Deep reinforcement learning for automatic run-time adaptation of UWB PHY radio settings

Dieter Coppens, Adnan Shahid, *Senior member , IEEE*, Eli De Poorter

***Abstract*—Ultra-wideband technology has become increasingly popular for indoor localization and location-based services. This has led recent advances to be focused on reducing the ranging errors, whilst research focusing on enabling more reliable and energy efficient communication has been largely unexplored. The IEEE 802.15.4 UWB physical layer allows for several settings to be selected that influence the energy consumption, range and reliability. Combined with the available link state diagnostics reported by UWB devices, there is an opportunity to dynamically select PHY settings based on the environment. To address this, we propose a deep Q-learning approach for enabling reliable UWB communication, maximizing packet reception rate (PRR) and minimizing energy consumption. Deep Q-learning is a good fit for this problem, as it is an inherently adaptive algorithm that responds to the environment. Validation in a realistic office environment showed that the algorithm outperforms traditional Q-learning, linear search and using fixed hard-coded UWB PHY settings. We found that deep Q-learning achieves a higher average PRR and also reduces the ranging error, as a side effect, while using only 14% of the energy compared to a fixed hard-coded UWB PHY setting in a dynamic office environment.**

***Index Terms*—UWB, localization, deep reinforcement learning**

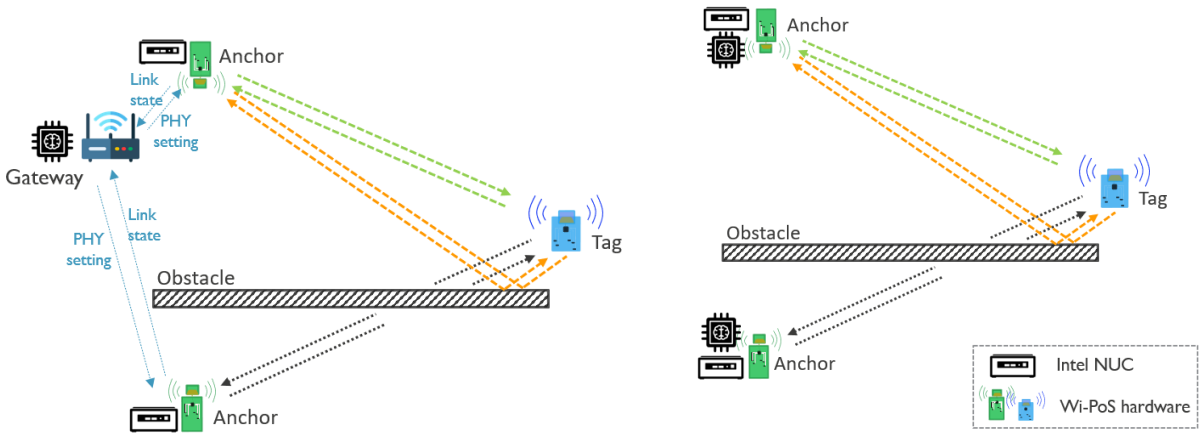
I. Introduction

ULTRA-WIDEBAND (UWB) technology has recently attracted the attention of not only the research community, but also the public. This is due to the addition of the technology to several smartphones and other consumer products [1]. UWB uses a much wider bandwidth i.e., more than 500 MHz, compared to traditional narrowband techniques such as Narrowband Internet of things (NB-IoT), Sigfox, LoRa, etc. This has benefits such as higher robustness to multipath effects, high temporal resolution and high channel capacity [2]. The high temporal resolution allows UWB to be used for precision ranging and enables applications like hands-free access-control and indoor navigation. While UWB is more robust to multipath effects (compared to narrowband techniques), the performance is still highly dependent on the environment. Having Line-of-sight (LOS) or Non-line-of-sight (NLOS) conditions and destructive interference influences the calculated ranges and reliability dramatically [3]. This has led to a large quantity of work to develop complex algorithms in order to maximize the ranging accuracy of UWB in these situations. Several approaches for this have been explored, such as auto-encoders [4], probabilistic learning [5] and path detection algorithms [6].

D. Coppens, A. Shahid, and E. De Poorter are with IDLab, Department of Information Technology, Ghent University—imec, 9052 Ghent, Belgium (e-mail: dieter.coppens@ugent.be; eli.depoorter@ugent.be; adnan.shahid@ugent.be

The IEEE 802.15.4 UWB physical layer (PHY) standard [7] contains several settings that can be selected: for example, channel, data rate (DR) and preamble symbol repetitions (PSR). These settings have shown to be highly influential on the radio sensitivity and energy consumption [8] and could enable more reliable and energy efficient UWB communication. However, it is challenging to find a reliable setting when a poor link is detected because it is not as straightforward as changing to a more energy consuming setting when a poor link is detected, as multipath reflections could require a channel (frequency) change. Currently, hard-coded UWB PHY settings are used in most scientific papers and practical UWB systems [8]. Hard-coded UWB PHY settings are never changed after deployment, they are chosen once based on the need of the user: a long range, high update rate, energy efficiency, etc. This means that they are not dynamically adapted with respect to a change in the environment, which makes them ineffective at sustaining a high reception rate [9] in dynamic environments. A second factor is that UWB radio devices report numerous link quality measures for each received packet, such as the received (RX) power, first path (FP) power, channel impulse response (CIR) and so on [10]. This information could be used to characterize the environment and aid in selecting the optimal PHY settings. This gives the opportunity to dynamically select PHY settings based on the environment. By doing this separately on different UWB links, we can enable a decentralized solution for adaptive UWB communication in complex, dynamic environments without the need for a central gateway to handle all communication. This dynamic link adaption is still mostly unexplored for UWB communication in scientific literature, certainly compared to accuracy improvement algorithms.

Since the aim is to solve the problem in a decentralized way, i.e., directly on anchors, reinforcement learning (RL) is well suited as it can reach an optimal solution for a control problem by interaction with the environment. A RL agent interacts with the environment in discrete time steps. At each time, it takes an action and then receives feedback for that action in the form of a reward. The agent learns by rewarding desired behavior and punishing undesired ones. In RL, the environment can be formulated as a Markov Decision Process (MDP), the value of the current state should suffice to determine transition probabilities and rewards following an action selection. Q-learning [11] is a commonly used RL algorithm that has already been used for several cognitive communication applications [12], [13], [14], [15], [16]. However, Q-learning is only suitable for problems with a low-dimensional state space because the learned



(a) Centralized architecture

(b) Proposed decentralized architecture

Fig. 1: An illustrative picture of a complex indoor environment causing multipath and attenuated communication with (a) a centralized solution running on the gateway and (b) our proposed decentralized solution running on the Intel NUC of each anchor.

policy is stored in a table. To mitigate this, the table can be replaced by a neural network that takes the state as input and approximates the Q-table values. This method is called deep Q-learning, and it has also shown its potential in cognitive communication applications [17], [18], [19].

The main contributions of the paper are the following:

- Identifying the UWB link state information parameters that most impact the packet reception ratio (PRR) and energy consumption
- Designing a decentralized Deep Q-learning architecture for automatically configuring UWB PHY radio settings that adapt to different channel conditions.
- Comparing the performance, algorithmic complexity, and time complexity for optimizing UWB PHY radio settings using linear search, Q-learning and deep Q-learning in static and dynamic environments.
- Making publicly available a UWB dataset in a realistic environment using 72 different UWB PHY settings for other researchers to design alternative solutions or further improvements. [1].

The remainder of this paper is organized as follows. Section II discusses the related work for both UWB PHY adaptation and deep Q-learning. In section III, the UWB communication problem and the system model are described. Section IV describes the proposed deep Q-learning and Q-learning algorithms. Section, V discusses the training of the RL agents. Next, sections VI-A and VI-B describe the environment in which the dataset is gathered and how the measurements are performed. Section VI-D discusses the performance of the developed algorithms in a static and dynamic environments. This is followed by the conclusion in Section VII

[1]The dataset will be made available together with the final version of the paper, and is currently available on request. https://github.com/dietercoppens/UWB-DRL-PHY-Runtime-Adaptation-dataset

II. Related Work

The related work is divided into a) adaptive UWB communication and b) Q-learning and Deep Q-learning for wireless communication.

A. Adaptive UWB communication

Although numerous papers aim to optimize the ranging accuracy by using advanced algorithms [5], [6], [4], e.g. by performing error correction or detecting LOS or NLOS, these solutions typically do not consider link reliability or energy consumption aspects. This section provides an overview of recent papers that investigate the influence UWB PHY settings on these metrics.

The authors of [8] present a study of the influence of different PHY settings on the reliability and energy efficiency of UWB communication. For example, increasing the pulse repetition frequency (PRF) provides a slight improvement in reliability, but it reduces the energy efficiency and increasing the PSR provides a greater increase in reliability, but it also has a worse effect on the energy efficiency. This study shows that by adjusting these settings, the radio sensitivity can be increased, but this generally comes at the cost of energy efficiency. Next, they estimate the link quality and try to extract characteristics of the surrounding environment from this. Finally, a scheme is designed to adapt the PHY setting based on the estimated link quality. The only used link state metric is CIR, and the whole logic is based on experimental data in only LOS conditions, which is not realistic for practical applications where NLOS conditions cannot be ignored. For characterization of LOS/NLOS situations to adapt the PHY setting, the authors refer to other papers using machine learning techniques. In [20], the energy consumption of UWB communication is analyzed based on the PHY layer setting. The authors show that more energy is consumed for lower data rates due to the higher number of pulses that need to be transmitted, which causes the system to be turned on longer. [21] focuses on the influence of the UWB PHY settings on the

ranging accuracy when using asynchronous two-way ranging. The results were obtained using an extensive measurement campaign where more than 200 UWB PHY settings were tested. The used radio channel, DR and PRF were found to strongly influence the ranging accuracy, but no algorithm was provided to dynamically modify the setting based on the environment. The authors of [22] present a method to give a reliability indication to an UWB distance measurement. For this, diagnostic information available on Qorvo's DW1000 UWB radio chip is used. The diagnostic information, such as CIR and leading edge detection (LDE)—threshold, are processed together to determine the estimated quality of the ranging measurement. In [23], a framework to make UWB more robust is proposed. The framework uses the distance between the first path and the ambient noise to determine low quality ranging. If low quality ranging is detected, a linear search algorithm that tries out all UWB settings within certain data rate, energy consumption, error rates, and robustness requirements. The best performing setting is then selected. This approach has some drawbacks, the UWB settings are only changed when a certain quality threshold is passed, which means that the performance is not continuously evaluated and optimized and the linear search algorithm means that every possible UWB PHY settings needs to be tried out first before the best performing one is found and configured.

