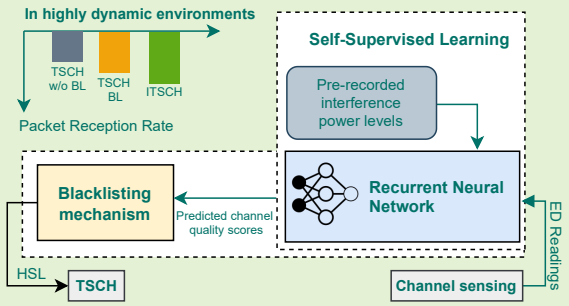
Channel Quality Prediction for TSCH Blacklisting in Highly Dynamic Networks: A Self-Supervised Deep Learning Approach

Mohammad Farahmand, Majid Nabi, *Member, IEEE*

***Abstract*— Cross-Technology Interference (CTI) is a severe issue for wireless networks operating in the license-free ISM bands. In particular, CTI can significantly affect the performance of low-power wireless links used in the Internet-of-Things (IoT). The IEEE 802.15.4 standard adopts a channel hopping scheme in its Time-Slotted Channel Hopping (TSCH) mode as a means to mitigating the adverse effect of CTI. However, the indiscriminate procedure by which TSCH nodes hop over different channels can suffer from severe interference in specific channels. Adaptive channel blacklisting is a technique to alleviate this issue by leaving out low-quality channels from the hopping list. To enable an effective blacklisting, especially in highly varying networks, an accurate real-time prediction of the quality of all available channels is of paramount importance. Previous studies rely on the past records of the channels as an indication of their quality in near future. Evidently, such approaches cannot extend to highly dynamic environments. This paper present a self-supervised approach for training deep neural networks capable of predicting the future behaviour of the frequency channels. The trained models can then substitute the quality assessment approaches in blacklisting schemas. Considering in-vehicle wireless networks as a target application, we evaluate this idea using a real-world experimental dataset, consisting of three measurement scenarios inside a moving vehicle. The experimental results show that using the proposed technique for TSCH blacklisting significantly improves the reliability of networks experiencing such highly dynamic interference and performs at least as good as the existing channel assessment methods in low-interference conditions.**



***Index Terms*— Internet-of-Things, IEEE 802.15.4, TSCH, Channel blacklisting, Deep neural networks, Deep learning, Channel quality prediction, Dynamic networks, In-vehicle networks**

I. Introduction

The unlicensed 2.4 GHz ISM band is one of the most widely used frequency bands for short-range wireless communications. WiFi [1] routers, Bluetooth [2] devices, and household appliance all occupy these frequencies, resulting in crowded communication channels with Cross-Technology Interference (CTI). Such CTI is a source of reliability issues attached to low-power wireless communication technologies such as the IEEE 802.15.4 standard [3]. Time-Slotted Channel Hopping (TSCH) is one of the Medium Access Control (MAC) operational modes of the IEEE 802.15.4 standard. TSCH is an attempt at increasing the robustness and reliability of IEEE 802.15.4 wireless links, with the primary goal of reducing the effects of interference and multi-path fading.

TSCH achieves reliability by providing diversity in both time and frequency. The protocol divides time into periods of fixed length called timeslots, which is long enough for exchanging a full-sized packet and its optional acknowledgement. Timeslots can be exclusively dedicated to wireless nodes by a scheduler, or they can be shared by several nodes in a neighborhood. In the frequency domain, the standard defines 16 channels in the 2.4 GHz band, all of which can be used by the nodes for parallel communications. The nodes in the network indiscriminately hop over the available channels in each timeslot. Such channel hopping technique eliminates long-term link blockage due to high interference in a particular channel or multi-path fading. This is done because a proper scheduler will schedule timeslots in such a way that packet transmissions over a particular link are performed in different frequency channels over time; a link hopefully uses all available frequency channels.

Although the plain channel hopping mechanism of TSCH greatly improves communication reliability, the network may still severely suffer from blindly using all channels for channel hopping. In the presence of external interference,

Manuscript received xxx; accepted xxx. Date of publication xxx; date of current version xxx. This work was supported by the SCOTT European project (www.scott-project.eu), that has received funding from the Electronic Component Systems for European Leadership Joint Undertaking under grant agreement No 737422.

Mohammad Farahmand is with the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran (e-mail: m.farahmand@ec.iut.ac.ir).

M. Nabi is with the Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands, and also with the Department of Electrical and Computer Engineering, Isfahan University of Technology, Isfahan 84156-83111, Iran (e-mail: m.nabi@tue.nl).

some channels could experience very strong interference for varying periods of time. Transmissions done in such noisy channels are very likely to fail leading to wasting energy and bandwidth, and degrading the reliability of the network. Therefore, detecting and leaving out the channels with a high level of interference from the hopping list can substantially increase the throughput and improve the reliability of the TSCH network.

Several blacklisting mechanisms have been proposed in literature to identify and exclude channels of low quality from the hopping list in TSCH. As an instance, our earlier work [4] presents Enhanced Time-Slotted Channel Hopping (ETSCH) as a blacklisting mechanism for TSCH. As prerequisite, a channel blacklisting mechanism needs to be fortified with a precise and agile channel quality prediction method to be able to detect the channels that are going to perform poorly for future packet transmissions due to CTI. ETSCH monitors the quality of all the 16 channels at run-time by exploiting the so-called silent periods in the timeslots, periods in a timeslot in which no radio activities within the TSCH network are expected. This channel quality estimation comes at no cost to the throughput of the network. ETSCH then estimates the quality of all channels using a moving average of the noise floors on each channel. While effective and straightforward, this approach assesses the channels based on their recent records, which might not properly reflect their future behavior. Some other techniques may even calculate metrics such as the Packet Reception Rate (PRR), the Received Signal Strength Indicator (RSSI), etc. as indicators for the quality of the channels. [5], shows that the chosen quality assessment approach greatly affects the performance of the blacklisting schema.

An appropriate channel quality assessment technique should have several features to be effective for channel blacklisting in TSCH. First, channel quality estimation needs to be done for all channels even if a channel is not being used and thus no packet transmission is being experienced. Therefore, existing link quality estimation techniques [6] are not directly applicable since they are based on the record of packet exchanges performed over a link. It is necessary to allow the mechanism to enable the use of previously blacklisted channels that turned to be of good quality. Also, in many applications like in-vehicle networks, the interference sources are highly dynamic (due to mobility of a vehicle or variation in traffic in the interferer technology). Thus, the channel quality assessment needs to be continuously performed and should be agile enough to predict long-term changes in the quality of channels.

