Neural Network-based Forecasting of Decodability for Early ARQ

Matthias Hummert, Dirk Wübben and Armin Dekorsy
Department of Communications Engineering
University of Bremen, 28359 Bremen, Germany
Email: {hummert, wuebben, dekorsy}@ant.uni-bremen.de

***Abstract*—Forecasting the decodability of a received packet of a given decoder is a hard task as many State-of-the-Art (SoTA) decoders are of high complexity and not easy to analyse in an analytical fashion. Gathering this forecast on the other hand would enable to save computational complexity and latency as a decoder execution can be saved if it is unlikely that the received packet is decoded correctly. On top, we can provide early feedback for Automatic Repeat Request (ARQ) schemes before actually running the decoding chain. Guided by this motivation, several approaches of classifying the received packet before the actual decoding process have been discussed. We propose to use neural networks (NN) in the context of forecasting received packets for a given receiver chain. We evaluate the performance of the NN by evaluating different performance metrics and perform an efficiency analysis of ARQ.**

***Index Terms*—Supervised Machine Learning, Link Abstraction, Belief Propagation, error-correcting codes, Early ARQ**

I. Introduction

In a time where ultra reliable low latency communication (URLLC) is getting more and more important, knowing in advance whether a packet is likely correctly decodable or not can save decoder executions and schedule retransmissions faster. Therefore, classification algorithms have been proposed to make a decision, if a packet is decocable or not before the actual decoding process takes place. This process of classification based on the received signal is discussed in the context of Early Automatic Repeat Request (E-ARQ). In case of a negative forecast, a negative acknowledgement (NACK) is immediately feed back to ask for a retransmission without executing the decoder. If the forecast is positive, the positive acknowledgment (ACK) is immediately feed back and the packet is decoded. A wrong E-ARQ feedback actually has the same impact as transmission errors on the feedback channel and are handled by additional mechanisms based on time references [1]. In the literature, E-ARQ has been discussed based on Bit-Error-Rate (BER) estimations using Log-Likelihood-Ratios (LLR) [2] and by exploiting subcode structures of Low-Density-Parity-Check (LDPC) codes [3]. Another approach to provide early feedback is the exploitation of different bounds for the error probability and the calculation of an outage rate [4]. Recently, machine learning techniques have been applied to E-ARQ [5] [6], where subcode structures have also been exploited and different algorithms like Random Forest (RF) or Linear Regression (LR) have been applied for the classification of packets. To our knowledge, the first NN based classifier has been proposed just recently, to predecide between 2 decoders or to ask for a retransmission [7].

Due to the difficulty of analyzing the decodability in an analytical faschion, we propose to apply NNs as a classifier in order to decide for a specific received signal whether it is correctly decodable or not. The received packet is used as the input and the NN will output a probability whether the packet is correctly decodable by the receiver chain. Furthermore, we perform an efficiency analysis for E-ARQ with the proposed NN. We name our NN-classifer as NN-FoC (NN ForeCast).

This work was partly funded by the German ministry of education and research (BMBF) under grant 16KIS1180K (FunKI).

II. Preliminaries

A. Nomenclature

We will note matrices by upper bold symbols $\boldsymbol{H}$, vectors by bold symbols $\boldsymbol{x}$ and an entry of the vector with subscripts, e.g., x_i for the ith entry. Hard decisions are marked with $\hat{\boldsymbol{x}}$ and soft estimates are noted as $\tilde{\boldsymbol{x}}$.

B. System model

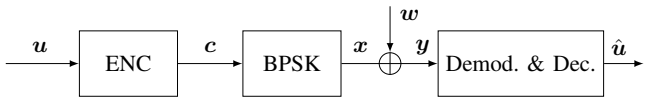

Fig. 1. System model of coded BPSK transmission over an AWGN channel and Demodulation & Decoding

Consider the given communication chain in Fig. 1 where a binary information word $\boldsymbol{u} \in \mathbb{F}_2^k$ of length k is encoded (ENC) into the codeword $\boldsymbol{c} \in \mathbb{F}_2^n$ of length n by a linear block code Γ of code rate $R_c = k/n$. The BPSK modulated codeword $\boldsymbol{x} = 1 - 2\boldsymbol{c}$ is transmitted over an Additive White Gaussian Noise (AWGN) channel leading to the receive vector $\boldsymbol{y}$ given by $\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{w}$. The received packet is then fed to demodulation and decoding to gather the estimated information word $\hat{\boldsymbol{u}}$. The decoding algorithms considered here are Belief Propagation [8] and the optimal maximum-likelihood (ML) decoder. For BP decoding LLRs of the receive signals need to be calculated by

$$L_{\boldsymbol{y}} = \log\left(\frac{P(y_i|x_i=1)}{P(y_i|x_i=-1)}\right) = \frac{2}{\sigma_w^2} y_i = L_{\text{ch}} y_i \qquad (1)$$

The input $L_{\boldsymbol{y}}$ of the BP decoder equals the receive signal $\boldsymbol{y}$ scaled by the so called channel reliability $L_{\text{ch}} = 2/\sigma_w^2$ with the noise variance σ_w^2.

