### D5.1: Machine learning algorithms development and implementation

**Revision:** v.1.0

<table>
<thead>
<tr>
<th>Work package</th>
<th>WP 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task</td>
<td>Task 5.1</td>
</tr>
<tr>
<td>Due date</td>
<td>30/12/2016</td>
</tr>
<tr>
<td>Submission date</td>
<td>23/12/2016</td>
</tr>
<tr>
<td>Deliverable lead</td>
<td>JSI</td>
</tr>
<tr>
<td>Version</td>
<td>4.1</td>
</tr>
<tr>
<td>Authors</td>
<td>Adnan Bekan (JSI), Klemen Bregar (JSI), Timotej Gale (JSI), Carolina Fortuna (JSI), Tomaz Solc (JSI), Merima Kulin (IMEC), Wei Liu (IMEC), Vincent Sercu (IMEC), Pieter Becue (IMEC), Bart Jooris (IMEC), Eli De Poorter (IMEC), Ingrid Moerman (IMEC), Yi Zhang (TCD), Anatolij Zubow (TUB), Niels Karowski (TUB), Filip Lemic (TUB)</td>
</tr>
<tr>
<td>Reviewers</td>
<td>Jerome Arokkiam (TCD), Adnan Shahid (IMEC)</td>
</tr>
</tbody>
</table>

**Abstract**

This deliverable reports on research enabling the efficient management of dense wireless networks. The work investigates automatizing aspects of wireless networks with the help of machine learning algorithms. Each section of the deliverable reports on a
D5.1: Machine learning algorithms development and implementation

| problem being solved and explains how the solution is included in the corresponding showcase. The provided solutions use both advanced and more traditional machine learning algorithms. An investigation on how to approach feature engineering and class unbalance challenges to finally deliver performant prediction and classification technology is also included. The software tools and datasets developed while carrying out the studies form part of the eWINE Intelligence Toolbox, will be released as open source in the Wireless Testbed Academy GitHub repository and used in the eWINE Grand Challenge. |
| Keywords | machine learning, wireless, classification, recognition, LOS, NLOS, OFDM, LQE |

**Disclaimer**

The information, documentation and figures available in this deliverable, is written by the eWINE (Elastic wireless networking experimentation) – project consortium under EC grant agreement 688116 and does not necessarily reflect the views of the European Commission. The European Commission is not liable for any use that may be made of the information contained herein.

**Copyright notice**

© 2016 - 2018 eWINE Consortium

**Acknowledgment**

This report has received funding from the EC under the grant agreement 688116

---

* R: document, report; DEM: demonstrator, pilot, prototype; DEC: websites, patent fillings, videos, etc.; OTHER; ETHICS: ethics requirement
EXECUTIVE SUMMARY

The eWINE project considers the emerging wireless world in which heterogeneous wireless devices will coexist in time, space and frequency. It investigates how to increase the efficiency of such a system (i.e. reduce transmission losses) through advanced context provisioning, optimization and machine learning algorithms. This deliverable reports on research enabling the efficient management of dense wireless networks. It investigates automatizing aspects of wireless networks with the help of machine learning algorithms. Machine Learning algorithms are able to increase the accuracy of existing technology, replace existing technology with components that enable more flexibility or add completely new components to a wireless system. For instance, an accurate LOS/NLOS classifier has the potential to increase the accuracy of localization technology. An accurate LTE-U duty cycle predictor can enable the development of a better scheduler. Finally, an effective OFDM technology detector can be an advanced building block on top of existing spectrum sensing technology.

All the work reported in this deliverable provides a building block for the three Showcases pursued in the project. For instance, the study of the automatic classification of line of sight and non-line of sight transmissions constitutes an enabling block for Showcase 1. The study of how to automatically recognize OFDM technologies, how to automatically determine the quality of a wireless link and how to predict the performance of a MAC protocol constitute enabling blocks for Showcase 2. Finally, the study of the predictor for the LTE-U duty cycle constitutes an enabling block for Showcase 3.

For these studies, advanced machine learning algorithms such as CNN also referred to as deep learning in the machine learning community, multilayer perceptrons and support vector machines were used alongside more traditional approaches such as J48 decision trees and k-means clustering. An investigation on how to approach feature engineering and class unbalance challenges to finally deliver performant prediction and classification technology is also presented.

The software tools and datasets developed while carrying out the studies form part of the eWINE Intelligence Toolbox, will be released as open source in the Wireless Testbed Academy GitHub repository and used in the eWINE Grand Challenge.
D5.1: Machine learning algorithms development and implementation

## LIST OF ACRONYMS AND ABBREVIATIONS

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ANN</td>
<td>Artificial Neural Network.</td>
</tr>
<tr>
<td>AP</td>
<td>Access Point.</td>
</tr>
<tr>
<td>CFR</td>
<td>Channel Frequency Response.</td>
</tr>
<tr>
<td>CIR</td>
<td>Channel Impulse Response.</td>
</tr>
<tr>
<td>CN</td>
<td>Cognitive networking.</td>
</tr>
<tr>
<td>CNN</td>
<td>Convolutional neural networks.</td>
</tr>
<tr>
<td>COTS</td>
<td>Commercial Off-The-Shelf.</td>
</tr>
<tr>
<td>CPU</td>
<td>Central processing unit.</td>
</tr>
<tr>
<td>CR</td>
<td>Cognitive radio.</td>
</tr>
<tr>
<td>CRN</td>
<td>Cognitive Radio Networking.</td>
</tr>
<tr>
<td>CS</td>
<td>Carrier Sensing.</td>
</tr>
<tr>
<td>CSAT</td>
<td>Carrier Sense Adaptive Transmission.</td>
</tr>
<tr>
<td>CSMA</td>
<td>Carrier sense multiple access.</td>
</tr>
<tr>
<td>CSMA/CA</td>
<td>Carrier sense multiple access with collision avoidance.</td>
</tr>
<tr>
<td>CSVC</td>
<td>C-Support Vector Classifier.</td>
</tr>
<tr>
<td>DSP</td>
<td>Digital Signal Processors.</td>
</tr>
<tr>
<td>ED</td>
<td>Energy Detection.</td>
</tr>
<tr>
<td>eWINE</td>
<td>Elastic wireless networking experimentation.</td>
</tr>
<tr>
<td>FN</td>
<td>False Negatives.</td>
</tr>
<tr>
<td>FP</td>
<td>False Positives.</td>
</tr>
<tr>
<td>FPGA</td>
<td>Field Programmable Gate Arrays.</td>
</tr>
<tr>
<td>GPP</td>
<td>General Purpose Processors.</td>
</tr>
<tr>
<td>HetSNets</td>
<td>Heterogeneous and small cell networks.</td>
</tr>
<tr>
<td>HL</td>
<td>Hidden Layers.</td>
</tr>
<tr>
<td>HW</td>
<td>Hardware.</td>
</tr>
<tr>
<td>IC</td>
<td>Integrated Circuit.</td>
</tr>
<tr>
<td>InPs</td>
<td>Infrastructure Providers.</td>
</tr>
<tr>
<td>IoT</td>
<td>Internet of Things.</td>
</tr>
<tr>
<td>IPI</td>
<td>Inter Packet Interval.</td>
</tr>
<tr>
<td>LOS</td>
<td>Line-of-sight.</td>
</tr>
<tr>
<td>LP</td>
<td>Low-Power.</td>
</tr>
<tr>
<td>LQI</td>
<td>Link Quality Indicator.</td>
</tr>
<tr>
<td>LTE-U</td>
<td>LTE Unlicensed.</td>
</tr>
<tr>
<td>MAC</td>
<td>Maximum Auto-correlation</td>
</tr>
<tr>
<td>Abbreviation</td>
<td>Full Form</td>
</tr>
<tr>
<td>--------------</td>
<td>-----------</td>
</tr>
<tr>
<td>MAC</td>
<td>Media Access Control.</td>
</tr>
<tr>
<td>MLP</td>
<td>Multilayer Perceptron.</td>
</tr>
<tr>
<td>mmWave</td>
<td>Millimeter wave.</td>
</tr>
<tr>
<td>MPC</td>
<td>Multipath Components.</td>
</tr>
<tr>
<td>MPPA</td>
<td>Multi-Purpose Processor Array.</td>
</tr>
<tr>
<td>MPSoc</td>
<td>Multi-Processor Systems-on-Chip.</td>
</tr>
<tr>
<td>MTC</td>
<td>Massive machine type communications.</td>
</tr>
<tr>
<td>NN</td>
<td>Neural Network.</td>
</tr>
<tr>
<td>OFDM</td>
<td>Orthogonal Frequency Division Multiplexing.</td>
</tr>
<tr>
<td>OMAP</td>
<td>Open Multimedia Applications Platform.</td>
</tr>
<tr>
<td>OpenCL</td>
<td>Open Computing Language.</td>
</tr>
<tr>
<td>OpenMP</td>
<td>Open Multi-Processing.</td>
</tr>
<tr>
<td>OTT</td>
<td>Over-the-Top.</td>
</tr>
<tr>
<td>PHY</td>
<td>Physical layer.</td>
</tr>
<tr>
<td>POSIX</td>
<td>Portable Operating System Interface X.</td>
</tr>
<tr>
<td>PRF</td>
<td>Pulse Repetition Frequency.</td>
</tr>
<tr>
<td>PRR</td>
<td>Packet Reception Ratio.</td>
</tr>
<tr>
<td>PTF</td>
<td>Platform.</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality of Service.</td>
</tr>
<tr>
<td>RAN</td>
<td>Radio Access Network.</td>
</tr>
<tr>
<td>RATs</td>
<td>Radio Access Technologies.</td>
</tr>
<tr>
<td>RBF</td>
<td>Radial Basis Function.</td>
</tr>
<tr>
<td>RELU</td>
<td>Rectified Linear Unit.</td>
</tr>
<tr>
<td>RISC</td>
<td>Reduced Instruction Set Computer.</td>
</tr>
<tr>
<td>RMSE</td>
<td>Root Mean Square Error.</td>
</tr>
<tr>
<td>RSSI</td>
<td>Received signal strength indication.</td>
</tr>
<tr>
<td>RTE</td>
<td>Real-Time Embedded.</td>
</tr>
<tr>
<td>SC-FDE</td>
<td>Single-Carrier Frequency-Domain Equalization.</td>
</tr>
<tr>
<td>SDR</td>
<td>Software Defined Radio.</td>
</tr>
<tr>
<td>SME</td>
<td>Small or Medium-sized Enterprise.</td>
</tr>
<tr>
<td>SNR</td>
<td>Signal to Noise Ratio.</td>
</tr>
<tr>
<td>SoC</td>
<td>System on Chip.</td>
</tr>
<tr>
<td>SOTA</td>
<td>State of the Art.</td>
</tr>
<tr>
<td>SP</td>
<td>Service Provider.</td>
</tr>
<tr>
<td>SRD</td>
<td>SRD Short range devices.</td>
</tr>
</tbody>
</table>
D5.1: Machine learning algorithms development and implementation

SRN           SRN Signal-to-noise Ratio.
SVM           Support Vector Machine.
SW            Software.
TN            True Negatives.
TP            True Positives.
UE            User Equipment.
UNB           Ultra Narrow Band.
USRP          Universal Software Radio Peripheral.
UWB           Ultra Wide Band.
VLIW          Very Long Instruction Word.
WRAN          Wireless regional area networks.
WSN           Wireless Sensor Network.
TABLE OF CONTENTS

EXECUTIVE SUMMARY .................................................................................................................. 3
LIST OF ACRONYMS AND ABBREVIATIONS .............................................................................. 4
TABLE OF CONTENTS .................................................................................................................... 7

1 INTRODUCTION .......................................................................................................................... 9
1.1 Supervised vs. Unsupervised vs. Semi-Supervised Learning ...................................................... 9
1.2 Offline vs. Online vs. Active Learning .......................................................................................... 9
1.3 The eWINE Intelligence Toolbox ............................................................................................. 10

2 UWB LOS/NLOS CLASSIFICATION ....................................................................................... 12
2.1 Dataset description .................................................................................................................. 12
2.2 Data exploration ..................................................................................................................... 13
2.3 Data pre-processing ................................................................................................................. 14
2.4 Model description ................................................................................................................... 14
2.4.1 Support Vector Machines (SVM) .......................................................................................... 14
2.4.2 Multilayer Perceptron (MLP) ............................................................................................. 15
2.4.3 Convolutional Neural Networks (CNN) ................................................................................ 15
2.5 Performance evaluation .......................................................................................................... 16
2.6 Conclusions ............................................................................................................................. 17
2.7 Inclusion in Showcase 1 ........................................................................................................... 17

3 DEVELOPMENT OF A MAC LEVEL PERFORMANCE PREDICTOR .................................. 18
3.1 Data collection methodology .................................................................................................. 18
3.2 Dataset description ................................................................................................................ 19
3.3 Algorithm description ............................................................................................................. 20
3.4 Implementation ....................................................................................................................... 22
3.5 Evaluation ............................................................................................................................... 22
3.5.1 Methodology ....................................................................................................................... 22
3.5.2 Results ................................................................................................................................. 24
3.6 Inclusion in Showcase 2 .......................................................................................................... 26

4 LEARNING TO CLASSIFY OFDM SIGNALS TO DIFFERENTIATE MULTIPLE TECHNOLOGIES .................................................................................................................. 29
4.1 RSSI-based signal classification using a machine learning algorithm ..................................... 29
4.1.1 Dataset description ............................................................................................................ 29
4.1.2 Data exploration ................................................................................................................. 30
4.1.3 Algorithm description ......................................................................................................... 31
4.1.4 Implementation .................................................................................................................. 32
4.1.5 Evaluation .......................................................................................................................... 32

© eWINE Consortium 2016-2018
D5.1: Machine learning algorithms development and implementation