In this paper, the application of Deep Neural Networks (DNNs) in channel quality prediction for TSCH blacklisting is studied. We propose a self-supervised approach for training DNNs, with the goal of predicting the *future* behaviour of the frequency channels. This approach allows us to train a DNN using a sequences of noise floors for individual frequency channels, and with no need for expert knowledge or manual labeling. A DNN trained using this approach can replace the current channel quality assessment metrics in existing blacklisting schemas. Using a real-world experimental dataset introduced in [7], in which the noise floors have been recorded inside a moving vehicle in various scenarios, we train recurrent neural networks and substitute them in the Enhanced TSCH (ETSCH) [4] schema. This schema does not deviate far from the original protocol, yet it manages to improve the reliability of the network by a considerable margin. After substituting the neural networks, the schema performs as well as it did before in low-interference environments, but manage to improve the reliability of the network in dynamic and high-interference scenarios significantly.

Although the proposed DNN-based approach for channel quality prediction is evaluated by replacing the channel quality assessment technique of a representative centralized TSCH blacklisting (ETSCH), its application is not limited to this protocol only. It can be used as a service for any kind of channel blacklisting scheme, especially those designed for highly dynamic networks. Although the developed model is able to predict frequent changes in the quality of channels, the action upon any predicted quality change is a decision that should be carefully made by the used channel blacklisting scheme. Overheads and complications involved in distribution of the blacklisted/whitelisted channels need to be taken into account. While it may be feasible to frequently change the hopping list in single-hop networks (e.g., ETSCH), reliably changing and distrusting hopping sequence list is challenging in multi-hop networks. Anyways, such quality predictions can help in making packet transmissions more efficient even if the change is considered to be transient upon which no action is taken by the channel blacklisting mechanism.

The paper is organized as follows. The next section is a brief introduction to the TSCH protocol and its fundamental concepts. Section III discusses the previous solutions for blacklisting in TSCH. Necessary background for the Deep Learning methods used in this work is provided in Section IV. Section V introduces our proposed solution, and the following section presents the experiments done with the model and their results. Section VIII concludes.

II. Time-Slotted Channel Hopping

TSCH is a MAC operational mode of the IEEE 802.15.4 standard that combines time-division multiplexing and frequency hopping, two techniques usually adopted individually by other wireless networking protocols, to improve the reliability of communications. The standard defines 16 non-overlapping channels over the 2.4 GHz ISM band, each with a 2 MHz bandwidth and a channel spacing of 5 MHz. Nodes in the network operate in discrete units of time called timeslots, which are long enough to transmit and acknowledge a single full packet (i.e., $10ms$ long by default). Multiple consecutive timeslots are grouped into *slotframes*, which continuously repeat over time. The number of timeslots in a slotframe, namely its length, is not under any strict regulations by the protocol, and it can be used to control trade-off between the energy consumption, latency, and duty cycle.

The TSCH mode comes with a particular frame known as an *Enhanced Beacon* (EB). TSCH uses these frames for setting up the network and synchronizing the nodes. PAN coordinators announce their presence periodically by transmitting an EB.

On the other hand, nodes that have not yet joined the network can synchronize to the network and join it by intercepting an EB packet.

When a node joins the network, it will receive the total number of timeslots passed since the network was set up, a parameter referred to as the *Absolute Sequence Number* (ASN), and it will also be assigned an arbitrary value in range 0 to 15, which will be called its *Channel Offset*. At any point in time, the node can identify its operating frequency channel using Eqn. 1.

$$CH = HSL[\ (ASN + CH\_Off)\ mod\ |HSL|\ ] \quad (1)$$

where CH represents the channel number from a Hopping Sequence List (HSL). HSL is a especially ordered subset of maximum 16 frequency channels in the 2.4 GHz ISM frequency band, and $|HSL|$ denotes the number of channels in HSL. Absolute Sequence Number (ASN) is a global variable synchronized in the whole TSCH network, which counts the timeslots. TSCH provides the possibility of parallel communication in the network by using different channel offset (CH_Off). The TSCH standard allows for HSL to only include a subset of the 16 available channels. [8] shows that limiting the HSL to a subset of the channels with the best qualities can increase the reliability of the network.

The TSCH standard protocol provide an appropriate machinery for implementing a Time-Division Multiple Access (TDMA) contention-free MAC mechanism by exclusively assigning the timeslots to wireless nodes in a neighborhood. However, the standard itself does not provide a scheduling scheme for timeslot and frequency (channel offset) assignment; it is left for the upper layers in the protocol stack.

III. Related Work

Single-channel communication protocols tend to experience long-term continuous link failure due to various phenomenons such as narrow-band interference and multipath fading [9]. The channel hopping technique can mitigate these effects by providing a wide selection of channels for nodes to transmit in. TSCH is not the only protocol to adopt this technique; Bluetooth, WirelessHART [10], and ISA 100.11a [11] all employ the same technique in order to improve the reliability and robustness of the network. Realizing the fact that all the channels on the frequency bands are not of the same quality, the specification of these protocols allow for user-configured white/black-listing. When setting up the networks, users can specify a static list of channels that the network can use in a predefined hopping sequence pattern. ISA 100.11a, which is a commercial option, provides an automatic technique that uses the history of the communications to whitelist the channels. Another example would be the Bluetooth Low Energy (BLE) technology, which is fortified with an adaptive blacklisting mechanism to exclude low-quality channels during the network operation. It is worth mentioning that adaptive blacklisting in BLE is centralized and performed by the master node in the network due to its star topology. However, in multi-hop large scale networks, the interference level of on

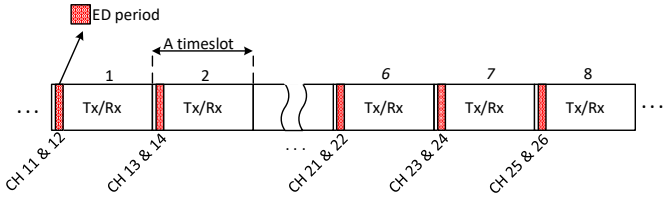


Fig. 1. General illustration of the operation of the NICE technique for channel noise probes.

different channels in various parts of the network may be quite different, making a decentralized or distributed channel blacklisting technique necessary.