III. Forecasting of decodability via NN

A. Predicition model

Since the decoder is the most complex part of the receiver chain when considering computational resources, and as iterative decoders introduce processing latency, it would be advantageous to allocate computational resources to decoding only when it is likely that decoding will be successful. Therefore, a classification algorithm can be used, which forecasts the decoder success, as shown in Fig. 2.

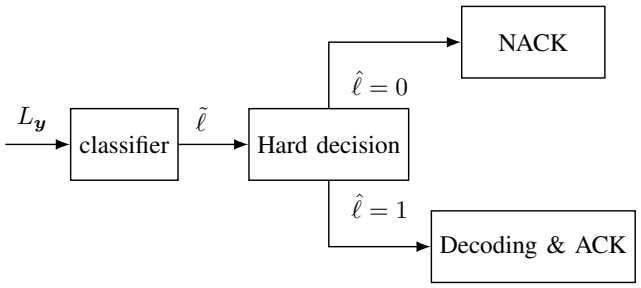

Fig. 2. Classificiation scheme forecasting whether a retransmission should be scheduled or run the receiver chain, depending on the likelihood of the classifier output

We propose to use the LLRs as an input of the classifier as they implicitly contain the SNR and the output is an estimated label $\tilde{\ell}$ between 0 and 1, i.e. $(0 \leq \tilde{\ell} \leq 1)$. The estimated label denotes the probability of a successfull packet decoding for the given receive signals. By a threshold decision $\hat{\ell} = Q_\alpha(\tilde{\ell})$ a hard forecast initiating either a NACK feedback ($\hat{\ell} = 0$ for $\tilde{\ell} < \alpha$) or the execution of the decoder and a ACK feedback ($\hat{\ell} = 1$ for $\tilde{\ell} > \alpha$). Here we use $\alpha = 0.5$, but notice that adaptation of α is currently under investigation. In this paper, we propose to use a NN as a classifier. Essentialy, NNs are nonlinear function combinations with trainable parameters $\boldsymbol{\Theta}$. For a detailed explanation of them we refer to [9]. The NN-FoC is trained using the supervised learning approach which is described hereafter.

B. Supervised Learning Approach

The general idea of supervised learning is to tune learnable parameters $\boldsymbol{\Theta}$ of a function (a NN in this case) using input data (observations) to match the corresponding labeled output data ($\tilde{\ell} = f(\boldsymbol{\Theta}, L_{\boldsymbol{y}})$) [9]. Here, the NN should mimic the decoder. Thus, the observations are the received LLRs $L_{\boldsymbol{y}}$ and the labels are determined by the outcome $\hat{\boldsymbol{u}}$ of the decoder. In case of a successful decoding $\hat{\boldsymbol{u}} = \boldsymbol{u}$ the label is $\ell = 1$ and otherwise $\ell = 0$, i.e. for $\hat{\boldsymbol{u}} \neq \boldsymbol{u}$.

Having the dataset at hand, we can minimize a loss $\mathcal{L}(\ell, \tilde{\ell})$ so that the estimated labels $\tilde{\ell}$ are as close as possible to the actual decoder success given by the labels ℓ. The loss reflects some form of distance between the output $\tilde{\ell}$ of the function and the labels ℓ. To formalize this, we define the training set consisting of T training samples as $(L_{\boldsymbol{y}}^t, \ell^t)$ for $t = \{1, \ldots, T\}$. Therefore the set containts T vectors $L_{\boldsymbol{y}} \in \mathbb{R}^n$ and T scalar labels ℓ, which are either 1 or 0. As loss function the binary crossentropy (BCE)

$$\mathcal{L}(\tilde{\ell}, \ell) = -\frac{1}{T}\sum_{t=1}^{T} \ell^t \cdot \log(\tilde{\ell}^t) + (1-\ell^t) \cdot \log(1-\tilde{\ell}^t) \qquad (2)$$

