the consequent drop in RSSI values) and interference, without the distance between source and receiver.

Section 4.1 describes the neural network model design, while Sections 4.2 and 4.3 deal with the generation and labeling of the training dataset, respectively. Section 4.4 describes the training process, while Section 4.5 addresses training and inference time of the Rel-ADR model. Finally, Section 4.6 presents an offline evaluation of the model performance on a test dataset.

4.1. Deep Neural Network model design

In this work, while designing the Rel-ADR Deep Neural Network (Rel-ADR DNN) model, the problem of dynamically predicting the most appropriate spreading factor and transmission power values to improve the performance of the ADR mechanism was considered as a multi-output time series regression. Regression models are particularly effective in scenarios where the outputs are not only dependent on the inputs, but also potentially correlated with each other, and were widely investigated in the literature [31,32]. The multi-output regression model proposed in this paper is summarized in Fig. 3 and exhibits a distinctive branching structure, with an initial Long Short-Term Memory (LSTM) layer followed by a bifurcation that forms two distinct branches, each responsible for one of the output variables.

This branching structure accounts for the different dynamics and scale of the two output variables, which require different regularization parameters and output layers.

The first LSTM layer processes the input sequence. It has 128 units and returns the full sequence of outputs (not just the last one), as the subsequent LSTM layers expect sequences as their inputs. Each of the two branches (named *SFbranch* and *TPbranch*, respectively) is composed of a further LSTM layer with 16 units, configured to process the output from the first LSTM layer. Here, for the LSTM units, Rectified Linear Unit (ReLu) activation functions were preferred for their slightly better performance over Tanh

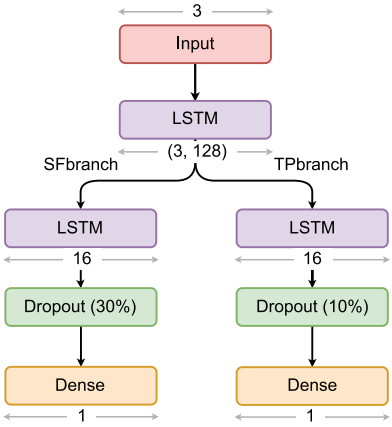

Fig. 3. Rel-ADR DNN model.

Table 2
Path-loss parameters.

Parameter	Value
d_0	1000 m
$\overline{PL}(d_0)$	128.95 dB
γ	2.32
σ	7.8 dB

activations during our preliminary tests (this choice is also supported in [33]). Dropout is used as a regularization technique to prevent overfitting. A dropout rate of 0.3 (30%) for *SFbranch* and of 0.1 (10%) for *TPbranch* were experimentally chosen as a trade-off between the overfitting risk and convergence speed while training. Each LSTM output in *SFbranch* and *TPbranch* is fed to a separate Dense layer with a single unit, initialized with a normal distribution and a linear activation function, in a widely adopted configuration for regression models. In particular, each Dense layer has a single output unit because the number of output units depends on the number of real-valued output variables and our model produces one output variable per branch. The linear activation function showed the best performance over direct competitors, as in [34].

The Rel-ADR DNN model, consisting in 85,154 parameters, has a memory footprint of 332.63 KiB when implemented using float32 weights.

4.2. Dataset generation

Several OMNeT++ simulations were performed in a 10000 m × 10000 m sensing area to generate a dataset that relates the values of SNR and RSSI to the transmission parameters here considered, i.e., the spreading factor and transmission power. The channel was modeled using a log-distance path loss model, expressed as:

$$PL(d) = \overline{PL}(d_0) + 10 \cdot \gamma \cdot \log_{10}\left(\frac{d}{d_0}\right) + X_\sigma \tag{2}$$

where:

- $\overline{PL}(d_0)$ is the path loss at the reference distance d_0
- γ is the path loss exponent
- X_σ is a zero-mean Gaussian distributed random variable with standard deviation σ
- d is the distance to calculate the path loss.

The parameters used in the simulations are those referred as LoRaPathLossOulu configuration implemented in OMNeT++ for the suburban scenario described in [35]. These values are summarized in Table 2.

Here the performance metrics used is the packet delivery ratio measured for a given end-device x (PDR_x), defined as the ratio between the number of messages delivered and the number of messages sent, expressed as a percentage.

The simulations aim to determine the combinations of spreading factor and transmission power that enable an end-device to achieve a packet delivery ratio above a specific threshold (PDR_{thr}) that depends on the application needs, while minimizing the end-device energy consumption, assuming that the gateway position is fixed in the center of the sensing area.

In this work, a suburban scenario was simulated in two configurations. In the first one, a single end-device is deployed in the LoRaWAN network. The position of the end-device varies in steps of 50 m. At each position, the end-device sends N_{TX} messages (in

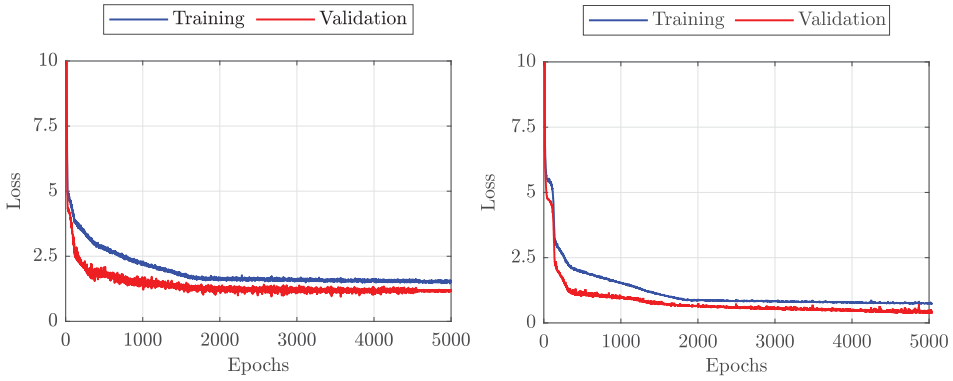

Fig. 4. Training and validation loss progress.

this case $N_{TX} = 1000$) for each combination of the considered transmission parameters, and the packet delivery ratio was measured. In addition, for every successfully received message, the RSSI and SNR values were recorded. In the second configuration, the LoRaWAN network includes 210 end-devices competing for the channel access. This configuration exploits the interference between end-devices, as they may transmit simultaneously. Multiple simulations were run to obtain results that consider different positions of the end-devices. For each position, all the different couples of spreading factor and transmission power (SF, TP) were tested.

4.3. Dataset elaboration and labeling

The simulations run in the scenarios described in Section 4.2 generated a large training dataset. The data obtained from the simulations were pre-processed as follows. For a given position of an end-device in the network, a single entry was stored for each (SF, TP) pair, including the TP value itself and the SNR and RSSI values, both computed as the median of the measurements collected at that position.

All the entries were labeled with the (SF_{Label}, TP_{Label}) pair that obtained a packet delivery ratio higher than the packet delivery ratio threshold (PDR_{thr}) while minimizing the energy consumption. As a result, all entries corresponding to a given end-device position share the same label. The energy consumption due to a message transmission was calculated as in (3),

$$E_{msg} = \text{ToA}(SF) \cdot V \cdot I(TP) \tag{3}$$

where $\text{ToA}(SF)$ is the Time on Air required for the transmissions with a given spreading factor calculated as in [9], V denotes the supply voltage, and $I(TP)$ is the current consumption experienced with a given transmission power [13,36].

For each combination of parameters, the energy consumption calculated as in (3) was used to identify the most energy-efficient configuration that meets the packet delivery ratio requirement. The labeling approach ensures that the dataset includes combinations optimized for both reliability and low energy consumption. The final dataset includes the normalized values of transmission power, the median SNR, the median RSSI, and the corresponding (SF_{Label}, TP_{Label}) labels. This dataset was used for the model training.

4.4. Model training

The DNN model described in Section 4.1 was built and trained using the Tensorflow 2 framework [37] for a maximum of 5000 epochs. The mean absolute error (MAE) was selected as the loss function and the Adam optimizer was used, according to the widely adopted common practice. The dataset was randomly split into training, validation, and test sets with an 80/10/10 ratio and no overlap. Fig. 4 shows the training and validation loss progress along epochs.

Although early stopping was always used during the model training to prevent overfitting and reduce training time, it was deliberately disabled to generate Fig. 4. This choice was made to show that the model does not overfit even after prolonged training, thanks to its architecture and the use of dropout layers. Both the training and validation loss values clearly reach their minimum after 1500 training epochs, where the validation loss is well and consistently below the training loss, thus showing that the model was not overfitting.

4.5. Training and inference computation time evaluation

This section presents training and inference time of the Rel-ADR model. The experiments were conducted on a computer system equipped with an 8-core (16 threads) AMD Ryzen™ 7 3700X CPU @ 3.6 GHz and 16 GB of DDR4 RAM running Linux (Ubuntu 22.04.2 LTS).

Table 3
Inference time statistics over 10,000 samples.