Most channel blacklisting techniques are suitable for static environments where the wireless conditions do not often change. Dynamic environments, however, require a more robust approach that can adapt to the varying conditions. Adaptive Time-Slotted Channel Hopping (ATSCH) [12] is one of the earliest attempts to improve the performance of TSCH networks by analyzing the quality of the channels and blacklisting the ones with the poorest conditions. This technique reserves two timeslots in each slotframe in which none of the nodes are allowed to transmit. The coordinator takes this time to perform energy detection (ED) samples on two channels. The sampling results are then used to assess the quality of the channels, which is then used to select a fixed-size subset of them as HSL. These values are processed using an exponential smoothing operator as an estimate of the quality of the channels. ATSCH is criticized for the throughput cost of the reserved timeslots, as well as the low sampling rate of the interference levels on the channels.

ETSCH [13] proposes a new method for measuring the interference power levels on the channels called Non-Intrusive Channel Quality Estimation (NICE) [4]. This mechanism recognizes a silent period within each TSCH timeslot in which no packet transmission is expected within the TSCH network. The mechanism then uses this period in each timeslot to perform ED on two frequency channels. Fig. 1 illustrates the high-level operation of this technique. Having sixteen channels in total, it always takes eight timeslots for NICE to provide one sample of the noise or interference level of each channel (frequent sampling at a fixed rate with uniform distribution over time). Since no timeslot is reserved for the sake of energy detection for channel assessment, this technique does not impose any throughput overhead.

More recent studies take the analysis a step further and employ well-known metrics as a measure of quality for the channels. In [5], the authors compare several metrics, including RSSI, PRR, also known as Packet Delivery Ratio (PDR), and Packet Error Rate (PER). In [14], nodes occasionally transmit probe packets in the blacklisted channels to be able to estimate the PDR over those channels. Based on that, a channel may go from blacklist to the whitelist or vive versa. [15] proposes a mechanism to develop whitelists per timeslot to avoid internal collisions in the TSCH network. The used channel quality metric in this work is again the PDR of the links. [16] is another metric study, which adds Expected Trans-

mission Count (ETX)-based metrics to the analysis, as well as propose a novel blacklisting-aware transmission scheduling schema. These studies show that the metric of choice has a significant impact on the performance of the schemas.

Another family of studies address channel selection as a Multi-Armed Bandit (MAB) problem, a famous and well-studies problem in Reinforcement Learning. Essentially, a successful transmission on a channel is a *reward* and we aim to maximize our overall reward, by identifying the best channels. The authors of [17] propose MABO-TSCH, a complete framework for utilizing algorithms designed for the MAB problem in TSCH scheduling. [18] expands upon this work by comparing the performance of different MAB algorithms in various communication scenarios. However, this approach requires the nodes physically close to one another to be assigned disjoint HSLs to avoid internal interference.

All these studies have one thing in common; they look at the *recent* quality of the channels as an assessment for their behaviour in *future*. This can be a major drawback, especially in dynamic environments. For instance, in an in-vehicle network deployed in a moving vehicle, a burst of WiFi frames will cause a substantial drop in quality of several adjacent IEEE 802.15.4 channels resulting in around three channels to be labeled as *bad*. When the vehicle moves out of the interference range of the WiFi access point, the three blacklisted channels become good again. The averaging metrics will take some time to recover and rise again, leaving perfectly usable channels unnecessarily blacklisted for a longer time than necessary. Moreover, a not-well-tuned MAB algorithm would either react too harshly to such changes or too slow to be effective. We believe that a model capable of recognizing behavioural and transmission patterns is better able to react to such changes. To test this hypothesis, we design and train a DNN using publicly available experimental data sets. Then the channel quality assessment approach of a blacklisting technique (i.e., ETSCH) is replaced with the trained model to evaluate and investigate its effectiveness in highly dynamic networks and environments.

Reliable distribution of new HSL after any change in the set of blacklisted channels is challenging and imposes overhead, which partially depends on the network topology and scale. ETSCH, for instance, embeds HSLs into the enhanced beacon packets and then uses a small set of strongest frequency channels to more reliably transmit such beacons. This technique together with the single-hop topology considered in this mechanism help it to be able to frequently change HSL. An in-vehicle TSCH network can be a representative application as we consider in this work. MABO-TSCH, on the other hand, considers multi-hop TSCH networks for which HSL distribution is more complicated. Given the network specifications, the corresponding blacklisting mechanism needs to develop a suitable strategy for taking action based on the results of the channel quality prediction model.

IV. Deep Neural Networks

DNNs unparalleled ability in capturing complex statistical patterns within data has turned them into a standard tool for addressing numerous problems across various fields. A DNN consists of several typical building blocks, often refereed to as layers, arranged in a specific manner. Layers are then followed by a scaler *activation function*, introducing non-linearity to the network. Each layer has a set of adjustable parameters, also known as weights, which are tuned during *training* to produce the desired results.

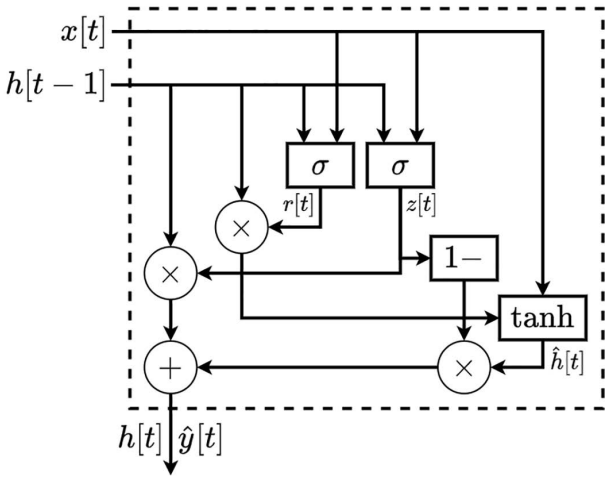


Fig. 2. The inner structure of a GRU neuron. The gates of the neuron, denoted as r and z, are adjusted in order to recognize the statistical patterns within the data that is fed to the layer.

If we denote the function represented by a neural network as $F(x)$ and the collective set of its parameters as W, training the network can be formulated as an optimization problem, shown in Eqn. 2. In this equation, J represents a *loss function*, a measure for how far off the outputs produced by the DNN are from the target values, y.

$$\min_{W} J\left(\hat{y}, y\right) = J\left(F\left(x\right), y\right) \tag{2}$$

Evidently, in cases where the data consists of x and y pairs, solving this problem is relatively straightforward and can be done using various gradient-based optimization algorithms. We refer to this method as a *supervised* training approach. However, acquiring the target values is an expensive process, requiring human intervention and expert knowledge to calculate. *Self-supervised* training is another approach that does away with target values. In a self-supervised setting, a *supervisory signal* is automatically generated (based on inputs), by baking (often problem-specific) constructs into the loss function, or the layers of the DNN itself.

