Neural Network Aided Decoding for Physical-Layer Network Coding Random Access

(Invited Paper)

Adriano Pastore*, Paul de Kerret†, Monica Navarro*, David Gregoratti*, and David Gesbert†

*CTTC/CERCA, Avinguda Carl Friedrich Gauss 7, 08860 Castelldefels, Spain
†EURECOM, 450 route des Chappes, 06410 Biot, Sophia Antipolis, France

***Abstract*—Hinging on ideas from physical-layer network coding, some promising proposals of coded random access systems seek to improve system performance (while preserving low complexity) by means of packet repetitions and decoding of *linear combinations* of colliding packets, whenever the decoding of individual packets fails. The resulting linear combinations are then temporarily stored in the hope of gathering enough linearly independent combinations so as to eventually recover all individual packets through the resolution of a linear system at the end of the contention frame. However, it is unclear which among the numerous linear combinations—whose number grows exponentially with the degree of collision—will have low probability of decoding error. Since no analytical framework exists to determine which combinations are easiest to decode, this makes the case for a machine learning algorithm to assist the receiver in deciding which linear combinations to target. For this purpose, we train neural networks that approximate the error probability for every possible linear combination based on the estimated channel gains and demonstrate the effectiveness of our approach by numerical simulations.**

I. Introduction

With the increasing importance of machine-to-machine communication in the context of the Internet of Things (IoT), many *ad hoc* wireless networks are witnessing a drastic densification. In typical massive machine-to-machine communication settings, numerous terminals with bursty traffic demands and scarce coordination seek to convey information packets to a central node. Early concepts of random access such as the legacy ALOHA scheme [1], [2] were, like most medium access schemes in packet-switched networks, based on the paradigm of collision avoidance [3], [4]. However, the more crowded the wireless medium, the more necessary it becomes to devise protocols that are capable of resolving collisions rather than discarding them. In addition, harsher delay constraints and limitations on feedback capabilities in many IoT-related applications (Industry 4.0, vehicle-to-vehicle communication, etc.) make retransmissions impractical or costly at best. As a result, rather than treating collisions as losses, modern proposals of grant-free random access protocols have evolved towards the use of more sophisticated forward error correction techniques, often referred to as multipacket reception (MPR) [5]–[10].

Of notable importance among well-known MPR techniques is *successive interference cancellation* (SIC) for its low complexity as compared to joint decoding: when one user experiences a substantially stronger channel gain than others, its packets can be decoded and subtracted (cancelled) from the received signal [11], [12], thereby potentially allowing more subsequent decoding operations or conflict resolutions. Liva's work [11] on Contention Resolution Diversity Slotted ALOHA (CRDSA) shows that the contention resolution with SIC is analogous to iterative belief-propagation (BP) erasure decoding and can thus be built upon established BP code designs. In the context of ultrareliable low-latency communications, [13] characterizes the reliability–latency performance of frameless ALOHA with SIC decoding.

Beyond SIC, another promising strategy, which is inspired by physical-layer network coding (PLNC), consists in encoding and decoding *linear combinations* of packets [14]–[17]. By collecting enough linearly independent packet combinations within a contention period (frame), the receiver might be able to resolve all individual packets. The motivation behind this approach is that, where SIC fails, the decoding of linear combinations might still succeed and unlock useful information for contention resolution. As another advantage over CRDSA, this PLNC random access scheme effectively implements some form of fast frame-level contention resolution, yet without the need of storing raw receive signals until the end of the contention period. The authors of [18], [19] also verified a notable performance advantage of PLNC random access over purely SIC based schemes.

However, given the large number of possible linear combinations—which is exponential in the number of colliding packets—and the fact it is unclear *a priori* which ones can be reliably decoded, the brute-force approach of attempting to decode them all is prohibitive. In this article, we propose to enhance the decoding architecture with a machine learning algorithm based on deep neural networks (DNN) that determines the linear combinations that can be most reliably decoded. Such an algorithm is a necessary step in making PLNC random access practically feasible.

This work was supported by the Catalan Government under grant 2017 SGR 1479 and by the Spanish Ministry of Economy and Competitiveness through projects TEC2014-59255-C3-1-R (ELISA) and TEC2015-69868-C2-2-R (ADVENTURE). D. Gesbert and P. de Kerret are supported by the European Research Council under the European Union's Horizon 2020 research and innovation program (Agreement no. 670896).