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Image-based signal classification using convolutional neural network algorithm</td>
<td>33</td>
</tr>
<tr>
<td>4.2.1</td>
<td>Dataset Description and Data Exploration</td>
<td>33</td>
</tr>
<tr>
<td>4.2.2</td>
<td>Algorithm Description</td>
<td>34</td>
</tr>
<tr>
<td>4.2.3</td>
<td>Implementation</td>
<td>34</td>
</tr>
<tr>
<td>4.2.4</td>
<td>Evaluation</td>
<td>34</td>
</tr>
<tr>
<td>4.3</td>
<td>Inclusion in Showcase 2</td>
<td>34</td>
</tr>
<tr>
<td>5</td>
<td>DEVELOPMENT OF PREDICTOR FOR LINK QUALITY</td>
<td>36</td>
</tr>
<tr>
<td>5.1</td>
<td>Dataset description</td>
<td>36</td>
</tr>
<tr>
<td>5.2</td>
<td>Algorithm description</td>
<td>54</td>
</tr>
<tr>
<td>5.3</td>
<td>Implementation</td>
<td>55</td>
</tr>
<tr>
<td>5.4</td>
<td>Evaluation</td>
<td>56</td>
</tr>
<tr>
<td>5.5</td>
<td>Inclusion in Showcase 2</td>
<td>58</td>
</tr>
<tr>
<td>6</td>
<td>DEVELOPMENT OF PREDICTOR FOR LTE-U DUTY CYCLE</td>
<td>60</td>
</tr>
<tr>
<td>6.1</td>
<td>Background</td>
<td>60</td>
</tr>
<tr>
<td>6.1.1</td>
<td>LTE-U Primer</td>
<td>60</td>
</tr>
<tr>
<td>6.1.2</td>
<td>Understanding the Impact of LTE-U Co-Channel Interference on WiFi</td>
<td>61</td>
</tr>
<tr>
<td>6.2</td>
<td>Problem statement</td>
<td>61</td>
</tr>
<tr>
<td>6.3</td>
<td>System description</td>
<td>63</td>
</tr>
<tr>
<td>6.4</td>
<td>Evaluation</td>
<td>64</td>
</tr>
<tr>
<td>6.4.1</td>
<td>Methodology</td>
<td>65</td>
</tr>
<tr>
<td>6.4.2</td>
<td>Results</td>
<td>65</td>
</tr>
<tr>
<td>6.5</td>
<td>Conclusions &amp; Future Work</td>
<td>66</td>
</tr>
<tr>
<td>6.6</td>
<td>Inclusion in Showcase 3</td>
<td>66</td>
</tr>
<tr>
<td>7</td>
<td>CONCLUSION</td>
<td>67</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>68</td>
</tr>
</tbody>
</table>
1 INTRODUCTION

Machine learning for wireless networks is a mechanism that enables context awareness and intelligence capabilities in different aspects of wireless communication. Over the last years, it has gained popularity due to its success in enhancing network-wide performance (i.e., the Quality of Service, QoS) [1], facilitating intelligent behavior by adapting to complex and dynamically changing (wireless) environments [2] and its ability to add automation for realizing concepts of self-healing and self-optimization [3]. During the past years, different learning approaches have been applied in various wireless networks schemes such as medium access control [4][5], routing [6][7], data aggregation and clustering [8][9], localization [10][11], energy harvesting communication [12], cognitive radio [13][14], etc. These schemes apply to a variety of wireless networks such as: mobile ad hoc networks [15], wireless sensor networks [16], wireless body area networks [17], cognitive radio networks [18][19] and cellular networks [20].

1.1 Supervised vs. Unsupervised vs. Semi-Supervised Learning

Machine learning is categorized by the amount of knowledge or feedback that is given to the learner into supervised, unsupervised or semi-supervised approaches.

Supervised Learning

Supervised learning utilizes predefined inputs and known outputs to build a system model. The set of inputs and outputs forms the labelled training dataset that is used to teach a learning algorithm how to predict future outputs for new inputs that were not part of the training set. Supervised learning algorithms are suitable for wireless network problems where prior knowledge about the environment exists and data can be labelled. For example, predict the location of a mobile node using an algorithm that is trained on signal propagation characteristics (inputs) at known locations (outputs).

Unsupervised Learning

Unsupervised learning algorithms try to find hidden structures in unlabelled data. The learner is provided only with inputs without known outputs, while learning is performed by finding similarities in the input data. These algorithms are suitable for wireless network problems where no prior knowledge about the outcomes exists, or annotating data (labelling) is difficult to realize in practice. For instance, automatic grouping of wireless sensor nodes into clusters based on their current sensed data values and geographical proximity (without knowing a priori the group membership of each node) can be solved using unsupervised learning algorithms.

Semi-Supervised Learning

Several mixes between the supervised and unsupervised learning methods exist and materialize into semi-supervised learning [21]. Semi-supervised learning is used in situations when a small amount of labelled data with a large amount of unlabelled data exists. It has great practical value because it may alleviate the cost of rendering a fully labelled training set, especially in situations where it is infeasible to label all instances.

1.2 Offline vs. Online vs. Active Learning

Learning can be categorized depending on the way the information is given to the learner as either offline or online learning. In offline learning the learner is trained on the entire training data at once, while in online learning the training data becomes available in a sequential order and is used to update the representation of the learner in each iteration. In active learning the model is being updated only occasionally.

Offline Learning

Offline learning is used when the system that is being modeled does not change its properties
D5.1: Machine learning algorithms development and implementation

dynamically. Offline learned models are easy to implement because the models do not have to keep on learning constantly, and they can be easily retrained and redeployed in production. For example, in [6] a learning-based link quality estimator is implemented by deploying an offline trained model into the network stack of Tmote Sky wireless nodes. The model is trained based on measurements about the current status of the wireless channel that are obtained from extensive experiment setups from a wireless testbed.

**Online Learning**

Online learning is useful for problems where training examples arrive one at a time or when due to limited resources it is computationally infeasible to train over the entire dataset. For instance, in [22] a decentralized learning approach for anomaly detection in wireless sensor networks is proposed. The authors concentrate on detection methods that can be applied online (i.e., without the need of an offline learning phase) and that are characterized by a limited computational footprint, so as to accommodate the stringent hardware limitations of wireless sensor network (WSN) nodes.

Another example can be found in [23], where the authors propose an online outlier detection technique that can sequentially update the model and detect measurements that do not conform to the normal behavioral pattern of the sensed data, while maintaining the resource consumption of the network to a minimum.

**Active Learning**

A special form of online learning is active learning where the learner first reasons about which examples would be most useful for training (taking as few examples as possible) and then collects those examples. Active learning has proven to be useful in situations when it is expensive to obtain samples from all variables of interest. Recently, the authors in [24] proposed a novel active learning approach (for graphical model selection problems), where the goal is to optimize the total number of scalar samples obtained by allowing the collection of samples from only subsets of the variables. This technique could for instance alleviate the need for synchronizing a large number of sensors to obtain samples from all the variables involved simultaneously.

### 1.3 The eWINE Intelligence Toolbox

The eWINE intelligence toolbox developed in this project aims to provide algorithms and tools for dense and heterogeneous wireless network optimization as depicted in Figure 1. For each of the three showcases defined in the project (see Deliverable 2.1), we identified optimization problems where machine learning can be valuable. In this deliverable, we report results from solving these problems. The resulting Data collection, processing, model training scripts are part of the accompanying software deliverable D5.2 and will be open sources on the Wireless Testbed Academy GitHub repository. These form part of the Intelligence repository developed in this project.

The Intelligence Toolbox contains algorithms that enable the experimenter to improve localization accuracy by automatically classifying line of sight (LOS) and non-LOS signals (Section 2), predict the performance of MAC protocols for pro-active network tuning and adaptation (Section 3), learning how to automatically recognize different OFDM technologies that could be used by cognitive radios on top of spectrum sensing to be able to recognize more than transmitter presence, but also the type of transmitter (Section 4), link quality classification to determine the fitness of the link and ultimately influence the selection by the MAC or network layer protocols (Section 5) and improve LTE-U and WiFi co-existence by predicting the LTE-U duty cycle (Section 6).

These algorithms solve problems in various showcases as summarized below and detailed through thesections of the deliverable:

---

1 [https://github.com/WirelessTestbedsAcademy](https://github.com/WirelessTestbedsAcademy)
Showcase 1: we study the automatic classification of LOS and non-LOS transmissions which helps improving the accuracy of localization.

Showcase 2: we study how to automatically recognize OFDM technologies, how to automatically determine the quality of a wireless link and how to predict the performance of a MAC protocol with the aim of enabling more efficient communication in dense setups.

Showcase 3: we study a predictor for the LTE-U duty cycle to be able to better schedule transmissions from other technologies.

Figure 1: eWINE Intelligent Toolbox for Cognitive Networking.
2 UWB LOS/NLOS CLASSIFICATION

The performance of range-based localization systems depends on the accuracy of measured or estimated ranges from unknown position to several reference points (anchors). Some algorithms are based on range estimation based on calculating ranges from path-loss model with RSSI measurements [28] but the most accurate ranging algorithms are based on some way of estimating time of flight of a radio signal or a packet (e.g. time of arrival, time difference of arrival, two-way-ranging, etc.) [29]. Ultra-wideband (UWB) radios transmit very short pulses which enables a receiver that it can distinguish between multipath components (MPC) and thus very precisely find the start of a signal or a packet [25]. This property of UWB radios enables that a precise time-of-flight-based ranging can be implemented when the receiver and transmitter are in a line-of-sight (LOS) positions.

In an indoor environment there are different obstacles for signal propagation. There are walls, furniture, electrical installation, steel construction elements, windows, etc. Radio wave emission from transmitting antenna depends on antenna characteristics but is in most cases of sensor nodes and personal devices near-omnidirectional. Energy of a transmitted pulse therefore travels on multiple paths where the shortest path is the direct LOS path. Other MPCs represent pulse energy which has travelled on paths affected by reflections, diffractions and propagation through surrounding environment (e.g. walls, obstacles, etc.). The more challenging is the environment the richer is the MPC information and the bigger is the signal delay spread [26][27]. But in all cases of LOS situations, the shortest direct path component is the strongest component (regarding magnitude) against all the MPCs.

In a NLOS signal propagation between transmitter and a receiver there is no path without obstruction. Every MPC travels through a number of obstacles or around them. The signal propagation speed is slower in materials other than air thus introducing positive bias to the true LOS (geographical) distance (signal propagation time) in any NLOS case. In most NLOS cases the shortest path component is also weaker than MPCs. MPCs usually travel on longer paths with weaker attenuation than signal on shortest path because they travel smaller distances in denser materials than signals on shorter paths [25]. MPCs in NLOS cases represent mostly signal paths travelling around obstacles with reflections.

To prevent big ranging errors correctly detecting NLOS propagation is crucial. In the following work description we propose machine learning approach where input data is raw channel impulse response (CIR) data.

2.1 Dataset description

The dataset was created during a measurement campaign in several stages held in different indoor environments (e.g. living room, garage, office, workshop etc.). For every data sample, the following packet/channel performance indicators were recorded:

- Measured distance
- Channel settings
- First path signal power
- Complete signal power
- Standard deviation of noise
- Maximum noise level
- Number of accumulated preamble symbols
- Complex channel impulse response estimate.

We collect labelled data from 6 indoor locations (rooms). Each data point is labelled with LOS or NLOS. The complete dataset comprises of 36000 samples in 12 files (2 for each indoor environment).
The hardware used for data collection during the campaign was SNPN-UWB board [30] equipped with a DecaWave DWM1000 IEEE 802.15.4-a UWB radio module [31]. The board performing data acquisition was connected to a computer while the other was equipped with a battery for mobility. The communication between Tx-Rx pair took place on channel 2 (3993.6 MHz) with 16 MHz pulse repetition frequency (PRF), bitrate of 6.8 Mbps, and preamble length of 128 symbols. This configuration is suitable for high data rate, low latency and short range communication.

### 2.2 Data exploration

We compute average LOS and NLOS CIR vectors to understand the difference between LOS and NLOS signals. Figure 2a presents the average CIR for LOS conditions. It can be seen that the average normalized magnitude is 1 while Figure 2b presents the average CIR for NLOS conditions and the average normalized magnitude is about 0.5. It can be clearly seen that in our dataset NLOS signals have lower magnitudes in average than LOS signals (most NLOS signals come from longer distances between sensor nodes). We can also observe that the envelope of CIR of average NLOS signal has slower descent (larger delay spread). This confirms our assumption that NLOS signals contain stronger reflections as there are more obstructions in the signal path. Signal path is also longer in most cases of NLOS propagation. Richer MPC content can also be confirmed with CIR in Figure 3b in comparison to CIR in Figure 3a.

![CIR comparison](image_url)
2.3 Data pre-processing

The CIR accumulator data length for our radio settings is 992 samples. There is a great part of accumulator data where there is just noise without information about channel quality. The full CIR accumulator example can be seen in Figure 4. To limit the input vector size used for the machine learning part, the useful information have to be cut out from the full accumulator length. DW1000 UWB radio provides first path CIR start index, which indicates the first detected rising edge where the actual CIR starts. The visual analysis of the data shows that all useful information describing the impulse response is no more than 150 points long, starting from the start index.

The produced feature set (labelled CIR samples with 150 points) needs to be scaled to improve performance of many machine learning algorithms (especially SVM and neural networks) [33]. Data scaling prevents feature vectors with disproportionately large values from dominating other feature vectors. For data standardization, we used StandardScaler from scikit-learn machine learning packet in Python [33].

2.4 Model description

In this section, we introduce the machine learning algorithms used for building and evaluating LOS/NLOS classification models. The algorithms listed in this section are suitable for building complex decision functions from large feature sets as it is the case with our dataset with raw CIR vectors.

2.4.1 Support Vector Machines (SVM)

Support vector machines (SVM) are a popular family of machine learning methods for classification, regression and clustering. They map the input vectors into higher dimensionality space through non-
linear mapping. In this hyper-dimensional space, a decision surface is constructed in a way that ensures high generalization ability of SVM [34].

Let the feature vectors $x_i \in R^n$ of length $n$ in a training set with $l$ instances. Each feature vector is labelled with class labels in $y \in R^l$. C-Support Vector Classifier (C-SVC) solves the following optimization algorithm:

$$
\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i
$$

subject to: $y_i (w^T \phi(x_i) + b) \geq 1 - \xi_i$, 

$\xi_i \geq 0, i = 1,...,l$, 

(1)

where $\phi(x_i)$ maps $x_i$ into a higher-dimensional (N) space and $C > 0$ is the regularization parameter [34]. $w$ and $b$ are linear separators in a N-dimensional feature space and a bias constructed for a set of transformed input vectors [35] respectively.

The kernel functions evaluated in the classification problem were linear kernel function $K(x_i, x_j) = x_i^T x_j$ and radial basis function (RBF) $K(x_i, x_j) = \exp \left(-\gamma \|x_i - x_j\|^2\right), \gamma > 0$.

We used the implementation from scikit-learn [33] which is based on LIBSVM support vector machine library [34].