Making predictions using a trained neural network, or as it is often called, *inference* through the network, requires far less resources than is needed for training it. With the typical resources accessible to low-end coordinators (edge devices), inference through small pre-trained networks that are loaded onto the module would be slow but possible. Training a network, however, cannot be expected from a typical coordinator and continuous training and refinement of a network would require custom hardware [19].

The most basic neural networks consist of fully-connected

layers, better known as feed-forward layers. Each neuron in a feed-forward layer returns a shifted (biased) linear combination of the output values of all the neurons in its preceding layer. The weights of this linear combination and the bias value can be learned from the data. This output is then passed through a smooth non-linear function, which is called the activation function of the layer. Activation functions enable ANNs to capture non-linear relations within the data.

The logistic function, also known as sigmoid, is a popular activation function, which is frequently used in the output layer of neural networks that are supposed to output a probability value. The definition of this function is given by Eqn. 3. The hyperbolic tangent function is another popular choice, but it is mostly used for the hidden layers of the network, i.e., the layers preceding the output layer.

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3}$$

The caveat to feed-forward layers is that a feed-forward layer can only be applied to an input vector of some fixed length. Our problem does not fit this criteria, since our goal is to predict the future behavior of the wireless links, using a sequence of vectors, describing the behaviour of the channels in the past. The model needs to be able to reason with regard to the changes in the channels' quality over time. Recurrent layers are a family of ANN layers that are specifically designed for working with sequential data. In recurrent layers, a feedback connection is added to each neuron, which allows it to access its output value for the data points that have come before when processing a new one.

Vanilla recurrent layers, the most basic type of recurrent layers, use multiplication for combining the results of the consecutive data points. This method can lead to uncontrollable fluctuations in the values in longer sequences, resulting in gradient signals that approach zero or gradient signals that rise to large numbers. Both cases are disastrous, and the optimizer fails to reach a solution if either of them is present.

Long Short-Term Memory (LSTM) [20] was one of the earliest and most successful attempts at tackling this issue. LSTM neurons use addition and several learnable gates over the data flow, which can mitigate the stated gradient problems. However, an LSTM layer is far more complicated than a vanilla recurrent layer and therefore requires much more computational resources for both training and inference. Gated Recurrent Units (GRUs) [21] are a simplified version of LSTMs that have fewer learnable weights. The structure of a GRU neuron is illustrated in Fig. 2.

In Fig. 2, r denotes the *reset gate* of the neuron, which is trained to drop any information they find useless in deriving the desired outputs and stopping them from flowing from previous data points into the next. z, on the other hand, is the *update gate* of the neuron, which specifies the amount of information the previous and current data points contribute to the output of the neuron. The exact formulations of these gates are given in Eqn. 4 [21]. In these equations, $\odot$ denotes the Hadamad product of the matrices. h and y can be used interchangeably, as they both represent the output value of the

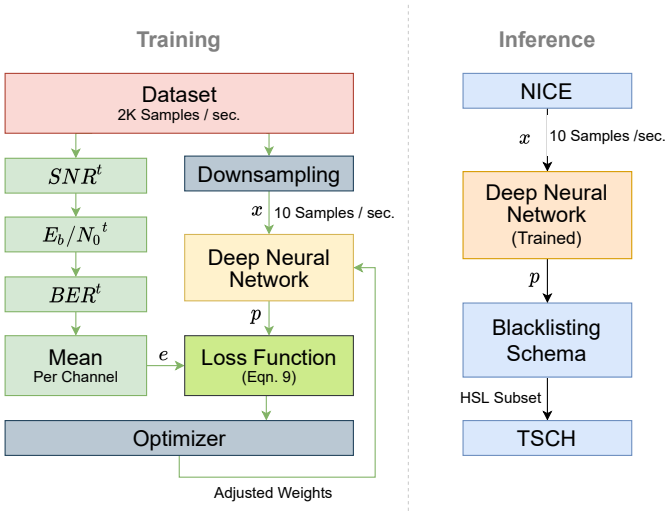


Fig. 3. The performed procedure during training and inference. $\boldsymbol{x}$ denotes the input matrix of the model, containing the interference power levels for the past $\boldsymbol{t_p}$ seconds. $\boldsymbol{p}$ and $\boldsymbol{e}$ denote two 16-D vectors, representing the outputs of the network and the supervisory signal, respectively.

neuron. W, U, and b are all adjustable weights that will be learned from the data.

$$\begin{aligned} z^t &= \sigma(W_z x^t + U_z h^{t-1} + b_z) \\ r^t &= \sigma(W_r x^t + U_r h^{t-1} + b_r) \\ \hat{h}^t &= \tanh(W_h x^t + U_h(r^t \odot h^{t-1}) + b_h) \\ h^t &= z^t \odot h^{t-1} + (1 - z^t) \odot \hat{h}^t \end{aligned} \tag{4}$$

It is important to note that any arrangement of layers and neurons cannot result in a working model. The structure of an ANN, which is often referred to as its architecture, must provide the model with the right amount of freedom to capture the patterns within the data. The process of finding the optimal architecture for a dataset and a problem is essentially a fine-tuning process; trial and error, and instinctive decisions.

V. Self-Supervised Model for Channel Quality Prediction

Our goal here is to train an ANN that can predict the channels' future behavior using measurements made prior to that moment. Fig. 3 illustrates the data flow through the model in training and inference. The model's input, x, is a sequence of interference power levels measured by the NICE mechanism, and its output, which we will denote as p, is a 16-D vector. The elements of this vector correspond to the TSCH channels. This design choice allows us to employ different blacklisting strategies. In scenarios where a certain quality of transmission is to be ensured, *bad* channels can be identified with respect to some threshold. On the other hand, if a certain number of channels need to be available at all times, the top channels can be chosen based on their scores.

To specify the length of x, we introduce a new parameter denoted as t_p. At time t_0, the x passed to the model contains the interference power levels recorded during the $[t_0 - t_p, t_0]$ interval. Larger values for t_p allow the model

to look further into the past. This might not be a desirable behaviour, especially in highly dynamic environments with rapid changes. Furthermore, the interference power levels need to be stored so that they can be processed by the network, and hence, increasing t_p increases the memory consumption of the system. Considering the sample rate of NICE (10 Hz), we found 5 seconds to be a suitable value for t_p.