Metrics	Min	Max	Average	Std. Dev.
Inference time (ms)	1.49	2.37	1.51	0.02

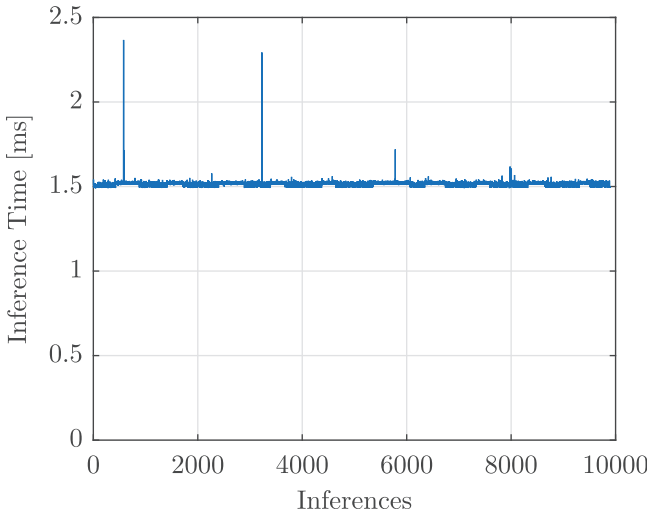

Fig. 5. Inference times.

The measured training time for the model was 450.91 s (approximately 7.5 min). This value highlights the efficiency of the training process, which is fast enough to be entirely executed using only the CPU, without GPU acceleration.

To evaluate the inference latency 10,000 inference operations were run using randomly generated input samples. The inferences were executed in C++ using the frugally-deep[1] library, a lightweight header-only library that enables inference of TensorFlow/Keras models in C++. The model structure was not modified and no optimizations such as quantization or pruning were applied.

The results are summarized in Table 3 and the distribution is shown in Fig. 5.

These results confirm that the model is lightweight and suitable for real-time execution on network servers managing large-scale LoRaWAN deployments.

4.6. Offline model evaluation

This section compares the performance of the Rel-ADR, AI-ERA, and CA-ADR Deep Neural models on their respective test datasets. To perform this comparison, the accuracy, precision, recall, and F1 score metrics are considered. This evaluation focuses on the accuracy of the models in predicting the potential best values for spreading factor and transmission power prior to deployment on the actual LoRaWAN network.

Since AI-ERA and CA-ADR are classification-based models, their performance are evaluated using the categorical cross-entropy loss (CEL). In contrast, Rel-ADR is a regression-based model, and the MAE is adopted Rel-ADR performance.

Moreover, since Rel-ADR performs regression, the accuracy, precision, recall, and F1 score are calculated after rounding predictions to the nearest allowed value, i.e., the nearest integer for SF and the nearest even integer for TP [12]. This conversion of continuous outputs into discrete classes allows the use of classification metrics and facilitates the comparison with AI-ERA [17] and CA-ADR [18], which are classification-based models.

Table 4 shows the accuracy, precision, recall, F1 score, and loss for all the considered models in two different situations, that is, with variance $\sigma = 7.8$ dB, that is the typical value for a suburban scenario [35,13], and with $\sigma = 0$ dB, that represents the ideal scenario [13] without shadowing, where the channel conditions are deterministic and the RSSI and SNR measurements are less noisy. In both scenarios, the Rel-ADR DNN model performs markedly better than its competitors. In particular, in the scenario with $\sigma = 0$ dB, Rel-ADR, AI-ERA, and CA-ADR achieve higher accuracy, as the mapping between input features and transmission parameters becomes more predictable.

These results provide a clear comparison before moving to a real-network assessment.

[1] Frugally-deep: https://github.com/Dobiasd/frugally-deep.

Table 4
Models performance.

Model		σ	Accuracy	Precision	Recall	F1 Score	Loss	
							CEL	MAE
AI-ERA [17]			42.17%	41.24%	42.17%	41.59%	0.676	
CA-ADR [18]			25.17%	24.20%	25.17%	24.35%	1.505	
	SF	7.8 dB	75.31%	79.86%	75.31%	75.54%		0.313
Rel-ADR	TP		79.01%	77.76%	79.01%	75.16%		0.937
	Combined		61.73%	58.17%	61.73%	57.79%		1.250
AI-ERA [17]			82.50%	82.38%	82.50%	82.30%	0.135	
CA-ADR [18]			64.33%	64.34%	64.33%	63.98%	0.973	
	SF	0 dB	97.20%	97.32%	97.20%	97.12%		0.067
Rel-ADR	TP		97.90%	98.00%	97.90%	97.92%		0.328
	Combined		95.45%	96.18%	95.45%	95.27%		0.396

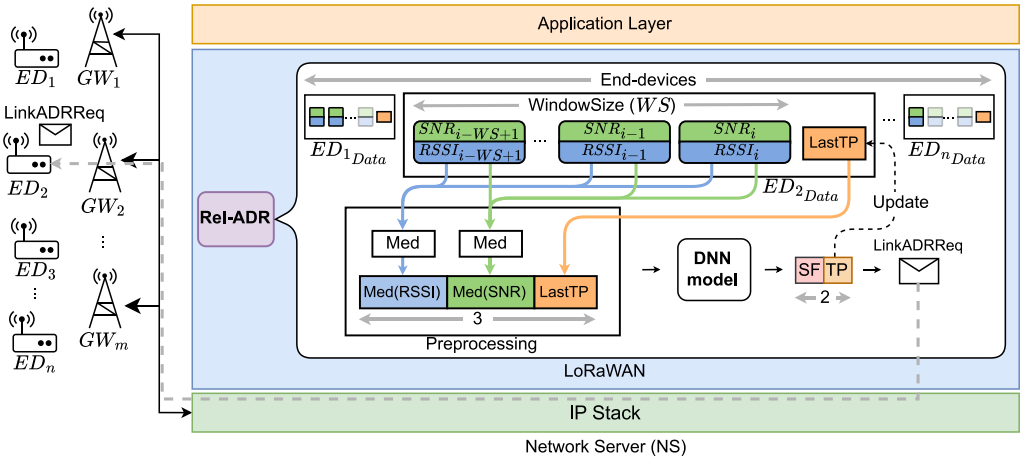

Fig. 6. The Rel-ADR architecture.

5. Rel-ADR architecture and operation

Rel-ADR is a software module designed to replace the ADR-NS software module proposed by Semtech and shown in Fig. 1. The Rel-ADR software architecture, shown in Fig. 6, is built around the dual-output regression model described in Section 4.6, named DNN model.

A LoRaWAN network that adopts Rel-ADR does not require any hardware modifications to the network nodes, i.e., end-devices and gateways (ED and GW, respectively, in Fig. 6). In fact, as shown in Fig. 6, the approach here presented proposes to extend only the network server software architecture adding a module, called Rel-ADR, that replaces the ADR-NS software module in Fig. 1. Rel-ADR manages a small database containing the time series data on the signal quality of the end-devices' transmissions. Rel-ADR preprocesses the data and feeds them to the DNN model to produce an estimate of SF and TP for each node at the current time. In Fig. 6 the database consists of the set of the n blocks called ($ED_{1_{Data}}$, $ED_{2_{Data}}$, ... $ED_{n_{Data}}$), where n is the number of end-devices in the network and $ED_{x_{Data}}$ represents the time series data of a given end-device x and stores, as recommended by Semtech [10], the SNR and RSSI values measured by the gateway for the latest WindowSize messages received from the end-device x. WindowSize (WS) is the time series length and can be individually configured for each end-device. The database also stores the transmission power used by each end-device during the last successful transmission. The amount of memory required for the database grows linearly with the number of end-devices and is the only contribution of the memory footprint that increases with the number of end-devices.

The Rel-ADR operational phase can be split into two subphases, i.e., transient and steady-state. During the transient subphase, the end-devices transmit using the default transmission parameters [3]. The transient subphase for each end-device ends upon its first reception of the transmission parameters generated by Rel-ADR, delivered through the LinkADRReq command. In particular, Rel-ADR generates the spreading factor and transmission power values when one of the following events occurs:

- An uplink with the ADRACKReq bit set is delivered.
- A number of messages equal to the configured WindowSize, sent by an end-device, are successfully received by the network server after the last Rel-ADR parameters generation for the end-device.

The transmission parameters are generated using the DNN model described in Section 4.1, which receives as input three data values from the output of the preprocessing block (see Fig. 6):

- The median SNR calculated from the SNR values of the latest WS messages received. This value provides significant insights on both the signal quality and interference levels.
- The median RSSI computed from the RSSI values of the latest WS messages received. This value provides significant insights about the signal strength at the receiver.
- The LastTP, i.e., the transmission power used by an end-device during the last successful transmission.

If the network server has not yet collected the WS messages from an end-device, but it receives a parameter update request from this end-device, the preprocessing block produces its output based on the available samples and the output is fed as input to the DNN model.

The DNN model generates as output two real numbers (i.e., the spreading factor and transmission power values) that are rounded to the nearest allowed integer value and sent to the end-device.