### 2.4.2 Multilayer Perceptron (MLP)

Multilayer perceptron is an implementation of artificial neural network. All input signals are connected to all neurons in the input layer and all outputs from those neurons are connected to all inputs of neurons in first hidden layer. The term for this type of connection is that a layer is fully connected. Typically, a three-layer perceptron is used where first layer is the input layer, the second layer is the hidden layer and the third layer is the output layer. The shortcoming of multilayer perceptron is high connectivity between neurons that result in over-fitting and high sensitivity to input shifting.

We used the implementation of MLP from scikit-learn machine learning package in Python [33] which is a user friendly and computationally-efficient machine learning library. Because most of the algorithms are actually implemented in C and C++ and Python is used only as a user interface the performance (execution time) is incomparably better than the performance of the algorithms implemented in Weka which. After several iterations of learning and evaluating models with different neural network topologies the best results were given by configuring MLP with 304 neurons in first layer, 150 neurons in second layer and 50 neurons in third layer. We used stochastic gradient descent optimization algorithm for learning error minimization with rectified linear unit activation function (RELU).

### 2.4.3 Convolutional Neural Networks (CNN)

The convolutional neural network [36] is a feed-forward artificial neural network with connectivity pattern between neurons inspired by the organization of animal visual cortex. They are mostly used for image recognition and the visual nature of our CIR data can benefit from using a network of this kind. CNNs are also known to be shift invariant artificial neural networks, which improve performance in case of possible small shifts of input signals.

For implementation of CNN we used TensorFlow machine intelligence library [37] because of its flexibility, scalability and large support and development community behind it.

The configuration of our CNN is:
• Convolutional layer 1: patch width = 4, depth = 10
• Rectified Linear Unit (RELU) layer
• Convolutional layer 2: patch width = 5, depth = 20
• RELU layer
• Pooling layer 1: stride = 2
• Convolutional layer 3: patch width = 4, depth = 20
• RELU layer
• Convolutional layer 4: patch width = 4, depth = 40
• RELU layer
• Pooling layer 2: stride = 2
• Fully connected layer: 128 neurons
• RELU layer
• Dropout layer: keep probability = 0.5
• Readout layer: algorithm = softmax

2.5 Performance evaluation

To evaluate the performance of all machine learning algorithms, we need to compute some standard metrics based on the confusion matrix calculated from classification results based on evaluation of the data set which is different from data set used during learning process. In the confusion matrix rows represent actual classes where the samples (instances) belong to and columns represent to which classes instances were assigned to during classification process. We have true positive (TP), true negative (TN), false positive (FP) and false negative (FN) assignments.

Some widely used classification performance metrics are:

- Accuracy: \( \frac{TP + TN}{TP + TN + FP + FN} \) (gives percentage of correctly classified instances).
- Precision: \( \frac{TP}{TP + FP} \) (represents percentage of correctly classified NLOS instances within all instances assigned as NLOS).
- Sensitivity: \( \frac{TP}{TP + FN} \) (true positive rate or fraction of correctly classified instances within NLOS class).
- F1 or harmonic mean of precision and sensitivity: \( \frac{2 \times \text{precision} \times \text{sensitivity}}{\text{precision} + \text{sensitivity}} \)

Table 1 presents the classification results on raw CIR data. As expected, the worst performer between four methods is SVM classifier with a linear kernel with overall accuracy of 81.4 % but with very high sensitivity of 88.6 %. Sensitivity tells us how many NLOS conditions were recognized as NLOS within true NLOS class. In other words, it tells us how good the classifier is at correctly detecting NLOS conditions. But the drawback of the model built with SVM with linear kernel in our case is the worst precision of all evaluated methods. Only 77.5 % of instances recognized as NLOS are true NLOS.

Second worst of evaluated algorithms goes MLP with accuracy of 82.8 % and precision of 81.8 %. This one is slightly more balanced but still worse than the better two. SVM with RBF kernel gets very good precision with better accuracy level (83.6 %) and the best overall performer is also the most complex model built with deep convolutional neural network.

Table 1 Classification results

<table>
<thead>
<tr>
<th></th>
<th>SVM (linear kernel)</th>
<th>SVM (RBF kernel)</th>
<th>MLP</th>
<th>CNN</th>
</tr>
</thead>
</table>
True positive  |  6401 |  6232 |  6051 |  6492  
True negative |  5316 |  5813 |  5878 |  5921  
False positive |  1858 |  1420 |  1349 |  1225  
False negative |   825 |   935 |  1122 |   762  
Accuracy        | 81.4 % | 83.6 % | 82.8 % | 86.2 %  
Precision       | 77.5 % | 81.4 % | 81.8 % | 84.1 %  
Sensitivity     | 88.6 % | 86.9 % | 84.4 % | 89.5 %  
F1              | 82.7 % | 84.1 % | 83.0 % | 86.7 %

### 2.6 Conclusions

The results of the classification with different machine learning algorithms promise accuracies close to 90%. One of reasons for lower values of accuracies is that dataset includes reasonably great part of measurements where differences between CIRs of LOS situations are similar to CIRs of NLOS situations. LOS condition can be similar to NLOS, when nodes are in LOS condition but at the same time very close to NLOS condition. They can be far apart and all the reflections and diffractions will come into account.

On the other hand, NLOS condition can be very similar to LOS condition. When nodes are close to each other and there is an obstacle in the way between radios also there are not so many reflections from surrounding environment as they would be in case of NLOS channel of nodes far apart. Signal attenuation is also low compared to NLOS condition where nodes are far apart.

The best channel model resulting in this work was used as channel classifier in other work reported in WP3.1 in section 2.3.2.1.6 Indoor Localization with Multilateration. The classifier was used to filter range measurements with NLOS channel conditions to improve localization accuracy.

### 2.7 Inclusion in Showcase 1

The LOS/NLOS channel model produced in this work will be used in Showcase 1 for localization accuracy improvement. Signal paths in NLOS channel are typically longer than paths in LOS channel. Radio waves propagate slower through hard materials than through air so the time of flight is longer. Time of flight is used as a measurement for signal path distance using speed of light as the speed of a signal. Because of slower propagation through materials other than air a range error is introduced which introduces localization error. Our approach to mitigation of localization error caused by NLOS range conditions is filtering NLOS ranges out of all measured ranges.
3 DEVELOPMENT OF A MAC LEVEL PERFORMANCE PREDICTOR

Wireless networks are notoriously unpredictable in nature in terms of QoS provisioning with frequent changes of communication quality. From a networking perspective, the most relevant aspect of wireless communication quality is the packet delivery performance and its dependency on the current wireless and physical environment [62]. In particular, for energy-constrained devices such as the battery-powered wireless sensor networks (WSN), one of the main goals is to reduce the number of radio transmissions and to simultaneously increase packet delivery, where the packet reception rate and the throughput are directly with the energy consumption. Efficient performance estimation in wireless networks is crucial as it can reduce the energy consumption by alleviating situations with cumulative failing transmissions (reflected by high packet loss) in a prescriptive manner, as they cause unnecessary energy consumption.

Accurate MAC layer performance estimation is a challenging task due to the notoriously dynamic and unpredictable wireless environment. On the network layer, performance is directly dependent on several environmental characteristics such as interference, traffic load, number of wireless nodes that might potentially contend for the same channel, etc. These factors will implicitly impact the performance of data acquisition in various applications of the currently growing the Internet of Things. One approach to improve the performance of wireless networks is to apply data-driven machine learning methods [63] that can learn the behaviour of the MAC layer based on historical information in order to make predictions that can facilitate selecting the optimal network configuration and/or MAC protocol.

In this study, we aim to propose a learning mechanism for selecting the MAC including CSMA, TDMA and TSCH in various environments. More specifically, the performance of the proposed learning mechanism is evaluated by various measures including packet loss, interference, etc.

3.1 Data collection methodology

In order to study the MAC layer performance, extensive experiments have been performed on the wilab2 testbed facility in Ghent\(^2\). The impact of different network topologies in varying environmental conditions and under different traffic load scenarios is evaluated. The experiments are performed using IEEE 802.15.4 RM090 sensor nodes. The basic topology is presented on Figure 5.

\(^2\) http://wilab2.ilabt.iminds.be/
The setup consists of up to 28 wireless nodes. The basic experiment consists of a variable number of sender nodes and one receiving node, i.e. sink node. All nodes run a CSMA/CA MAC protocol that is developed in the TAISC framework [64]. They have been configured to periodically generate a 100B message to a single receiver located in the centre of the topology. The transmission power is set to the maximum, i.e. 5dBm, to ensure that all nodes are in communication range.

To facilitate the experimental control and data collection from the nodes a WiSHFUL (1.0) UPI global control program [65] has been implemented. The global control program sets up the MAC and application level parameters (e.g. MAC protocol, packet size…) on all nodes and controls the duration for monitoring the measurements. During the experiment the density and network load has been controlled, while interference has been introduced by a synthetic interferer with an interference duty cycle of 20%. The sink node collects statistics for each received packet and forwards them to the global controller for storage (i.e. event-based monitoring).

The following traffic load setups have been investigated: 1pckt/2s, 1pckt/s, 2pckts/s, 4pckts/s, 8pckts/s, 16pckts/s and 64pckts/s. While some of the traffic load values are common for typical wireless sensor network applications, others have been selected for experimental purposes and the simulation of a burst traffic scenario.

Interference due to transmission from other wireless technologies has been simulated by generating an interference pattern with a duty cycle of 20% from a USRP N210, i.e. by transmitting a modulated carrier for 2 ms, followed by an 8 ms idle period (repeating this over time).

### 3.2 Dataset description

In order to obtain a representative population of observations for predicting the MAC layer performance several traces of MAC level statistics have been captured during the experiments [66]. The captured raw data consists of: Sender ID, 802.15.4 Sequencing Number, Topology (Node density), Interference Indication, and Traffic Load.

Extensive experiments have been performed? for topologies with node densities 3, 6, 9, 12, 15, 18, 21, 24, and 26, while a smaller set of experiments has been performed with the remaining node densities. For most of the setup configurations data was collected for a duration of? 10minutes, resulting in a total dataset of ~5hours. A histogram of measurements including node density, traffic load and interference from different setups can be seen on Figure 6.
In short, the histograms show that most of the data was collected for scenarios with the aforementioned node densities, with higher traffic loads, and without interference. A smaller sample of data was gathered for setups with additional interference and with low traffic load (less packets are generated in scenarios with lower traffic load).

### 3.3 Algorithm description

Several learning algorithms have been trained and evaluated for the MAC performance predictor. Figure 7 shows a flowchart that describes the process of creating the MAC performance predictor model.
The raw data consists of per-packet MAC-level reports and is the input for the feature extraction module. The feature extraction module generates the following feature vectors as output [52]:

\[ x^{(i)} = [d, IPI, rP, errP]^T, \]  

(2)

where:

- \( d \) is the total number of neighboring nodes, within a given time frame,
- \( IPI \) is the inter-packet-interval,
- \( rP \) is the number of received packets,
- \( errP \) is the number of erroneous packets/frames.

The corresponding communication reliability in terms of packet loss rate is defined by,

\[ y^{(i)} = plr. \]  

(3)

Those are representative features for predicting the performance of a MAC protocol in terms of overall communication reliability. The density, \( d \), is a good indicator about the number of contending nodes and potential intra-technology interference. The \( IPI \) is a good feature for reasoning about the current application demand and traffic load. Finally, last two features, \( rP \) and \( errP \), are representative for inferring about the interference level and congestion in the network.

Based on the extracted feature vectors a training set is created that is used to train several machine learning algorithms. The model that showed best performance with a particular parameter configuration is selected and retrained on the full training set and gives the final model as result.

The machine learning model for MAC performance prediction and can be easily exported from standard machine learning toolboxes/libraries (e.g. Weka, scikit-learn…).
3.4 Implementation

The proposed algorithm for MAC performance estimation consists of the following components developed in Python:

- Component for loading the data in a standard data structure.
- Feature extraction component. The goal of this module is to turn the raw MAC-level measurements into features/attributes that can be used for training a machine learning algorithm.
- A trained machine learning model. The goal of this component is to predict the MAC performance in new previously unseen environmental conditions, based on a structured representation of the raw measurements obtained through the UPI as input.

3.5 Evaluation

3.5.1 Methodology

The MAC performance prediction problem can be translated into a data mining regression problem. Regression is a data mining method that is suitable for problems that aim to predict a real-valued output variable. It is a supervised learning method, which models (i.e., fits) a set of known inputs (i.e., explanatory or independent variables) and corresponding outputs (i.e., dependent variable) with the most suitable mathematical representation (i.e., function or model) [62].

As the distribution of the process that has generated the analysed sample is a priori unknown, we trained and tested both linear and non-linear regression machine learning algorithms. The learning algorithms were trained and validated on the dataset collected from the experiments described in Section 3.1. We used the following machine learning algorithms: linear regression, regression trees and neural networks.

Linear regression

Linear regression is a technique for modeling the relationship between the input \( x \) and output variable \( y \) so that the output is a linear combination of the input variables (dependent variable).

\[
y(x) = \theta_0 + \theta_1 x_1 + \ldots + \theta_n x_n = \theta_0 + \sum_{i=1}^{n} \theta_i x_i
\]

(4)

We used the linear regression implementation available in Weka.

Regression trees

A regression tree is a tree-based learning algorithms that is used to predict a variable that takes continuous or ordered values. Regression trees is a supervised learning algorithm that creates a tree-like graph or model that represents the possible outcomes or consequences of using certain input values. The tree consists of one root node, internal nodes called decision nodes which test its input against a learned expression, and leaf nodes which correspond to a final class or decision. The learning tree can be used to derive simple decision rules that can be used for decision problems by starting at the root node and moving through the tree until a leaf node is reached where a prediction outcome is assigned. We used the M5P regression tree algorithm implementation available in Weka.

Neural networks
Neural Networks (NN) or artificial neural networks (ANN) is a supervised learning algorithm inspired on the working of the brain, that is typically used to derive complex, non-linear decision boundaries for building a classification model, but are also suitable for training regression models when the goal is to predict real-valued outputs. Neural networks are known for their ability to identify complex trends and detect complex non-linear relationships among the input variables at the cost of higher computational burden. A neural network model consists of one input, a number of hidden layers and one output layer. The input layer corresponds to the input data variables. Each hidden layer consists of a number of processing elements called neurons that process its inputs (the data from the previous layer) using an activation or transfer function that translates the input signals to an output signal. We used the multilayer perceptron implementation available in Weka to train the neural network.

To evaluate and select the best performing machine learning model, a $k$-fold cross validation was used. In particular, a 5-fold cross-validation algorithms is used for model selection and tuning the hyperparameters of the aforementioned learning algorithms.