The authors of these papers discuss the influence of the PHY settings on the reliability and energy consumption and how to give a quality indication to a range. However, they do not elaborate on how to use the link state information and link diagnostics to dynamically improve the UWB reliability and energy consumption of UWB communication. This shows that there is a clear research gap in dynamic UWB settings selection, and that this research defines and tries to solve a novel research problem. We will address these limitations by developing a RL algorithm that dynamically selects the best performing setting without a) the need to try out every possible setting and b) requiring additional knowledge of the link state parameters. The best performing setting is continuously chosen by the algorithm.

B. Q-learning and Deep Q-learning for wireless communication

Q-learning and Deep Q-learning have been used for several resource management wireless communication problems. The authors of [17] proposed Deep Q-learning for a resource allocation mechanism for V2V communications. In [19] it is used for downlink power allocation in multicell systems and the authors of [18] proposed Deep Q-learning for circumstance-independent resource allocation with efficient scheduling and power allocation. These papers show that RL, and in particular Q- and deep Q-learning, can be successful in wireless communication resource management. However, it has not yet been used for UWB communication, as it requires a different system and algorithm design and the proposed algorithms in wireless communication resource management problems cannot be directly applied for solving the adaptation of UWB PHY settings problem. Which means we are the first to define actions, states, rewards, and features to use in reinforcement learning for this problem. To summarize, there is a clear research gap in (1) developing a decentralized algorithm that adapts the UWB PHY setting in runtime using deep Q-learning while taking into account both reliability (PRR) and energy consumption and (2) evaluating the algorithm on a realistic dataset containing both LOS and NLOS situations.

III. Problem and system description

Figure 6 illustrates environmental challenges that cause problems for UWB communication in indoor environments. Obstacles in the environment can cause attenuated NLOS paths between the devices, and reflections can cause multipath and interference. These effects have a major impact on both the reliability and ranging accuracy.

A. UWB link state information

UWB devices, such as the Qorvo DW1000 UWB radio chip used in this research, report several diagnostic link state parameters that can be used to characterize the environment and determine the state of the link. Table I gives an overview of the link state parameters that are available. All diagnostics can be found in the user manual [24] as well. Using these parameters, estimates can be made of link quality [8], [22] and the presence of LOS/NLOS [25], [10].

SYNC	SFD	PHR	PHY payload

Fig. 2: Structure of a UWB PHY frame in the IEEE 802.15.4 standard [7]. The Synchronization (SYNC) field is used to synchronize the sender and receiver. The start-of-frame (SFD) delimiter indicates the end of the SYNC field and start of PHY header (PHR) which contains information about the payload and how it is transmitted to the receiver.

B. UWB PHY settings

In UWB communication, several PHY settings can be configured that have an influence on the sensitivity and energy consumption. Important to mention is that in this section, we focus on the settings that are supported by the Qorvo DW1000 chip that is used to gather the dataset.

- **Channel**: The IEEE 802.15.4 UWB PHY defines 16 different channels. A channel is a combination of center frequency and bandwidth. Changing the center frequency and bandwidth influences the perceived noise and can be an important parameter to increase the robustness.
- **PSR**: The UWB PHY packet (shown in Figure 2) starts with a synchronization header containing a preamble (used for frame synchronization and detection) and a start-of-frame delimiter (SFD) that indicates the end of the preamble and the time of arrival used for ranging. The length of the preamble can be changed by changing the amount of repetitions of the preamble symbol that are sent. A higher PSR increases the robustness because there is a higher chance of reception, but it increases the

TABLE I: List of DW1000 provided link state information parameters

Features (Abbreviation)	Description
CIR	The array of accumulated complex Channel Impulse Response (CIR)
F_1	The amplitude of the first harmonic of the first path (FP) signal
F_2	The amplitude of the second harmonic of the FP signal
F_3	The amplitude of the third harmonic of the FP signal
CIR_power	The CIR power
N_c	The standard deviation of the noise reported in the CIR accumulator
RX_{pacc}	The preamble accumulation count at the receiver
Pre_{ratio}	The ratio between the accumulated preamble symbols and total transmitted preamble symbols $Pre_{ratio} = RX_{pacc}/Preamble\_length$ Comparing the RX_{pacc} with $Preamble\_length$ gives additional indication of the preamble accumulation conditions
$fpindex$	The index of the detected FP
FP_p	The estimated FP power level in dBm and can be calculated as FP power level$= 10\log_{10}\left(\frac{F_1^2+F_2^2+F_3^2}{N^2}\right) - A$ where A is 113.77 for a PRF of $16MHz$ and 121.74 for a PRF of $64MHz$, N represents preamble accumulation count
RX_p	The estimated RX power level in dBm and can be calculated as RX power level $= 10\log_{10}\left(\frac{CIR_{power}\times 2^{17}}{N^2}\right) - A$
$NLOS$	The power difference between the RX power and FP power and can be calculated as NLOS $=$ RX power level $-$ FP power level Can be used as an indicator for NLOS. If greater than 10 dB, likely to be in NLOS
$\hat{d}$	The estimated range and can be calculated as $\hat{d} = c \times \tau$ where c represents the speed of light in m/s and τ the signal propagation time from tag to anchor.
LDE	Leading edge detection (LDE) threshold used to find the first path, based on an estimate of the noise.
PP	Amplitude of the peak path (PP) in the CIR.
PP_index	Index of the PP in the CIR.
Q_1	The ratio between F_2 and N_c $Q_1 = F_2/N_c$ Comparing the noise with F_2 can give additional indication of the quality of the FP measurement.
Q_2	The ratio between F_2 and $LDE\_threshold$ $Q_2 = F_2/LDE\_threshold$ Comparing the $LDE\_threshold$ with F_2 gives additional indication of the quality of the $LDE\_threshold$

radio's energy expenditure due to the longer length. The PSR is the only configurable setting of the UWB PHY frame shown in 2 which we consider.

- **PRF**: In UWB communication, a bit is transmitted using a train of pulses. The speed at which these pulses are sent is a configurable setting. By increasing the PRF, the amount of pulses sent per unit of time increases. This means that there are more threshold decision events during, and thus a higher accuracy and robustness. This comes at the cost of higher energy consumption.
- **Data rate (DR)**: Lowering the DR increases the reliability of the communication, but causes a longer transmission time and thus higher energy consumption.
- **Transmit power gain** (P_{tx}): The transmit power gain can be adjusted, a higher power gain means reliable communication can be achieved for longer distance between the devices, but this also comes at the cost of higher energy consumption.

Summarizing the information from these settings, we can see that there is a trade-off between the reliability and energy consumption [8].

C. System description

1) Centralized vs Decentralized: Centralized and decentralized solutions each have their advantages and disadvantages. The centralized solution, illustrated in Figure 1a, collects the data from all anchors at the gateway. This information is used to improve the learned model and to determine the best UWB PHY setting to be used for all anchors. This means that it has access to more data (from all anchors) than the decentralized solution (only one anchor). However, this approach requires the data to be transmitted between the anchor and the gateway, with high latency as a clear drawback. This is especially the case when considering environments where there is a low capacity uplink and a large sending interval, for example, industrial environments that use LoRa. In addition, the centralized approach takes up an unnecessary high bandwidth in the local area network (LAN) and wide area network (WAN) which could become unfeasible in practical use cases. Most importantly, the centrally learned model is not tailored to a specific anchor or anchor environment. UWB radio performance depends heavily on environmental factors. A learned model, trained on data from one environment, may not generalize well to new environments. To compensate, the centralized system would require storing unique information for every possible link pair. The number of stored contexts could become enormous as more anchors and tags are introduced. In contrast, the decentralized solution, illustrated in Figure 1b, determines the best setting to be selected at the anchor itself using link state information available there

regardless. Leading to reduced data transmission and enabling more quick response times to changes in the environment. Additionally, the learned model is specific to each anchor, resulting in better performance. Based on the above comparison, we propose to develop a decentralized architecture because we think it is the better fitting choice for this problem and the centralized approach makes less sense. The developed system is thus represented by Figure 1b. Each anchor will run the developed RL algorithm on its Intel NUC and use its link state information, eliminating the need for link state data transmission between anchor and gateway as well as tag and anchor before setting selection can be performed. The PHY setting selected is communicated to the tag UWB device and finally configured on both.

2) Decentralized system: The list of notations used in the paper is given in Table II. We consider a link between two UWB devices, a tag and an anchor (as shown in Figure 1b) using the IEEE 802.15.4 UWB PHY standard implemented on the Wi-Pos, which is a Low-Cost UWB Hardware Platform with Long Range Sub-GHz Backbone [26]. The goal is to evaluate the link state and adjust the PHY settings (in real time) on the anchor to ensure that the UWB communication is always as good as possible, in terms of reliability (PRR) and energy consumption, even in dynamic and changing environments. This PHY setting is then communicated to the tag UWB device and finally configured on both. The PHY setting can be combined into a single settings variable A as shown in (1).

$$A = \{C, PSR, PRF, DR, P_{tx}\}, \tag{1}$$

where $C \in \{3, 5, 7\}$, $PSR \in \{128, 1024, 4096\}$, $PRF \in \{16, 64\}$, $DR \in \{110, 6800\}$ and $P_{tx} \in \{0, 10.5\}$. These values are determined by our collected dataset.

The environment is described by parameter e, this indicates the influence of the environment on the system. The parameter is adjusted when changes happen in the environment in which the UWB ranging is taking place. The ranging method used in the system is called Asymmetric Double Sided TWR (ADS-TWR) [27], which means that for one range estimation, there are three packets transmitted (TX) and received (RX). Combining this with the configured setting A, the energy consumption can be determined. Table III shows the current consumption (I) for the different settings during TX and RX for both the preamble and data parts in mA. The preamble includes the SYNC and SFD field and the data part consists of the PHR and PHY payload shown in Figure 2. Using this information and the constant supply voltage of 3.3 V for the DW1000 chip [28], the power P can be calculated using (2). Note that we use $P(A)$ which means the power with respect to the PHY setting A. This notation is used for other quantities as well.

$$P(A) = 3.3 \cdot I(A). \tag{2}$$

The number of symbols in the preamble $S_p(A)$ and data parts $S_d(A)$ can be calculated by (3) and (4), using the 12 data bytes (B_p), the number of bits in the PHR (b_p) the forward error correction rate ($R_{FEC} = 0.87$) and the SFD symbols (S_s) (64 for 110 kbps and 8 for other data rates).