However, if the generated parameters are identical to those currently used by the end-device, they are not transmitted unless explicitly requested by an uplink with the ADRACKReq bit set. This decision is made to reduce the downlink workload on the gateway, which may lead to packet loss due to the duty cycle limitations. If an end-device loses the connection with the network server (i.e., the network server does not reply to any uplink message for a given time interval), the end-device progressively changes the values of the spreading factor and transmission power to increase the coverage range following the ADR Backoff algorithm defined in the LoRaWAN specification [3]. To prevent the activation of the ADR Backoff mechanism when the lack of downlink is due to unchanged transmission parameters rather than to actual connectivity loss, Rel-ADR is designed to respond to uplink with the ADRACKReq bit set, even when the parameters have not changed. This ensures that Rel-ADR maintains control over the transmission parameters configuration. This way, on one hand, Rel-ADR reduces downlink overhead, on the other hand, it avoids unnecessary increase of spreading factor and transmission power due to the ADR Backoff that runs on the end-device. It is worth noting that the Rel-ADR mechanism minimizes the use of downlink messages by design, sending an ADR MAC command only when necessary to adjust transmission parameters. As a result, the impact on the gateway duty-cycle remains limited. In dense deployments, where downlink capacity could become a concern, the use of multiple gateways, common in real-world LoRaWAN networks, effectively distributes the load and increases the overall downlink capacity of the network.

6. Evaluation

This section presents the methodology adopted to evaluate Rel-ADR. In particular, in Section 6.1 both the framework adopted for the assessments and the performance metrics are introduced, while Section 6.2 deals with the description of the simulation setup.

6.1. Evaluation methodology

To assess the Rel-ADR performance, a two-step methodology is adopted. In the first step the impact of the Rel-ADR parameters, i.e., PDR_{thr} and WindowSize, on the communication reliability is evaluated through simulations, varying the number of end-devices, with the aim of selecting the best values for these parameters to be used in the second step.

The PDR_{thr} and WindowSize values thus obtained are adopted in the second step simulations, to compare the Rel-ADR performance with that of other approaches in the literature, varying the number of end-devices, the mobility patterns and channel conditions.

The simulations were performed using OMNeT++. In particular, the Framework for LoRa (FLoRa) [13] was adopted to simulate the main features of the LoRaWAN end-devices, the LoRa physical layer, and the wireless channel. Moreover, the pre-trained DNN model (described in 4.1) was developed in the software module called Rel-ADR, as shown in Fig. 6. Before each message transmission, a duty cycle check function runs on both the end-devices and the gateway to ensure the compliance with the ETSI limitations [38,39]. The imperfect orthogonality of LoRa spreading factors was modeled as in [40].

In the simulations, two performance metrics were used, i.e., the packet delivery ratio (PDR) and energy consumption.

The packet delivery ratio, measured at the application layer, is expressed as a percentage according to (4),

$$PDR = \left(\frac{\mathcal{N}_{RX}}{\mathcal{N}_{TX}} \right) \cdot 100 \tag{4}$$

where $\mathcal{N}_{RX}$ and $\mathcal{N}_{TX}$ are, respectively, the total number of packets correctly received by the network server and the total number of packets sent by all the end-devices during the entire simulation.

The energy consumption (EC), measured in Joule, is defined in (5) as the total energy consumption divided by the packet delivery ratio, as done in [29], i.e.

$$EC = \frac{Energy}{PDR} \tag{5}$$

where $Energy$ is the energy consumed by all the end-devices in the network and the PDR is calculated using (4).

Table 5
Simulation parameters.

Parameter	Value
Mobility Model	Random Waypoint
Application payload	20 bytes
Bandwidth	125 kHz
Simulations time	12 days [13,29]
Number of end-devices	{100, …, 1000}
Propagation model	log-distance (Eq. (2))
Path-loss parameters	LoRaPathLossOulu [35] (see Table 2)
Spreading factor	{7, 8, 9, 10, 11, 12}
Transmission power	{2, 4, 6, 8, 10, 12, 14} dBm [12]
Carrier frequency	{868.1, 868.3, 868.5} MHz
Coding rate	$\frac{4}{5}$
Message generation pattern	exp(1000 s) [13]
Rel-ADR WindowSize (WS)	20
ADR and ADR+margins	30 dB [41]

6.2. Simulation setup

The assessment was performed in a scenario that refers to a realistic suburban use case, similar to the one adopted in [13]. The simulated network includes a variable number of end-devices (e.g., sensors) uniformly distributed over a square sensing area of 9800 m × 9800 m, a network server and a gateway located at the center of the sensing area. Each end-device transmits, according to an exponential distribution with mean 1000 s, a message with a 20-byte long application payload as in [13]. The simulation time was set to 12 days to collect a significant amount of data.

Table 5 shows the configuration parameters used in the simulations.

7. Preliminary assessment

This section presents the results of preliminary assessments aimed at tuning the Rel-ADR parameters. In particular, Section 7.1 assesses the impact of different PDR_{thr} values on the packet delivery ratio and energy consumption with a varying number of (static or mobile) end-devices. Section 7.2 presents the packet delivery ratio results obtained as a function of the WindowSize with a fixed number of static and mobile end-devices, respectively.

7.1. Impact of PDR_{thr}

This section assesses the impact of different PDR_{thr} values on the packet delivery ratio and energy consumption with a varying number of (static or mobile) end-devices. In particular, three different PDR_{thr} values were considered, i.e., 80%, 85%, and 90%. The evaluation was performed in a suburban scenario by varying the number of end-devices from 100 to 1000, in steps of 100 end-devices.

The choice of these thresholds reflects different application requirements. For instance, the 90% threshold is adopted in [42] to represent reliability targets in monitoring applications. Lower thresholds such as 85% and 80% were also considered to explore the behavior of the network under less stringent reliability constraints, which may be acceptable in non-critical scenarios.

To investigate the impact of different PDR_{thr} values on the network performance the Rel-ADR model was trained three times. Each training run used a distinct dataset, created preprocessing the original data with one of the three PDR_{thr} values.

Static scenario. Fig. 7 shows the impact of the PDR threshold on the packet delivery ratio and energy consumption with a varying number of static end-devices.

Fig. 7(a) shows that, as expected, the packet delivery ratio decreases as the number of end-devices increases. For networks with a low number of end-devices (i.e., up to about 300), a high PDR_{thr} (e.g., 90%) results in a higher packet delivery ratio than in the case a lower PDR_{thr} is used. Conversely, in networks with a high number of end-devices (i.e., more than 500 in the considered scenario), a lower PDR_{thr}, e.g., 80%–85%, results in a higher packet delivery ratio than with a higher threshold. This behavior can be explained as follows. When the number of end-devices is limited (i.e., up to about 300), a packet loss is mainly due to the path loss. Consequently, using a higher PDR_{thr} (that results in training the DNN to generate higher spreading factor and transmission power values) leads to a higher packet delivery ratio. Conversely, when the network consists of many end-devices, packet loss is mostly due to collisions. In this case, with a lower PDR_{thr} the DNN model is trained to generate lower values for the parameters, thus generally reducing the ToA and therefore the probability of collisions. Moreover, the reduced transmission power helps minimize the interference levels.

Fig. 7(b) shows the energy consumption obtained varying the number of static end-devices for each considered PDR_{thr} value. In all the considered scenarios, the energy consumption is higher when the PDR_{thr} is set to 90% and progressively decreases when the DNN models is trained with PDR_{thr} equal to 85% and 80%. These results are expected because, as discussed before, the use

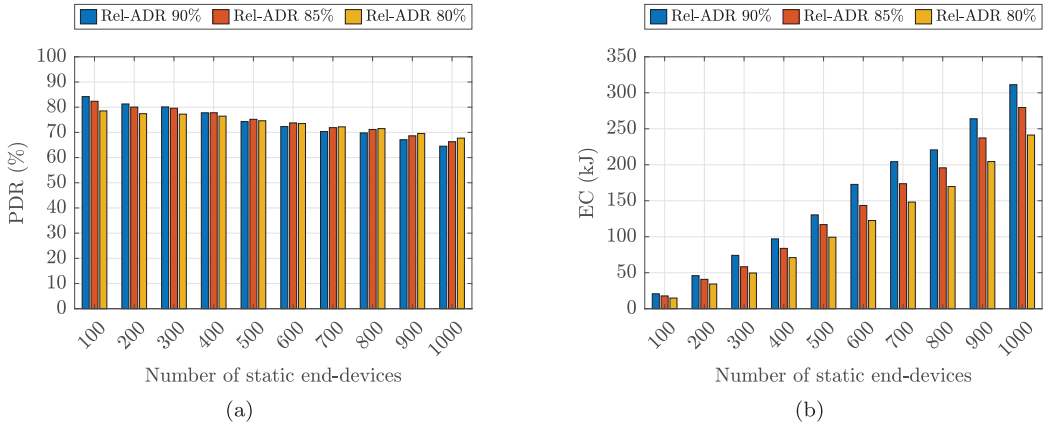

Fig. 7. Packet delivery ratio (a) and energy consumption (b) results varying PDR_{thr} values and the number of static end-devices.