Figure 17 explains the $k$-fold cross-validation algorithm [63]. As it can be seen, the dataset is first permuted and split into five folds. Then, cross-validation is run through five rounds. In each round, one of the folds is kept for validation while the others are used for training. The validation error, $c_1$, is calculated and used as an estimate of the prediction error for that round. At the end, the average error over all folds is computed as:

$$\hat{\epsilon} = \frac{1}{k} \sum_{i=1}^{k} \hat{\epsilon}_i$$  \hspace{1cm} (5)

The selected performance metrics for evaluating the regression models are the root mean squared error (RMSE):

$$RMSE = \frac{1}{N} \sum_{i=1}^{N} (\hat{y}_i - y_i)^2$$  \hspace{1cm} (6)

where, $y_i$ are the real outcome, while $\hat{y}_i$ the estimations, and the correlation coefficient:

$$\rho_{y, \hat{y}} = \frac{\text{cov}(y, \hat{y})}{\sigma_y \sigma_{\hat{y}}}$$  \hspace{1cm} (7)
3.5.2 Results

Figure 8 and Figure 9 present the correlation coefficient and root mean squared error (RMSE) for several trained regression models for predicting the packet delivery, \( plr \). The performance was evaluated with regard to the observation time window, spanning from 5 s to 60 s, on the \( x \)-axes. Observations are measurements collected from the wireless network as explained on Figure 20 in following Section 4.5. The performance of models of three machine learning algorithms trained on the same training set is illustrated. The choice of the observation time window for measurements monitoring has to be selected with regard to a meaningful period for checking the state of the network and selecting a MAC protocol, i.e. selecting a MAC protocol every 10s makes more sense than selecting a new MAC every 10ms.

Figure 8 Prediction performance for linear regression, regression trees and neural networks

It can be seen from Figure 8 and Figure 9 that the neural network model showed best performance over a variety of observation time windows. In particular, a neural network with 10 hidden layers (HL), 2000 training iterations, and a learning rate of \( \alpha = 0.1 \), captured best the underlying distribution of the data and as such was selected as the model that will be used for making accurate future MAC performance predictions.
In order to estimate how well the trained neural network model generalizes to previously unseen data, a separate new set of experiments has been executed with a shorter duration of monitoring (around 8 hours of experimentation). From the new measurements a test set has been extracted. The goal of this dataset is to find the test error, i.e. the estimate of the prediction error of the model in the future. Figure 19 presents results for the model’s accuracy on the test set, i.e. how well the model predicts $plr$ on new instances that were not part of the training set that was collected as explained in Section 4.1.
As it can be seen from Figure 19, the actual and the values predicted by the model are close to each other, suggesting that the neural network model will make good prediction on the MAC performance in a dynamic wireless network with changing environmental properties. The integration of the model in a real system setup is explained in the following section.

### 3.6 Inclusion in Showcase 2

In the last decade, there have been many research efforts in designing an optimal MAC protocol for wireless sensor networks. Typically, each of these MAC protocols were designed to optimally perform for a target environment and application demands. They tend to either achieve low latency, power consumption, high throughput or robustness to interference. None of the existing designs can deliver optimal performance in varying conditions and changing network scale.

For instance, a CSMA/CA MAC protocol can achieve low latency in low data rate applications, while in high-data rate applications and under high interference it might significantly underperform due to too many channel contentions. On the other hand, a time-slotted channel hopping (TSCH) MAC can deliver high throughput in high-data-rate applications even under interfering transmission from other technologies by avoiding channel contentions and changing the central frequency of the operating channel. However, the channel hopping mechanism that is performed in each subsequent time slot may downgrade the performance in another dimension.

It is clear that one MAC cannot meet the challenges of the dynamic wireless environment, the changing network traffic demands, and changing network scale.

Figure 20 presents the conceptual system architecture to accomplish a cognitive loop for cognitive MAC selection [52]. There are two main components, the:

- **Sensor network** - a set of wireless nodes that generate information and are capable of reconfiguring its transmission parameters at runtime.
- **Global controller** - the central entity that collects and uses information from the wireless nodes to predict the MAC-level performance. Based on the predictions, it dynamically decides...
how to configure the MAC layer so as to improve the overall network performance (e.g. to cope with cross-technology interference it may decide to configure a more interference robust MAC protocol, e.g. TSCH). Finally, it disseminates the new configuration to the nodes.

This cognitive loop in Figure 20 can be realize in two approaches.

**Approach 1**

To meet the aforementioned challenges, the network has to intelligently accommodate to large amounts of connected devices and avoid interference from other technologies using a user-defined threshold. Figure 21-a illustrates the proposed approach.

![Figure 21-a Showcase MAC performance estimator](image)

The showcase presents a machine learning based MAC performance predictor, which can be used to design a new intelligent MAC layer that can detect a poorly behaving MAC protocol (e.g. CSMA) by predicting its performance in the future. This may be the basis for more advanced utilization of the shared medium in hyper-dense heterogeneous wireless networks which can improve the overall performance. For instance, an alternative MAC (e.g. TSCH) can be used in case the current operating MAC is performing poorly. The decision will be propagated through the Decision/Optimization interface.

**Approach 2**

A slightly modified version of the previous approach can take into account dynamic changes of the wireless environment and dynamically adapt the performance thresholds used for decision making. Figure 11-b presents this idea. Instead of statically providing the thresholds as in approach 1, approach 2 uses additional mining algorithms to find the boundaries through learning.

![Figure 11-b Machine learning-based MAC performance estimator with dynamic threshold selection](image)

This work is ongoing and the following steps are planned:

- Collecting data using WiSHFUL UPI v2.0 (initial tests showed an increase in throughput compared to showcases using WiSHFUL UPI v1.0)
D5.1: Machine learning algorithms development and implementation

- Train more machine learning algorithms
- Feature engineering
- Execute experiments for another MAC protocols, e.g. TSCH
4 LEARNING TO CLASSIFY OFDM SIGNALS TO DIFFERENTIATE MULTIPLE TECHNOLOGIES

OFDM is used in a number of systems including LTE, WiFi, WiMAX, DVB and DAB. Driven by the fast growth of wireless communication, the trend of sharing spectrum among heterogeneous technologies which uses OFDMA as its signal transmission mode has become increasingly dominant. Hence, identifying concurrent technologies is an important step towards efficient spectrum sharing. However, due to the complexity of recognition algorithms, communication systems capable of recognizing signals other than its own type are rare.

The literature gives several examples where machine learning algorithm is used to extract features from OFDM signals belonging to single technology. Rehman et al. [67] present machine learning as a tool to provide RF fingerprinting by extracting features (number of paths, path gains, etc.) in the wireless channel under several multipath conditions. Sen et al. [68] provide an experimental perspective in identifying and classifying Channel Frequency Response (CFR), using the machine learning algorithm PinLoc, in order to 'spot localize' the users using WiFi. PinLoc extracts the CFR from the signals obtained using Intel 5330 cards, which support the WiFi frequencies (2.4 or 5GHz). Huang et al. [69] present the use of machine learning in identifying hand-gesture behavior in the 802.11a-based wireless environment. Luo et al. [70] use a Q-learning-based machine learning technique to select the most suitable channel and power values (those which improve the average channel capacity in the system) to be assigned for device-to-device communication in LTE, in the presence of traditional uplink in LTE; however, the evaluations are limited to simulation-based analysis.

The above examples use the single-technology-based OFDM signals to be the input of the machine learning algorithm to extract specific features of the signals. However, using the machine learning algorithms to distinguish OFDM signals from multiple technologies remains an open area of research. To perform efficient classification of signals from multiple technologies: first the real and imaginary (I&Q) components of the OFDM signals transmitted from known technologies are acquired; these I and Q samples are then converted into a base format, which is used as the input for the machine learning algorithm to extract features; finally, the extracted features and the learning algorithm are used together to identify features from the real-time OFDM signals, so that the transmission technology associated with the OFDM signals can be identified with reasonable accuracy levels (this is called the real-time classification phase). Such usage of application of machine learning algorithms, which are generally classified as ‘supervised’ learning algorithms, are used in the following two distinct ways, to identify OFDM signals from multiple technologies:

1. RSSI based signal classification using a machine learning algorithm
2. Image-based signal classification using convolutional neural network algorithm

4.1 RSSI-based signal classification using a machine learning algorithm

4.1.1 Dataset description

RSSI can be obtained from variety of approaches. Since wireless chipsets, which calculates RSSI based upon the individual received packets, RSSI values are technology or device specific. Here, RSSI values are calculated as the sum of the squared magnitude of samples in logarithmic scale and is represented as:

$$y[n] = 10 \times \log_{10} \left( \sum_{i=1}^{N} x^2[i] \right)$$  \hspace{1cm} (8)
The probability density function of RSSI is approximated by normalized histogram.

A set of experiments are performed to explore how RSSI can be used to characterize real-life signals. The experiment target on three representative signals: 1) the IEEE compliant Wi-Fi signal, 2) the downlink signal of the Long-term evaluation (LTE) technology and 3) Digital Video Broadcasting-Terrestrial (DVB-T) signals. The concern of considering the afore-mentioned technologies is that they are operating in unlicensed band and compete with each other in various aspects. Although LTE was traditionally designed for operating in licensed band, a recent enhancement of LTE i.e., LTE-U boosts the performance of LTE via operation in 5GHz ISM band. In this regard, it is indeed of critical importance for identifying the coexisting technologies so that the overall performance is enhanced by taking intelligent decisions including channel assignment, power allocation, interference management, etc.

The characteristics of the signals are described below:

**Wi-Fi (IEEE 802.11a/g):** Signals transmitted in random bursts, modulated with constant amount of carriers;

**LTE:** Signal transmitted in very fine and regular intervals, modulated with constant amount of carriers;

**DVB-T:** Signals transmitted continuously and modulated with constant amount of carriers

More precisely, the Wi-Fi signal is captured in an office environment, including two access points at 5540 MHz (channel 108), and on average 20 associated work stations. The LTE signal is captured from a nearby base station, operating in FDD mode at 806 MHz, around the Ghent area of Belgium. Finally, the DVB-T signal is collected from the local TV broadcasting station at 482 MHz.

An Anritsu MS 2690A spectrum analyser [61] is used to capture samples of each of the aforementioned signal types. The samples are collected at the rate of 10 MHz for a duration of 1 second. The RSSI is calculated using Equation 1 for \( N = 200 \). In total 50 k RSSI values are computed.

### 4.1.2 Data exploration

The first row of Figure 12 contains the normalized RSSI histograms of the three selected technologies, and the spectrograms are shown at the corresponding position of the second row.

![Figure 12 The normalized histogram and spectrogram.](image-url)
As expected, the Wi-Fi signal appears as short and random bursts in the spectrogram. Since the signals belong to multiple stations and AP’s, the histogram contains several peaks between -50 dB and -75 dB, corresponding to signals transmitted by nearby and remote stations respectively. The LTE signal has the most versatile spectrogram, which is primarily occupied by the frequently occurred reference signal in the idle resource blocks. The 1 MHz wide signal occurring every 5 ms at the center of the LTE band is the synchronization sequence. The remaining parts of the spectrogram with higher intensity are the resource blocks active for data transmission. The histogram of the LTE signal contains multiple peaks spreading out from -80 dB to -55 dB. The DVB-T’s spectrogram shows that it is highly stable in both time and frequency domain. This is because the current DVB-T standard only allows continuous transmission, and does not contain flexibility in carrier allocation. Therefore, its histogram simply matches the characteristic of Gaussian distribution — only one narrow peak is present.

All measurements are conducted in an office building of 12x80m. To increase the diversity of signal strength, the measurement locations are placed on the north, east, and west side of the building respectively. On each day and at each location, 10 traces are collected per technology. A trace contains 1×106 IQ samples, obtained by USRP for a duration of 1 second, at the ADC sample rate of 1 MHz.

4.1.3 Algorithm description

The algorithm for signal classification is summarized in the pseudo code in Figure 13. First, Wi-Fi is recognized if the RSSI contains large amount of noise and has sufficient standard deviation, the latter condition is used to distinguish Wi-Fi signal from noise. Next, LTE signal is identified when the standard deviation is above certain threshold. Then, when the previous condition is not met, and the number of peaks in the RSSI histogram is below certain threshold, the signal is considered to be DVB-T; otherwise it is classified as unknown.

<table>
<thead>
<tr>
<th>Algorithm 1 RSSI-based technology identification</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Input:</strong> $\vec{R}$ // a vector of RSSI measurements</td>
</tr>
<tr>
<td><strong>Output:</strong> sig // the identified signal type</td>
</tr>
<tr>
<td><strong>Variables:</strong></td>
</tr>
<tr>
<td>$\vec{H}$ ← Histogram($\vec{R}$)</td>
</tr>
<tr>
<td>stddev ← StdDeviation($\vec{H}$)</td>
</tr>
<tr>
<td>noiseloc ← LocOfLeftmostPeak($\vec{H}$)</td>
</tr>
<tr>
<td>noisepeak ← AmpOfLeftmostPeak($\vec{H}$)</td>
</tr>
<tr>
<td>npks ← TotalNumOfPeaks($\vec{H}$)</td>
</tr>
<tr>
<td><strong>Function:</strong></td>
</tr>
<tr>
<td>if noiseloc ≤ THR_NL &amp; noisepeak ≥ THR_NP then</td>
</tr>
<tr>
<td>if stddev ≥ Wi-Fi.mindev then</td>
</tr>
<tr>
<td>sig ← Wi-Fi</td>
</tr>
<tr>
<td>else</td>
</tr>
</tbody>
</table>
Figure 13 Algorithm for OFDM signal classification

The algorithm involves several thresholds in the decision-making process, which are determined as follows:

- **THR NL** is the upper bound of average noise level, obtained by the maximum ‘noise loc’ in the collected Wi-Fi traces plus the standard deviation of the ‘noise loc’.
- **THR NP** determines the minimal amount of noise present in the Wi-Fi’s RSSI measurements. It is calculated by the smallest ‘noise peak’ minus the standard deviation of the noise peaks in the Wi-Fi’s RSSI measurements.
- **Wi-Fi.mindev** denotes the minimum standard deviation among the collected RSSI measurements of Wi-Fi, which is used to differentiate Wi-Fi from noise.
- **(LTE.mindev + DVB-T.maxdev )/2** denotes the medium of the minimum and the maximum standard deviation of LTE and DVB-T’s RSSI measurements, respectively. It is used to differentiate LTE and DVB-T signal.
- **DVB-T.maxpks** denotes the maximum number of peaks in the histograms of the RSSI measurements of DVB-T. It is used to exclude unknown signals from the DVB-T signals.