II. System description

A. PLNC coded random access scheme

We focus on the coded random access scheme from [15], which was adapted and proposed for massive access connectivity within European project [20], [21]. We generally adopt the corresponding numerology, though all considerations in this paper can be straightforwardly extended to different parameter values.

Consider an L-user slotted random access channel with a user index set $[L] = \{1, \ldots, L\}$. All signals are in real-valued baseband representation and indexed by discrete time indices $t \in \mathbb{Z}$ which are structured as follows: a *slot* is a group of $T_\mathsf{s} = 168$ consecutive time instants, whereas a *frame* is a group of $T_\mathsf{f} = 10$ consecutive slots. We assume that a control loop allows the terminals to time their transmissions in such way that the received signals are precisely synchronized on symbol, slot and frame level.

In each frame, a random subset $\mathcal{A} \subseteq [L]$ of so-called *active* users attempt to convey their payload data, while other users remain silent. Each active user is set to transmit r replicas of a *packet*[1] in r randomly chosen slots out of the T_f slots that compose the frame, while remaining silent in other slots.[2] A packet consists of a preamble of length $T_\mathsf{p} = 40$ symbols that carry a unique user signature that serves simultaneously for user identification and channel estimation[3] (e.g., via orthogonal matching pursuit at the receiver), and a payload of length $n = 128$, which contains the codeword to be transmitted. As a result, in each slot, a random subset $\mathcal{T} \subseteq \mathcal{A}$ of active users will be simultaneously transmitting their packets.[4] We say that $|\mathcal{T}|$ is the *collision degree* in that slot, where $|\cdot|$ denotes set cardinality. The resulting transmission pattern is depicted in Figure 1.

At time instant $t \in \mathbb{Z}$ pertaining to a slot in which users $\mathcal{T}$ are transmitting payload data, we have that, conditioned on transmit symbols $X_\ell = x_\ell \in \mathbb{R}$, $\ell \in \mathcal{T}$, the channel output is given by

$$Y(t) = \sum_{\ell \in \mathcal{T}} h_\ell x_\ell(t) + Z(t) \tag{1}$$

where $Z(t) \sim \mathcal{N}(0, 1)$ is additive white Gaussian noise. We assume BPSK signaling with power P, hence $X_\ell \in \{-\sqrt{P}, +\sqrt{P}\}$. The channel gains $h_\ell \in \mathbb{R}$ are drawn from some random distribution (to be specified) and are assumed to stay constant over the duration of a slot, but they vary independently from one slot to the next.

[1]The number r of packet repetitions can be optimized in line with [11].

[2]We talk of *coded* random access due to this repetition coding, which the receiver can exploit at frame level to resolve collisions.

[3]In general, the tasks related to the preamble processing (multiuser detection, user identification, collision detection, channel estimation) impact overall performance and resource allocation strategies. Thus, some works treat them as an essential part of the random access scheme. For simplicity however, in this work we assume that they are *genie-aided*, so we can focus entirely on payload decoding and leave a joint detection–decoding perspective for future work.

[4]Hence $\mathcal{T}$ varies from slot to slot, and $\mathcal{A}$ from frame to frame, yet we refrain from specifying the slot/frame indices so as to not encumber notation.

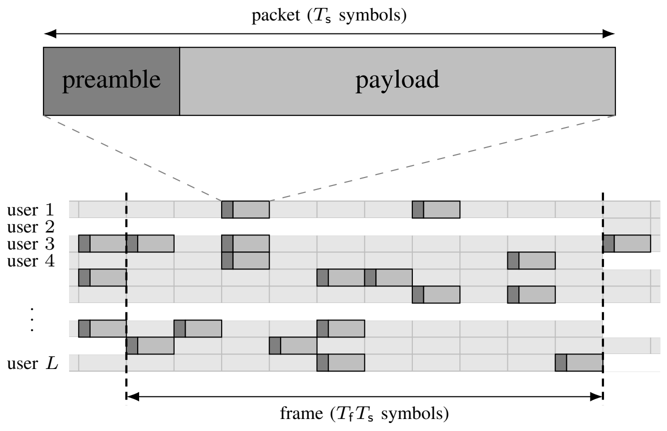
packet (T_s symbols)
preamble
payload
user 1
user 2
user 3
user 4
user L
frame ($T_\mathsf{f}T_\mathsf{s}$ symbols)

Fig. 1. Exemplary illustration of the slot and frame traffic pattern in the PLNC coded random access system under study. For this picture, we have set $T_\mathsf{f} = 10$ and $r = 2$. Every row in the above grid-like structure depicts the activity of one user over time. Frames in which a user is idle are blanked out.