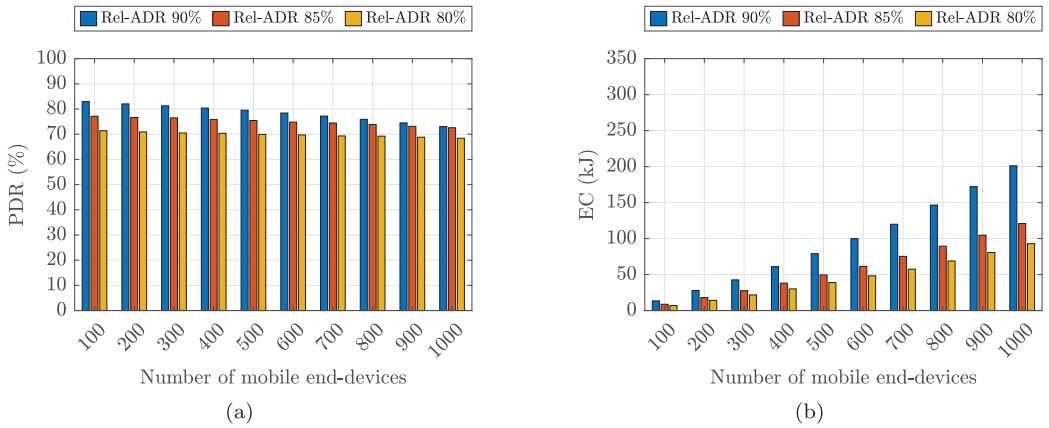

Fig. 8. Packet delivery ratio (a) and energy consumption (b) results varying PDR_{thr} values and the number of mobile end-devices (Randow Waypoint).

of a lower PDR_{thr} value during dataset preprocessing leads to the selection of lower spreading factors, and therefore to shortened ToAs. As a result, the time spent by the end-devices in transmit mode is reduced and the energy consumption decreases.

In Section 8, the best performing PDR_{thr} was used for each scenario. In particular, for the static scenario, a PDR_{thr} of 80% was adopted.

Mobile scenario. To assess the robustness of Rel-ADR under dynamic network conditions, additional simulations were run with mobile end-devices. In this scenario, the same suburban area and simulation parameters were considered while varying the mobility pattern of the end-devices. In particular, two widely adopted mobility models were tested, i.e., the Random Waypoint and Chiang mobility.

In both mobility configurations, the number of end-devices varies from 100 to 1000 (in steps of 100), and each end-device moves during the simulation according to the selected pattern. The speed of each end-device is randomly selected from a uniform distribution in the range of 1 to 5 meters per second.

Figs. 8 and 9 show the results obtained considering the Random Waypoint and Chiang mobility patterns, respectively.

When adopting a more realistic mobility pattern, i.e., the Chiang Mobility, the results in Fig. 9 show a slightly lower packet delivery ratio and generally a higher energy consumption compared to the static scenario (Fig. 7). This behavior was expected, as the mobility of the end-devices causes larger fluctuations in SNR and RSSI, thus achieving reliable transmission becomes more challenging. For this reason, higher values of spreading factor and transmission power are used. However, the performance degradation remains limited.

Conversely, with the Random Waypoint model, a different trend emerges. As shown in Fig. 8, when the Rel-ADR model is trained with PDR_{thr} equal to the 80% it achieves a more stable packet delivery ratio as the number of end-devices increases. Surprisingly, the packet delivery ratio in the Random Waypoint scenario is higher than in the static case (Fig. 7(a)) and the energy consumption is significantly lower, especially in dense networks. These effects are explained by the inherent spatial bias of the Random Waypoint model, which tends to concentrate the end-devices toward the center of the simulation area, i.e., closer to the

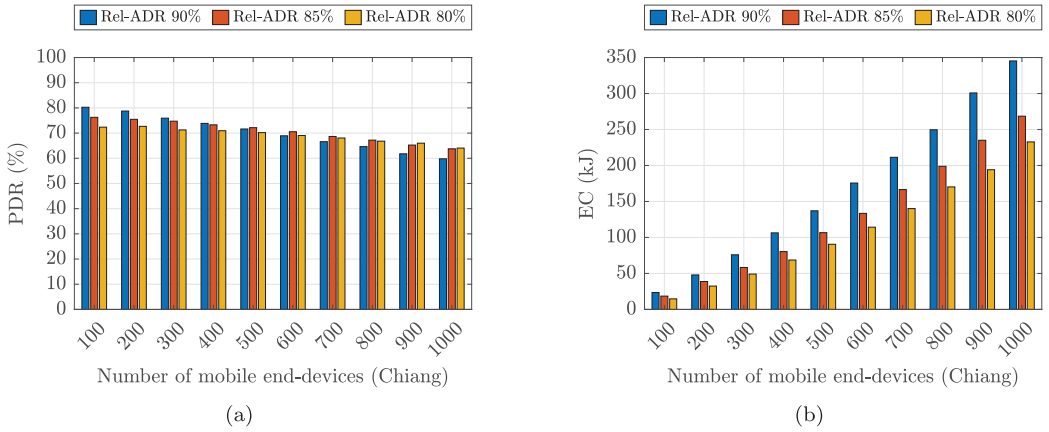

Fig. 9. Packet delivery ratio (a) and energy consumption (b) results varying PDR_{thr} values and the number of mobile end-devices (Chiang).

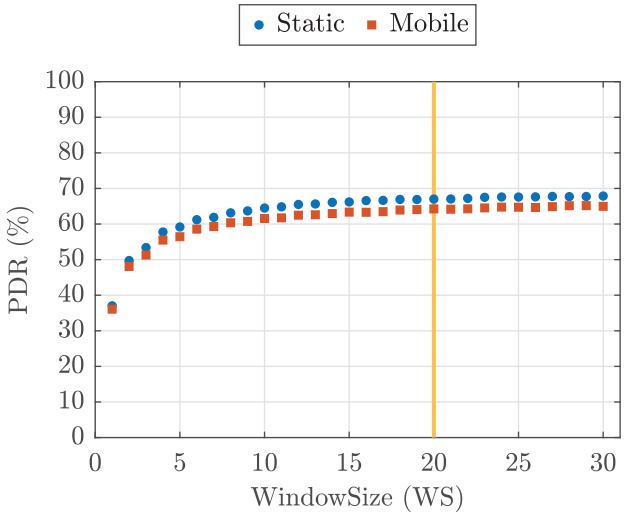

Fig. 10. Packet delivery ratio varying the WindowSize.

gateway. This aggregation around the gateway reduces the average communication range, thus resulting in improved link quality and reduced need for high spreading factors or transmission powers. This behavior is consistent with the prior studies in [43–45], which analytically and empirically demonstrate that Random Waypoint mobility leads to non-uniform node distributions with higher densities around the center of the area.

In Section 8, for the mobile scenarios the PDR_{thr} depends on the adopted mobility model. For example, in scenarios in which the end-devices move according to the Random Waypoint model, a PDR_{thr} equal to 90% is adopted, as it consistently ensures the highest reliability across all considered network densities. While this choice determines the highest energy consumption, the overall consumption remains lower than the one measured in the static scenario.

7.2. Evaluation of packet delivery ratio and energy consumption varying the WindowSize

In this assessment, two different scenarios were considered, i.e., a network consisting of 1000 static end-devices and one with 1000 mobile end-devices. Fig. 10 shows the impact of the Rel-ADR WindowSize on the packet delivery ratio.

The results show that, in both scenarios, the packet delivery ratio increases as the WindowSize grows. This behavior is due to two key factors. First, low WindowSize values make the Rel-ADR DNN model more sensitive to outliers and short-term fluctuations in network conditions, thus resulting in unstable transmission parameter assignments and lower packet delivery ratios. Second, low WindowSize values determine more frequent transmission parameter updates, thereby increasing the number of downlink messages. The downlink transmissions may collide with the uplink ones, thus reducing the packet delivery ratio further. Conversely, high WindowSize values help mitigate both issues, but at the expenses of a reduced DNN model responsiveness under highly variable network conditions. As a result, a trade-off between the transmission robustness and Rel-ADR responsiveness is required.

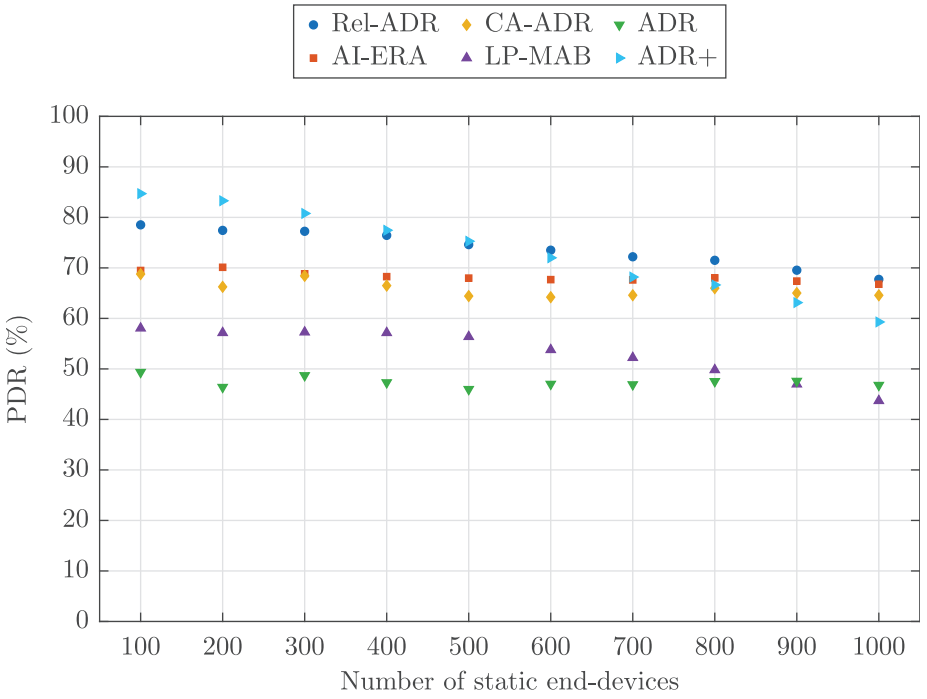

Fig. 11. Static scenario — Packet delivery ratio versus the number of static end-devices.