Let e denote a 16-D vector, the *supervisory* signal for training the model. The elements of this vector are an indicator for the quality of their corresponding channels in near future. We introduce another timing parameter, to control how far into the future we look when calculating e. We denote this parameter as t_f, so that the interval would be $[t_0, t_0 + t_f]$. This parameter sets an upper bound for how long an HSL suggested by the network can be expected to be valid. Increasing t_f forces the network to make broad predictions corresponding to longer periods of time. This would reduce the frequency of HSL updates, which can be worth the reduced energy consumption depending on how dynamic the environment is. We found 5 seconds to also be a reasonable value for t_f in vehicular environments.

In order to calculate the vector e using the raw interference power levels, we resort to a *differentiable* simulation framework introduced in [7]. This would allow the optimization algorithm to interact with the simulation framework, and hopefully, better understand the dynamics involved in channel selection. Let us denote the transmission power for the communication as P_{tx}, and the interference power on the communication channel at time t as P^t_{int}. Like in [7], the path-loss model for short-range communications is used for estimating the strength of the transmitted signal at the location of the receivers. The Signal-to-Noise Ratio (SNR) for the given IEEE 802.15.4 link at time t can be formulated as in Eqn. 5.

$$\text{SNR}^t[dB] = P_{tx}[dBm] - \alpha\,(20.1 + 10\log d) - P^t_{int}[dBm] \tag{5}$$

In Eqn. 5, d is the distance between the nodes, and P_{tx} is the transmission power of the transmitter node. The parameter α is the path-loss exponent, related to the environment the link is operating in. In free space, the value of the path-loss exponent is $\alpha = 2$, while it gets some higher values for other environments. For intra-vehicular environments, in which the experiments are conducted and the dataset is collected, α has been reported to be around 3.5 [22]. We use this value in our calculations. We then need to estimate the Bit Error Rate (BER) of links, for which the notion of E_b/N_0 is used. For some given SNR, the E_b/N_0 of the transmission is calculated using Eqn. 6.

$${E_b/N_0}^t[dB] = \text{SNR}^t[dB] - 10\log\frac{f_b}{B} \tag{6}$$

where, f_b and B are bit-rate (in bit per second) and the bandwidth of the channel (in Hertz). Given this ratio, in the case of a QPSK modulation and AWGN channel, the BER at time t can be estimated using Eqn. 7.

$$\text{BER}^t = \frac{1}{2}\text{erfc}\left(\sqrt{{E_b/N_0}^t}\right) \tag{7}$$

In Eqn. 7, erfc denotes the complementary error function. If the transmission of a packet of length L (in bytes) starts at time t, we can calculate the Packet Reception Probability (PRP) using the formula in Eqn. 8.

$$\text{PRP}^t = \prod_{k=0}^{8L-1}\left(1 - \text{BER}^{t+4k[\mu s]}\right) \tag{8}$$

With the BERs of the link for the $[t_0, t_0 + t_f]$ time window, we can calculate a 16-D vector containing either the **maximum** or the **average** BER experienced when each channel was being used. Using the maximum values pushes the network towards predicting the worst possible behaviors of the channels, while an average encourages it to take the general behavior of the channel into perspective. If we denote the output vector of the neural network as p, the loss function for training the model can now be formulated as in Eqn. 9.

$$\min_W H\left(\ (1-p)\odot e\ ,\ 0\right) + \lambda\sum_{i=1}^{16} p_i \tag{9}$$

The bivariate function H presented in Eqn. 9 is known as Binary Cross-Entropy (BCE) which would rise in its value as its arguments diverge in theirs. The first argument of this function is expected to be a vector of probabilistic values, limited to the $[0, 1]$ range, while its second argument are often discrete labels, either 0 or 1. The mathematical definition of this function is provided in Eqn. 10.

$$H(p, y) = \sum_{i=1}^{n} y_i \log(p_i) + (1 - y_i)\log(1 - p_i) \tag{10}$$

In Eqn. 9, the first term of the loss function encourages the model to output higher scores for channels with bad qualities, to ensure that the dot-product approaches 0. However, the neural network can blindly output a vector of 1s as p for any given input to minimize the first term, meaning that the network would not actually learn to predict channels' behavior. Consequently, the second term of the loss function is added to penalize the network when it outputs high scores for too many channels. This penalty can be adjusted using a scalar coefficient, denoted as λ in Eqn. 9. With this penalty, the neural network is forced to output solutions that strike a balance between the number of channels that will be blacklisted, which would limit the number of parallel communications we can have, and the quality of the remaining channels, which would affect the probabilities of successful transmissions.

With this formulation, we can train a neural network with the criteria established earlier in this section. The neural network architecture consists of recurrent layers, for analyzing the temporal nature of the input sequences and extracting useful features from them, followed by feed-forward layers, which then have to map the extracted features into our desired scores. The details of the architecture might vary slightly from one case to another, e.g., when the environment's dynamics are more chaotic than usual and if the computation resources are available.

The model can be trained using any dataset that contains noise floor measurements on all 16 channels of TSCH. However, the exact amount of data required for training the model is hard to specify, as it depends on the data itself, as well as the architecture of the network, and the optimizer used for training it. As we previously stated, we downsample these measurements before feeding them to the network during training, to match the sample rate of NICE. However, we use the original measurements when calculating e for better precision. The trained model can be loaded onto the edge nodes in the TSCH network (coordinators) and may be invoked every t_f (or less) seconds. When invoked, it will receive the noise floors for the past t_p seconds, and return a 16-D vector, predicting how poor each channel is expected to perform. The blacklisting schema can use these scores to work out the new HSL.

VI. Training the Model

In our experiments, we used the publicly available real-world experimental dataset that we presented in [7]. The dataset is collected inside a moving vehicle in four different scenarios; while the vehicle was passing near some apartments, along an office area downtown, in a suburb area, and on a highway with no buildings around. The noise floors have been measured for 300 seconds, at a fixed sample rate of 2000 Hz, on all the 16 channels of TSCH in the 2.4 GHz band. The measurement setup consists of two stationary IEEE 802.15.4 nodes placed 3 meters apart. These diverse measurement scenarios can expose the strengths and weaknesses of our proposed technique in different environments. We train and evaluate three individual networks, one for each measurement scenario, to analyze our proposed approach against varying levels of interference and dynamic behavior.