TABLE II: List of notations and abbreviations used in the paper.

Notation/Abbreviation	Description
e	Environment
C	Channel
DR	Data rate
A	UWB PHY setting variable
TX	Transmit
RX	Receive
P	Power consumption
I	current consumption
S_p	The number of symbols in the complete preamble
S_s	The number of symbols in the SFD
S_d	The number of symbols in the data part of the frame
b_p	The number of bits in the PHR
B_p	The number of bytes in the payload
R_{FEC}	The forward error correction rate
T_{sym}	The duration of a symbol
T_p	The total duration of the preamble
T_d	The total duration of the data part
E_p	Energy consumption of the preamble
E_d	Energy consumption of the data part
E_{rx}	Energy consumption during transmission
E_{tx}	Energy consumption during reception
P_p	Power consumption during the preamble
P_d	Power consumption during the data part
E	The total energy consumption for a range
E_{min}	The minimum total energy consumption of the radio for a single range, determined over all possible PHY layer settings
E_{max}	The maximum total energy consumption of the radio for a single range, determined over all possible PHY layer settings
p_{rec}	The number of received packets
p_{total}	The total number of transmitted packets
G	The complete system with respect to A-th setting and e-th environment
s_t	State at time t
f	List of all available features
$Xcor$	The cross correlation between two vectors.
α	Learning rate in Bellman equation
γ	Discount factor in Bellman equation
F	F-value of a feature
R_t	The reward at time t
ϵ	The epsilon-greedy parameter
θ	Weights of main Deep Q-network
$\hat{\theta}$	Weights of target Deep Q-network
ζ	Importance sampling prioritization parameter
β	Importance sampling bias compensation parameter
p_i	Priority or TD-error of experience i
$Prob(i)$	Sampling probability of experience i
w_i	Importance sampling weight of experience i

$$S_p(A) = PSR + S_s(A). \tag{3}$$

$$S_d(A) = (b_p + \frac{B_p \cdot 8}{R_{FEC}}). \tag{4}$$

The duration of the preamble $T_p(A)$ and data $T_d(A)$ parts (in seconds) can then be calculated using (5) and (6) where the symbol duration (T_{sym}) is computed from Table IV.

$$T_p(A) = S_p(A) \cdot T_{sym}(A). \tag{5}$$

$$T_d(A) = S_d(A) \cdot T_{sym}(A). \tag{6}$$

Using the power and duration of the separate parts, we can calculate the energy consumption of the preamble $E_p(A)$ and data $E_d(A)$ parts as shown in (7) and (8).

$$E_p(A) = P_p(A) \cdot T_p(A). \tag{7}$$

$$E_d(A) = P_d(A) \cdot T_d(A). \tag{8}$$

TABLE III: Overview of I(A) for the different modes in the system [10]

Channel	PRF (MHz)	Data rate (Mbps)	Preamble TX current (mA)	Preamble RX current (mA)	Data TX current (mA)	Data RX current (mA)
3	16	0.11	68	113	35	59
3	16	6.8	68	113	50	118
3	64	0.11	83	113	40	72
3	64	6.8	83	113	52	118
5	16	0.11	74	118	42	62
5	16	6.8	74	118	57	123
5	64	0.11	89	118	46	75
5	64	6.8	89	118	59	123
7	16	0.11	74	118	42	62
7	16	6.8	74	118	57	123
7	64	0.11	95	124	52	81
7	64	6.8	95	124	65	129

TABLE IV: DW1000 symbol durations in the separate parts of the frame [10]

PRF (MHz)	Data Rate (Mbps)	T_{sym} SHR (ns)	T_{sym} Data & PHR (ns)
16	0.11	993.59	8205.13
16	6.8	993.59	1025.64
64	0.11	1017.63	8205.13
64	6.8	1017.63	128.12

The energy consumption is the combination of the data and preamble parts as shown in (9).

$$E(A) = E_p(A) + E_d(A). \tag{9}$$

Finally, the total energy consumption (in Joules) for a range using a certain setting is found using (10). Here, ADS-TWR is used, which means three packets are needed for one range estimate. Each packet requires one node to use the receiving energy consumption and one node to use the transmitting energy consumption. For the energy consumption during TX, an extra factor $10^{\frac{P_{tx}}{10}}$ is added to account for the P_{tx} gain in the setting.

$$E(A) = 3 \cdot (E_{rx}(A) + E_{tx}(A) \cdot 10^{\frac{P_{tx}}{10}}). \tag{10}$$

Applying (2)-(10), we can calculate the total energy consumption for a range of all possible PHY settings A and determine the fixed minimum E_{min} and maximum E_{max}, to allow normalization of the energy consumption later.

The second factor to optimize in the system is the PRR, this factor depends on both the environment e and used setting A. The complete system $G(A, e)$ with respect to the A-th setting and e-th environment can be described by (12). It is a combination of the PRR and the scaled and normalized energy consumption (E).

$$PRR = \frac{p_{rec}}{p_{total}}. \tag{11}$$

$$G(A, e) = PRR(A, e) + \left(1 - \frac{E(A) - E_{min}}{E_{max} - E_{min}}\right). \tag{12}$$

D. Problem description

The goal is to enable reliable UWB communication while consuming as little energy as possible with the available settings. This is depicted below.

Maximize:

$$G(A, e) = PRR(A, e) + \left(1 - \frac{E(A) - E_{min}}{E_{max} - E_{min}}\right)$$

Constrained by:

$$A$$

IV. Algorithm Design

In this section, the proposed deep Q-learning algorithm for runtime adaptation of UWB PHY settings while maximizing $G(A, e)$ as introduced in Section III is discussed. First, the key parts of RL are introduced. Then, the feature selection step is shown. Next, we will discuss why the discrete nature of Q-learning limits to what extent link estimation can be optimized. Finally, a more advanced deep Q-learning algorithm is proposed.

A. Reinforcement learning

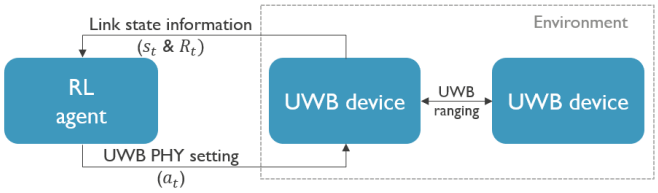


Fig. 3: High-level overview of RL system for UWB PHY runtime adaptation

As shown in Figure 3, the RL framework is composed of an agent and an environment interacting with each other. An agent considers the link between an anchor and a tag UWB device and is executed at the anchor. Anything in the area around the two UWB devices that could possibly affect communications is regarded as the environment. As in Figure 3, at each time t, the UWB link observes a state s_t, and takes an action a_t. The action selection is determined by a state-action function, $Q(s_t, a_t)$. The environment transitions to the

next state s_{t+1} and receives a reward R_t based on the action taken. Specifically for this problem, the objective function G can be seen as a first approximation of the reward. The UWB PHY setting, expressed as:

$$A = \{C, PSR, PRF, DR, P_{tx}\},$$

is the action space from which action a_t is selected. The state is a combination of the link state parameters from Table I, expressed as:

$$s_t = \{CIR, F_1, F_2..., Q_2\}.$$

B. State feature selection

Using all the available diagnostics of the used Qorvo DW1000 for the state s_t would considerably increase the complexity of the proposed algorithms. Therefore, a feature selection step is necessary. Reinforcement learning consists of two phases, namely an initial exploration phase, where many actions are tried out to find out in general which actions perform well and the optimization phase, where the system tries to find the optimal action to select. To select features that are representative for both phases, there are two factors based upon which features can be prioritized: (1) for exploration, their impact on selecting which action to take and (2) to find the optimal action, their influence on the objective function G. To reflect these two factors, feature selection is performed for both classifying which setting to select and predicting the value of G. To select the best features to predict G, we use the F-value [29]. This calculation is performed in two steps:

1) The cross-correlation $Xcor$ between each feature $f(i)$ in list of features f and target G is calculated using (13).
2) The cross correlation is converted to the F-value using equation 14.

$$Xcor(i) = \frac{E\left[(f[i] - mean(f[i]) \cdot (G - mean(G))\right]}{std(f[i]) \cdot std(G)} \quad (13)$$

$$F(i) = \frac{Xcor(i)^2}{1 - Xcor(i)^2} \cdot \nu, \quad (14)$$

where ν is the degrees of freedom. ($\nu = n - 1$ with n is the number of features f in the dataset). The features with the highest F-value have the highest correlation to the value of G and will thus be the most important features. To select the best features to classify which setting is the best, the Analysis of variance (ANOVA) F-value can be used. This F-value is similar to the previously explained F-value, only now the result is a categorical value (the setting) instead of a numerical value. The F-values of the features for both F-value calculations are shown in Figure 4. From this, we can select the most important features to be used in our system. Important to note is that the number of features to use is also a trade-off. Adding all available features causes higher complexity to the system and several features have a very low F-value. However, the system needs enough features to be able to learn and distinguish between different states. The selected features are: RX_p , FP_p, $NLOS$, N_c, Q_1, RX_{pacc}, LDE, Q_2. PRR is also a state feature but not added in the feature comparison because it added in any case due to it being central to this problem and the best indicator for the performance. The selection of these features and why this number of features are used is discussed in the first part of the evaluation results (section VI-D).
These will be used as the state of the RL system, expressed as:

$$s_t = \{RX_p, FP_p, NLOS, N_c, Q_1, RX_{pacc}, LDE, Q_2, PRR\},$$

C. Q-learning

Q-learning is a model-free RL algorithm that learns the value of an action in a certain state. The most important part of the algorithm is the Q-table. Each row in the table represents a state of the system, and each column represents an action. Each value in the table represents the 'quality' of a particular state-action pair. A more detailed schematic of the Q-learning algorithm is given in Figure 5.