Henceforth, a WindowSize equal to twenty was considered, as suggested in [10].

8. Comparative assessment

Here, Rel-ADR performance are compared with those obtained by the plain ADR [10] mechanism adopted by LoRaWAN and other approaches in the literature for tuning the LoRaWAN transmission parameters, i.e., ADR+ [13], Blind ADR [11] (only in mobile scenarios), LP-MAB [29], AI-ERA [17], and CA-ADR [18].

In this assessment, two scenarios were considered, i.e., a static scenario and a mobile scenario, where the number of end-devices varies between 100 and 1000. These scenarios are addressed to assess the impact on the network performance of the end-device density in two different cases, i.e., when the network consists of static end-devices only (Static case in Section 8.1) and when the network consists of mobile end-devices only (Mobile case in Section 8.2). This choice allows a comparison with the results in [29,13].

In addition, Section 8.3 presents a further assessment performed to evaluate the effect of the wireless channel variability by changing the standard deviation σ of the shadowing component in the path loss model.

To assess the ADR and ADR+ performance, a margin_db value equal to 30 dB was adopted. This margin_db is the highest recommended value according to the work in [41]. Such a choice is made because adopting a margin dB value of 10 dB, as suggested by the recommendations [10], leads to poor performance for ADR and ADR+ in the assessed scenarios.

8.1. Static end-devices

In this scenario, the packet delivery ratio and the energy consumption were evaluated while increasing the number of end-devices and, consequently, their density.

Packet delivery ratio. Fig. 11 shows the packet delivery ratio obtained by all the approaches under consideration.

In general, the packet delivery ratio decreases when the number of end-devices increases. This is because, when the number of end-devices in the network increases, the workload grows and the probability of collision between transmissions also increases.

Among the considered approaches, the supervised learning-based methods, i.e., Rel-ADR, AI-ERA, and CA-ADR, consistently achieve higher packet delivery ratios compared to LP-MAB and ADR, which are based on reinforcement learning and rule-based heuristics, respectively. For example, with 100 end-devices, Rel-ADR achieves a packet delivery ratio of 78.52%, compared to 58.05% and 49.35% obtained, respectively, by LP-MAB and ADR. These results represent a 35.26% improvement over LP-MAB and a 59.11% improvement over ADR. Similarly, with 1000 end-devices, Rel-ADR achieves a packet delivery ratio of 67.73%, which is an improvement of 55.02% over LP-MAB and 19.37% over ADR. The lower performance of LP-MAB compared with Rel-ADR is due to the LP-MAB alternating exploration and exploitation phases. In fact, during the exploration phase all the possible parameter

combinations are tested and some of them actually lead to packet loss. Even in the exploitation phase, in which the transmission parameters are selected through a probability distribution, there is still a small chance of selecting a combination of parameters that results in a lower packet loss ratio than Rel-ADR. Similarly, ADR obtains lower packet delivery ratio values than Rel-ADR due to its simple rule-based nature.

The other considered rule-based approach, i.e., ADR+, obtains the highest PDR values up to 300 end-devices, while it performs similar to Rel-ADR in networks with a number of end-devices between 400 and 600. When the number of end-devices is higher than 600, the ADR+ performance degrades compared to the supervised learning-based methods here considered. In fact, ADR+ uses the average of the last twenty SNR values to estimate the link quality (instead of the maximum SNR used by ADR). This approach overcomes ADR in case of fast-fading conditions, but it does not avoid the effect of extreme values. In particular, in networks with a high number of end-devices, low SNR values are often measured, leading to the selection of high SF values, which results in long ToAs and in a high number of collisions in dense networks, i.e., a degradation of the PDR performance. In fact, with 1000 end-devices ADR+ achieves a packet delivery ratio 12.46% lower than the one obtained by Rel-ADR, which maintains higher reliability in dense deployments, confirming its scalability and robustness. This result shows that supervised learning effectively captures the complex relationships between the input variables (e.g., RSSI and SNR) and the most suitable transmission parameters.

Among the supervised learning methods, Rel-ADR achieves the highest packet delivery ratio in all evaluated scenarios. For instance, with 100 end-devices Rel-ADR achieves a 78.52% packet delivery ratio, compared with the 69.49% and 68.77% values obtained by AI-ERA and CA-ADR, respectively. This represents a relative improvement of 12.99% and 14.18%, respectively. Similarly, in the scenario with 1000 end-devices, Rel-ADR reaches a packet delivery ratio of 67.73%, compared with 66.77% of AI-ERA and 64.57% of CA-ADR.

Unlike AI-ERA and CA-ADR, Rel-ADR achieves these performance without relying on GPS-based location data. The obtained results suggest that Rel-ADR, thanks to its training process, generalizes more effectively than the other approaches here considered. Consequently, Rel-ADR enables the network server to configure spreading factor and transmission power values that are more accurately aligned with the dynamics of the channel conditions. This consideration reflects the results discussed in Section 4.6, where Table 4 clearly shows that Rel-ADR achieves higher accuracy compared to AI-ERA and CA-ADR.

Energy consumption. Fig. 12 shows the energy consumption obtained by the approaches under consideration with a varying number of end-devices in the network. As it can be seen, for all the approaches, the energy consumption increases with the number of end-devices. This is because, when the number of end-devices grows, the SNR decreases, and therefore, to ensure that transmissions are successfully demodulated even at lower SNRs, the considered approaches select higher values of spreading factor or transmission power. Either way, the energy consumption, calculated as in (3), increases. For a limited number of devices (e.g., 100 end-devices), all approaches exhibit comparable energy consumption. This indicates that, under a low network load, the impact on energy consumption of the transmission parameter selection algorithm is minimal. However, as the number of end-devices increases, the differences in energy consumption between the approaches become more pronounced.

Fig. 12 shows that ADR achieves the lowest energy consumption, as expected. In fact, as discussed in [46] ADR targets to minimize energy consumption while optimizing data rates. LP-MAB consumes approximately 72.99% more energy than Rel-ADR in the configuration with 1000 end-devices, thus highlighting the advantage of the proposed supervised learning over the reinforcement learning approach adopted by LP-MAB. ADR+ achieves a energy consumption slightly lower than LP-MAB, but significantly higher than the ones measured by the other approaches. This result is due to the selection of high SF values, which results in long ToAs, thus in higher energy consumption.

In the configuration with 1000 end-devices, AI-ERA and CA-ADR consume approximately 53% less energy than Rel-ADR. These results are because Rel-ADR, to ensure more reliable transmissions, tends to select more robust spreading factors that increase the ToA and, consequently, the energy consumption. This behavior highlights a trade-off between packet delivery ratio and energy consumption, i.e., that the higher reliability achieved by Rel-ADR, shown in Fig. 11, comes at the expense of a higher energy consumption. However, both AI-ERA and CA-ADR require to run a neural network directly on the end-devices and rely on GPS to obtain the positional data used as input, thus introducing additional computational overhead and hardware requirements.

8.2. Mobile end-devices

In this scenario, a variable number of mobile end-devices that adopt the Random Waypoint Mobility model, as in [29], was considered. The speed of the end-devices is modeled as a uniformly distributed random variable ranging from 0 to 5 meters per second.

Here, the evaluation includes a comparison with the Blind ADR strategy that is specifically devised for networks supporting mobility. Instead, ADR and ADR+ were not considered as they are specifically designed for static end-devices and are not applicable in mobile or dynamic environments [10]. As the original Blind ADR algorithm typically operates over a one-hour window, here, to ensure a fair comparison with the other approaches under test, a modified version was adopted. The adopted version allows Blind ADR to operate at a timescale consistent with the simulated use case.