In order to assess the generalization of the trained models to unseen data, the networks are trained using 80% of the measurement signals, corresponding to the first 240 seconds of each. The remaining 20% is left for the evaluation phase. We do not use the measurements made while driving on the highway due to a lack of sufficient interference that can have any significant adverse effect on the performance of TSCH. The interference signals are linearly downsampled to 10 Hz to match the sampling rate of the NICE mechanism. Furthermore, the values fed into the neural network were re-scaled so that the mean and the standard deviation of them approaches 0 and 1, respectively. This practice, which is better known as standardization, is often necessary for stable training. Having the input values distributed in a small neighborhood of zero results in better gradient signals, especially in the presence of activation functions such as the logistic sigmoid or the hyperbolic tangent [23].

We use the same architecture for all the measurement scenarios; two GRU layers, each containing 50 neurons, stacked on top of one another. They are then followed by a sigmoid-activated feed-forward layer with 16 neurons as the output of the network. The models are trained using the RMSprop optimizer [24], a special variant of gradient descent that speeds up the training process. The optimizer is set up with a learning rate of 10^{-4}, a smoothing constant of 0.99, and no momentum. We also experimented with the Adam optimizer [25], which is a famous enhancement of the gradient descent algorithm, but RMSprop proved to be more stable for our cases, compared to Adam. For models trained with mean BERs as e, $\lambda = 0.05$ properly balanced the two terms of the loss function. However, when using the maximum BERs of the channels as e, the BCE loss rose to signifacntly higher values. Consequently, we had to increase λ to 0.55. The models are trained for 1000 iterations, with a batch size of 32 samples. In the calculation of the BER values of the link, we assumed a low transmission power of -10 dBm, in order to study the performance of the protocol in low-power scenarios. All of our experiments were implemented in Python, using the open-source PyTorch library. The neural networks were trained on an NVIDIA GTX 1660Ti GPU (with 6GB of VRAM).

VII. Evaluation and Performance Results

To investigate the performance of our deep learning-based channel blacklisting, we test it in various network setups. Fig. 4 shows the performance of TSCH, ETSCH, and our method, Intelligent TSCH (ITSCH) over time, trained using both variations of the loss function discussed in Section V. Using the data recorded under the three measurement scenarios, we calculate the PRPs of the channels established by the four techniques, strictly in the last 20% of each recording. This period of time, which we refer to as the testing period, has not been observed by the neural networks during their training, and therefore, the networks' performance in this period is a good indicator of their generalization to unseen data. We further summarize these results in Fig. 5, where we provide the average PRP of the channels, a metric often referred to as the PRR of the channel.

In arguably the most dynamic measurement scenario in the dataset, downtown, both variations of ITSCH outperform standard TSCH and the ETSCH technique, improving the PRR of the channels by at most 21% and 8%, respectively. The same can be said for the suburb, although the fluctuations in this environment do not quite resemble those in downtown. The "bad" periods in the suburb measurements, periods of time where the performance of TSCH plummets, do not last as long as they do in downtown. Interestingly, unlike the downtown environment where the "mean" variate of the loss function performed the best, in the suburb, it is the "maximum" variate that achieves the highest PRR, 5% and 3% more than standard TSCH and ETSCH, respectively. Finally, in the apartments measurements where wild fluctuations in performance are far less common, ITSCH does not surpass ETSCH in PRR, falling short by 0.16%. However, the PRR of the channel is still 10% more than the one established by standard TSCH.

Overall, it can be concluded that in highly stochastic environments with high levels of interference, ITSCH is able to surpass ETSCH by a considerable margin. In scenarios where the fluctuations in channel qualities are relatively low, ITSCH manages to perform at least as well as ETSCH, and both of them far exceed the standard TSCH (no blacklisting). To further compare the three techniques, we estimate their

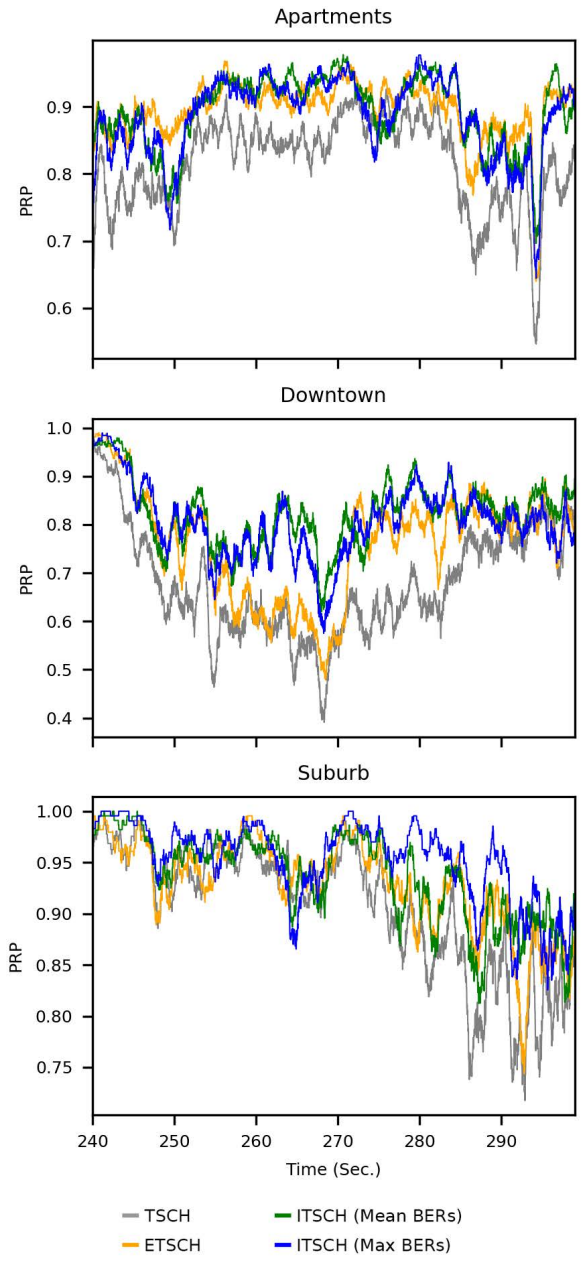


Fig. 4. The PRP values of the links established by each protocol over time. The schemas powered by a DNN manage to generally perform as well or better than TSCH and ETSCH. These values are further summarized in Fig. 5.