### 4.1.4 Implementation

The analysis of the raw data which is in the form of I and Q samples is carried in MATLAB. The classification algorithm presented above is also implemented in MATLAB and the analysis is carried out in various respects presented in the next section.

### 4.1.5 Evaluation

First, the performance for \( N = 20 \), during the observation time of 1 second, is analyzed. Then we extend the evaluation for \( N \in [20,320] \), incremented at the step of 20. In our solution, large \( N \) corresponds to a longer average interval, hence lower update rate of the RSSI samples. The analysis for variable \( N \) is important, as developers on constrained devices usually do not have direct access to the raw IQ samples, RSSI is provided by accessing a register or other types of interface towards hardware modules, which have a limited access rate. Also, how often RSSI needs to be updated, has a strong impact on the feasibility of implementing the solution on small scale devices. Additionally, we also evaluate the impact of the total observation time — the time interval during which the RSSI series are derived, denoted as \( T \) — on the detection accuracy. This is helpful to explore the minimum waiting time needed for the algorithm to achieve a given recognition accuracy.
We use 3-fold cross-validation to evaluate the basic algorithm, for all selected N and T settings. In each round of the validation, the RSSI traces of a given technology are divided into two parts: 70% of the data are used to derive thresholds used in the algorithm, while the remaining 30% are used to validate if the signal is correctly classified. This process is repeated three times, each time different portions of the traces are used for training and validation, respectively. The recognition results are averaged over three rounds, and presented in the form of a confusion matrix in Table 2, where a row represents the actual technology type of a given RSSI trace, and a column represents a technology type determined by the algorithm. For all three technologies, the probability of true positive in the confusion matrix is above 90%, which proves that the predicted technology types are highly consistent with the actual ones. All RSSI traces of LTE signals are correctly classified, whereas 1.85% of the DVB-T traces are falsely determined as LTE. This indicates that the threshold of the standard deviation to distinguish LTE and DVB-T could be improved to reduce the false positive detection of LTE.

### 4.2 Image-based signal classification using convolutional neural network algorithm

In this work a CNN algorithm is used to classify the OFDM signals transmitted in LTE and WiFi. The OFDM signals, after being converted into images (base format) are used as input to the CNN algorithm, to be classified into the appropriate technology being used. To efficiently classify the images into its original technology, the algorithm, in its training-phase, obtains a list of features from the images (with the know technology) so that the same features are used for identifying the technology behind the images belonging to signals obtained in the real-time classification phase. Our experiment is designed to investigate both the adaptive nature of OFDM signal transmission/reception by the USRPs and the efficiency of the deep learning algorithm to classify the OFDM signals based on the spectrogram images of the signals.

#### 4.2.1 Dataset Description and Data Exploration

In this experiment, a set of images (base format) are first generated from the I&Q samples captured from the technology and then used as an input for the CNN algorithm. In the training phase, the I&Q samples are collected for a signal duration of $T_D$ seconds, from the commercial deployment of LTE and WiFi, individually. For robustness of the feature extraction in the training phase and signal classification in the real-time classification phase, each OFDM signal is captured for the duration of $n*T_D$ seconds where $n$ is the number of samples collected for each technology. Each image, which is a conversion of the I&Q samples of duration $T_D$, will be used by the CNN algorithm to extract unique features belonging to the individual technologies. In the real-time classification phase, the I&Q samples from an OFDM signal captured for a duration of $T_D$ seconds, are first converted into the base format (images) before being used by the CNN algorithm to identify the technology associated with the OFDM signals.
4.2.2 Algorithm Description

For image processing applications, CNN is the most commonly used class of Deep Neural Networks. CNN algorithms consist of a set of convolutional layers for extracting features from the input images. In the training stage, CNNs learn image representations that become more abstract along the processing hierarchy. These abstract representations are used to perform classification, regression, etc. based on the application implementation [71].

4.2.3 Implementation

We plan to use the USRP hardware (model – Ettus B210) and the open-source SDR platform, gnuRadio, to capture the I&Q samples from commercial sources of LTE and WiFi. USRPs, with their (dynamically configurable) ability to support a large frequency/spectrum range and the ability to integrate with open-source SDR platforms, are able to transmit and receive OFDM signals under a range of frequencies. Hence, by integrating SDR and CNN algorithms in the USRP-based experimental evaluation, we have the advantage of characterizing OFDM signal features and improving the wireless transmission performance in a more dynamic (spectrum range) manner.

4.2.4 Evaluation

We will perform evaluation in the following manner to accurately identify the technology associated with the OFDM signals captured in section 4.2.1:

- During the training phase, base-format (image) converted I&Q sample set of both LTE and WiFi will be used individually for collecting a set of unique features ($F_L$ for LTE and $F_w$ for WiFi) as identified by the CNN algorithm.
- These unique features ($F_L$ and $F_w$) will be stored in a central (accessible by both the open source SDR platform and the CNN algorithm) storage space, along with the identification of the corresponding technology.
- During the real-time classification phase, the I&Q samples of new OFDM signal (let us assume of technology A) will be captured at the USRP for a duration of $T_D$ seconds; the I&Q samples, after being converted to its base-format (image), will be analyzed by the CNN algorithm so that the unique features ($F_A$) associated with the base-format of the OFDM signal of technology A can be identified.
- The CNN will finally compare $F_A$ against $F_L$ and $F_w$ individually to identify the technical (probabilistic) closeness of the OFDM signal of technology A to LTE and WiFi as $P_{A,L}$ and $P_{A,W}$ respectively (with $P_{A,L} + P_{A,W} = 1$).

4.3 Inclusion in Showcase 2

Due to the quick growth of wireless communications, radio spectrum are either already claimed by licensed users, or heavily loaded by radio applications in the unlicensed bands. The shortage of spectrum resource has become a key limitation of wireless communications. One way to address the spectrum scarcity is to improve the efficiency of the underutilized licensed spectrum by allowing dynamic spectrum access, by means of vertical and horizontal sharing schemes: vertical spectrum sharing refers to the opportunistic access of licensed spectrum without compromising the incumbents’ communication quality, while horizontal spectrum sharing refers to the access of unlicensed spectrum by multiple technologies with equal privileges.

Competition for spectrum sharing in the ISM band is on a steep rise: several technologies (e.g., Wi-Fi, Zigbee, Bluetooth) are already sharing the ISM bands in the horizontal approach while many new licensed technologies, such as LTE-U are also considering to operate in the unlicensed bands; in the near future, communication above 60 GHz is foreseen to meet the throughput demand of the next
D5.1: Machine learning algorithms development and implementation

generation wireless networks. With numerous technologies competing for limited spectrum in the ISM band, hyper dense deployment of small cells will be necessary to compensate for higher path loss in these frequency ranges. In this scenario, exclusive frequency assignment based on technologies is beyond feasibility, thus, the need for efficient and autonomous spectrum sharing. Using effective machine learning techniques to classify the types of technologies that compete for dynamic spectrum in the ISM band, by intelligently identifying unique features associated with the technologies can lead towards efficient spectrum utilization among several competing technologies.
5 DEVELOPMENT OF PREDICTOR FOR LINK QUALITY

Link quality estimation is a long-researched topic in wireless sensor networks and it has been shown to be relevant also for routing decisions in multi-hop networks [47]. More recently, machine learning approaches have been proposed for estimating or predicting the future behaviour of a link based on historical physical and link level data [48]. Existing work using machine learning focuses on predicting the actual values of a metric (i.e. packet reception ratio), this solving a regression problem. One arising challenge is the accuracy for predicting the short-term behaviour of intermediate links [49]. In this study we aim to formulate the prediction as a classification task and assess the feasibility of the prediction for intermediate links.

5.1 Dataset description

Description and analysis of three publicly available datasets and a dataset generated within this project. In particular, we consider the IEEE 802.11 dataset from Rutgers University and Colorado University, the 802.15.4 dataset from the University of Michigan which are publicly available on CRAWDAD [50]. We also generate our own SigFox dataset with the support of our eWINE project partner. After the thorough analysis of the datasets, we were able to fully use the 802.11/Rutgers and SigFox/JSI datasets, to a small extent the WiFi Colorado University dataset (incomplete documentation and missing sequence numbers) and we had to discard the 802.15.4/University of Michigan dataset (unable to generate labels for the classification tasks for the available data).

5.1.1 The Rutgers/noise dataset (v.2007-04-20)

This dataset was collected on the Open Access Research Testbed for Next-Generation Wireless Networks (ORBIT) and is publicly shared through the CRAWDAD repository [50]. The researchers aimed to use a physical testbed in a constrained space to create real world multi-hop network. They proposed noise injection as a more flexible option than hardware attenuation and considered methods for mapping real world wireless network topologies onto the testbed [51].

ORBIT compromises of 128 IEEE 802.11a/b/g radio interfaces attached to 64 static nodes arranged on an 8 by 8 grid. For this experiment, 32 nodes fitted with two Atheros 5212-based IEEE 802.11a/b/g cards each were used. The interference was produced at 4 randomly selected locations with signal generator and omni-directional noise antenna. The interference levels were configurable at the interferers and were used to vary the link conditions between any two nodes. The receivers’ driver provided all received MAC frames encapsulated with a so-called Prism header that contained bitrate, RSSI, and other physical layer information. Employing Perl scripts on the receiver side, sequence number and RSSI for each correctly received frame were extracted from the logs. RSSI is a relative index and we can infer that the higher the RSSI value is, the better the signal per link is.

The duration of transmission period per node was equal to 30 seconds. Since the transmitter sends one beacon per 100ms, N_t = d/100ms, where d is the duration of the transmission period in milliseconds this resulted in 300 transmitted packet per node. The experimenters were also able to control bitrate and transmission power on the senders. Therefore, in the dataset there are 5 experiments, where in each experiment transmission power was increased by 5 dBm, starting at 0 dBm and ending at 20dBm of transmission power. The traces of three nodes are missing from the publicly available dataset thus it contains 29 traces and 812 links per experiment.

5.1.1.1 Data cleaning

The Atheros cards return an RSSI index value of 0 to 127 (0x7f) with 128 (0x80) indicating an invalid value. We cleaned the dataset by filtering RSSI values in range of RSSI_MIN and RSSI_MAX value, for each trace/link. From the received sequence numbers, we saw that some packets were not successfully received. To better reflect this, we created to versions of the dataset. In one version we
D5.1: Machine learning algorithms development and implementation

insert low RSSI (-1) for missing packets. As we know from the dataset description, the maximum number of packets per transmission was 300 so we filtered all traces with MAX_SEQUENCE_NUMBER of 300.

5.1.1.2 Data Statistics

The dataset consists of 5 experiments, for each experiment 812 links connecting 29 nodes were monitored and traces were collected. Considering that, each node was transmitting in broadcast mode, there should have been 8400 measurements per transmission recorded by the 28 listening nodes, totaling of 243600 packets per experiment. However, some packets were lost, the number of missing packets depends on the transmit power and how far the interference generator is from the actual link.

We statistically describe the dataset by computing the five number summary per link. In Table 3, we illustrate these numbers for three representative links. The links are between node 1_2 (the transmitter, 0dBm) and nodes 5_4, 7_2 and 2_5 as listeners. From the table it can be seen that packet reception ratio (PRR) per link can vary from 22% for very bad link and up to 94.66% for very good links. Additionally, we can see that link with highest mean RSSI value of 4.010638 and lowest standard deviation value of 0.780018 results in the best PRR. In other words, stable links with high mean RSSI are the best in terms of PRR while unstable links with high average RSSI often have lower PRR ratio and behave as transitional links [55].

Table 3 Five number summary of three selected links when node 1_2 was transmitter.

<table>
<thead>
<tr>
<th></th>
<th>Node 5_4</th>
<th>Node 7_2</th>
<th>Node 2_5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of missing values</td>
<td>234.0</td>
<td>113.0</td>
<td>18.0</td>
</tr>
<tr>
<td>Packet count</td>
<td>66.000000</td>
<td>187.000000</td>
<td>282.000000</td>
</tr>
<tr>
<td>Mean RSSI</td>
<td>1.075758</td>
<td>1.967914</td>
<td>4.010638</td>
</tr>
<tr>
<td>Std RSSI</td>
<td>1.071456</td>
<td>1.067122</td>
<td>0.780018</td>
</tr>
<tr>
<td>Min RSSI</td>
<td>0.000000</td>
<td>0.000000</td>
<td>2.000000</td>
</tr>
<tr>
<td>25% RSSI</td>
<td>0.000000</td>
<td>1.000000</td>
<td>4.000000</td>
</tr>
<tr>
<td>50% RSSI</td>
<td>1.000000</td>
<td>2.000000</td>
<td>4.000000</td>
</tr>
<tr>
<td>75% RSSI</td>
<td>2.000000</td>
<td>3.000000</td>
<td>4.000000</td>
</tr>
<tr>
<td>Max RSSI</td>
<td>5.000000</td>
<td>5.000000</td>
<td>9.000000</td>
</tr>
<tr>
<td>PRR</td>
<td>22.00%</td>
<td>62.33%</td>
<td>94.66%</td>
</tr>
</tbody>
</table>

In addition to the five number summary, we did link symmetry analyses by plotting directional link behaviour of one pair of nodes, as depicted in Figure 14 and Figure 15. Figure 14 depicts RSSI values per link of each successfully received packet for node 2_5. Since we had 0% PRR when node 2_5 was transmitting and node 1_2 listening we were not able to depict this link. Other way around we had most of the packets received and this link is depicted in Figure 14. From this particular example it can be noticed that links are not always symmetric in the analysed dataset. This can be due to the interference generator and possibly also transceiver, antenna or other hardware particularities. Figure 15 depicts analyses of same pair of links, in this case we plotted PRR gain of directional links at
different transmission power levels. Here it can be noticed that directional link marked with the blue color, already at 0 dBm is approaching to 100% PRR, while directional link marked with green color at power level less than 10 dBm is not show any PRR changes.

Figure 14 RSSI over time for link between node 1_2 and node 2_5.

Figure 15. PRR gain of links between, node 1_2 and node 2_5, at various noise power level.

5.1.1.3 Preliminary exploration

As preliminary exploration, we calculated and stored vector of normalized histogram values with PRR and RSSI mean average values per link. For each link in our dataset we defined 40 bins with bins width of 1.0. Figure 16 depicts graphical comparison of histograms with 30 and 40 bins, when node 2_1 was transmitting while noise level was 20 dBm. It can be seen that graph on the left-hand side is slightly shifted right in comparison to the graph on the right-hand side. Therefore, in case we had a lower number of bins than 40, as result we could expect data loss per each link. While if we had set higher number of bins than 40 per link, we would have additional unnecessary empty bins.

