Autoencoder-Based Characterisation of Passive IEEE 802.11 Link Level Measurements

Priyanka Neuhaus§1, Marcus Henninger*, Andreas Frotzscher§, and Ulf Wetzker§
§Fraunhofer Institute for Integrated Circuits, Division EAS, 01069 Dresden, Germany
*Nokia Bell Labs, 70435 Stuttgart, Germany

Abstract—Wireless networks are indispensable in today’s industrial manufacturing and automation. Due to harsh signal propagation conditions as well as co-existing wireless networks, transmission failures resulting in severe application malfunctions are often difficult to diagnose. Remote wireless monitoring systems are extremely useful tools for troubleshooting such failures.

However, the completeness of data captured by a remote wireless monitor is highly dependent on the temporal, e.g., short-term interference, and spatial characteristics of its environment. It is necessary to first ensure that the data was completely captured at the remote monitor in order to maintain the integrity of the failure analysis, i.e., to avoid false positives. In this paper, we propose an autoencoder-based framework to evaluate the quality of wireless data captured at a remote wireless monitor. The algorithm is trained using data generated under controlled laboratory conditions and validated on testbed as well as real-world measurement data.

Index Terms—Wireless Network Analysis, Industrial Wireless Communications, Passive Monitoring, Anomaly Detection, Machine Learning, Autoencoder

I. INTRODUCTION

Wireless networks are currently ubiquitously deployed in various industrial automation and control applications where they must meet stringent latency and reliability constraints [1]. Usually, in a smart factory setting, sub-optimal signal propagation conditions such as path loss, non-line-of-sight (NLOS) and multi-path propagation, shadowing and fading due to movement of the transmitter (Tx), receiver (Rx), people or machinery combined with interference from intra- and inter-technology wireless networks operating on the same frequency may lead to a significant degradation in the quality of service experienced by the user. Additionally, depending on the position of the communication devices and other influencing variables (such as the radiation characteristics of the antenna, transmission power or reflection and attenuation properties of the environment), situations arise in which the interference within a network can vary greatly depending on the location.

Currently, wireless transceivers do not have sufficient resources to monitor the state of the communication link because of highly limited hardware resources. In the event of transmission errors, radio components are not able to independently identify the cause of such failures and initiate remedial measures or send an error report to the user. Since troubleshooting such application failures is both time- and cost-intensive, it is essential to automate the troubleshooting process.

External monitoring systems, which can be fitted with the necessary hardware resources, are usually remote, stand-alone monitoring devices that do not actively communicate with the wireless network being monitored. The remote monitor captures the traffic and analyses the frame without generating additional wireless traffic and thereby interference itself. The captured frames can be analysed at the monitor to detect anomalies in the traffic pattern and, if possible, identify the root-cause of these anomalies. However, since the radio propagation environment is unique at each device including the monitor, there can be significant differences in the reception and interference conditions with regard to the network to be monitored and at the monitor itself. These differences may complicate and falsify the results of the intended analyses (e.g., analyses of transmission quality, stability, reliability and availability of all radio links). It is therefore necessary to validate that the wireless image captured at the monitor is close enough to the original wireless link under analysis in order to avoid false positives and negatives during troubleshooting.

The problem of evaluating the capture conditions of the wireless monitor is non-trivial, since the reception of a transmitted frame is strongly dependent on device-specific interference, resulting in highly individual error patterns of the nodes even for densely co-located nodes. Furthermore, mobility of the network nodes and of objects in their vicinity (e.g., persons and vehicles) in combination with environmental influences (e.g., temperature changes, air humidity fluctuations, metal doors, vibrations caused by heavy machinery) cause a strong temporal dependency of the connection quality of the individual radio links. Using threshold-based heuristics such as determining well-captured links using Received Signal Strength Indicator (RSSI) is unsuitable since it depends heavily on the hardware and software characteristics such as Rx sensitivity of the monitor. Additionally, while RSSI can be a good indicator for problems due to pathloss and network coverage, it cannot be used to detect frame collisions. For a more systematic and robust approach to the problem, we propose to view incompletely captured links as anomalies and apply state-of-the-art anomaly detection algorithms to determine the validity of the captured
links. More specifically, in this paper we use a Long Short-Term Memory Autoencoder (LSTM-AE) to detect the completeness of the captured data. With this unsupervised learning approach, the prohibitively high costs of labeling data can be avoided. Furthermore, the solution is generic and can be implemented for other wireless standards (IEEE 802.11, Bluetooth, Zigbee etc.) as well as proprietary protocols.