1) Updating the Q-table: The foundation of the Q-learning algorithm is the Bellman equation that is a value update function using the newly received information and weighted old value:

$$Q_{new}(s_t, a_t) \leftarrow Q_{old}(s_t, a_t) + \alpha \left[R_t + \gamma \max_a Q(s_{t+1}, a_t) - Q_{old}(s_t, a_t)\right]. \quad (15)$$

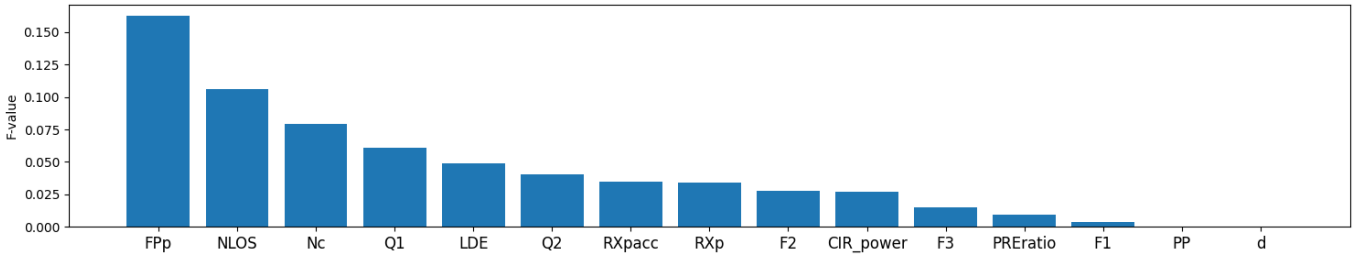
The following parameters are used in the equation:

- α: learning rate. This factor determines the weight that is given to the newly acquired information and how much old information can be overridden.
- γ: discount factor. This factor determines the weight that is given to newly acquired information.

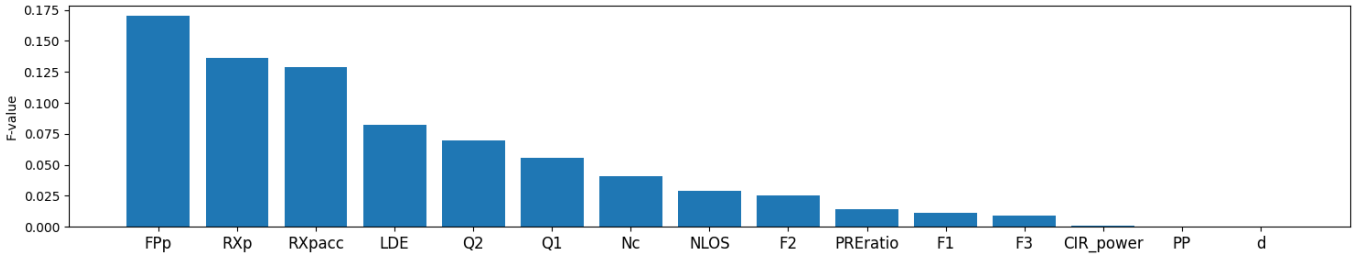
Using this function, the values in the Q-table are filled in. After enough iterations, the values in the table will reflect the quality of the state-action pairs and the expected value of the total received rewards will be maximized. The reward clearly drives the behavior of the algorithm and the reward function, a mapping of a state-action pair to a numerical value that indicates the desirability. The objective function G discussed before is a potential reward function because the goal of the algorithm is to maximize this objective. However, using this function does not deliver good results. It could lead to the selection of a certain setting based purely on a low energy consumption. If the setting with the lowest energy consumption is used, but it causes a PRR of 0 the reward is 1. Using the highest energy setting with a PRR of 1 also results in a reward of 1, while this is a more desirable setting (because there is communication possible). Having a low energy consumption is meaningless when no communication is possible using that low energy setting. To mitigate this problem, the following reward function is proposed:

$$R_t = PRR + PRR \cdot \left(1 - \frac{E - E_{min}}{E_{max} - E_{min}}\right). \quad (16)$$

In this function, the energy consumption factor is multiplied by the PRR. This causes the energy consumption to influence the reward proportionally to the reliability of the communication while still prioritizing the PRR. Different reward function are possible, however we have found this reward function to



(a) Action



(b) Objective (G)

Fig. 4: Overview of the F-values for all features, for (a) classification of PHY setting (or action) and (b) prediction of the G used in the run-time adaptation of the UWB PHY layer.

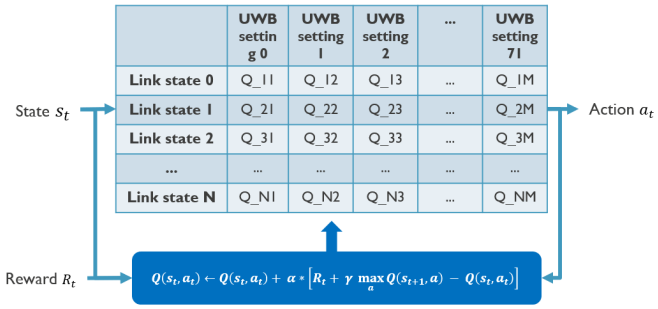


Fig. 5: Schematic showing the Q-table and Bellman update equation for the Q-learning run-time adaptation of UWB PHY settings

be a good performing one for the goals we set for the system. Nonetheless, different reward function are possible. Because reward function can be used to change the behavior of the algorithm and there is not one singular 'best' we will evaluate and discuss the performance under different reward function in section VI-D3.

At each point in time, the agent selects an action to be taken using the Q-table. This action selection is a fundamental trade-off in RL. Initially, the agent does not yet know the outcome of the possible actions. Hence, high enough exploration is required. Once the agent has learned more information, the exploration can be reduced by exploiting (selecting the action with highest Q-value) the learned information.

During the training stage, the epsilon-greedy policy is used. In this policy, the best action (highest Q-value) is selected with a probability of 1-ϵ. With a probability of ϵ a random action is chosen uniformly. For evaluation, the epsilon-greedy policy is modified. Now, in exploration, the random selection is changed to a random selection among the 10 actions with the highest Q-value.

2) State determination: The biggest drawback of using Q-learning is the need to determine a discrete state number from continuous state variables. This introduces a trade-off between granularity and size of the Q-table. The state features are each split up in 3 categories: low, middle and high. This gives a total of 19,683 possible link states or rows in the Q-table. Multiplying this with the number of actions, there is a total of 1,417,176 cells. The Q-table is huge, while we have only introduced very limited granularity on the features and performed feature selection. Without feature selection, there would be $83.7e^9$ cells in the Q-table. This demonstrates the need for the feature selection step. Additionally, determining the low, middle, and high separation for each feature requires expert knowledge of UWB diagnostics, which can greatly impact the algorithm's performance. The pseudocode for the complete Q-learning algorithm is given in Algorithm 1.

D. Deep Q-learning

To mitigate the previously mentioned drawbacks, the Q-learning algorithm can be modified to a Deep Q-learning algorithm. In Deep Q-learning, the Q-table is replaced by a neural network that approximates the Q-value function. The state is given as input and the Q-value of all possible actions is the output. Using a neural network instead of a table has several benefits: (1) The continuous values of the link state measurements can now be used directly as input, (2) no 'expert knowledge' is necessary to determine categories for the inputs, (3) no loss of information due to discretization and (4) no need for a table containing millions of cells.

1) Target network: In supervised learning, the label of an input does not change over time. This stable condition for input

Algorithm 1 Q-learning for UWB PHY

Require:
Initialize Q-table
Initialize parameters α, γ and ϵ
Ensure:
Select random UWB link and action (setting)
Determine start state s_t
while step $<$ training steps **do**
  Every 100 steps: select new random UWB link
  With probability ϵ select random action a_t
  Otherwise $a_t = arg\max_a(Q(s,a))$
  Perform action a_t (configure setting)
  Measure link quality indicators
  Determine reward R_t and new state s_{t+1}
  $Q_{new}(s_t,a_t) \leftarrow Q_{old}(s_t,a_t)$ $+\alpha\ [R_t + \gamma \cdot \max_a Q(s_{t+1},a_t) - Q(s_t,a_t)]$
  Update ϵ
  s_t=s_{t+1}
end while

and output allows it to perform well. In RL, both the input and the target to obtain change constantly during the learning process. This makes training unstable because the target or 'label' of the output Q depends on Q itself. (17) shows the target value based on the Bellman equation with θ, the weights of the Q neural network.

$$target = R_t + \gamma \cdot \max_a Q(s_{t+1},:,\theta). \tag{17}$$

This means that the target will move as the Q approximation improves. The moving target could cause the 'chasing your own tail' problem [30]. To mitigate this issue, a target network can be introduced. The target network is used to calculate the Q-values and fix the target for as long as we fix the target network. The main network updates its weights θ using the training data to minimize the following loss function (18) on a dataset D.

$$Loss(\theta) = \sum_{s_t,a_t \in D} (target - Q(s_t,a_t,\theta))^2. \tag{18}$$

After several updates of the main network, the weights of the main network are copied to the weights of the target network $\hat{\theta}$ [31]. The neural network architecture is given in Table V. The input to the neural network consists of the state and current settings, The output is an approximation of the Q-values for all actions in this state.

TABLE V: Neural network architecture

DNN	
Layer	**Output dimension**
Input	14
Dense(128), ReLu	128
Dense(256), ReLu	256
Dense(512), ReLu	512
Dense(256), ReLu	256
Dense(128), ReLu	128
Dense(72), linear	72
Output	72

2) Weighted importance sampling: Updating the neural network at every time step with one sample would be very inefficient. Therefore, the network is updated on batches of data that are sampled from a replay memory containing experiences (s_t, a_t, R_t, s_{t+1}) of the Q-agent. There are several methods to sample from the memory, such as random sampling, prioritized sampling, and weighted importance sampling. [32]. For this problem, we opted for weighted importance sampling. The optimal criterion to select on would be the amount the Q-agent can learn from using an experience. While this measure is not available, a reasonable approximation is the temporal-difference (TD)-error, which indicates how surprising a certain transition is. However, pure greedy prioritization can result in overfitting and a lack of diversity in the sampled experiences. To address these issues, a stochastic sampling method is used that interpolates between pure greedy prioritization and uniform random sampling [32]. The priority or TD-error p_i of an experience is calculated in (19) and the sampling probability of each experience is given in (20).

$$p_i = R_t + \gamma \cdot \max_a \hat{Q}(s_{t+1},:,\hat{\theta})) - Q(s_t,a_t). \tag{19}$$

$$Prob(i) = \frac{p_i^\zeta}{\sum_k p_k^\zeta}, \tag{20}$$

where ζ is a hyperparameter that determines how much prioritization is used. This priority is saved with the experience. Prioritized learning in this way introduces bias, as it changes the solution that the estimates will converge to [32]. This bias can be corrected by using importance sampling weights (w) shown in (21).