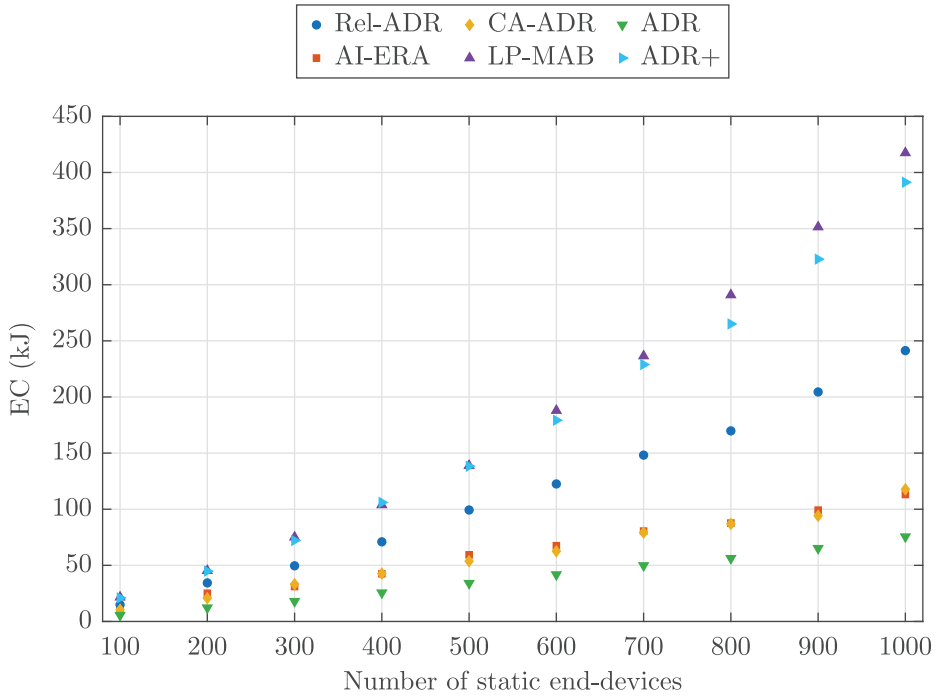

Fig. 12. Static scenario — Energy consumption versus the number of static end-devices.

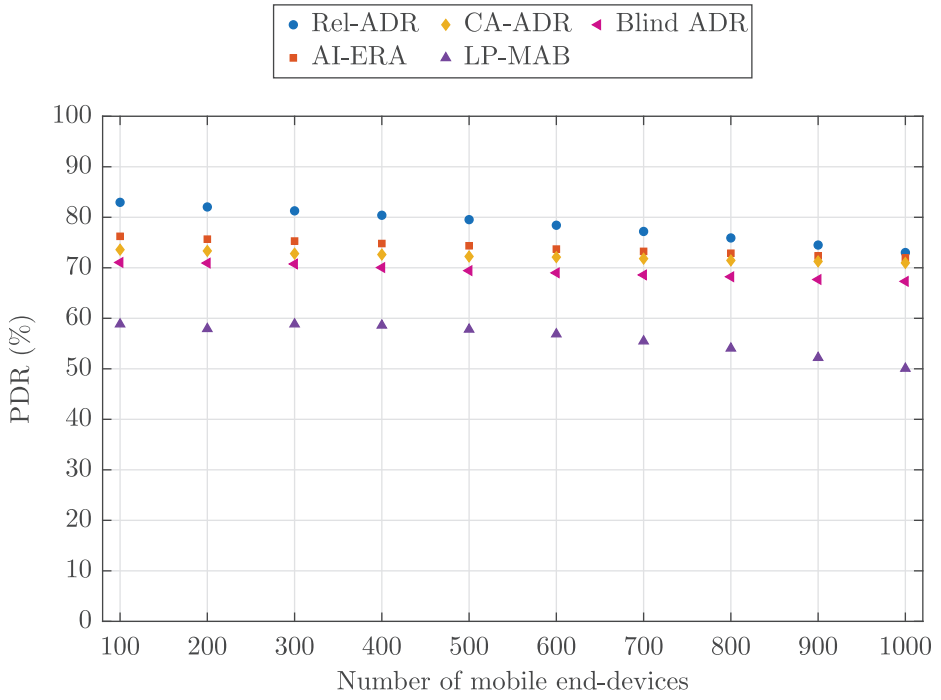

Fig. 13. Mobile scenario — Packet delivery ratio versus the number of mobile end-devices.

Packet delivery ratio. Fig. 13 presents the packet delivery ratio as a function of the number of mobile end-devices.

Fig. 13 shows a trend similar to the one obtained in the static scenario (Fig. 11). In fact, the higher the end-device density, the lower the packet delivery ratio. Rel-ADR outperforms the other approaches also in the case of mobile end-devices, maintaining a higher packet delivery ratio in dense and dynamic networks.

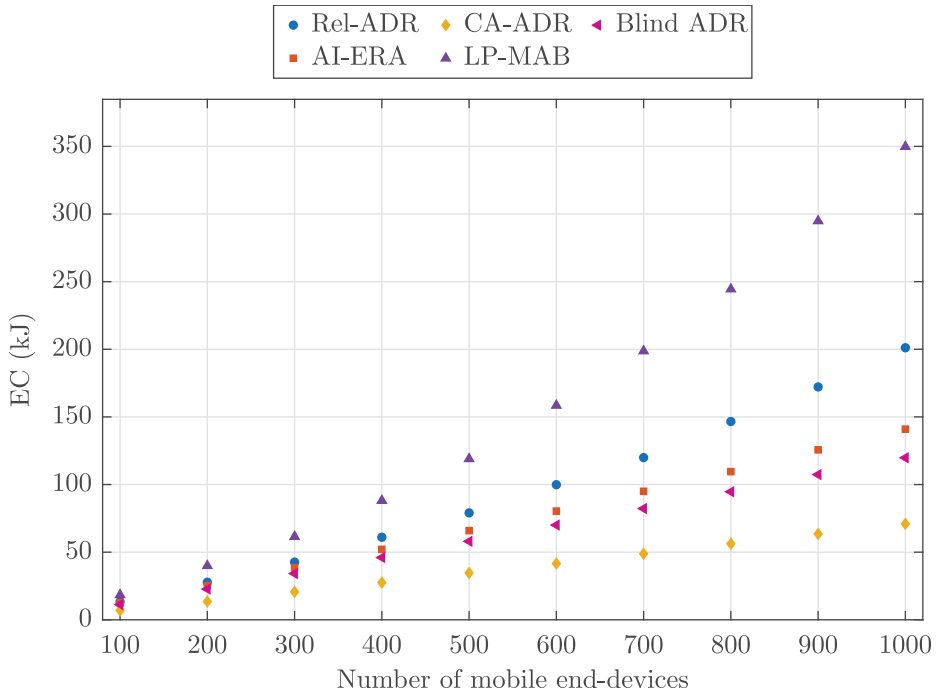

Fig. 14. Mobile scenario — Energy consumption versus the number of mobile end-devices.

Blind ADR does not react to link quality variations. For example, with 1000 mobile end-devices, Blind ADR achieves a packet delivery ratio of 67.31%, while Rel-ADR improves this result by 8.4%. This result confirms that while Blind ADR is a lightweight option in mobile and dense deployments it does not provide the adaptability and reliability of learning-based approaches such as Rel-ADR.

LP-MAB shows lower packet delivery ratio values than Rel-ADR due to the exploration phases, during which LP-MAB tests different parameter combinations. This approach does not work well with mobile end-devices as their varying physical position may influence learning and, consequently, the obtained parameter combinations. In fact, while exploring the parameter space, the end-devices also move around in the sensing area, so the values used by LP-MAB in the learning phase are bound to different physical locations. For instance, with 1000 mobile end-devices, Rel-ADR reaches a packet delivery ratio of 73.02%, while LP-MAB obtains 50.06%, which corresponds to a relative improvement of 45.86% in favor of Rel-ADR.

As far as the comparison with the supervised learning methods is concerned, with 100 mobile end-devices, Rel-ADR reaches a packet delivery ratio of 82.96%, while AI-ERA and CA-ADR achieve a packet delivery ratio of 76.23% and 73.59%, respectively. As a result, Rel-ADR shows a packet delivery ratio improvement of 8.83% and 12.74%, respectively. In the configuration with the highest density, i.e., the one with 1000 mobile end-devices, Rel-ADR obtains a packet delivery ratio of 73.02% that confirms its robustness under high traffic even in mobile scenarios. For the sake of comparison, AI-ERA and CA-ADR achieve a packet delivery ratio of 71.97% and 70.96%, respectively, thus the relative improvements of Rel-ADR are 1.78% and 3.22%, respectively.

This consistent advantage across different network densities highlights the ability of the Rel-ADR DNN model to generalize well and adapt to dynamic environments without relying on GPS data or device-side computation.

Energy consumption. Fig. 14 shows the energy consumption of all evaluated approaches as the number of mobile end-devices increases.

As shown in Fig. 14, in all the assessed approaches, the energy consumption increases when the density of end-devices grows. The obtained results are similar to the ones obtained in the static scenario.

Rel-ADR shows a reduced energy consumption compared with LP-MAB. In the configuration with 100 mobile end-devices, Rel-ADR consumes 26.97% less energy than LP-MAB. These savings are achieved thanks to the Rel-ADR ability to select efficient transmission parameters that minimize the Time on Air and transmission power, even in dynamic conditions. With 1000 mobile end-devices, the better energy consumption result of Rel-ADR over LP-MAB becomes more evident. In fact, Rel-ADR consumes 42.51% less energy than LP-MAB.

Blind ADR consumes less energy than Rel-ADR, with savings of about 18.8% with 100 end-devices and 40.45% with 1000 end-devices. However, this energy efficiency comes at the cost of lower reliability, as Blind ADR cannot adapt to varying channel conditions.