Most of the related work in this area is focused on the development of algorithms and tools for passive wireless analysis. In [2], a monitoring network of 150 nodes is deployed for a large-scale wireless network and gaps in the data acquisition are determined by identifying missing logical sequences of protocol data. This approach is highly limited to the wireless protocol used and the hidden node problem. In [3], a combination of protocol and spectral analysis to identify IEEE 802.11 hosts has been studied whereas [4] addresses a toolchain for the passive analysis of IEEE 802.11 networks and identifies suitable metrics for the evaluation of communication link quality. However, none of these works addresses the issue of incomplete data acquisition due to the location of the passive monitor.

While commercially available passive monitoring tools [5], provide detailed insight into the structure of individual radio frames, they are usually dependent on the expert knowledge of the user for further evaluations and root-cause analyses. Corrupted data frames received by the monitor, resulting from transmission errors are usually immediately discarded or simply displayed to the user as errors. An evaluation of the observation position is carried out indirectly, if at all. In these cases, only the RSSI of the received frames from each individual Tx is displayed to the user. To the best of the authors’ knowledge, no evaluation, interpolation or inclusion of this information in further analyses is performed. Furthermore, to the best of our knowledge, there is no current solution that takes into consideration the quality of the captured data as a function of the propagation environment at the monitor.

This work illustrates a specific execution of our conceptual solution described in [6], which addresses the detection of data capture anomalies in a passive monitoring scenario. Furthermore, this work is based on the IEEE 802.11 protocol, but the solution proposed here can be transferred to other wireless protocols.

II. SYSTEM MODEL

A. Problem Description

Fig. 1 illustrates the communication between a wireless Access Point (AP) and a wireless mobile Station (STA) in an ideal monitoring scenario. The unidirectional link between the AP and the STA needs to be analysed by the wireless monitoring system. In this example, the monitoring system comprises of a single wireless Monitor (M). In a real application with several networks deployed, the monitoring system would usually comprise several monitoring nodes for better area coverage. In the ideal scenario, the radio link between the AP and STA is identical to the link between the AP and the monitor such that the sequence of wireless frames received at the monitoring node is an identical copy of the data received at the STA. In this case, the monitoring node has complete knowledge of the wireless Link under Analysis (LuA) and can proceed to inspect the communication on the link for potential problems.

However, in more realistic indoor channel environments, the channel impulse response varies even with a small displacement (>1 m) of the wireless Rx [7]. Additionally, since the wireless channel on the factory floor is constantly varying, the data captured at the monitor can vary significantly from the data received on the LuA. In the exemplary scenario shown in Fig. 1, the link between the Tx i.e., AP and the monitor is temporarily affected by some short-term wireless interference whereas the LuA remains intact. In this case, without further inspection and treatment of the acquired data, the monitor may arrive at the incorrect conclusion that the frames were lost at the LuA too. Hence, in this work we tackle the problem of identifying LuAs which can be analysed reliably by the monitor.