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{Prob(i)}\right)^\beta, \tag{21}$$

where N is the amount of experiences in the memory, and β a hyperparameter that determines how much the non-uniform probabilities $P(i)$ are compensated.

3) Update function: Combining this, the update function to generate the training data is given by equation (22)

$$Q(s,a_t,\theta) = Q(s,a_t,\theta) + \alpha \cdot p_i \cdot w_i \tag{22}$$

The pseudocode for the Deep Q-learning algorithm is given in algorithm 2.

V. Training

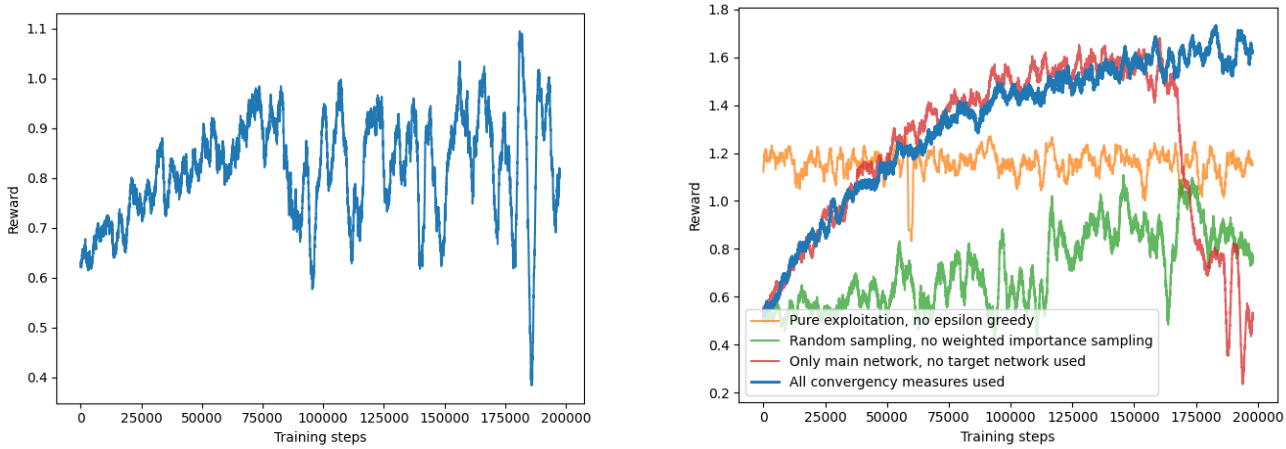
A. Q-learning

The Q-learning algorithm was trained for 200,000 steps with $\alpha = 0.8$ and $\gamma = 0.5$. ϵ (from the epsilon-greedy policy) changed during the training, following an exponential decay as shown in (23). The symbol λ represents the decay constant, 'step' refers to the number of training steps that have been completed thus far.

$$\epsilon = \epsilon_{min} + (\epsilon_{max} - \epsilon_{min}) \cdot e^{-\lambda \cdot step} \tag{23}$$

where $\epsilon_{min} = 0.01$, $\epsilon_{max} = 1$ and $\lambda = 3.91e^{-5}$.

Figure 6a shows the received rewards during training. Initially, the received rewards increase, however after more than 100,000 steps, the received rewards become more noisy



(a) Q-learning

(b) Deep Q-learning

Fig. 6: The received rewards (averaged over 200 training steps) during training of Q and deep Q-learning. In 6b the blue shows the deep Q-learning using all discussed convergence measures, and the three others each have one measure disabled or switched to a worse performing one.

Algorithm 2 Deep Q-learning for UWB PHY

Require:
Initialize replay memory D to capacity N
Initialize action-value function Q with random weights θ
Initialize target action-value function $\hat{Q}$ with $\hat{\theta} = \theta$

Ensure:
Select random UWB link and start action (setting)
Determine start state s_t
while step $<$ training steps **do**
  Every 100 steps: select new random UWB link
  With probability ϵ select random action a_t
  Otherwise $a_t = arg\max_a(Q(s, a, \theta))$
  Perform action a_t (configure setting)
  Measure link quality indicators
  Determine reward R_t, new state s_{t+1} and priority p_t
  Store $e_t = (s_t, a_t, R_t, s_{t+1})$ with priority p_t in D
  Every M steps:
    Sample minibatch b of $e_j \sim Prob(j) = \frac{p_j^\zeta}{\sum_k p_k^\zeta}$
    For all e_j in b: current_Qs: $Q(s, :, \theta)$
    For all e_j in b: future_Q: $\hat{Q}(s_{t+1}, :, \hat{\theta}))$
    X = s_t for all e_j in b
    for all e_j in b:
      $max\_future\_q = R_t + \gamma \cdot \max(future\_Qs)$
      $p_i =$ |max_future_q - current_Qs[a_t]|
      $w_i = \left(\frac{1}{Prob(i)} \cdot \frac{1}{N}\right)^\beta$
      current_Qs[a_t] = current_Qs[a_t] + $\alpha \cdot p_i \cdot w_i$
    Y = current_Qs for all e_j in b
    Update weights θ: fit Q using X and Y.
  Every T steps: reset $\hat{Q} = Q$, i.e., $\hat{\theta} = \theta$
end while

with outliers in both the positive and negative directions. This indicates that the Q-learing algorithm has difficulties with learning the actions with the highest reward in a state, this could be due to determination of discrete states from continuous state variables.The Q-learning agent after these 200,000 training steps is used in the experimental evaluation section.

B. Deep Q-learning

The deep Q-learning algorithm was trained for 200,000 steps, with $\alpha = 0.8$ and $\gamma = 0.5$. The neural network was trained using Huber loss [33] and the Adam optimizer [34]. ϵ changed during training following (23) with $\epsilon_{min} = 0.01$, $\epsilon_{max} = 1$ and $\lambda = 1.96e^{-5}$.

1) Convergence improvements: Reinforcement learning algorithms are known to be challenging to converge, to improve the convergence three measures were adopted and are explained in Section IV: (1) Decaying epsilon-greedy exploration (2) Weighted importance sampling and (3) Target network. The need for these three measures is illustrated in Figure 6b. The figure shows the received rewards during training when these three convergence measures are used and when one of the measures is disabled. When all convergence measures are used, the reward increases steadily during training. When pure exploration is used, the reward starts off higher because there are no random actions selected, but because there is no exploration, the Q-agent does not learn the best possible actions and the reward does not increase. Without weighted importance sampling, (with random sampling), the received reward increases slightly, but there is more noise and the reached reward level is significantly lower. When the target network is not used, the received rewards have a similar pattern as when all convergence measures are used. However, after around 160,000 training steps, the received rewards suddenly drop dramatically (this does not always happen at this specific point). This effect is also known as 'catastrophic forgetting'

[35] and the target network is a measure specially to overcome this problem. The deep Q-learning agent (with all convergence measures) after 200 000 training steps is used in the evaluation section.

VI. Experimental evaluation

In this section, we evaluate the proposed Q-learning and deep Q-learning algorithms to assess their performance compared to a linear search algorithms and fixed hard-coded UWB PHY settings, and we also compare the difference in behavior based on the used reward function. First, the environment in which the data for the experiments is gathered is described. Then, we describe how the measurements are performed, and finally we discuss and analyze the results.

A. Office lab

The OfficeLab from imec - IDLab - Ghent University [36] offers a test environment which includes 3 floors that are equipped with 40 Intel NUC nodes, supporting several Wi-Fi and sensor technologies including UWB. In our evaluation, we use a single floor that has a total area of around 41 x 26 m^2 and 15 UWB nodes placed in corridors, meeting rooms, and offices. All nodes are placed at the same height of around 2.6 m above the floor. The walls separating the rooms consist of heterogeneous materials, ranging from plywood to reinforced concrete, resulting in a very heterogeneous environment. An overview of the placement of the nodes is shown in Figure 7.

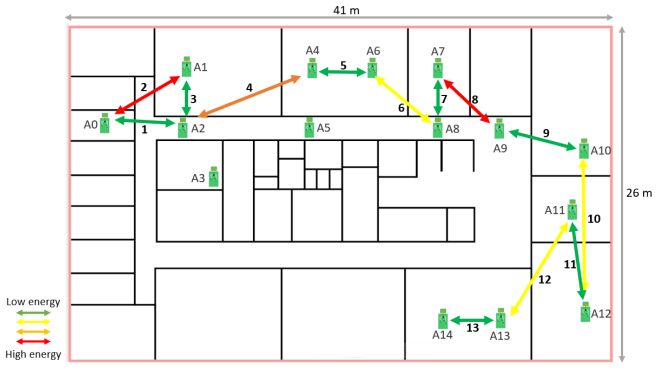


Fig. 7: Data gathering setup with 15 UWB nodes (A0-A14) and the anchor-tag links selected for dynamic evaluation distributed over the 9th floor of the OfficeLab at imec - IDLab - Ghent University

B. Dataset

The dataset was gathered using Wi-PoS: as UWB hardware connected to an Intel NUC as shown on the proposed architecture (Figure 1b). All 15 nodes are programmed once as tag and try to range with the other 14 nodes (anchors) using the settings shown in Table VI.

The combination of these settings give a total of 72 different UWB PHY settings. ADS-TWR [27] is used to enable accurate ranging. All devices have ranged with each other for a total of 500 range attempts per combination. This resulted in a total of more than 605 thousand ranges overall. This is considerably less than the total attempts, as on average each node could only range with 7 other nodes due to obstacles like walls.

TABLE VI: UWB PHY settings used in the dataset

Parameter	Values
Channel	3,5,7
PSR	128, 1024, 4096
PRF	16, 64 MHz
Data rate	110, 6800 kbps
Transmit power gain	0,10.5 dB

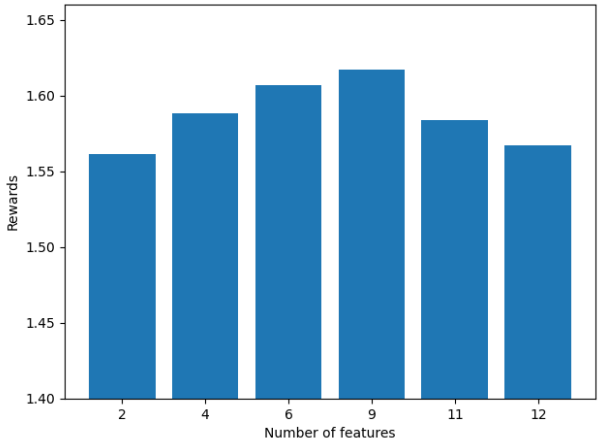


Fig. 8: The average received rewards during the last 10,000 steps of training for a different number of selected state features, showing that using the 9 most important features (determined in IV-B) results in the best performance.