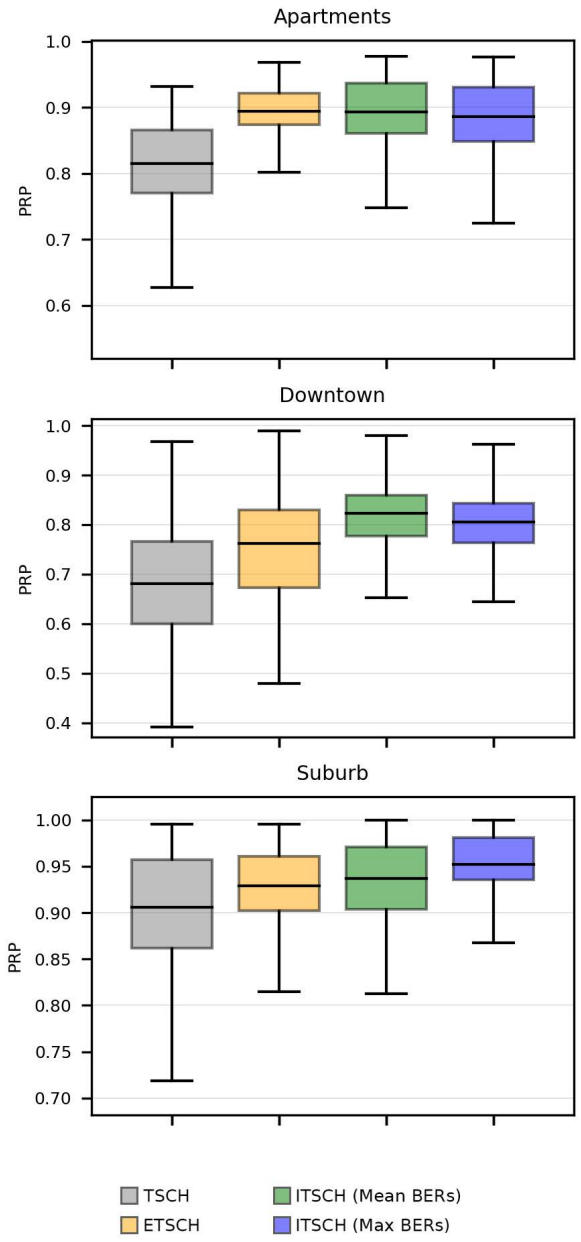


Fig. 5. The performance of standard TSCH, ETSCH, and ITSCH, using both variations of the loss function, under different simulation scenarios. Each box-plot displays the average PRP value, along with the quartiles of the values. The average PRP of the links is often refereed to as the PRR or the PDR of the link.

energy consumption, an important factor when we consider the networks in which TSCH is primarily used. In this regard, ITSCH seems to be at a disadvantage, having to carry out the calculations needed for a neural network. However, the coordinator of the network is the only node that has to deal with the model, and coordinators are often not under the same constraints as the other low-power wireless sensor nodes. The architectures used in this study are small enough so that the calculations can take place with the limited resources a coordinator has access to. As such, we can omit the impact of the neural network in our calculations, leaving only the transceivers in the nodes. We can estimate the energy consumed by a transceiver during a single slotframe using Eqn. 11 [13].

$$E = \left[I_{ED} N_{ED} T_{ED} + \frac{1}{\text{PRR}} \left[(I_{Rx} N_{Rx} T_{Tx}) + (I_{Tx} N_{Tx} T_{Tx}) \right] \right] \times V_{cc} \tag{11}$$

In this equation, the first term in the brackets corresponds to the energy consumption related to the ED measurements, which are done for the sake of channel quality estimation. I_{ED}, N_{ED}, and T_{ED} denote the current drawn by the transceiver during an ED, the number of EDs performed in

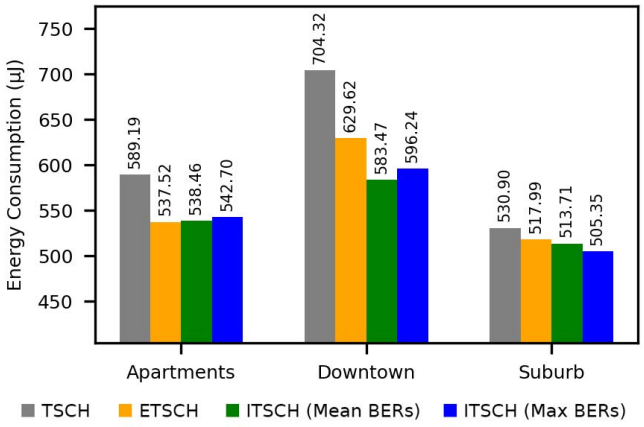


Fig. 6. The energy consumption of TSCH, ETSCH, and our work (labeled ITSCH), both using mean BERs and maximum BERs in the loss function, under various simulation scenarios.

a slotframe, and the duration of time it takes to perform an ED, respectively. The second term in the brackets estimates the energy consumed by both ends of a transmission, using similar notations (I_{Rx} and I_{Tx} stands for current consumption in the receive and transmit modes, N_{Rx} and N_{Tx} are the number of received and transmitted packets in a slotframe, and T_{Tx} is the transmission time of a packet). Evidently, channels with lower PRRs consume more energy due to the re-transmissions. The term V_{CC} denotes the operating voltage of the transceiver. We use the same values for the currents, time durations and voltages as in [13]. The results are presented in Fig. 6.

Since, standard TSCH does not perform EDs, N_{ED} is set to zero for it. However, both ETSCH and ITSCH, which perform EDs regularly with the same setting, consume less energy than TSCH, showing that the extra energy consumption is more than made up for by the boost to channel PRR. As a matter of fact, the only different variable when calculating the energy consumption of ETSCH and ITSCH variants is the PRR of the link, meaning that the differences in energy consumption between the two follow the same trend as their reception rates. Consequently, in downtown and suburb, where ITSCH variants outperform ETSCH with regard to average PRP, the energy consumption of the technique is lower as well. A similar argument can be made for the third scenario, apartments, for which ITSCH performs as good as ETSCH, and hits the same energy consumption levels as ETSCH, as expected.

VIII. Conclusion

In this paper, we present a self-supervised learning approach for training a Deep Neural Network (DNN) that can replace traditional channel quality assessment metrics used in adaptive channel blacklisting techniques for IEEE 802.15.4 Time-Slotted Channel Hopping (TSCH) networks. To predict the future behavior of wireless frequency channels, deep neural networks are designed and trained. The presented approach allows one to train a DNN for any environment using raw recordings of interference power levels on all 16 channels of TSCH on the 2.4GHz ISM band. Using the neural network predictions, the coordinator can periodically select a subset of the channels for the hopping mechanism.