B. System Concept

The remote monitor consists of a commercial off-the-shelf Radio Frequency (RF) interface which captures wireless frames and forwards them to a generic protocol analyser such as Wireshark [8]. Raw data from the protocol analyser is fed into the analysis toolchain which begins with classical feature engineering and data preprocessing steps. The detection of deviations in received frame sequences due to the physical location and environment of the monitor is integral to the preprocessing stage. Here, the measured values provided by the preceding data acquisition block are examined for conspicuous patterns and inconsistencies in state transitions, which indicate a deviation from the standard state-machine of the wireless communication protocol. In insufficient data capture scenarios, where there are significant diversions from standard state-machine transitions, the data must either be augmented, for e.g., through interpolation, or eliminated from being used further down the analysis toolchain, in order to preserve the integrity of the link quality analysis.
As illustrated in Fig. 2, sequences of length \( l \) of the input data are fed to an Autoencoder (AE) which tries to reconstruct the output sequence from the input sequence. The cost function or reconstruction error \( \varepsilon \) indicates how well the AE could reconstruct the input sequence. If \( \varepsilon > T \), where \( T \) is a pre-defined, application-dependent threshold, the decoder was unable to reconstruct the data correctly. In this case, the data cannot reliably be used for further analyses without some further treatment of the data. Depending on how severe the error is, the data can be augmented using interpolation techniques or must be discarded before subsequent analysis.

III. AUTOENCODER APPROACH TO ANOMALY DETECTION

In general terms, an AE is a type of Neural Network (NN) that tries to find an efficient coding to represent the respective input data. The network can be thought of as a system consisting of two separate parts: an encoder trying to find a suitable coding, and a decoder with the aim of reconstructing this code so that the output corresponds to the input [9].

Anomaly detection is one of various fields of application AEs have been employed for [10]. The underlying idea is to train the network with a dataset from regular conditions such that it is able to learn the intrinsic characteristics and can encode and decode similar data with a very low reconstruction error. During deployment, feeding the AE anomalous data with different and previously unseen inherent statistical properties should lead to (comparatively) higher reconstruction errors and, hence, enable the detection of outliers.

Relating this principle to the problem considered in this work, wireless frames captured in an environment where the monitoring device is able to capture virtually all of the traffic between an AP and a STA represent regular data and, accordingly, the kind of data the AE is trained on. On the contrary, data from unfavorable monitoring positions, where the passive device misses a lot of frames, should lead to higher reconstruction errors and, consequently, to the detection of such unfavorable positions.

A. Data Preprocessing

As is generally the case in machine learning tasks, a great share of the work to be done is consumed by handling the input data. Initially, it needs to be determined which parts (i.e., features) of a WiFi frame we utilize at all for training. After this selection, the remaining features must then be processed in a way that they can be fed to the AE as input. This comprised scaling numerical values (e.g., sequence number or RSSI) between a range of -1 and 1, but also finding suitable encodings for the categorical variables of a WiFi frame, such as the frame type (e.g., Data or Beacon). For the latter we utilised entity embeddings [11], which proved superior to other approaches like simple one-hot or ordinal encoding. The main benefit of using embeddings is that, instead of assigning numerical values to categorical variables, the network can learn suitable ways of encoding the data and thereby also has the possibility to exploit relationships between the variables. Moreover, we also overcome two major shortcomings of the two other mentioned methods, namely introducing high dimensionality (for one-hot encoding) and imposing a non-existent ordinal relationship between the variables (for ordinal encoding).

B. Proposed LSTM-AE Architecture

Long Short-Term Memory (LSTM) [12] units are especially equipped to exploit long-term dependencies between data points and have thus been chosen as the foundation of our AE. While we also tested and trained AEs based on conventional fully-connected layers and convolutional layers employing 1D filters, we observed the framework shown here to achieve the best results with respect to the capability of detecting missing frames. Fig. 3 shows the proposed architecture of our LSTM-AE.

As can be seen from the illustration, the proposed network structure is of low-complexity. The encoder that transforms the input \( x \) to the latent representation \( z \) consists of a single layer with 100 LSTM units, whereas the decoder is comprised of an LSTM layer with the same number of units plus an additional dense layer to force the output \( y \) to have the same length as...
the input. Moreover, even though the data is encoded, it is not compressed as is usually the case for AEs. However, this architecture showed better capability as compared to a deep network with several layers and the typical bottleneck which forces the latent space representation to be of much lower dimensionality than the input.