C. Feature selection evaluation

In section IV-B the importance of the available state features was determined using the F-value statistics. Here, we determine how many features need to be used as the state to enable the highest performance. This is done by training the deep Q-learning model using different numbers of state features and comparing their average reward at the end of the training. This allows us to assess the relative performance, as the ultimate goal of reinforcement learning is to maximize the cumulative reward over time, and the average reward received at the end of training is a good indicator of how well it is achieving this goal.

In Figure 8, the average received rewards during the last 10,000 steps of training are shown for different numbers of selected state features. Features are added based on previously determined importance. The figure shows that adding more features increases the received rewards up to 9 features, once 11 or more features are used, the rewards start decreasing again. This shows that adding irrelevant or redundant features, i.e. features with very low F-values (the ones added last), can increase noise and make it harder for the model to identify the relevant patterns in the data. This shows that using 9 features should yield the optimal performance, and the selected state features are:

$$s_t = \{RX_p, FP_p, NLOS, N_c, Q_1, RX_{pacc}, LDE, Q_2, PRR\},$$

TABLE VII: Parameters used during evaluation of the runtime adaptation of UWB PHY settings

Parameter	Static scenario	Dynamic scenario
Ranging update rate	50 Hz	50 Hz
Update rate main network	5	20
Update rate target network	25	100
Learning rate	0.7	0.55
Discount factor	0.7	0.55
Batch size	10	256

D. Results

In this section, we evaluate the proposed Q-learning and deep Q-learning algorithms to assess their impact on (1) the time it takes to find the best setting in a static situation (the environment and link is fixed) compared to a linear search algorithm, (2) the reliability (PRR) and energy consumption in a dynamic or changing environment compared to using fixed hard-coded UWB PHY settings and (3) the reliability and energy consumption of deep Q-learning in a dynamic or changing environment for different reward functions.

1) Static environment: This experiment tries to evaluate the ability of the proposed model to quickly, in terms of RL iterations (or amount of settings configured), determine the best PHY layer setting when UWB communication is started between two devices, deployed in new conditions, using a pre-trained RL algorithm (see Section V). Since the environment is different, the RL agents require input about the new link state conditions (PRR, RX_p, FP_p, etc...). This information is based on statistics that are not yet available in new deployments. As such, at iteration 0, the neural network cannot yet make any decision and instead selects a random PHY layer setting. Over time, state information that relies on statistical calculations becomes more reliable, allowing the system to increasingly make use of the neural network for optimal decisions. The parameters of the models used during this evaluation are shown in Table VII.

In Figure 9, the percentage of UWB links that found settings within 5% of the highest possible reward (for that link) in function of the number of iterations is shown for Q-learning and Deep Q-learning. The y-axis value is explained mathematically in (24), with D the complete dataset, l a link in between two UWB devices, R_selected(l) the reward of the UWB PHY setting selected by the algorithm for link l and R_best(l) the reward of the setting with the highest reward for link l. A link is considered 'optimal' if the reward of the selected UWB PHY setting is within 5% of the setting with the highest possible reward, $Optimal(l)$ returns 1 in this case. This 5% comes from minor changes in the environment (airco, humidity, etc.) which result in slight variations of about 5% performance impact, even when using fixed settings. Due to these fluctuations, even using fixed best performing settings shows random variations.

$$\text{Optimal(l)} = \begin{cases} 1 & \text{R\_selected(l)} \ \geq \text{R\_best(l)} \times 0.95 \\ 0 & \text{otherwise} \end{cases} \quad (24)$$

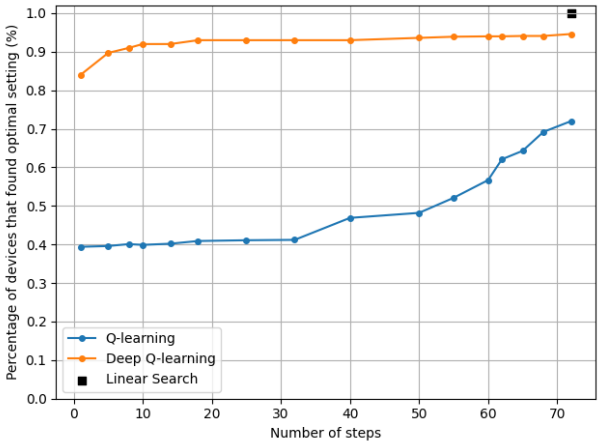


Fig. 9: The percentage of optimal configurations selected in terms of number of evaluation steps

$$\text{y} = \frac{\sum\limits_{\forall l \in D} \text{Optimal(l)}}{\sum\limits_{\forall l \in D} 1} \quad (25)$$

The performance of a linear search algorithm is indicated on the figure as well. This algorithm tries out every possible setting and selects the best performing one in the end. In this scenario, there are 72 possible settings, which means that the algorithm needs 72 iterations but is 100% sure that the best setting is selected. Q- and deep Q-learning try to predict the best setting based on the link state parameters. Figure 9 clearly shows that Deep Q-learning performs better than Q-learning. After the first iteration, which means starting at a random action (iteration 0) measuring the link state and then configuring a setting, the Q-learning algorithm selects an optimal setting for 39% of the links between UWB devices and Deep Q-learning for 84%. This demonstrates the ability of the Deep Q-learning algorithm to select an optimal action using only link state measurements from an initial random action. After 10 iterations, with more reliable link state measurements, the percentage has increased to 92% for Deep Q-learning, while the Q-learning is still only around 40%. Comparing the algorithms when they have used the same amount of iterations as linear search shows that Deep Q-learning finds the best setting in 95% of the cases and Q-learning 70%. These results show the importance of using Deep Q-learning instead of Q-learning. The drawbacks of using Q-learning as discussed in IV-C are clearly visible in these results. Deep Q-learning shows a clear advantage over the linear search algorithm, as it can select an optimal setting with a high accuracy after only a few iterations. While the linear search algorithm requires 72 iterations.

2) Dynamic environment: To test the performance of the algorithms in a dynamic environment, a test scenario is defined as shown in Figure 7. To simulate a walk around the office, every 5 seconds, one of the anchor-tag UWB nodes is switched to abruptly change the situation and test how well the

algorithm can adapt. Harder links, containing more obstacles, require more energy. The color used for the tag-anchor link indicates the minimal amount of energy necessary to enable communication, as shown by the legend. The parameters used during this experiment are shown in Table VII.

In Figure 10, the PRR and the energy consumption during this test scenario are shown for different cases, namely deep Q-learning, Q-learning and three commonly used fixed hard-coded UWB PHY settings. These were chosen because they represent the difference in performance of high and low energy settings and the influence of the channel.The fixed hard-coded UWB PHY settings are (1) a high energy consuming setting on channel 7 {7, 4096, 64, 6800, 10.5}, (2) a high energy consuming setting on channel 3 {7, 4096, 64, 6800, 10.5} and (3) a low energy consuming setting on channel 7 {7, 128, 64, 6800, 0}. Figure 10 shows that using a fixed hard-coded UWB PHY setting does not guarantee good reliability. Communication drops off completely at several links for the fixed hard-coded UWB PHY settings on channel 7, even for the high energy consuming one. The high energy consuming setting on channel 3 has better reliability, except for the single drop to 40%.

Q-learning has no links where the communication drops off completely, but fails to provide true reliability as the PRR is never 100%. This means that frames are continuously lost. However, it still uses considerably less energy than the two high energy consuming ones.

Figure 10 shows that the Deep Q-learning algorithm can ensure reliable communication, by keeping the PRR constantly very close to 100%, while dynamically selecting settings with higher energy consumption when necessary. This is well demonstrated by the sharp increase in energy consumption at time intervals 5-10 and 35–40 seconds of the evaluation. In these time intervals, the UWB communication is happening over a red link (number 2 and 8) in 7. Between 40 and 65 seconds, we can clearly see that the algorithm can also make the distinction between links with a smaller difference in required energy consumption (green vs. yellow). However, deep Q-learning never selects a PHY setting with the lowest energy setting, even though this setting can enable reliable communication for certain links.

Table VIII compares the average performance of the five different cases in the figure in terms of PRR, energy consumption and ranging error. Although ranging error is included in the table, it is important to note that it is not the objective of the developed RL systems. While the energy consumption and packet reception rate are directly related to the system's objective, the ranging error is included in the table to provide a more complete picture of the system's performance, rather than as a metric to be optimized. Improved ranging error could be a side effect of better communication. Deep Q-learning clearly performs best, it has the highest average PRR, the lowest energy consumption which also results in good ranging error performance. The high energy consuming setting on channel 3 comes closest based on PRR and has slightly better ranging error performance, but it consumes more than seven times the amount of energy. While Q-learning performs considerably worse than deep Q-learning, it has a similar PRR and ranging performance as the high energy consuming setting on channel 3 and consumes less than half the amount of energy. The fixed settings on channel 7 perform the worst as the PRR is low, even when consuming a lot of energy.

3) Influence of reward function: In this section, the same experimental evaluation is performed as in VI-D2 but instead of comparing different algorithms, we will be comparing the performance of the deep Q-learning algorithm under different reward functions to show that the behavior can be shaped using this reward function and that our proposed reward function is the best one for minimizing energy consumption while prioritizing the PRR. However, different systems can have different priorities and our proposed deep Q-learning algorithm can be adapted to this by changing the reward function. In Figure 11 and Table IX, the PRR and the energy consumption during the dynamic test scenario are shown for different reward functions: (1) the previously proposed reward function (2) $R_t = PRR$, to show the performance when the energy consumption is not considered in the system (3) $R_t = PRR + \left(1 - \frac{E - E_{min}}{E_{max} - E_{min}}\right)$ to illustrate that this function does not lead to the desired performance (4) $R_t = PRR \cdot \left(1 - \frac{E - E_{min}}{E_{max} - E_{min}}\right)$.