We substituted the quality assessment metric used in Enhanced TSCH (ETSCH), an improvement to TSCH that stays close to the original protocol and still manages to significantly improve the reliability of the network, with a DNN. We then train and evaluate the modified blacklisting schema using real-world experimental data collected in three different environments. The schema performs as well as ETSCH in the environments with lower average interference levels, but improves the reliability of the network by 8% and 21% over ETSCH and TSCH, respectively, in a highly dynamic environment.

References

[1] IEEE, "IEEE Standard - Part 11: Wireless LAN MAC and PHY Specifications," IEEE, Tech. Rep., 2007, IEEE Std 802.11.

[2] ——, "IEEE Standard - Part 15.1: MAC and PHY Specifications for Wireless Personal Area Networks (WPANs)," IEEE, Tech. Rep., 2005, IEEE Std 802.15.1.

[3] ——, "IEEE Standard for Low-Rate Wireless Networks," IEEE, Tech. Rep., 2020, IEEE Std 802.15.4-2020 (Revision of IEEE Std 802.15.4-2015).

[4] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, "Enhanced Time-Slotted Channel Hopping in WSNs Using Non-intrusive Channel-Quality Estimation," in *2015 IEEE 12th International Conference on Mobile Ad Hoc and Sensor Systems*. IEEE, oct 2015.

[5] A. Elsts, X. Fafoutis, R. Piechocki, and I. Craddock, "Adaptive channel selection in IEEE 802.15.4 TSCH networks," in *2017 Global Internet of Things Summit (GIoTS)*. IEEE, 2017, pp. 1–6.

[6] N. Baccour et al, "Radio link quality estimation in wireless sensor networks: A survey," *ACM Trans. Sen. Netw. (TOSN)*, vol. 8, no. 4, 2012.

[7] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, "An Experimental Study of Cross-technology Interference in In-Vehicle Wireless Sensor Networks," in *Proceedings of the 19th ACM/IEEE International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. ACM, 2016.

[8] T. Watteyne, A. Mehta, and K. Pister, "Reliability through frequency diversity," in *Proceedings of the 6th ACM symposium on Performance evaluation of wireless ad hoc sensor, and ubiquitous networks - PE-WASUN '09*. ACM Press, 2009.

[9] K. Pister and L. Doherty, "TSMP: Time synchronized mesh protocol," *IASTED Distributed Sensor Networks*, vol. 391, p. 398, 2008.

[10] H. C. F. Std., "HART Field Communication Protocol Specification, Revision 7.1," IEEE, Tech. Rep., 2008.

[11] I. S. of Automation Std., "ISA100.11a: 2009 Wireless systems for industrial automation: Process Control and Related Applications," ISA, Tech. Rep., 2009.

[12] P. Du and G. Roussos, "Adaptive time slotted channel hopping for wireless sensor networks," in *4th Computer Science and Electronic Engineering Conference (CEEC)*. IEEE, sep 2012.

[13] R. Tavakoli, M. Nabi, T. Basten, and K. Goossens, "Dependable interference-aware time-slotted channel hopping for wireless sensor networks," *ACM Transactions on Sensor Networks (TOSN)*, vol. 14, no. 1, January 2018.

[14] P. C. V. Kotsiou, G. Z. Papadopoulos and F. Theoleyre, "Label: Link-based adaptive blacklisting technique for 6tisch wireless industrial networks," in *Proceedings of the 20th ACM/IEEE International Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems (MSWiM)*. ACM, 2017.

[15] ——, "Whitelisting without collisions for centralized scheduling in wireless industrial networks," *IEEE Internet of Things Journal*, vol. 6, no. 3, pp. 5713–5721, 2019.

[16] V. Kotsiou, G. Z. Papadopoulos, D. Zorbas, P. Chatzimisios, and F. Theoleyre, "Blacklisting-based channel hopping approaches in low-power and lossy networks," *IEEE Communications Magazine*, vol. 57, no. 2, pp. 48–53, 2019.

[17] P. H. Gomes, T. Watteyne, and B. Krishnamachari, "MABO-TSCH: Multihop and blacklist-based optimized time synchronized channel hopping," *Transactions on Emerging Telecommunications Technologies*, vol. 29, no. 7, 2017.

[18] H. Dakdouk, E. Tarazona, R. Alami, R. Féraud, G. Z. Papadopoulos, and P. Maillé, "Reinforcement Learning Techniques for Optimized Channel Hopping in IEEE 802.15.4-TSCH Networks," in *Proceedings of the 21st ACM International Conference on Modeling Analysis and Simulation of Wireless and Mobile Systems*. ACM, oct 2018.
[19] S. Branco, A. G. Ferreira, and J. Cabral, "Machine learning in resource-scarce embedded systems, fpgas, and end-devices: a survey," p. 1289, 2019.
[20] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, nov 1997.
[21] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, November 2014.
[22] C. U. Bas and S. C. Ergen, "Ultra-wideband Channel Model for Intra-vehicular Wireless Sensor Networks Beneath the Chassis: From Statistical Model to Simulations," *IEEE Transactions on Vehicular Technology*, vol. 62, no. 1, pp. 14–25, January 2013.
[23] Y. A. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Efficient BackProp," in *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, 2012, pp. 9–48.
[24] T. Tieleman and G. Hinton, "Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude," *COURSERA: Neural networks for machine learning*, vol. 4, no. 2, pp. 26–31, 2012.
[25] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

Mohammad Farahmand is a student of Computer Engineering at Isfahan University of Technology. Throughout his years, he has been a research assistant in the field of Generative Adversarial Networks, as well as practical applications of deep Learning in cancer treatment. During an internship program, he collaborated with Isfahan University of Medical Sciences. For his thesis, he has investigated SOTA convolution-based single-image super-resolution techniques in a survey study. His research interests include deep learning and its applications in image processing, healthcare, etc. as well as theoretical deep learning, mainly optimization algorithms, transformers (in vision) and generative adversarial networks.

Majid Nabi (S08-M13) received the B.Sc. and M.Sc. degrees both in computer engineering from Isfahan University of Technology and Tehran University, Respectively. He received the Ph.D. degree in electrical and computer engineering from Eindhoven University of Technology (TU/e), Eindhoven, the Netherlands in 2013. He is currently an assistant professor with the Department of Electrical Engineering at TU/e, and Isfahan University of Technology. His research interests include efficient and reliable networked embedded systems, low-power wireless sensor networks, and internet-of-things. He is a member of IEEE.