B. Coding and modulation

All users employ the same rate-$1/2$ short LDPC code with message size $k = 64$ bits and codeword length $n = 128$ bits [22]. Since linear binary codebooks are closed under modulo-2 addition, any linear combination

$$\boldsymbol{w}_{\boldsymbol{a}} = \bigoplus_{\ell \in \mathcal{T}} a_\ell \boldsymbol{u}_\ell \tag{2}$$

of codewords $\boldsymbol{u}_\ell \in \{0, 1\}^n$ (as row vectors) with binary coefficients $a_\ell \in \{0, 1\}$, $\ell \in \mathcal{T}$, belongs to the codebook. The received payload signal can be expressed in vector notation as

$$\boldsymbol{y} = \sum_{\ell \in \mathcal{T}} h_\ell \boldsymbol{x}(\boldsymbol{u}_\ell) + \boldsymbol{z}. \tag{3}$$

The modulation mapping $\boldsymbol{x}(\boldsymbol{u}_\ell) = \big[x(u_{\ell,1}), \ldots, x(u_{\ell,n})\big]$ is such that code bits are mapped to BPSK symbols as $x(0) = -\sqrt{P}$ and $x(1) = +\sqrt{P}$.

C. Decoding

The decoding procedure is performed at slot level and at frame level.

1) Slot-level decoding: The slot-level decoding is performed in two successive stages:

a) Successive interference cancellation (SIC): Within each time slot, based on a vector of channel gains $\boldsymbol{h}$ and a receive vector $\boldsymbol{y}$, the slot-level decoder attempts to recover as many individual codewords $\boldsymbol{u}_\ell$ as possible by means of BP decoding. The users are decoded in order of decreasing channel gains and their interference is successively subtracted from the observation $\boldsymbol{y}$.

b) Sum decoding (SD): Suppose that after some SIC iterations, a subset $\mathcal{T}' \subseteq \mathcal{T}$ of packets remains undecoded and the BP decoding fails for the next packet in line. Based on the channel coefficients $\boldsymbol{h}$ and the remaining signal $\boldsymbol{y}' = \boldsymbol{y} - \sum_{\ell \in \mathcal{T} \setminus \mathcal{T}'} h_\ell \boldsymbol{x}(\hat{\boldsymbol{u}}_\ell)$, the sum decoder now attempts to recover as many codeword combinations $\boldsymbol{w}_{\boldsymbol{a}}$ as possible, where the weight vector $\boldsymbol{a} = [a_1, \dots, a_{\mathcal{T}'}] \in \{0,1\}^{|\mathcal{T}'|}$ is varied through all possibilities. There are $2^{|\mathcal{T}'|} - |\mathcal{T}'| - 1$ such weight vectors to be considered, since we exclude the all-zero vector and the $|\mathcal{T}'|$ singleton vectors (which are not decodable due to the SIC routine having terminated). This sum decoding is done by a variation of the BP decoder (see Section 4.4 in [15] and Section III in [23] for details). The usefulness of sum decoding is best visualized by the histogram in Figure 2, which makes it evident that even when SIC fails, a large amount of codeword combinations can still be reliably decoded and are potentially helpful for frame-level decoding.

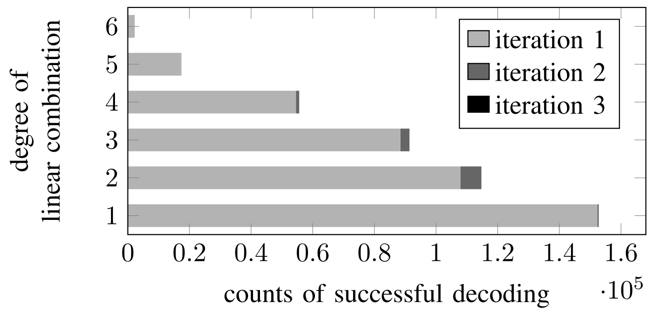

Fig. 2. Counts of successful decoding for slot-level decoding of $|\mathcal{T}| = 6$ colliding packets, based on a simulation with 400 000 slots, user-symmetric Rayleigh fading and average SNR of 15 dB. The *degree of linear combination* (ordinate axis label) stands for the Hamming weight of $\boldsymbol{a}$.