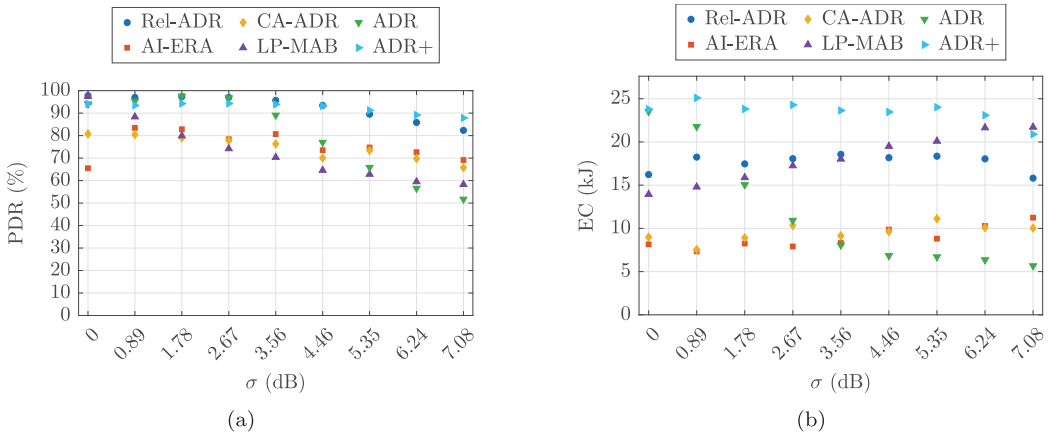

Fig. 15. Packet delivery ratio (a) and energy consumption (b) versus different values of channel shadowing (σ).

Compared to AI-ERA, Rel-ADR consumes slightly more energy in the 100 end-device scenario (about 6.71% more), and approximately 42.68% more energy than AI-ERA in the 1000 end-devices. This suggests that Rel-ADR uses higher values of spreading factor and transmission power that allow it to obtain a higher packet delivery ratio as shown in Fig. 13.

CA-ADR achieves the lowest energy consumption overall. However, this comes at the cost of a reduced packet delivery ratio (Fig. 13) and the need for GPS and end-device-side inference.

Likewise Figs. 10 and 11, Figs. 12 and 13 also confirm that Rel-ADR achieves a strong balance between energy consumption and communication reliability, placing a greater emphasis on reliability. This feature makes Rel-ADR a suitable solution for mobile IoT deployments where both reliability and energy consumption are critical.

8.3. Impact of shadowing variability on performance

This section analyzes the impact of channel shadowing on the performance of the considered approaches. The variability of the wireless channel is modeled through the standard deviation σ of the log-normal shadowing component in the path loss model (Eq. (2)). By varying σ, it is possible to emulate different levels of environmental unpredictability and assess the robustness of the considered approaches under increasingly challenging propagation conditions. In this scenario, the channel noise, represented by the standard deviation σ of the shadowing component, takes values in the set {0, 0.89, 1.78, 2.67, 3.56, 4.46, 5.36, 6.24, 7.08} as in [29], thus allowing to evaluate the performance of the Rel-ADR DNN model under increasing levels of channel variability. To maintain a controlled environment by eliminating the mobility effects for a more precise analysis of the noise impact on the performance metrics, 100 static end-devices are deployed in the network.

Fig. 15 shows the packet delivery ratio and energy consumption obtained by the considered approaches under varying channel conditions.

The results in Fig. 15(a) show that Rel-ADR achieves the highest packet delivery ratio compared to other approaches, even in scenarios not included in the training set, suggesting a good generalization capability of the Rel-ADR DNN model. AI-ERA and CA-ADR obtain lower packet delivery ratio values with respect to Rel-ADR. These results are consistent with the findings from previous scenarios and the accuracy values reported in Table 4. The PDR obtained by ADR is similar to the one obtained by Rel-ADR with low value of sigma, but it degrades when sigma increases as transmission parameters more robust than the one suggested by ADR are required. Conversely, ADR+ imposes the use of transmission parameters more robust than the one used by ADR. This enables ADR+ to obtain high PDRs with all the considered sigma values as the considered scenario includes 100 end-devices (see Fig. 11).

Fig. 15(b) shows the energy consumption of the considered approaches as the channel conditions vary. As expected, the energy consumption increases with σ due to the need for more robust transmission parameters (i.e., higher values of spreading factor and transmission power) to cope with worsening signal conditions.

Despite this, Rel-ADR maintains energy consumption levels comparable to those of LP-MAB and ADR+ and slightly higher than those of AI-ERA and CA-ADR, while significantly outperforming ADR. It is important to note that the Rel-ADR, AI-ERA and CA-ADR DNN models were trained on data with $\sigma = 7.8$ dB and were not retrained with all the different sigma values here considered. Instead, throughout the evaluation tests with all the sigma values, the same DNN model weights were used for each inference phase, thus allowing the evaluation of the ability of the DNN models to generalize to "unseen" channel conditions, i.e., channel conditions that were not present in the training set. These findings confirm the robustness and ability of the Rel-ADR DNN model to generalize across different conditions.

9. Conclusions and future work

This paper proposed Rel-ADR, a deep learning-based adaptive data rate approach for LoRaWAN networks that changes both the spreading factor and transmission power in parallel, thus pursuing high packet delivery ratio and low power consumption at the same time. In particular, Rel-ADR considers both energy efficiency and communication reliability, with a stronger emphasis on improving reliability, as reliability is more important than energy efficiency for several IoT applications. However, this can be a limitation of Rel-ADR for some specific applications that may have stricter energy constraints. To cope with this limitation, Rel-ADR can be adapted to support an adaptive parameter tuning that considers per-node reliability and energy consumption requirements. This aspect will be investigated in future work.

Rel-ADR runs on the network server, thus avoiding any computational overhead in the LoRaWAN end-devices that would increase their power consumption. Moreover, Rel-ADR does not use either the confirmed mode, which would introduce additional traffic overhead, or the end-devices position, thus avoiding the need for costly and energy-greedy localization technologies (e.g., GPS).

The simulation results here presented are promising and confirm that Rel-ADR represents a good tradeoff between communication reliability and energy consumption, albeit with a better preference for reliability. In particular, in all the considered scenarios (both static and mobile) Rel-ADR always obtained PDR values higher than 67% while considering a number of end-devices up to 1000.

Moreover, the outcomes of the assessment demonstrate the robustness of the Rel-ADR DNN model and its ability to generalize across different conditions. In fact, in the scenario in which the impact of the shadowing was assessed, the PDR achieved by Rel-ADR is always higher than 82%.

Future work will address a methodology for the dynamic configuration of the WindowSize parameter of Rel-ADR for each end-device in the LoRaWAN network. Moreover, the implementation of Rel-ADR on COTS devices and its experimental evaluations will be tackled.

CRediT authorship contribution statement

Luca Leonardi: Writing – original draft, Methodology, Investigation, Conceptualization. **Giancarlo Iannizzotto:** Writing – review & editing, Software, Methodology, Conceptualization. **Mattia Pirri:** Writing – original draft, Software, Investigation, Conceptualization. **Gaetano Patti:** Writing – review & editing, Methodology, Conceptualization. **Alessio Pirri:** Writing – review & editing, Investigation, Conceptualization. **Lucia Lo Bello:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The work of Luca Leonardi is funded by the European Union (NextGenerationEU), through the MUR-PNRR project SAMOTHRACE (ECS00000022).

Data availability

No data was used for the research described in the article.

References

[1] E. Sisinni, A. Mahmood, Wireless communications for industrial Internet of Things: The LPWAN solutions, in: Wireless Networks and Industrial IoT, Springer, 2021, pp. 79–103.
[2] L. Leonardi, L. Lo Bello, G. Patti, Lora support for long-range real-time inter-cluster communications over bluetooth low energy industrial networks, Comput. Commun. 192 (2022) 57–65, http://dx.doi.org/10.1016/j.comcom.2022.05.026.
[3] LoRa Alliance Technical Committee, LoRaWAN™ 1.0.4 Specification, LoRa Alliance, 2020.
[4] M. Jouhari, N. Saeed, M.-S. Alouini, E.M. Amhoud, A survey on scalable LoRaWAN for massive IoT: Recent advances, potentials, and challenges, IEEE Commun. Surv. & Tutorials 25 (3) (2023) 1841–1876.
[5] M. Alipio, M. Bures, Current testing and performance evaluation methodologies of LoRa and LoRaWAN in IoT applications: Classification, issues, and future directives, Internet Things (2023) 101053.
[6] L. Leonardi, L. Lo Bello, G. Patti, MRT-LoRa: A multi-hop real-time communication protocol for industrial IoT applications over LoRa networks, Comput. Commun. 199 (2022) 72–86, http://dx.doi.org/10.1016/j.comcom.2022.12.013.
[7] M. Bor, U. Roedig, LoRa transmission parameter selection, in: 2017 13th International Conference on Distributed Computing in Sensor Systems, DCOSS, IEEE, 2017, pp. 27–34.
[8] Semtech Corporation, SX1272/73 860 MHz to 1020 MHz low power long range transceiver, 2017, SX1272/73 Datasheet.
[9] K. Mikhaylov, J. Petaejaejaervi, T. Haenninen, Analysis of capacity and scalability of the LoRa low power wide area network technology, in: European Wireless; 22th European Wireless Conference, 2016, pp. 1–6.
[10] S. Corporation, LoRaWAN—Simple Rate Adaptation Recommended Algorithm, Semtech Camarillo, CA, USA, 2016.
[11] Blind ADR, 2025, URL https://www.semtech.com/company/video/technology-in-depth/blind-adr. (Online; Accessed 02 January 2025).
[12] LoRa Alliance Technical Committee Regional Parameters Workgroup, RP002-1.0.4 LoRaWAN Regional Parameters, LoRa Alliance, 2022.