Figure 11 indicates that relying solely on PRR as a reward yields consistently high PRR during evaluation, but incurs a significant increase in energy consumption compared to our proposed reward function. Specifically, Table IX illustrates that it provides the highest PRR, which is virtually the same as our proposed reward function, it comes at the expense of a considerable increase in energy consumption, which is approximately 4.5 times greater than that of our proposed reward function. Adding normalized and scaled energy consumption to the reward function can improve energy consumption, as shown in Figure 11. However, this reward function favors low-energy settings even when high-energy is necessary for successful communication, leading to significant packet loss. Although this approach leads to the lowest average energy consumption (Table IX), it results in a significantly lower PRR and which depicts the issue of this reward function: receiving relatively high rewards for low-energy settings with PRR of 0, which removes the incentive for the algorithm to switch to higher-energy settings for better PRR. Multiplying the PRR and energy consumption factors in the reward function (shown in red on Figure 11) leads to better PRR performance, as low energy consumption is rewarded proportionally to the PRR. This causes the deep Q-learning algorithm to change to more energy consuming settings when necessary. At first glance, this reward functions seems to lead to the best performance as it leads to lower energy consumption compared to the proposed one, and it has high PRR. However, there are a few caveats, mainly if the reliability has the highest priority. this is most clearly represented in the figure between 5 and 10 seconds. The algorithm using this reward function switches to a setting with a higher energy consumption of around 5000 $W\mu s$) that lead to a PRR of around 0.94 while the algorithm with our proposed reward function switches to a setting that consumes more than 3 times more energy but importantly keeps the PRR at 1. This depicts the behavior of this reward function: it tries to balance both the energy consumption and

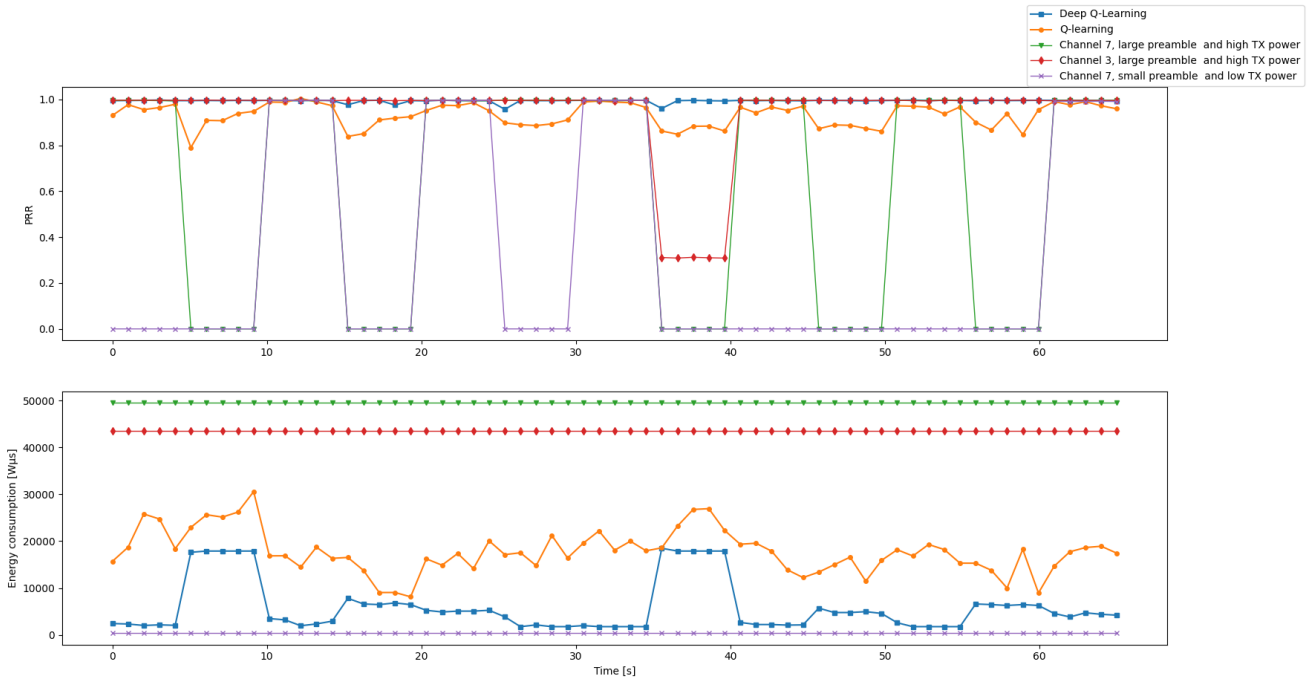


Fig. 10: Comparison of the proposed Q-learning and deep Q-learning algorithms with constant high energy and low energy PHY settings in terms of PRR and energy consumption.

TABLE VIII: Average performance comparison of the different methods in the dynamic test scenario

fixed hard-coded UWB PHY settings / Proposed algorithm	**Average energy consumption (Wμ s)**	**Average PRR (%)**	**Average ranging error (mm)**
Channel 7, small preamble and low P_{tx}	373.92	30.59	295.018
Channel 7, large preamble and high P_{tx}	49572.15	61.23	318.02
Channel 3, large preamble and high P_{tx}	43497.42	94.25	235.92
Q-learning	17782.15	93.26	243.45
Deep Q-learning	5898.92	99.31	240.65

TABLE IX: Average performance comparison of different reward functions in the dynamic test scenario

Reward function	**Average energy consumption (Wμ s)**	**Average PRR (%)**
$R_t = PRR + PRR \cdot \left(1 - \frac{E - E_{min}}{E_{max} - E_{min}}\right)$ **(proposed)**	5898.92	99.31
$R_t = PRR$	27066.88	99.32
$R_t = PRR + \left(1 - \frac{E - E_{min}}{E_{max} - E_{min}}\right)$	1763.92	83.96
$R_t = PRR \cdot \left(1 - \frac{E - E_{min}}{E_{max} - E_{min}}\right)$	3924.63	98.45

reliability objectives by selecting a low energy setting that has an 'acceptable' PRR, while the addition of the extra PRR term in our proposed reward function leads to the highest priority being on the PRR remaining as close as possible to 1 at all times. This effect is also visible (to a lesser extent) between seconds 15 and 25 and 45 and 50 and is also reflected in Table IX. These results show that our proposed reward function makes PRR the highest priority and energy consumption should be as low as possible to not negatively impact the PRR. $R_t = PRR \cdot \left(1 - \frac{E - E_{min}}{E_{max} - E_{min}}\right)$ allows the system to accept a slight decrease in PRR if it leads to lower energy consumption.

E. Complexity analysis

1) Algorithmic complexity: The complexity of Q-learning and Deep Q-learning is mainly based on determining and updating the Q-value approximation of a state [37]. Q-learning selects a table row O(1) and modifies one value. Deep Q-learning's complexity depends on neural network structure and hyperparameters. For fully-connected layers, complexity is $O(mn \log n)$, with m the number of layers and n the number of units per layer [38].

2) Time complexity: Q-learning's optimal setting determination and Q-table update time depends on selecting the maximum value in a row and updating one Q-table value. It takes 6.89 ms on an Intel(R) Xeon(R) E3-1200/1500 v5/6th Gen laptop. Deep Q-learning's time depends on model size, batch size, update rate, and retraining. A higher update rate makes the model more dynamic but increases time complexity. For the settings shown in Table VII, the average inference time is 23.18 ms, and one batch training takes 320 ms on average.

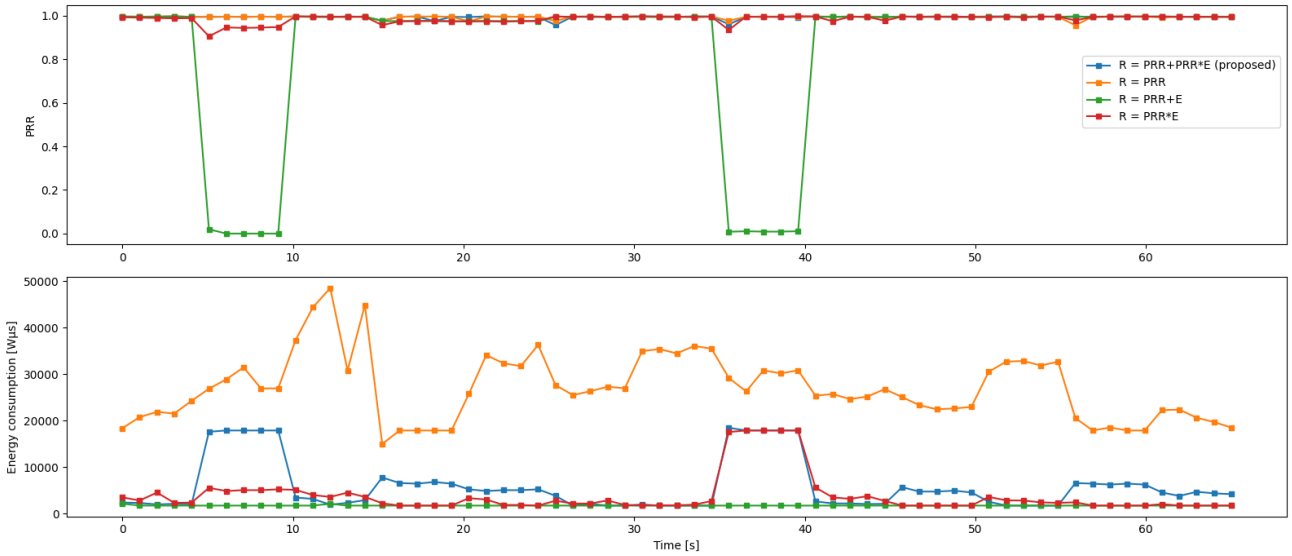


Fig. 11: Comparison of different reward functions for the proposed deep Q-learning algorithm in terms of PRR and energy consumption. The E in the legend represents the scaled and normalized energy consumption.

VII. Conclusion

For the increasingly prominent indoor localization technology UWB, accurate ranging has been extensively researched. However, enabling reliable, low energy consuming UWB communication in dynamic environments is largely unexplored. This work proposes a deep Q-learning algorithm for improved reliability in dynamic environments while minimizing energy consumption by changing the PHY layer settings based on link state measurements. This method outperforms using a fixed PHY layer setting and an exhaustive search, which are currently the most common ways to set the PHY setting. We found that deep Q-learning can achieve higher PRR while using considerably less energy, and whilst also reducing the ranging error as a side effect. We also concluded that using traditional Q-learning does not suffice to solve this problem, and that different reward function can be used if different requirements are placed on the system.In addition, future work could employ RL for UWB localization systems using Time Difference of Arrival (TDoA) instead of two-way ranging or multi-agent localization problems.