In the experiment from Figure 2, the receiver proceeds as follows: in the first iteration, it exhaustively attempts to decode all $2^{|\mathcal{T}|} - 1 = 63$ combinations (including singleton vectors), then cancels whatever packets he might have fully decoded (corresponding to singleton vectors), and moves on to repeating the exhaustive decoding attempts on the remaining set of packets (second iteration), etc. We observe that the third iteration provides negligible decoding progress if any.[5] In turn, the mass of linear combinations counts (degrees 2 through 6 combined) is rather substantial when compared to degree-1 counts, with degree-2 combinations yielding highest counts. This hints at the significant potential of the PLNC approach.

2) Frame-level decoding: After the slot-level decoder has been run for all T_{f} slots that constitute a frame, a collection of decoded codeword combinations $\hat{\boldsymbol{w}}_{\boldsymbol{a}^{(1)}}, \hat{\boldsymbol{w}}_{\boldsymbol{a}^{(2)}}, \dots$ are available, with different weight vectors $\boldsymbol{a}^{(j)} = \{0,1\}^{|\mathcal{A}|}$. Let us stack these row vectors into a weight matrix

$$\boldsymbol{A} = \begin{bmatrix} \boldsymbol{a}^{(1)} \\ \boldsymbol{a}^{(2)} \\ \vdots \end{bmatrix} \tag{4}$$

and denote $\mathcal{A}(i)$ the i-th ordered entry of the set of active users $\mathcal{A} \subseteq [L]$, e.g., $\mathcal{A} = \{2, 4, 5, \dots\}$ would yield $\mathcal{A}(1) = 2$, $\mathcal{A}(2) = 4$, $\mathcal{A}(3) = 5$, etc. Choosing the matrix of decoded linear combinations $\hat{\boldsymbol{W}} = [\hat{\boldsymbol{w}}^{\mathsf{T}}_{\boldsymbol{a}^{(1)}}, \hat{\boldsymbol{w}}^{\mathsf{T}}_{\boldsymbol{a}^{(2)}}, \dots]^{\mathsf{T}}$ as an estimator for the true linear combinations

$$\boldsymbol{W} = \begin{bmatrix} \boldsymbol{w}_{\boldsymbol{a}^{(1)}} \\ \boldsymbol{w}_{\boldsymbol{a}^{(2)}} \\ \vdots \end{bmatrix} = \boldsymbol{A} \begin{bmatrix} \boldsymbol{u}_{\mathcal{A}(1)} \\ \boldsymbol{u}_{\mathcal{A}(2)} \\ \vdots \end{bmatrix}, \tag{5}$$

the frame-level decoder attempts to recover the individual messages. If $\boldsymbol{A}$ has rank $|\mathcal{A}|$, the linear system can be fully solved by means of

$$\hat{\boldsymbol{u}} = (\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathsf{T}}\hat{\boldsymbol{W}}, \tag{6}$$

all operations being performed in the binary field.

In general, even if the rank of $\boldsymbol{A}$ is less than $|\mathcal{A}|$, the decoder may still be able to recover some *subset* of the codewords. For a systematic treatment of these situations, it is convenient to put $\boldsymbol{A}$ into reduced row echelon form $\boldsymbol{A}_{\mathrm{RREF}} = \boldsymbol{R}\boldsymbol{A}$ by Gaussian elimination via elementary row operations described by a square full-rank matrix $\boldsymbol{R}$. In RREF, the first 1 of each row (called the *pivot*) is located strictly to the right of the pivot of the previous row, and every pivot is the only non-zero entry in its corresponding column. For example,

$$\boldsymbol{A}_{\mathrm{RREF}} = \begin{bmatrix} 1 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \tag{7}$$

is in RREF. If in addition, a pivot is the only non-zero entry of its row, then the corresponding codeword can be recovered. In the above example, $\boldsymbol{u}_{\mathcal{A}(3)}$ and $\boldsymbol{u}_{\mathcal{A}(6)}$ can be recovered, whereas $\boldsymbol{u}_{\mathcal{A}(1)}, \boldsymbol{u}_{\mathcal{A}(2)}, \boldsymbol{u}_{\mathcal{A}(4)}, \boldsymbol{u}_{\mathcal{A}(5)}, \boldsymbol{u}_{\mathcal{A}(7)}$ cannot.