Figure 16 Histogram of 30 bins vs 40 bins

Figure 17 depicts PRR value against time-average RSSI value of all links, when noise power was set to 0 dBm and window size to 50 packets. To create plot of PRR vs RSSI, we took unidirectional links among 29 nodes and for each link per window we computed PRR and RSSI value of all successfully received packets. From the plot it is clear that links with RSSI more than 2 of index value, PRR value is at least 90 %, while links with RSSI value between 0,5 and 2 of index value corresponds to PRR value between 20 % and 90%. As a third groups are links with PRR less than 20% and RSSI less than 0,5 of index value. This plot confirms the well-known shape of the three region link quality [55].
Next, as further exploration, we would like to see if, using automated methods such as unsupervised machine learning, would be able to detect the three zones (with bad, average and good link quality) of the link behavior manually determined in existing work [55]. For this, using the SimpleKMeans clustering algorithm, we performed clustering of data sets in 3 clusters. The model for clustering was built using only the vector of normalized RSSI values. When the transmission was done with 0 dBm noise level we got 3 clusters depicted in Figure 18 and Figure 19. Figure 19 depicts similar link distribution over RSSI and PRR attributes as we had in Figure 17. This implies that the algorithm is able to detect automatically the three regions.

Figure 17. PRR vs RSSI of 812 links, when noise level was 0 dBm.

Figure 18. Link clusters plotted on graph, PRR vs cluster when noise power was 0 dBm.
Using same machine learning algorithm and same configuration like in previous step, we performed clustering of dataset when noise level was -15 dBm. Depicted in Figure 20 and Figure 21 we can see that clusters 2 and 3 become very similar each to other, this can be due to the increased transmission power level that improved potential links of average link quality. In other words, it seems that previously intermediate links become good links while the previously bad links do to improve significantly.
5.1.2 The Colorado University/rssi dataset (v. 2009-05-28)

This data set provides a comprehensive set of RSSI readings from within an indoor office building from several perspectives. The experiment was performed on a roughly 180 nodes that are 802.11 compliant and deployed at 180 distinct physical locations throughout a large office building. The office building environment is a single floor building measuring roughly 50 x 70 meters. The interior consists of small offices, cubicles, long hallways, and large warehouse-like rooms. A floor plan of this environment with measurement points and the passive monitors locations labelled is provided in Figure 22. More into details the monitors were using D-Link DWL-AG530 card with omnidirectional dipole antenna 2-4 dBi, while transmitters were using WNC WLAN Cardbus Adaptor CB9 card with omnidirectional dipole antenna 2-4 dBi.

The data set was collected by researchers at the University of Colorado over the course of one day in August 2007. They use transmitters from each of the 180 physical positions to transmit 500 packets per transmitter. All measurement packets are recorded by 5 passive monitors, which are commodity linux machines with 802.11 cards and each RSSI measurement is labelled with the transmitter’s physical location. The experiment was repeated at each transmit power level between 10-20 dBm.

---

http://crawdad.org/cu/rssi/
5.1.2.1 Data cleaning

On this data, we took the following data cleaning and analysis steps. In the available log files description, the “device name” column was missing, therefore, we omitted this column from our analysis. For each transmission session we expected 500 packets, but for unknown reason in some experiments we had records of only 200 packets. Therefore, we optimized our code to handle this exception.

As for our study we need information per each link, we performed transformation of records to form unidirectional links “transmitter to monitor”. Additionally, we abstracted each link per transmission power level.

After thorough analysis of the log files, we cleaned the dataset by filtering antenna direction column with values (up, down, left, right), per each trace/link. Regarding the antenna direction changes during the experiment, we abstracted data traces of each transmission session per direction change for short term analysis and we merged all together for long term analyses. In some case we had 500 packets per link for short term analysis and up to 2000 (500 * 4) of packets for long term analysis.

We also stored missing packets in separated column (missing value) to be able to use information about missing packets later in the study.

5.1.2.2 Data Statistics

The dataset consists of 3 experiments, for each experiment different antenna type and different number of nodes were used. For each experiment there were unidirectional links, therefore when 18 nodes were transmitting there were total of 90 links, 70 nodes 350 links and 179 nodes 895 links respectively. Considering that, each node was transmitting in broadcast mode, there should have been 10000 measurements per transmission recorded by the 5 listening monitor nodes, totaling of 179000 packets per experiment when 179 nodes were used. However, some packets were lost, the number of missing packets depends on the transmit power and interference within the building.

We statistically describe the dataset by computing the five number summary per link. In Table 32, we illustrate these numbers for three representative links. The links are between nodes 0, 11, 6 (the transmitters, 12dBm) and node Monitor 2 as listeners. From the table it can be seen that PRR per link can vary from 28% for very bad link and up to 100% for very good links. Additionally, we can see that link with highest mean RSSI value of -40.355000 results in the best packet reception ratio (PRR). In other words, stable links with high mean RSSI are the best in terms of PRR while unstable links with high average RSSI often have lower PRR ratio and behave as transitional links.
Table 4 Five number summary of three selected links when Monitor 2 was receiver.

<table>
<thead>
<tr>
<th></th>
<th>Node 0</th>
<th>Node 11</th>
<th>Node 6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of missing values</td>
<td>144.0</td>
<td>58.0</td>
<td>0.0</td>
</tr>
<tr>
<td>Packet count</td>
<td>56.000000</td>
<td>142.000000</td>
<td>200.000000</td>
</tr>
<tr>
<td>Mean RSSI</td>
<td>-89.696429</td>
<td>-86.718310</td>
<td>-40.355000</td>
</tr>
<tr>
<td>Std RSSI</td>
<td>1.278062</td>
<td>1.531542</td>
<td>1.479839</td>
</tr>
<tr>
<td>Min RSSI</td>
<td>-93.000000</td>
<td>-94.000000</td>
<td>-54.000000</td>
</tr>
<tr>
<td>25% RSSI</td>
<td>-90.000000</td>
<td>-87.000000</td>
<td>-41.000000</td>
</tr>
<tr>
<td>50% RSSI</td>
<td>-90.000000</td>
<td>-86.000000</td>
<td>-40.000000</td>
</tr>
<tr>
<td>75% RSSI</td>
<td>-89.000000</td>
<td>-86.000000</td>
<td>-40.000000</td>
</tr>
<tr>
<td>Max RSSI</td>
<td>-87.000000</td>
<td>-85.000000</td>
<td>-37.000000</td>
</tr>
<tr>
<td>PRR</td>
<td>28.0 %</td>
<td>71.0 %</td>
<td>100.0 %</td>
</tr>
</tbody>
</table>

Since we had only one-way unidirectional links between the transmitter and monitor provided in the dataset, we could not provide link symmetry analyses. Depicted in Figure 11, we plotted RSSI gain over time when different antenna directions were switching, each period of 500 packets refers to one antenna direction. It can be seen that the antenna direction affects the quality of the link.

Depicted in Figure 23, we can see the PRR of unidirectional link between Node 0 and Monitor 1, transmission power 16 dBm.

Figure 23 Node 0 to Monitor 1, transmission power 16 dBm.

Depicted in Figure 24, we can see the PRR of unidirectional link between Node 0 and Monitor 2 at different transmission power levels. PRR is increasing respectively with each next step of transmission power level.
5.1.2.3 Preliminary exploration

As preliminary exploration, we computed the PRR vs RSSI plot as depicted in Figure 26. While the documentation of this dataset is not fully aligned with the actual traces and there are some questions with respect to the significance of the missing lines and the sequence numbers, it can be seen that for our interpretation of this dataset, only good and intermediate links are available. The analysis doesn’t show any bad quality links.

Figure 26 The PRR vs RSSI plot for the UColorado dataset.

5.1.3 The umich/rss dataset (v. 2011-08-10)

The experiment was performed on a 14 Mica2 sensor nodes randomly deployed inside and outside a lab room. The site of the experiment was the 4th floor of the EECS building, University of Michigan,
Ann Arbor. During the measuring period, students walked into and out of lab at random times, which caused anomaly patterns in the RSSI measurements.

During the experiment sensor nodes communicate each to other by broadcasting packets periodically and the recording signal strength (RSS), which is defined as the voltage measured by a receiver’s received signal strength indicator circuit (RSSI) is stored into a log file. For each pair of transmitting and receiving nodes RSSI values were recorded, totaling in $14 \times 13 = 182$ links/pairs of RSSI measurements over a 30-minute period, with a sample rate of 0.5 second. Additionally, for ground truth a web camera was employed to record activity of people entering the lab room.

5.1.3.1 Data cleaning

On this data, we took the following data cleaning and analysis steps. As for our study we need information per each link, we performed transformation of records to form unidirectional links “transmitter to receiver”. Additionally, to the mentioned process experimenter provided code samples with this dataset for the data preprocessing and cleaning.

5.1.3.2 Data Statistics

The dataset consists of experiment when each node was transmitting in a broadcast session, totaling in $14 \times 13 = 182$ links/pairs of RSSI measurements over a 30-minute period, with a sample rate of 0.5 second. With the defined sample rate over 30 minutes we should have 3600 packets per unidirectional link. Unfortunately, for unknown reason from the dataset we have only 3127 packets per link.

We statistically describe the dataset by computing the five number summary per link. In Table 33, we illustrate these numbers for three representative links. The links are between nodes 3, 5, 12 (the receivers) and node 1 as transmitter. Since we did not have information about sequence number of each packet, we were not able to calculate PRR value of link. From the table it can be seen that based on the MAX RSSI value and Mean RSSI value Node 12 had the best physical location in comparison to the other two nodes.

<table>
<thead>
<tr>
<th></th>
<th>Node 3</th>
<th>Node 5</th>
<th>Node 12</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Packet count</strong></td>
<td>3127.000000</td>
<td>3127.000000</td>
<td>3127.000000</td>
</tr>
<tr>
<td><strong>Mean RSSI</strong></td>
<td>94.067436</td>
<td>83.527540</td>
<td>107.104417</td>
</tr>
<tr>
<td><strong>Std RSSI</strong></td>
<td>14.598350</td>
<td>11.253193</td>
<td>16.920852</td>
</tr>
<tr>
<td><strong>Min RSSI</strong></td>
<td>63.000000</td>
<td>61.000000</td>
<td>85.000000</td>
</tr>
<tr>
<td><strong>25% RSSI</strong></td>
<td>84.412658</td>
<td>74.567246</td>
<td>93.304767</td>
</tr>
<tr>
<td><strong>50% RSSI</strong></td>
<td>89.198365</td>
<td>84.444444</td>
<td>102.756335</td>
</tr>
<tr>
<td><strong>75% RSSI</strong></td>
<td>99.550062</td>
<td>91.885500</td>
<td>114.966843</td>
</tr>
<tr>
<td><strong>Max RSSI</strong></td>
<td>146.000000</td>
<td>120.000000</td>
<td>156.000000</td>
</tr>
</tbody>
</table>

5.1.3.3 Preliminary exploration

In addition to the five number summary, we did link symmetry analyses by plotting directional link behavior of one pair of nodes, as depicted in Figure 27. Figure 27 depicts RSSI values per link of each

http://crawdad.org/umich/rss
successfully received packet for node 5. From this particular example it can be noticed that links are not always symmetric in the analyzed dataset. Similar as for other datasets we further explored the distribution, however, due to the unavailability of sequence numbers, PRR vs RSSI analysis is not possible for this dataset.

![Figure 27 RSSI over time for link between node 1 and node 5.](image)

### 5.1.4 The SigFox dataset

This dataset contains a comprehensive set of RSSI and SNR measurements, collected from SIGFOX uplink communication experiments.

In the course of the experiments, a SIGFOX base station was mounted on the roof of JSI building C, coupled with a transmitter, which was moved on a trolley through four different in-door locations, as can be seen in Figure 28.

The transmitter used in all experiments was USRP N200 with SBX daughterboard and VERT900 antenna, which was in vertical position. Front-end PA gain was varied from 0 dB to 30 dB in steps of 10 dB. Moreover, measurements were made with 30 dB Mini-Circuits attenuator inserted amid the USRP N200 and the antenna.

At each location, 100 packets were sent for each of the 4 gain settings, this resulted in total of 1600 measurements. The packet transmission frequency was defined by proprietary SigFox library, however only the first of three packet repetitions was actually transmitted.
5.1.4.1 Data cleaning

After parsing the log files, which contain data for all experiments, we coupled the corresponding data and acknowledgement packets. Some transmissions were not successful, therefore not all acknowledgement packets were logged. After sorting the packets by their sequence numbers accordingly, we extracted packets for each experiment and marked the failed transmissions (missing packets). Information about missing packets will be used later in the study.

There were no errors or particularities in the data whatsoever.

5.1.4.2 Data Statistics

Considering that experiments were conducted with four gain setting on four different locations, the dataset effectively contains 16 links, connecting two nodes, one transmitting device and one base station. All data transmissions were uplink. The transmitting node sent 100 packets in total for every distinct location and gain configuration, totalling 1600 measurements. However, some packets were lost, failures in transmission happened due to interferences in addition to changes in parameters and SigFox protocol imposed by the researchers.

We statistically describe the data by computing five number summaries for each link’s SNR and RSSI.
measurements. In Table 5, we illustrate a portion of these numbers for a few representative links. We can see that PRR varies from 61% (intermediate link) to 94% (good link). Additionally, the link with highest mean RSSI is also the best in terms of PRR, while the link with lowest mean RSSI is the worst in terms of PRR. Links with high average RSSI and low variation of RSSI perform worse than links with high average RSSI and low variation. In other words, stable links with high mean RSSI are the best in terms of PRR while unstable links with high average RSSI often have lower PRR ratio and behave as transitional links.