[13] M. Slabicki, G. Premsankar, M. Di Francesco, Adaptive configuration of LoRa networks for dense IoT deployments, in: NOMS 2018-2018 IEEE/IFIP Network Operations and Management Symposium, IEEE, 2018, pp. 1–9.
[14] M. Ali Lodhi, L. Wang, K. Mahmood, A. Farhad, J. Chen, S. Kumari, Enhanced adaptive data rate strategies for energy-efficient internet of things communication in LoRaWAN, Int. J. Commun. Syst. (2024) e5966.
[15] A. Farhad, J.-Y. Pyun, LoRaWAN meets ML: A survey on enhancing performance with machine learning, Sensors 23 (15) (2023) 6851.
[16] A. Farhad, D.-H. Kim, J.-S. Yoon, J.-Y. Pyun, Deep learning-based channel adaptive resource allocation in LoRaWAN, in: 2022 International Conference on Electronics, Information, and Communication, ICEIC, 2022, pp. 1–5, http://dx.doi.org/10.1109/ICEIC54506.2022.9748580.
[17] A. Farhad, J.-Y. Pyun, AI-ERA: Artificial intelligence-empowered resource allocation for LoRa-enabled IoT applications, IEEE Trans. Ind. Informatics 19 (12) (2023) 11640–11652, http://dx.doi.org/10.1109/TII.2023.3248074.
[18] M.A. Lodhi, L. Wang, A. Farhad, K.I. Qureshi, J. Chen, K. Mahmood, A.K. Das, A contextual aware enhanced LoRaWAN adaptive data rate for mobile IoT applications, Comput. Commun. 232 (2025) 108042.
[19] LiteRT for Microcontrollers | Google AI Edge | Google AI for Developers, 2024, URL https://ai.google.dev/edge/litert/microcontrollers/overview. (Online; accessed 24 December 2024).
[20] D.-H. Kim, J.-Y. Pyun, AI-driven adaptive data rate for LoRaWAN location-based services, IEEE Access (2024).
[21] M.A. Lodhi, X. Sun, K. Mahmood, A. Lodhi, Y. Park, M. Hussain, AI-enhanced resource allocation for LPWAN-based LoRaWAN: A hybrid TinyML and deep learning approach, IEEE Internet Things J. (2025).
[22] S.I.A. Elkarim, M. Elsherbini, O. Mohammed, W.U. Khan, O. Waqar, B.M. ElHalawany, Deep learning based joint collision detection and spreading factor allocation in LoRaWAN, in: 2022 IEEE 42nd International Conference on Distributed Computing Systems Workshops, ICDCSW, 2022, pp. 187–192, http://dx.doi.org/10.1109/ICDCSW56584.2022.00043.
[23] M. González-Palacio, D. Tobón-Vallejo, L.M. Sepúlveda-Cano, M. Mauricio, C. Röehrig, L. Bao Le, Machine-learning-assisted transmission power control for LoRaWAN considering environments with high signal- to -noise variation, IEEE Access 12 (2024) 54449–54470, http://dx.doi.org/10.1109/ACCESS.2024.3387457.
[24] M. González-Palacio, D. Tobón-Vallejo, L.M. Sepúlveda-Cano, S. Rúa, G. Pau, L.B. Le, LoRaWAN path loss measurements in an urban scenario including environmental effects, Data 8 (1) (2022) 4.
[25] S.U. Minhaj, A. Mahmood, S.F. Abedin, S.A. Hassan, M.T. Bhatti, S.H. Ali, M. Gidlund, Intelligent resource allocation in LoRaWAN using machine learning techniques, IEEE Access 11 (2023) 10092–10106, http://dx.doi.org/10.1109/ACCESS.2023.3240308.
[26] N. Aihara, K. Adachi, O. Takyu, M. Ohta, T. Fujii, Q-learning aided resource allocation and environment recognition in LoRaWAN with CSMA/CA, IEEE Access 7 (2019) 152126–152137, http://dx.doi.org/10.1109/ACCESS.2019.2948111.
[27] I. Ilahi, M. Usama, M.O. Farooq, M.U. Janjua, J. Qadir, LoRaDRL: Deep reinforcement learning based adaptive PHY layer transmission parameters selection for LoRaWAN, in: 2020 IEEE 45th Conference on Local Computer Networks, LCN, 2020, pp. 457–460.
[28] G. Zhao, K. Lin, D. Chapman, N. Metje, T. Hao, Optimizing energy efficiency of LoRaWAN-based wireless underground sensor networks: A multi-agent reinforcement learning approach, Internet Things 22 (2023) 100776.
[29] B. Teymuri, R. Serati, N.A. Anagnostopoulos, M. Rasti, LP-MAB: Improving the energy efficiency of LoRaWAN using a reinforcement-learning-based adaptive configuration algorithm, Sensors 23 (4) (2023) 2363.
[30] F.Z. Mardi, Y. Hadjadj-Aoul, M. Bagaa, N. Benamar, Resource allocation for LoRaWAN network slicing: Multi-armed bandit-based approaches, Internet Things 26 (2024) 101195.
[31] H. Borchani, G. Varando, C. Bielza, P. Larranaga, A survey on multi-output regression, Wiley Interdiscip. Rev.: Data Min. Knowledge Discov. 5 (5) (2015) 216–233.
[32] D. Xu, Y. Shi, I.W. Tsang, Y.-S. Ong, C. Gong, X. Shen, A survey on multi-output learning, 2019, arXiv:1901.00248.
[33] T. Szandała, Review and comparison of commonly used activation functions for deep neural networks, in: Bio-Inspired Neurocomputing, Springer, 2020, pp. 203–224.
[34] M. Rana, M.M. Uddin, M.M. Hoque, Effects of activation functions and optimizers on stock price prediction using LSTM recurrent networks, in: Proceedings of the 2019 3rd International Conference on Computer Science and Artificial Intelligence, CSAI '19, Association for Computing Machinery, New York, NY, USA, 2020, pp. 354–358, http://dx.doi.org/10.1145/3374587.3374622.
[35] J. Petajajarvi, K. Mikhaylov, A. Roivainen, T. Hanninen, M. Pettissalo, On the coverage of LPWANs: Range evaluation and channel attenuation model for LoRa technology, in: 2015 14th International Conference on ITS Telecommunications, ITST, IEEE, 2015, pp. 55–59.
[36] M.C. Bor, U. Roedig, T. Voigt, J.M. Alonso, Do LoRa low-power wide-area networks scale? in: Proceedings of the 19th ACM International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2016, pp. 59–67.
[37] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.
[38] ETSI, Short Range Devices (SRD) Operating in the Frequency Range 25 MHz to 1 000 MHz; Part 2: Harmonised Standard for Access to Radio Spectrum for Non Specific Radio Equipment, ETSI, 2018.
[39] L. Leonardi, L. Lo Bello, F. Battaglia, G. Patti, Comparative assessment of the LoRaWAN medium access control protocols for IoT: Does listen before talk perform better than ALOHA? Electronics 9 (4) (2020) http://dx.doi.org/10.3390/electronics9040553.
[40] D. Croce, M. Gucciardo, S. Mangione, G. Santaromita, I. Tinnirello, Impact of LoRa imperfect orthogonality: Analysis of link-level performance, IEEE Commun. Lett. 22 (4) (2018) 796–799.
[41] G.G.M. de Jesus, R.D. Souza, C. Montez, A. Hoeller, Lorawan adaptive data rate with flexible link margin, IEEE Internet Things J. 8 (7) (2021) 6053–6061, http://dx.doi.org/10.1109/JIOT.2020.3033797.
[42] S. Thepphaeng, C. Pirak, A study of LoRaWAN network performance for advanced metering infrastructure (AMI) applications, in: 2025 22nd International Conference on Electrical Engineering/Electronics, Computer, Telecommunications and Information Technology, ECTI-CON, IEEE, 2025, pp. 1–4.
[43] C. Bettstetter, Smooth is better than sharp: A random mobility model for simulation of wireless networks, in: Proceedings of the 4th ACM International Workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, 2001, pp. 19–27.
[44] C. Bettstetter, H. Hartenstein, X. Pérez-Costa, Stochastic properties of the random waypoint mobility model, Wirel. Netw. 10 (5) (2004) 555–567.
[45] E.M. Royer, P.M. Melliar-Smith, L.E. Moser, An analysis of the optimum node density for ad hoc mobile networks, in: ICC 2001. IEEE International Conference on Communications. Conference Record (Cat. No. 01CH37240), Vol. 3, IEEE, 2001, pp. 857–861.
[46] L. Acosta-Garcia, J. Aznar-Poveda, A.J. Garcia-Sanchez, J. Hollenstein, J. Garcia-Haro, T. Fahringer, ADRL: A reconfigurable energy-efficient transmission policy for mobile LoRa devices based on reinforcement learning, Internet Things 31 (2025) 101577, http://dx.doi.org/10.1016/j.iot.2025.101577, URL https://www.sciencedirect.com/science/article/pii/S2542660525000903.