III. Decoding decisions via Deep Neural Networks

The main difficulty in the SD step stems from the fact that the number of linear combinations to attempt increases exponentially with the degree of collision. As argued before, there are $2^{|\mathcal{T}'|} - |\mathcal{T}'| - 1$ relevant combinations, which makes exhaustive trials impractical.

Unlike SIC, in which it is well known that users should be decoded in order of decreasing channel gains, for SD there is no such simple rule of thumb, nor any known analytical way to evaluate or sort the probabilities of decoding success. Thus, using a machine learning algorithm to *learn* to predict which linear combinations can be reliably decoded appears as a promising approach to this problem. The machine learning approach is suitable in this context for several reasons: (i) the computation complexity is shifted to the training phase and allows for highly efficient real-time implementation, (ii) the structure of the problem will remain very similar when the number of users or the distribution of the channel changes, thus offering an opportunity to *transfer* the trained models,[6] (iii) the function to be approximated is expected to be fairly smooth and well-behaved, and hence easy to learn.

We introduce a function $\boldsymbol{p} = g(\boldsymbol{h})$ which takes the channel gains $h_\ell,\ \ell \in \mathcal{T}'$ as inputs and returns the vector $\boldsymbol{p}$ containing

[5]For linear combination degrees 1 through 6, the third iteration counts were 2, 19, 4, 0, 0, 0, respectively, and are thus barely visible on the histogram.

[6]Note however that this aspect lies beyond the scope of this paper and will be investigated in future work

the $2^{|\mathcal{T}'|}-1$ probabilities of successful decoding corresponding to each linear combination, i.e.,

$$p_i = \mathsf{Pr}\{\hat{\boldsymbol{w}}_{\boldsymbol{a}(i)} = \boldsymbol{w}_{\boldsymbol{a}(i)} | \boldsymbol{h}\}, \quad i = 1, \ldots, 2^{|\mathcal{T}'|} - 1 \tag{8}$$

where $\boldsymbol{a}(i)$ denotes the length-$\mathcal{T}'$ binary expansion (as a row vector) of i.[7]

If cognizant of $\boldsymbol{p}$, the decoder will only attempt to decode the most promising combinations according to some heuristic. For instance, the decoder could choose those combinations $\boldsymbol{a}(i)$ whose probabilities p_i lie above a given threshold τ, or it could choose the topmost (in terms of high p_i) ν combinations (where ν is a fixed parameter) which are guaranteed to increase the rank of $\boldsymbol{A}$ by ν.

We will approximate g using a DNN with parameter $\boldsymbol{\theta}$ and we will denote the obtained function by $g^{\boldsymbol{\theta}}$. Beforehand, we will give a very concise introduction to supervised learning using DNNs in order to explain the general approach, before describing the training procedure in further detail.

A. Refresher on supervised learning with DNNs

A DNN is a function that can be expressed as the concatenation of several non-linear functions—so-called *layers*, where the j-th layer produces n_j real-valued outputs (nodes). Each of these n_j outputs is obtained from a so-called *neuron*, which takes a linear combination of the outputs of the previous layer, followed by the application of a non-linear *activation function*. Specifically, let us denote the output of the i-th node of layer j by y_i^j and the activation function by Φ. The output y_i^j is then given by

$$y_i^j = \Phi\left(\sum_{i=1}^{n_{j-1}} \alpha_i^{j-1} y_i^{j-1} + \beta_i^{j-1}\right) \tag{9}$$

where α_i^j and β_i^j are the so-called *weights* are *biases*, which constitute the parameters of the DNN (collectively referenced by $\boldsymbol{\theta}$) that need to be *trained*.