Table 6 Five number summaries for three selected links

<table>
<thead>
<tr>
<th></th>
<th>Location 0, gain 20</th>
<th>Location 2, gain 0</th>
<th>Location 0, gain 30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of missing values</td>
<td>6</td>
<td>39</td>
<td>17</td>
</tr>
<tr>
<td>Packet count</td>
<td>94</td>
<td>61</td>
<td>83</td>
</tr>
<tr>
<td>Mean RSSI</td>
<td>9.012158</td>
<td>1.639344</td>
<td>8.013769</td>
</tr>
<tr>
<td>Std RSSI</td>
<td>1.624729</td>
<td>0.386513</td>
<td>2.351790</td>
</tr>
<tr>
<td>Min RSSI</td>
<td>1.142857</td>
<td>0.000000</td>
<td>2.857143</td>
</tr>
<tr>
<td>25% RSSI</td>
<td>9.142857</td>
<td>1.428571</td>
<td>8.000000</td>
</tr>
<tr>
<td>50% RSSI</td>
<td>9.428571</td>
<td>1.714286</td>
<td>9.142857</td>
</tr>
<tr>
<td>75% RSSI</td>
<td>9.428571</td>
<td>1.714286</td>
<td>9.428571</td>
</tr>
<tr>
<td>Max RSSI</td>
<td>10.000000</td>
<td>2.285714</td>
<td>9.714286</td>
</tr>
<tr>
<td>PRR</td>
<td>94 %</td>
<td>61 %</td>
<td>83 %</td>
</tr>
</tbody>
</table>

5.1.4.3 Preliminary exploration

We explore the interrelation of transmission power with RSSI and SNR. After plotting every pair of attributes, we notice that, as expected, the average of RSSI and SNR in general increases with intensification of transmission power. For brevity, we only show the graph of RSSI versus transmission power gain in Figure 29. Graph also shows an unexplained drop in RSSI, we may attribute that to having insufficiently large dataset additionally to other hardware and software particularities.
D5.1: Machine learning algorithms development and implementation

Figure 29 RSSI versus transmission power gain for all locations

Figure 30 depicts PRR against average RSSI for all links. The figure shows that link quality and physical parameters, namely RSSI, appear to be correlated. Physical layer parameters are directly linked to current characteristics of a wireless channel, so link quality and physical parameters are usually tightly coupled [56].

5.1.4.4 Feature generation

Existing link quality estimators that use adaptive algorithms based on machine learning to better predict the link use a subset or a linear combination of the following wireless parameters (also called features in machine learning parlance): PRR, RSSI, SNR and LQI [56][57]. Other well performing estimators using more traditional algorithms use PRR [58] and PRR, ETX and LQI [59].

In the remaining of the study, we will consider only 3 of the four analysed datasets: WiFi from
Rutgers University, WiFi from University of Colorado and SigFox from JSI. We omit the ZigBee dataset from the University of Michigan because, the available traces do not contain enough information to compute the PRR and thus the labels necessary for the classification problem. This dataset is suitable for a regression problem.

For the WiFi from Rutgers we are able to use RSSI and PRR for generating the features, while for SigFox we are able to use RSSI, SNR and PRR. PRR is computed based on the available sequence numbers. We compute all possible combinations of feature vectors and generate corresponding training datasets. The number of possible combination is larger than we can list here (i.e. 127 different combinations for the SigFox dataset). Examples of computed feature vectors are listed below.

1) Instant RSSI (vector with dimension 1)
2) Instant RSSI + Avg RSSI over a time window of 5 packets (vector with dimension 2)
3) Instant RSSI + Avg RSSI over a time window of 10 packets (vector with dimension 2)
4) Instant RSSI + Avg RSSI over a time window of 20 packets (vector with dimension 2)
5) Instant RSSI + Avg RSSI over a time window of 5 packets + Std RSSI over a time window of 5 packets (vector with dimension 3)
6) Instant RSSI + Avg RSSI over a time window of 10 packets + Std RSSI over a time window of 10 packets (vector with dimension 3)
7) Instant RSSI + Avg RSSI over a time window of 20 packets + Std RSSI over a time window of 20 packets (vector with dimension 3)
8) Instant RSSI + Avg RSSI over a time window of 5 packets + Std RSSI over a time window of 5 packets + Instant SNR (vector with dimension 4)
9) Instant RSSI + Avg RSSI over a time window of 5 packets + Std RSSI over a time window of 5 packets + Instant SNR + Avg SNR 5 packets (vector with dimension 5)
10) Instant RSSI + Avg RSSI over a time window of 10 packets + Std RSSI over a time window of 10 packets + Instant SNR + Avg SNR 10 packets (vector with dimension 5)
11) Instant RSSI + Avg RSSI over a time window of 20 packets + Std RSSI over a time window of 20 packets + Instant SNR + Avg SNR 20 packets (vector with dimension 5)
12) Instant RSSI + Avg RSSI over a time window of 5 packets + Std RSSI over a time window of 5 packets + Instant SNR + Avg SNR 5 packets + Std SNR 5 packets (vector with dimension 6)
13) Instant RSSI + Avg RSSI over a time window of 10 packets + Std RSSI over a time window of 10 packets + Instant SNR + Avg SNR 10 packets + Std SNR 10 packets (vector with dimension 6)
14) Instant RSSI + Avg RSSI over a time window of 20 packets + Std RSSI over a time window of 20 packets + Instant SNR + Avg SNR 20 packets + Std SNR 20 packets (vector with dimension 6)

Using the PRR, we generate the labels. For each instance of a vector we can tell something about the link while will be its label. For instance, we can say that all vectors in which PRR is below 10% are bad links (label “bad”), all between 10%-90% are intermediate links (label “intermediate”) and all above 90% are good links (label “good”). Example configuration for generating the feature vectors as explained here is presented in Figure 31 for the SigFox dataset.
We noticed that the resulting training data is significantly unbalanced in favour of good links. This is natural, the recorded datasets contain no information for lost packets. In the Rutgers dataset, for instance, there are 699831 data points for good links (96% of the dataset), 31067 for intermediate links (4% of the dataset) and 321 for bad links (less than 1% of the dataset) – see Figure 32. Training a decision model on such a dataset leads to a strong bias for good links. With a simple threshold rule, the model would correctly classify more than 90% of the dataset. In cases when such a biased distribution exists, there are two options: 1) subsample the dominant class (use a random subset of the data points available for good links) or 2) oversample the minority classes (intermediate and bad links). In the following we refer to Resampled and Interpolated corresponding to the two options.

**Resampled**: we use the standard built-in approach in Weka for resampling the dataset in order to bias the class distribution toward a uniform distribution. We randomly sample with replacement, in this manner we acquire a new dataset with uniform class distribution, which is the same size as original dataset.

**Interpolated**: To add more data for bad and intermediate links, we take the following approach. We know the number of packets N that was sent over each link and the sequence numbers of the received packets. As a result, we can identify gaps. Given received sequence numbers $S_{1}$ (RSSI_{S1}) and $S_{M}$ (RSSI_{SM}), we compute the average RSSIavg = (RSSI_{S1} + RSSI_{SM})/2, and standard deviation RSSIstd = STD (for the 2 (RSSI_{S1} and RSSI_{SM}) packets). Then, for each missing packet with sequences between 1 to M, we insert in the dataset the sequence number and white noise with RSSIavg and RSSIstd.
D5.1: Machine learning algorithms development and implementation

Figure 32 Distribution of the classes in the dataset (class no 1 with PRR > 90% – good links, class no 2 with PRR between 10% and 90% - intermediate links and class 6 with PRR < 10% - bad links).

As can be seen in Figure 33, this results in 701086 data points for good links (60% of data points), 67670 data points for intermediate links (5% of data points) and 388344 for bad links (33% of data points). This training set is more balanced; however, the intermediate links seem to be underrepresented so it is likely the model won’t capture them too well (i.e. misclassify intermediate links). To account for that, we should either record a new dataset with more intermediate links or under sample the dominating two classes to improve the accuracy. We do so by keeping all data points for average links and randomly sampling data points for other two classes. This results in a new dataset with uniform class distribution.
Figure 33 Distribution of the classes in the Rutgers dataset (class no 1 with PRR > 90% – good links, class no 2 with PRR between 10% and 90% - intermediate links and class 6 with PRR < 10% - bad links).

The Colorado dataset contains only good and intermediate links as depicted in Figure 34 with 77% of the links being good and 33% intermediate. The situation is similar for the SigFox dataset where the percentages are 37% good links and 63% intermediate links.
5.2 Algorithm description

We use the J48 decision trees from Weka to build the link quality estimation models. These decision trees are based on the classical C4.5 algorithm for machine learning [60]. This algorithm uses the entropy, and information theoretic metric, to select the attribute which discriminate the most in the dataset to be higher in the tree. In other words, the splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. Besides performing the classification task, this algorithm is also useful for understanding which features are more relevant than others in explaining the dataset.

There are several settings (see Figure 35) we can choose when running the algorithm, the most important refers to pruning the three. By pruning, we avoid overfitting the model to the available dataset, thus creating a model that is likely to perform similarly on other, previously unseen, data points. The lower the value of the confidence Factor, the more pruned the tree is. We evaluated the model with several values (0.25 and below).
5.3 Implementation

The classification system is implemented using a set of Python data cleaning and data transformation scripts, a set of scripts specific for each dataset. Then, it uses a feature generator that is common for all datasets and is also written in Python. Finally, the models are trained using a Java program that calls Weka. The block diagram in Figure 36 depicts the interaction between the scripts modules of the system. All source code is released as open source in the corresponding D5.2.

Figure 35 Settings for the decision tree classifier.
5.4 Evaluation

On the performance:

The classification results for the WiFi dataset from Rutgers University are presented in Table 7. It can be seen that using interpolation for the missing dataset enables developing the most accurate classifier which correctly classifies of the test instances 95%. Undersampling the majority classes in an effort to balance the dataset and enable better classification for the minority class, represented by the intermediate link, decreases the performance of the classifier to 88%. Standard statistical resampling is less useful than interpolation, resulting in a classifier that classifies only 77% of the test instances correctly.

In terms of feature vectors, it can be seen that the largest feature vector (RSSI, RSSI avg, RSSI std) gives the best classification results and having smaller feature vectors (RSSI avg, RSSI std or RSSI, RSSI std) costs about 1% in performance.

Rutgers

Table 7 Classification results for the WiFi dataset from Rutgers University.

<table>
<thead>
<tr>
<th></th>
<th>Resampled</th>
<th>Interpolated</th>
<th>Interpolated and undersampled</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSSI</td>
<td>Correct class</td>
<td>65.1868 %</td>
<td>93.4898 %</td>
</tr>
<tr>
<td></td>
<td>Incorrect class</td>
<td>34.8132 %</td>
<td>6.5102 %</td>
</tr>
<tr>
<td>RSSI avg</td>
<td>Correct class</td>
<td>69.2663 %</td>
<td>94.0005 %</td>
</tr>
<tr>
<td></td>
<td>Incorrect class</td>
<td>30.7337 %</td>
<td>5.9995 %</td>
</tr>
<tr>
<td>RSSI std</td>
<td>Correct class</td>
<td>46.4307 %</td>
<td>90.9933 %</td>
</tr>
<tr>
<td></td>
<td>Incorrect class</td>
<td>53.5693 %</td>
<td>9.0067 %</td>
</tr>
</tbody>
</table>
### SigFox

The classification models for the SigFox/JSI dataset perform significantly poorer than for the WiFi/Rutgers dataset with the best performing model classifying only 78.8% of the test examples correctly. This can be partly explained by the fact that in the SigFox dataset we have only good and intermediate links (37% good links and 63% intermediate links). Intermediate links tend to be quite unstable, sometimes temporarily behaving as good links, other times as bad links so they might be confused by the model.

Also in this experiment, if we just consider RSSI based feature vectors, it can be seen that the combination of RSSI, RSSI avg, RSSI std performs the best, similar to the WiFi/Rutgers case. If we also consider SNR based feature vectors, it can be seen that they perform poorly with up to 69.7059% accuracy. However, RSSI and SNR based vectors such as snr_std snr_avg rssi_std rssi_avg rssi avgSnr lead to the best results.

It’s worth noticing that the model resulting from a 1 dimensional feature vector RSSI std, behaves about the same as random guessing for SigFox, about 50% correct classification, while the same for WiFi/Rutgers correctly classified 80-89% of the test instances.

Table 8 Classification results for the SigFox dataset from JSI.

<table>
<thead>
<tr>
<th>Feature</th>
<th>Correct class</th>
<th>Incorrect class</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSSI</td>
<td>65.1868 %</td>
<td>34.8132 %</td>
</tr>
<tr>
<td>RSSI avg</td>
<td>69.2663 %</td>
<td>30.7337 %</td>
</tr>
<tr>
<td>RSSI std</td>
<td>46.4307 %</td>
<td>53.5693 %</td>
</tr>
<tr>
<td>RSSI, RSSI avg</td>
<td>71.6481 %</td>
<td>28.3519 %</td>
</tr>
<tr>
<td>RSSI, RSSI std</td>
<td>70.6768 %</td>
<td>29.3232 %</td>
</tr>
<tr>
<td>RSSI avg, RSSI std</td>
<td>72.5581 %</td>
<td>27.4419 %</td>
</tr>
<tr>
<td>RSSI, RSSI avg, RSSI std</td>
<td>77.1956 %</td>
<td>22.8044 %</td>
</tr>
<tr>
<td>SNR</td>
<td>62.5735 %</td>
<td>37.4265 %</td>
</tr>
<tr>
<td>SNR avg</td>
<td>62.2059 %</td>
<td>37.7941 %</td>
</tr>
<tr>
<td>SNR std</td>
<td>62.5735 %</td>
<td>37.4265 %</td>
</tr>
<tr>
<td>SNR, SNR avg</td>
<td>68.2353 %</td>
<td>31.7647 %</td>
</tr>
<tr>
<td>SNR, SNR std</td>
<td>69.7059 %</td>
<td>30.2941 %</td>
</tr>
<tr>
<td>SNR avg, SNR std</td>
<td>62.5735 %</td>
<td>37.4265 %</td>
</tr>
</tbody>
</table>
Colorado

Due to missing sequence numbers in the Colorado dataset, we are unable to generate time windowed feature vectors for historical packets as in the other cases. We are only able to generate one feature vector per link and see how the classifier works.

Using the full dataset, J48 classifier puts everything in majority class (good links) no matter what features are used, therefore resulting in 77% correctly classified instances (i.e. all the good links are classified as good) and 33% incorrectly classified instances (i.e. all the intermediate links are also classified as good).

We also attempted to take all instances that are labelled 'intermediate' and randomly sample from 'good' links to establish uniformly distributed classes in the dataset. However, as shown in the table below, this model behaves almost as random guess with around 50% correctly classified instances.