VIII. Acknowledgment

This work was supported by the imec ICON WISH project HBC.2021.0664.

References

[1] D. Coppens, E. De Poorter, A. Shahid, S. Lemey, B. Van Herbruggen, and C. Marshall, "An overview of uwb standards and organizations (ieee 802.15. 4, fira, apple): Interoperability aspects and future research directions," *IEEE Access*, 2022.

[2] A. Alarifi, A. Al-Salman, M. Alsaleh, A. Alnafessah, S. Al-Hadhrami, M. A. Al-Ammar, and H. S. Al-Khalifa, "Ultra wideband indoor positioning technologies: Analysis and recent advances," 2016.

[3] B. Denis, J. Keignart, and N. Daniele, "Impact of nlos propagation upon ranging precision in uwb systems," in *IEEE Conference on Ultra Wideband Systems and Technologies, 2003*, 2003, pp. 379–383.

[4] J. Fontaine, M. Ridolfi, B. Van Herbruggen, A. Shahid, and E. De Poorter, "Edge inference for uwb ranging error correction using autoencoders," *IEEE access*, vol. 8, pp. 139 143–139 155, 2020.

[5] C. Mao, K. Lin, T. Yu, and Y. Shen, "A probabilistic learning approach to uwb ranging error mitigation," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6.

[6] S. Wu, Y. Ma, Q. Zhang, and N. Zhang, "Nlos error mitigation for uwb ranging in dense multipath environments," in *2007 IEEE Wireless Communications and Networking Conference*, 2007, pp. 1565–1570.

[7] "Ieee standard for low-rate wireless networks," *IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015)*, pp. 1–800, 2020.

[8] B. Großwindhager, C. Alberto Boano, M. Rath, and K. Römer, "Enabling runtime adaptation of physical layer settings for dependable uwb communications," in *2018 IEEE 19th International Symposium on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2018, pp. 01–11.

[9] B. Kempke, P. Pannuto, and P. Dutta, "Polypoint: Guiding indoor quadrotors with ultra-wideband localization," in *Proceedings of the 2nd International Workshop on Hot Topics in Wireless*, 2015, pp. 16–20.

[10] D. Ltd, *DW1000 User Manual Version 2.18*, 2017.

[11] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[12] X.-L. Huang, Y.-X. Li, Y. Gao, and X.-W. Tang, "Q-learning-based spectrum access for multimedia transmission over cognitive radio networks," *IEEE Transactions on Cognitive Communications and Networking*, vol. 7, no. 1, pp. 110–119, 2021.

[13] O. J. Pandey, T. Yuvaraj, J. K. Paul, H. H. Nguyen, K. Gundepudi, and M. K. Shukla, "Improving energy efficiency and qos of lpwans for iot using q-learning based data routing," *IEEE Transactions on Cognitive Communications and Networking*, vol. 8, no. 1, pp. 365–379, 2022.

[14] A. Shahid, S. Aslam, H. S. Kim, and K.-G. Lee, "A docitive q-learning approach towards joint resource allocation and power control in self-organised femtocell networks," *Transactions on Emerging Telecommunications Technologies*, vol. 26, no. 2, pp. 216–230, 2015.

[15] M. Girmay, V. Maglogiannis, D. Naudts, A. Shahid, and I. Moerman, "Coexistence scheme for uncoordinated lte and wifi networks using experience replay based q-learning," *Sensors*, vol. 21, no. 21, p. 6977, 2021.

[16] V. Maglogiannis, D. Naudts, A. Shahid, and I. Moerman, "A q-learning scheme for fair coexistence between lte and wi-fi in unlicensed spectrum," *IEEE Access*, vol. 6, pp. 27 278–27 293, 2018.

[17] H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep reinforcement learning based resource allocation for v2v communications," *IEEE Transactions on Vehicular Technology*, vol. 68, no. 4, pp. 3163–3173, 2019.

[18] H.-S. Lee, J.-Y. Kim, and J.-W. Lee, "Resource allocation in wireless networks with deep reinforcement learning: A circumstance-independent approach," *IEEE Systems Journal*, vol. 14, no. 2, pp. 2589–2592, 2020.

[19] K. I. Ahmed and E. Hossain, "A deep q-learning method for downlink power allocation in multi-cell networks," 2019. [Online]. Available: https://arxiv.org/abs/1904.13032

[20] P. Losco, S. Bourdel, J. Gaubert, N. Dehaese, S. Meillère, O. Ramos, R. Vauche, and H. Barthelemy, "Analysis of the ieee 802.15.4a uwb phy layer for optimizing the power consumption of the transmitter," in *2013 IEEE International Conference on Ultra-Wideband (ICUWB)*, 2013, pp. 189–194.

[21] K. Mikhaylov, J. Petäjäjärvi, M. Hämäläinen, A. Tikanmäki, and R. Kohno, "Impact of ieee 802.15.4 communication settings on performance in asynchronous two way uwb ranging," *International Journal of Wireless Information Networks*, 2017.

[22] F. Hartmann, C. Enders, and W. Stork, "Ranging errors in uwb networks and their detectability," in *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, 2016, pp. 194–198.

[23] H. Mohammadmoradi, M. Heydariaan, and O. Gnawali, "Uwb physical layer adaptation for best ranging performance within application constraints," *Proceedings of the 2nd International Conference on Smart Digital Environment*, 2018. [Online]. Available: https://doi.org/10.1145/3289100.3289120

[24] D. Ltd., "Dw1000 user manual version 2.18."

[25] J. Schroeder, S. Galler, K. Kyamakya, and K. Jobmann, "Nlos detection algorithms for ultra-wideband localization," in *2007 4th Workshop on Positioning, Navigation and Communication*, 2007, pp. 159–166.

[26] B. Van Herbruggen, B. Jooris, J. Rossey, M. Ridolfi, N. Macoir, Q. Van den Brande, S. Lemey, and E. De Poorter, "Wi-pos: A low-cost, open source ultra-wideband (uwb) hardware platform with long range sub-ghz backbone," *Sensors*, vol. 19, no. 7, 2019. [Online]. Available: https://www.mdpi.com/1424-8220/19/7/1548

[27] Y. Jiang and V. C. Leung, "An asymmetric double sided two-way ranging for crystal offset," in *2007 International Symposium on Signals, Systems and Electronics*, 2007, pp. 525–528.

[28] D. Ltd., "Dw1000 dw1000 datasheet."

[29] W. G. Cochran, "Fisher and the analysis of variance," in *R.A. Fisher: An Appreciation*, S. E. Fienberg and D. V. Hinkley, Eds. New York, NY: Springer New York, 1980, pp. 17–34.

[30] O. M. Omisore, T. Akinyemi, W. Duan, W. Du, and L. Wang, "A novel sample-efficient deep reinforcement learning with episodic policy transfer for pid-based control in cardiac catheterization robots," *arXiv preprint arXiv:2110.14941*, 2021.

[31] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[32] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[33] P. J. Huber, "Robust estimation of a location parameter," New York, NY, pp. 492–518, 1992. [Online]. Available: https://doi.org/10.1007/978-1-4612-4380-9_35

[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[35] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences*, vol. 3, no. 4, pp. 128–135, 1999.

[36] "Ghent university - idlab research infrastructure: Officelab." [Online]. Available: https://www.ugent.be/ea/idlab/en/research/research-infrastructure/officelab.htm

[37] H.-S. Lee, J.-Y. Kim, and J.-W. Lee, "Resource allocation in wireless networks with deep reinforcement learning: A circumstance-independent approach," *IEEE Systems Journal*, vol. 14, no. 2, pp. 2589–2592, 2020.

[38] Y. Wang, C. Ding, Z. Li, G. Yuan, S. Liao, X. Ma, B. Yuan, X. Qian, J. Tang, Q. Qiu, and X. Lin, "Towards ultra-high performance and energy efficiency of deep learning systems: An algorithm-hardware co-optimization framework," 2018. [Online]. Available: https://arxiv.org/abs/1802.06402

Dieter Coppens received his master's degree in electrical engineering from Ghent University, Belgium, in 2021. He is currently a PhD candidate with the IDLab Research Group at Ghent University, where his research focuses on wireless networking and indoor localization systems using Ultra-wideband technology. Specifically, he is applying reinforcement learning techniques to optimize the localization and communication performance of these systems.

Prof. dr. Adnan Shahid (M'15 - SM'17) received the B.Sc. and M.Sc. degrees in computer engineering from the University of Engineering and Technology, Taxila, Pakistan, in 2006 and 2010, respectively, and the Ph.D. degree in information and communication engineering from Sejong University, South Korea, in 2015. From Mar 2015 to Aug 2015, he worked as a Postdoctoral Researcher at Yonsei University, South Korea. Following that, from Sep 2015 to Jun 2016, he worked as an Assistant Professor at Taif University, Kingdom of Saudi Arabia. Currently, he is Professor in the Internet Technology and Data Science Lab (IDLab) of Ghent University and imec. He is the Secretary and a Voting Member of IEEE P1900.8 (active PAR)— Standard for Training, Testing, and Evaluating Machine-Learned Spectrum Awareness Models. He has been involved in several projects such as DARPA Spectrum Collaboration Challenge (SC2), European H2020 (eWINE, WiSHFUL), and ESA (CODYSUN, MRC100). He is currently leading several European and national projects (imec ICON, FWO). Within IDLab-intelligent Wireless Networking (iWINe), he leads the 'AI/ML for Wireless' subgroup. His research interests include machine learning and artificial intelligence for wireless communications and networks, decentralized learning, radio resource management, the Internet of Things, 5G/6G networks, localization, connected healthcare, etc.

Prof. dr. ir. Eli De Poorter is professor at the IDLab research group from Ghent University and imec (https://idlab.technology). He received his master degree in Computer Science Engineering from Ghent University, Belgium, in 2006, his Ph.D. degree in 2011 at the Department of Information Technology at Ghent University, and he became professor at IDLab in 2015. Since 2017, he is also affiliated with the IMEC research institute. His team performs research on wireless communication technologies such as (indoor) localization solutions, wireless IoT solutions, connected healthcare and machine learning for wireless systems. He performs both fundamental and applied research. For his fundamental research he is currently the coordinator of several research projects (SBO, FWO, GOA, etc.) and has over 200 publications in international journals or in the proceedings of international conferences. For his applied research, he collaborates with (Flemish) industry partners to transfer research results to industrial applications, as well as to solve challenging industrial research problems.