In supervised learning, we aim to *learn* a mapping $\boldsymbol{x}_i \mapsto g(\boldsymbol{x}_i)$ from a training data set $\{(\boldsymbol{x}_i, g(\boldsymbol{x}_i))\}_{i=1,2,\ldots}$. For supervised deep learning, we choose a neural network size (number of layers and nodes) and activation function(s) and then seek to adjust (train) parameters $\boldsymbol{\theta}$ so that $g^{\boldsymbol{\theta}}$ approximates g well. Clearly, the input dimension of the first layer and the output dimension of the last layer have to be compatible with the input-output dimensions of the function g to be approximated. The inner layers are referred to as *hidden layers*. Ideally, activation functions are chosen so as to reach the desired approximation accuracy at the fastest rate. A very popular activation function is the so-called rectified linear unit function

$$\mathrm{ReLU}(z) = \max(z, 0) \tag{10}$$

due to the simplicity of its derivative (which facilitates training via the backpropagation algorithm) and the fact that it has proven to be highly efficient in a large number of applications. In fact, DNNs have been known for many years but were notoriously difficult to train until recent breakthroughs both in terms of hardware and algorithmic efficiency of training [24].

It is important to understand that a DNN contains many parameters (so-called *hyperparameters*) whose tuning is essential for an efficient training (e.g., number of nodes, layers, learning step, etc.). How to optimally design a DNN to learn a specific task is the focus of current research efforts in the field. In the present work, we show a new interesting application of DNNs and showcase a first working scheme. Yet, our design choices are led by a limited trial-and-error approach to improve the accuracy of training. Fine-tuning of the method herein presented will be the focus of future research.

B. Training Parameters

In the following, the function $g^{\boldsymbol{\theta}}$ will be obtained from a DNN with 3 hidden layers each containing 50 neurons each, initialized independently with samples from a zero-mean Gaussian distribution with variance 0.05 to avoid saturation of the coefficients. For convenience, since we will fix the number of active users to 6 for our simulations, we train 6 separate DNNs, one for each degree of collision d (from 1 to 6). Each DNN is trained with a training set of 10^5 channel realizations, each consisting of a real-valued vector $\boldsymbol{h}_i \in \mathbb{R}^d$, $i = 1, \ldots, 10^5$, $d = 1, \ldots, 6$. The channel gains are distributed according to a Rician distribution with a Rician factor of $|\mathbb{E}[h]|^2/\mathbb{E}[|h|^2] = 0.9$. Each label in the training data consists of a binary vector $\boldsymbol{s}_i \in \{0,1\}^{2^d-1}$ which records success (one) or failure (zero) of one encoding–decoding operation for each of the $2^d - 1$ possible linear codeword combinations. Hence our training data set is $\{(\boldsymbol{h}_i, \boldsymbol{s}_i)\}_{i=1}^{10^5}$.

Note that this training set does not contain the output values of the function g to be approximated. Instead, the training samples contain *realizations* of a random binary vector whose marginal probabilities are described by the outputs of g. By large sampling, we manage to approximate the function g.

We have implemented the DNN using the Tensorflow library which allows for an easy implementation of the DNN training, in particular thanks to automated differentiation routines. The training was carried out using the Adam-gradient descent algorithm with 10^5 iterations and a gradient step size of 1.001.

To assess the DNN's performance, we have computed the empirical frequencies of false alarm and missed detection (based on the training set statistics) as

$$\mathsf{P}_{\mathrm{FA}} = \frac{|\{(i,j)\colon s_{ij} = 0, g_j^{\boldsymbol{\theta}}(\boldsymbol{h}_i) = 1\}|}{|\{(i,j)\colon s_{ij} = 0\}|} \tag{11a}$$

$$\mathsf{P}_{\mathrm{MD}} = \frac{|\{(i,j)\colon s_{ij} = 1, g_j^{\boldsymbol{\theta}}(\boldsymbol{h}_i) = 0\}|}{|\{(i,j)\colon s_{ij} = 1\}|} \tag{11b}$$

respectively. The frequencies obtained are shown in the following table.

	training	validation
P_{FA}	6%	7 %
P_{MD}	18.21%	20.01%

[7]Note that this representation of successful decoding probabilities only accounts for *marginal* probabilities, although the binary variates of the underlying random vector may *not* be independent. We limit this analysis to marginal probabilities for the sake of simplicity.