<table>
<thead>
<tr>
<th>Feature Vector</th>
<th>Correctly Classified</th>
<th>Incorrectly Classified</th>
</tr>
</thead>
<tbody>
<tr>
<td>RSSI avg</td>
<td>51.3682 %</td>
<td>48.6318 %</td>
</tr>
<tr>
<td>RSSI std</td>
<td>53.5473 %</td>
<td>46.4527 %</td>
</tr>
<tr>
<td>RSSI avg, RSSI std</td>
<td>53.2601 %</td>
<td>46.7399 %</td>
</tr>
</tbody>
</table>

On the feature vectors:

In all the experiments, average RSSI was the most reliable predictor for the link quality. The second most important predictor was the standard deviation of the RSSI while the third only came the instant RSSI.

5.5 Inclusion in Showcase 2

In Showcase 2 we aim to show the effects of advanced transmission control using channel information as depicted in Figure 37. An advanced detector collects information about how occupied are the channels and can control then the transmission. The occupation of the channels can be decided randomly, purely on historical evidence using a probability table or can be predicted using measurements from the past. All three options are investigated in the showcase. The link quality prediction module aims to pro-actively enable channel selection.
Figure 37 Showcase two demonstrator using SigFox.
6 DEVELOPMENT OF PREDICTOR FOR LTE-U DUTY CYCLE

Cellular network operators are offloading traffic in unlicensed 5 GHz spectrum using LTE Unlicensed (LTE-U). But this part of the radio spectrum is also used by the existing IEEE 802.11 standards, e.g. 802.11ac/ax, or will be used by or may be used by future WiFi standards. Hence, the situation is similar to the one in 2.4 GHz ISM band where radio spectrum is shared by different radio technologies, such as WiFi and Bluetooth.

Knowing the available radio spectrum at each Access Point (AP) in a managed WiFi network can be used to load-balance client stations across the APs in order to maximize overall throughput/service quality. Therefore, existing algorithms need to be extended to incorporate external (non-WLAN) interference.

We propose LTEUSpotter - a system that detects LTE-U RF devices and their duty-cycles in real-time and using only commodity WiFi hardware. We present the design and implementation of LTEUSpotter. LTEUSpotter makes use of functionality provided by WiFi cards using Atheros chipsets to detect LTE-U devices and estimating their duty-cycle. Our approach is passive and is based on monitoring the state of the WiFi MAC protocol (state machine) at the Access Point. Experiment results reveal that LTEUSpotter is able to accurately estimate the duty-cycle of co-located LTE-U signal source at a wide range of LTE-U signal strengths allowing the AP to assess the available airtime for WiFi communication.

![Source: Korea Communication Review, Jan. 2015](image)

Figure 38. Radio spectrum shared by WiFi and LTE-U networks.

6.1 Background

6.1.1 LTE-U Primer

This section gives a brief overview of the relevant parts of LTE-U. LTE-U is created by LTE-U forum (http://www.lteuforum.org) is expected to be the first technology to be deployed where the unlicensed band is directly integrated into the LTE lower layers. The LTE-U attempts to achieve fair coexistence via Carrier Sense Adaptive Transmission (CSAT). Therefore, the LTE-U transmission is adaptively (ON/OFF) duty-cycled, based on active sensing and decoding of 802.11 frames. LTE-U transmission further introduces frequent gaps in the ON-cycle, which shall help WiFi to transmit delay-sensitive data (subframe puncturing). Note that, in LTE-U, listen-before-talk (carrier sensing) is not applied before transmission of packets in ON-cycle [38].

WiFi on the other hand cannot decode LTE-U frames, but can only rely on energy-based carrier sensing.
6.1.2 Understanding the Impact of LTE-U Co-Channel Interference on WiFi

Our system model is shown in Figure 40. Here we have a WiFi AP serving a client station. In addition there is a LTE-U signal source, which is sharing the same radio spectrum. The LTE-U signal can impact the WiFi communications in two ways [39], namely, i) affecting the carrier sensing mechanism of WiFi or ii) corrupting packets due to co-channel interference. Whether the first or the second has an impact depends on the received LTE-U signal strength at the WiFi transmitter, i.e. (AP. At high power levels the AP will be able to sense the LTE-U signal using its carrier sensing (CS) mechanism and hence defer from the channel during the LTE-U on period. Hence only during the off periods the channel can be used by WiFi. For lower LTE-U signal power the CS mechanism is unable to detect any ongoing LTE-U transmission and hence will also transmit during the LTE-U on period resulting in potential packet corruption due to co-channel interference, i.e. inter-technology hidden node problem. Hence, any WiFi packet being transmitted during the on-period will get lost (also retransmissions) and only packet transmissions during the on period are successful.

Note, WiFi has to rely on the less sensitive energy-based carrier sensing for detecting LTE-U signal which is around -65 dBm for today’s WiFi chipsets, e.g. Atheros [40]-[45].

6.2 Problem statement

Our objective is to estimate the available medium airtime, which is available for WiFi transmissions. Hence we have to quantify the amount of airtime utilized by LTE-U, i.e. the duty cycle In particular we aim for a low-complexity passive solution using Commercial off-the-shelf WiFi hardware. Further properties of the detector are: i) online algorithm running on AP, ii) passive and low-complexity (w.r.t. computational complexity), iii) realized with commodity 802.11 hardware (no additional hardware), iv) covering the whole LTE-U interference range (low & high). We assume a single LTE-U source (assume multiple sources are time-synchronized).
We consider IEEE 802.11 infrastructure networks, i.e. set of BSS (APs) each serving a set of client stations. Our approach is based on monitoring the state of the WiFi MAC protocol at each AP. Widely used WiFi chipsets based on Atheros AR93xx allow monitoring of the MAC state machine. Figure 41 illustrates the most important blocks associated with packet reception and carrier sensing: i) the first two blocks address signal detection whereas ii) the third block is an energy detection block. The following MAC state registers are available: i) total MAC clock ticks, ii) tx_busy state in clock ticks, iii) rx_busy state in clock ticks, iv) energy_detection state in clock ticks. With the help of those registers we are able to calculate the percentage of the time spent in each MAC state during a given observation duration.

Our observation is as follows. As WiFi cannot decode LTE-U frames it has to rely on energy-based carrier sensing. Hence, the LTE-U’s share of medium time equals the busy states that correspond to energy detection without triggering packet reception, i.e. interference. Observing the busy states is sufficient as long as the LTE-U signal received at the WiFi AP is sufficiently strong, i.e. above -62 dBm in case of Atheros chips. Unfortunately, even a weaker LTE-U can distort ongoing WiFi transmissions by corrupting packets. In case of Atheros, we observed that for weak LTE-U signals the busy states are very low and we saw sporadic peaks in the TX counter which represents transmissions attempts being not successful, i.e. not acknowledged. Therefore, in addition we monitored the ARQ state (MAC layer retransmissions) for each downlink WiFi transmission at the AP. TX states suffering from high retransmissions are being considered as busy states.

The related work for detection of non-WiFi devices is divided in two parts:

**Classical approaches:** Detection of non-WiFi signals like LTE-U is done either using custom hardware systems that integrate unique spectrum analyser functionality to perform non-WiFi device detection. Here commercial spectrum analysers or software radio platforms have specialized capability of providing “raw signal samples” for large chunks of radio spectrum (e.g., 100 MHz) at a fine-grained sampling resolution in both time and frequency domain. Another option is to use commodity WiFi hardware instead of more sophisticated capabilities available in dedicated spectrum analyzers, expensive software radios, or any additional specialized hardware.
Other approaches: Huehn [46] developed a tool RegMon for monitoring the state of the MAC of WiFi devices based on Atheros AR93xx chipsets. As WiFi cannot decode non-WiFi frames the non-WiFi's share of medium time equals the MAC busy states that correspond to energy detection without triggering packet reception. Observing the busy states is sufficient as long as the non-WiFi signal received at the WiFi NIC is sufficient strong, i.e. above -62 dBm in case of Atheros chips. RegMon serves also as baseline for our evaluations. In our work we extended the RegMon tool to provide information about retransmitted radio frames as well.

### 6.3 System description

A flow chart describing our LTE-U duty-cycle detector is depicted in Figure 42.

![Flow chart of signal processing in WiPLUS.](image)

**Step 1 – raw data acquisition:** The input is data from RegMon tool [46] where the MAC state registers (TX, RX, Other (= energy_detection - RX), ACK fail) are sampled at a rate of 2 kHz (Figure 43). Processing is done in chunks of 1 second windows size, i.e. W= 2000 samples.

**Step 2 – spurious signal extraction:** The next step is to extract signal which indicates for interference. Therefore, we construct a new signal by taking the filtered busy dwell time. In addition we have to account for unsuccessful transmission attempts, i.e. unsuccessful transmission due to corruption of either datapacket or acknowledgement frame, as they indicate interference. For those we take the TX instead of busy dwell time. We have to abort in case not enough samples could be extracted, 1% of samples.
D5.1: Machine learning algorithms development and implementation

**Step 3 – FFT / PWM signal detection:** In next step we compute the normalized power spectrum of signal using FFT calculation. This is used as input for peak detector to find the fundamental frequency and second/third harmonics. We abort in case no periodic spectrum can be detected.

**Step 4 – k-Means clustering:** We use KMeans clustering to detect cluster centers in time domain. Therefore, we perform clustering for different k values around the fundamental frequency. We determine the optimal K by silhouette analysis around desired K. Finally, we set all signal parts outside 2x median(cluster,) around cluster centers to zero in order to suppress outliers.

**Step 5 – low pass filtering:** We low pass filter the remaining signal to bridge possible gaps, i.e. smoothing. Therefore the fundamental frequency is used as filter cutoff frequency. We estimate the LTE-U ON time by computing median segment duration weighted by energy (area) of the segments (suppress outliers).

**Step 6 – calculate duty cycle:** Finally, we are able to compute the LTE-U duty cycle from LTE-U ON time and fundamental frequency.

Figure 43. Illustration of the steps involved.

**6.4 Evaluation**

The evaluation section is divided into two parts. First, we present the system model and methodology. Thereafter, the results are described.
6.4.1 Methodology

Our experiment set-up is as shown in the system model in Figure 40 and Figure 44. We set-up a single WiFi AP and client station and a LTE-U signal source. For WiFi we used Atheros AR93xx network cards. The LTE-U signal was generated using Matlab and R&S signal generator. The distance between each pair of nodes was set to 3m, i.e. triangle. We set-up a saturated UDP packet flow from AP to client station. Moreover, during the experiment the transmit power of the LTE-U signal was varied.

![Figure 44 Experiment set-up for LTE-U WiFi.](image)

For the WiFi configuration we used ATH9k driver and hostapd. The WiFi mode was set to 802.11a and the channel 48 (5240 MHz) was used. Furthermore, we used only a single antenna for WiFi (no antenna diversity, MIMO, etc.) and also disabled Atheros Adaptive Noise Immunity (ANI). The TX power for WiFi was set to 15 dBm for both AP and client station STA. The TX power of LTE-U was varied from +15 to -33 dBm. The LTE-U signal was generated using Matlab.

For performance evaluation we identified the root-mean-square error (RMSE) between the predicted and actually observed available airtime for WiFi. The latter is obtained by normalizing the actual UDP throughput with the maximum UDP throughput, i.e. the throughput in absence of LTE-U signal.

We compared our approach, **WiPLUS**, with a **Simple Detector**, i.e. a duty-cycle detection based on analysing the MAC busy state registers only, i.e. pure energy detection.

6.4.2 Results

The result for a LTE-U duty cycle of 20% is shown in Figure 46. We can clearly see that our approach (WiPLUS, green) is able to accurately estimate the LTE-duty-cycle from which the available airtime for WiFi can be derived. The simple detector is able to correctly estimate the LTE-U duty cycle for very high LTE-U signal power, i.e. up to 6 dB below the WiFi TX power. In contrast the proposed approach estimates the correct duty cycle at even very low LTE-U power levels, i.e. -36 dB.
6.5 Conclusions & Future Work

We presented the design of a passive, low-complexity LTE-U duty cycle estimator using COTS 802.11 hardware. At the moment, the algorithm is running in offline mode, i.e. the raw MAC state values are sampled and processed afterwards offline using Python scripts. We are currently working on an online implementation of the algorithm.

Another issue is that in absence of any DL WiFi traffic the passive mechanism is not working properly. We tackle this by injecting null function probe packets send to neighboring APs on the same channel in regular intervals. Note that we are not probing STAs due to battery constraints. The optimal probe packet injection rate (which depends on DL traffic) is an open research question.

6.6 Inclusion in Showcase 3

In showcase 3 we consider a cognitive network consisting of primary users (PU) and secondary users (SU). In particular, LTE-U will be used a technology for PU while the SU will be based on the envisioned GFDM system. The objective is to use WiPLUS component to detect the LTE-U PU signal and its duty cycle for different parts of the spectrum. The information about PU’s spectrum utilization is used as input for the spectrum shaping component of the GFDM SU.
7 CONCLUSION

This deliverable reported on research enabling the efficient management of dense wireless networks using machine learning technology.

Section 2 investigated LOS/NLOS classification to help improve the accuracy of localization technology in Showcase 1. Our current results show that convolutional neural network correctly detect NLOS/LOS conditions 90% of the time.

Section 3 investigated a predictor for MAC performance in which aimed to predict future packet loss ratio. The results so far show that a complex neural network seems to be the best performing with root mean square error below 10%. Section 4, investigated the suitability of automatically recognizing OFDM transmitters from spectrum data aiming to distinguish between WiFi, LTE and DVB-T. The results are very promising with correctly classifying the technologies more than 90% of the times. Section 5, investigated how well the quality of a link (good, intermediate or bad) can be estimated using physical and link level data across several technologies (WiFi, ZigBee and SigFox). The results showed that, when the dataset is of sufficient quality, with properly pre-processed data and engineered feature vectors we it is possible to correctly classify up to 95% of the time. Good and bad links are easy to recognize while intermediate links are more challenging. Sections 2-5 resulted in building blocks supporting Showcase 2.

Finally, section 6, investigated an LTE-U duty cycle predictor. It was shown that a simple detector is able to correctly estimate the LTE-U duty cycle for very high LTE-U signal power, i.e. up to 6 dB below the WiFi TX power. The approach we proposed is able to estimate the correct duty cycle at even very low LTE-U power levels, i.e. -36 dB. This building block supports the transmission of the secondary user in Showcase 3.

The software tools and datasets developed while carrying out the studies in this deliverable form part of the eWINE Intelligence Toolbox, will be released as open source in the Wireless Testbed Academy GitHub repository and will be used in the eWINE Grand Challenge.
REFERENCES


D5.1: Machine learning algorithms development and implementation


D5.1: Machine learning algorithms development and implementation


[37] TensorFlow is an Open Source Software Library for Machine Intelligence, https://www.tensorflow.org/