C. Training Approach

For the AE to be able to exploit the characteristics of the underlying transmission, it self-evidently needs to see multiple frames at once. Therefore, it is necessary to feed the AE a certain number of consecutive frames, which we hereinafter refer to as a frame sequence. We found a number of 5 frames to be a good trade-off, where one frame sequence already contains a lot of information and interdependencies between subsequent frames, but can also be reconstructed reasonably well due to its still manageable length. This is also the frame sequence length that was used for the results part in Section V and leads to 50 numerical values (or features) the AE has to process at once. Moreover, the batch size (i.e., number of frame sequences for performing a single training step) was 256 and we used a decaying learning rate with a starting value of 0.005. As the loss function, we chose the mean-squared error (MSE) between elements of the input frame sequence \( x \) and the AE output frame sequence \( y \). AE training was conducted employing stochastic gradient descent and the backpropagation algorithm as well as the Adam optimizer [13]. For implementing the network, we used the TensorFlow machine learning library [14].

IV. Dataset Generation

In an AE-based approach for detecting unreceived frames, it is of primary importance that the training data represents all characteristic frame sequences without reception errors. Simulators for wireless networks [15] represent an easy way to generate these required datasets. However, due to the complex interaction of radio channel, hardware, application software and network stack, there are noticeable differences in a simulated dataset with respect to a real application.

Training and test data generation. A wireless testbed was used to obtain a representative packet capture (pcap) of the network traffic for the training and test datasets. The network topology replicated in these setup represents a typical application as found in industrial plants, which consists of one or more STAs connected to a central AP. A monitoring node positioned within the testbed captured the wireless communication between all nodes. In order to obtain datasets with a large diversity, all hardware and software components were selected according to their adaptability. The final dataset included over 14.4 million frames created in 20 different configurations of the testbed. Given the size of the data set obtained, a split was made into 80% training data and 20% test data.

Testbed hardware setup. The design of the above outlined testbed mainly used amd64-based hardware platforms equipped with Qualcomm Atheros AR9380 PCIe WLAN cards as wireless network nodes. In order to represent different performance classes of STAs, these amd64 nodes were completed by an embedded system based on the Raspberry Pi 3 model B+ with a Cypress CYW43455 WLAN chipset. To avoid any external interference, the entire wireless network was housed in an EMI/RFI-shielded faraday tent [16] of 2.5 m width, 2 m depth and 2.5 m height. An unintended increase of the noise floor by reflections was suppressed by lining the interior with RF absorbers [17].

Testbed software setup. The AP was implemented on an Ubuntu LTS using the software hostapd [18], which allows a very fine-grained setting of the WLAN network capabilities. A typical client-server scenario was chosen to emulate the application software running on the STAs, using the traffic generators iperf3 [19] and D-ITG [20]. The acquisition of training data in form of pcap file format was carried out on the monitoring nodes via the software tsghark [8]. All individual software components of the testbed, such as the settings of the software AP, the parameters of the traffic generators and the recording of the pcaps, were orchestrated by a central control software.

Validation data generation. Datasets with deterministic frame gaps were used to specifically validate the detection accuracy of non-received frames. The lack of a reliable method for generating single-packet errors within the testbed required the use of an emulator to generate a validation dataset. Based on a packet loss model, frames were reproducibly removed from datasets recorded in the testbed. The number and pattern (bursts or single frames) of missing frames can be determined by setting the parameters of the selected model. In the current implementation, a simple model consisting of burst error generation via the Erlang distribution as well as of normally distributed single frame errors has been realised. Using this emulative approach, data generated in the testbed could be supplemented with a real-world dataset captured in an automotive factory. The wireless communication of a production line offers a significantly larger variance of traffic and interference patterns and is therefore very suitable as a validation dataset.

Feature selection. All datasets are multi-dimensional, non-equidistant time series. As we are dealing with an unsupervised learning problem, the selection of features was based on a mixture of domain knowledge based on several years of experience in protocol analysis for wireless networks and experimenting with different setups. Within the scope of this work, the following features were selected:

- **Received timestamp**: The timestamp at which the frame arrives at the monitor. This is useful in determining the inter-frame spacing between consecutive data frames. A large value of inter-frame spacing is a possible indicator of a capture gap.
- **Sequence number**: The sequence number is incremented on the transmission of each data frame so they can be clearly identified by the recipient. Unexpected leaps in the sequence numbers are an useful indication of capture problems.
- **RSSI**: The signal strength in dBm represents the attenuation between the Tx and the monitor node. A low value of
RSSI could be indicative of pathloss problems.