Note that false alarms lead to wasted decoding efforts, whereas missed detections negatively impact the system performance in terms of increased packet loss. In Figure 3, we have plotted the packet loss probabilities resulting from exhaustive decoding against those obtained from a DNN-assisted receiver. The number of active users is fixed to $|\mathcal{A}| = 6$ for all frames. The SNR is set to $10\,\mathrm{dB}$ and the fading distribution is Rician with factor 0.9, both for training data generation and in the validation phase.

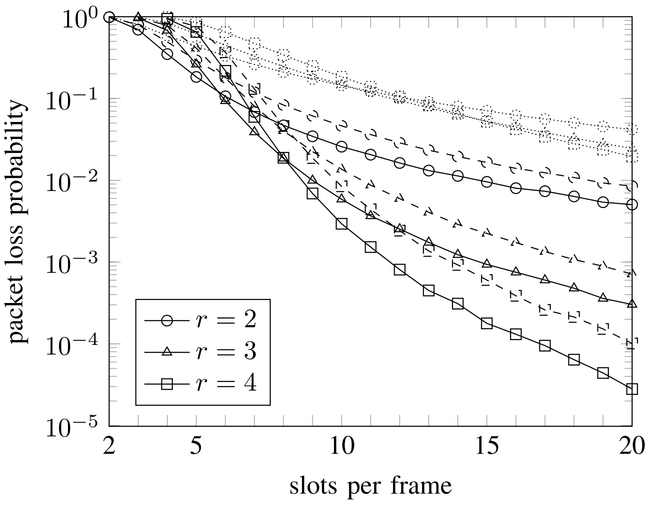

Fig. 3. Packet loss probability of the PLNC coded random access scheme for varying number of repetitions r and slots per frame, with exhaustive decoding attempts (solid curves), with DNN-aided decoding decisions (dashed curves) and with SIC decoding only (dotted curves).

As expected, the performance of a DNN-aided receiver does not fully match exhaustive search, but it still distinctly outperforms pure SIC decoding while keeping the number of decoding steps very low.

IV. Conclusion

In this work, we have demonstrated how a machine learning algorithm based on deep learning can be a key enabler of a PLNC coded random access system by helping the receiver in making efficient decoding decisions. The main contribution of this work is to showcase the potential of this machine learning approach and to clarify the technical challenges still ahead in order to harvest the full benefits of this DNN-aided decoding approach. In extensions of this work, besides fine-tuning the deep learning setup, it will be interesting to account for more realistic conditions and factor in additional considerations, such as the robustness to parameter variations (SNR, fading distribution, imperfect channel knowledge), the adaptation of code rates and modulation schemes, retransmission requests, the scalability to very large collision degrees (in the context of massive machine-type communication), joint slot- and frame-level decoding decisions, etc. These are left for future investigation.

References

[1] N. Abramson, "The ALOHA system: Another alternative for computer communications," in *Proceedings of the Joint Computer Conference*, ser. AFIPS 1970 (Fall). New York, NY, USA: ACM, 1970, pp. 281–285.

[2] L. G. Roberts, "ALOHA packet system with and without slots and capture," *SIGCOMM Comput. Commun. Rev.*, vol. 5, no. 2, pp. 28–42, Apr. 1975.

[3] L. Kleinrock and M. Scholl, "Packet switching in radio channels: New conflict-free multiple access schemes," *IEEE Trans. Commun.*, vol. 28, no. 7, pp. 1015–1029, Jul. 1980.

[4] Y. C. Tay, K. Jamieson, and H. Balakrishnan, "Collision-minimizing CSMA and its applications to wireless sensor networks," *IEEE J. Sel. Areas Commun.*, vol. 22, no. 6, pp. 1048–1057, Aug. 2004.

[5] S. Ghez, S. Verdú, and S. C. Schwartz, "Stability properties of slotted Aloha with multipacket reception capability," *IEEE Trans. Autom. Control*, vol. 33, no. 7, pp. 640–649, Jul. 1988.

[6] L. Tong, Q. Zhao, and G. Mergen, "Multipacket reception in random access wireless networks: from signal processing to optimal medium access control," *IEEE Commun. Mag.*, vol. 39, no. 11, pp. 108–112, Nov. 2001.

[7] V. Naware, G. Mergen, and L. Tong, "Stability and delay of finite-user slotted ALOHA with multipacket reception," *IEEE Trans. Inf. Theory*, vol. 51, no. 7, pp. 2636–2656, Jul. 2005.

[8] J. Luo and A. Ephremides, "Power levels and packet lengths in random multiple access with multiple-packet reception capability," *IEEE Trans. Inf. Theory*, vol. 52, no. 2, pp. 414–420, Feb. 2006.

[9] D. S. Chan, T. Berger, and L. Tong, "Carrier sense multiple access communications on multipacket reception channels: Theory and applications to IEEE 802.11 wireless networks," *IEEE Trans. Commun.*, vol. 61, no. 1, pp. 266–278, Jan. 2013.

[10] Y. H. Bae, B. D. Choi, and A. S. Alfa, "Achieving maximum throughput in random access protocols with multipacket reception," *IEEE Trans. Mobile Comput.*, vol. 13, no. 3, pp. 497–511, Mar. 2014.

[11] G. Liva, "Graph-based analysis and optimization of contention resolution diversity slotted ALOHA," *IEEE Trans. Commun.*, vol. 59, no. 2, pp. 477–487, Feb. 2011.

[12] A. Zanella and M. Zorzi, "Theoretical analysis of the capture probability in wireless systems with multiple packet reception capabilities," *IEEE Trans. Commun.*, vol. 60, no. 4, pp. 1058–1071, Apr. 2012.

[13] C. Stefanovic, F. Lazaro, and P. Popovski, "Frameless ALOHA with reliability-latency guarantees," *arXiv:1709.02177 [cs, math]*, Sep. 2017.

[14] G. Cocco and S. Pfletschinger, "Seek and decode: Random multiple access with multiuser detection and physical-layer network coding," in *IEEE Int. Conf. on Comm. Workshops (ICC)*, Jun. 2014, pp. 501–506.

[15] G. Cocco, S. Pfletschinger, and M. Navarro, "Seek and decode: Random access with physical-layer network coding and multiuser detection," *Transactions on Emerging Telecommunications Technologies*, Dec. 2016.

[16] J. Goseling, M. Gastpar, and J. H. Weber, "Random access with physical-layer network coding," *IEEE Trans. Inf. Theory*, vol. 61, no. 7, pp. 3670–3681, Jul. 2015.

[17] S. Ashrafi, C. Feng, S. Roy, and F. R. Kschischang, "Slotted ALOHA with compute-and-forward," in *IEEE International Symposium on Information Theory (ISIT)*, Jun. 2015, pp. 571–575.

[18] J. Goseling, C. Stefanovic, and P. Popovski, "Sign-compute-resolve for random access," in *52nd Annual Allerton Conference on Communication, Control and Computing (Allerton)*, Sep. 2014, pp. 675–682.

[19] S. Ashrafi, C. Feng, and S. Roy, "Performance analysis of CSMA with multi-packet reception: The inhomogeneous case," *IEEE Trans. Commun.*, vol. 65, no. 1, pp. 230–243, Jan. 2017.

[20] M. Navarro *et al.*, "D4.1 Technical Results for Service Specific Multi-Node/Multi-Antenna Solutions," FANTASTIC5G, Public Deliverable, Jun. 2016. [Online]. Available: http://fantastic5g.com/?page_id=676

[21] ——, "D4.2 Final results for the flexible 5G air interface multinode/multi-antenna solution," FANTASTIC5G, Public Deliverable, Apr. 2017. [Online]. Available: http://fantastic5g.com/?page_id=676

[22] CCSDS Secretariat, "CCSDS 231.0-B-3 TC Synchronization and Channel Coding," Consultative Committee for Space Data Systems (CCSDS), Blue Book, Sep. 2017. [Online]. Available: https://public.ccsds.org/Pubs/231x0b3.pdf

[23] S. Pfletschinger, "Joint decoding of multiple non-binary LDPC codewords," in *IEEE Int. Conf. on Comm. (ICC)*, Jun. 2014, pp. 513–519.

[24] Y. LeCun, Y. Bengio, and G. Hinton, "Deep Learning," *Nature International Weekly Journal of Science*, vol. 521, pp. 436–444, Apr. 2015.