- **Modulation and coding scheme (MCS):** The MCS indicates the datarate of transmission on the LuA. This feature is an indicator of the stability of the LuA and is useful in distinguishing problems at the LuA from capture problems at the monitor.
- **Frame length and frame duration:** The frame length in bytes and the calculated duration in µs.
- **MAC addresses of the Tx and Rx:** To identify transmissions on unidirectional and bidirectional links.
- **Frame type and subtype:** IEEE 802.11 distinguishes between management, data and control frames. This is useful for completing logical protocol sequences and identifying gaps in them.
- **Retransmission flag:** The retransmission flag in the IEEE 802.11 frame header indicates whether the frame is being transmitted for the first time or not. This helps to identify transmission problems on the LuA as opposed to problems with the data capture.
- **Acknowledgement frame:** Every successfully received data
frame must be acknowledged by the Rx. Tracking Data and ACK sequences on a bidirectional link helps to identify asymmetrical capture problems at the monitor.

It must be noted that these features are specific to the IEEE 802.11 standard but similar features could be used for other protocols.

V. RESULTS

A. Validation on testbed data

Fig. 4 illustrates the performance of the trained LSTM-AE on the validation dataset from the testbed. It can be seen that the AE manages to decode almost all frames correctly under good capture conditions. There is some loss due to encoding even under good capture conditions, because of which some frames display a high reconstruction loss. Although there is diversity in the training data, the AE overfits to at least some degree to it, leading to a higher reconstruction error floor for frames from unseen environments. While this makes setting the anomaly threshold application-specific, a quick visual comparison of Fig. 4a and Fig. 4b shows a distinct difference in good and bad data captures. The scenario illustrated in Fig. 4b with insufficient capture conditions at the monitor where 10% of frames were lost at the monitor, it can be seen that the average frame error increases by a factor of $10^2$ as compared to the scenario illustrated in Fig. 4a.

However, a better measure is provided by investigating the error distribution of the reconstruction errors as seen in Fig. 5. Fig. 5a and Fig. 5b display the reconstruction error distribution on the test data for good and insufficient capture conditions respectively. There is a clear distinction in the spread of the distribution for the good and insufficient capture scenarios. Numerically, this might be captured by the index of dispersion [21] which is defined as $\sigma^2/\mu$, where $\sigma$ is the standard deviation and $\mu$ is the mean of the distribution. The index of dispersion is almost 4 times higher for insufficient capture conditions as compared to good conditions. In other words, links with a high index of dispersion cannot be reliably analysed by the monitor.

B. Validation on real-world measurement data

We further validated the models on data recorded at an automobile factory. Since the data is from a completely different wireless environment (RF propagation conditions, hardware and software of the communicating devices) as compared to the training and test data generated in our lab, it is interesting to see that even on data from previously unseen environments, the index of dispersion is much lower for good (Fig. 5c) monitor positions as compared to insufficient (Fig. 5d) capture conditions. Comparing Fig. 5c and Fig. 5d clearly highlights the practical applicability of the proposed solution. It shows that the algorithm is capable of distinguishing between good and bad capture conditions even on real-world datasets and therefore improves the reliability of the wireless monitor.

VI. CONCLUSIONS

In this paper, we proposed a method to detect anomalies arising in wireless data captured by an external monitor. We trained a LSTM-AE to learn the wireless protocol sequences under good capture conditions and showed that the LSTM-AE fails to reconstruct the sequences under conditions of high frame loss, thereby detecting these as anomalies. In future works, we aim to extend the training dataset and explore AE-ensembles to encompass a variety of wireless environments, anomalies and user applications. Additionally, we will explore more complex packet loss models to model the loss at the monitor more accurately. Furthermore, the index of dispersion must be benchmarked for good capture positions in every new environment, in order to be used to automatically detect scenarios with capture problems.

REFERENCES