is used. To minimize the loss function, the learnable parameters $\boldsymbol{\Theta}$ are adapted, e.g. by using a form of Stochastic Gradient Descent (SGD) approach based on Steepest Descent. For initial values of $\boldsymbol{\Theta}$ the whole NN is executed and soft predicisions $\tilde{\ell}$ are calculated leading to the loss $\mathcal{L}(\ell^t, \tilde{\ell}^t)$. From the loss the derivative w.r.t. $\boldsymbol{\Theta}$ can be determined and the parameter $\boldsymbol{\Theta}$ updates via SGD. The most basic form of SGD is given by the iteration

$$\boldsymbol{\Theta}^{i+1} = \boldsymbol{\Theta}^i - \frac{\eta}{|\mathcal{S}_r|} \sum_{t \in \mathcal{S}_r} \nabla_{\boldsymbol{\Theta}} \mathcal{L}(\ell^t, \tilde{\ell}^t) \qquad (3)$$

where i denotes the iteration index, η is the step size or learning rate and $\nabla_{\boldsymbol{\Theta}} \mathcal{L}(\ell^t, \tilde{\ell}^t)$ is the gradient of the loss function $\mathcal{L}(\ell^t, \tilde{\ell}^t)$ w.r.t. the learning paramter $\boldsymbol{\Theta}$. The training set is divided into subsets $|\mathcal{S}_r| < T$ of equal size in the way that no training data is present twice in any subset. These subsets are called batches and $|\mathcal{S}_r|$ is named batch size. When all subsets $\mathcal{S}_r$ are processed the whole training set T has been used once which is called an epoch. For implementation, Keras [10] and Tensorflow [11] have been used here.

IV. Simulation Results

A. Basic Parameters

We investigate the performance of the NN-FoC classifier for two codes, the (7, 4) Hamming code and a (32, 16) LDPC code [12] to get a first impression of the performance. For all testing purposes we use the Adam optimizer [13], which is a form of SGD and a learning rate of $\eta = 0.001$ is applied. The BP decoder uses a maximum of 5 iterations and we note the total number of received packets as N for our Monte Carlo simulations. We choose $N = 10^6$ if not mentioned otherwise. Please note that the NNs are trained offline once and only used for inference afterwards. In order to define our performance metrics, we need the indicator function given by

$$\mathbb{1}_{\{\hat{\ell} \in \mathbb{A}\}}(\hat{\ell}) = \begin{cases} 1, & \hat{\ell} \in \mathbb{A} \\ 0, & \text{else} \end{cases} . \qquad (4)$$

where $\mathbb{A}$ is the set or the condition when the indicator function is 1. As performance metrics we use the false forecast accuracy P_{FF} indicating how many wrong forecast are made

$$P_{\text{FF}} = \frac{\sum_t \mathbb{1}_{\{\hat{\ell}^t=0|\ell^t=1\}}(\hat{\ell}^t) + \sum_t \mathbb{1}_{\{\hat{\ell}^t=1|\ell^t=0\}}(\hat{\ell}^t)}{N}, \qquad (5)$$

and the false negative P_{FN}, the false positive P_{FP}, true negative P_{TN} and true positive P_{TP} probabilities defined as

$$P_{\text{FN}} = \frac{\sum_t \mathbb{1}_{\{\hat{\ell}=0|\ell=1\}}(\hat{\ell}_t)}{\sum_t \mathbb{1}_{\{\ell=1\}}(\ell_t)}, P_{\text{FP}} = \frac{\sum_t \mathbb{1}_{\{\hat{\ell}=1|\ell=0\}}(\hat{\ell}_t)}{\sum_t \mathbb{1}_{\{\ell=0\}}(\ell_t)}, \quad (6)$$

$$P_{\text{TN}} = \frac{\sum_t \mathbb{1}_{\{\hat{\ell}=0|\ell=0\}}(\hat{\ell}_t)}{\sum_t \mathbb{1}_{\{\ell=0\}}(\ell_t)}, P_{\text{TP}} = \frac{\sum_t \mathbb{1}_{\{\hat{\ell}=1|\ell=1\}}(\hat{\ell}_t)}{\sum_t \mathbb{1}_{\{\ell=1\}}(\ell_t)}. \quad (7)$$

For example, the false negative probability reflects a non decodable forecast $\hat{\ell} = 0$, whereas the packet was actually decodable ($\ell = 1$) normalized on the number of actually decodable packets.

We will focus on the false probabilites, P_{FP} and P_{FN}, as false decisions highly influence the performance of the overall proposed scheme. For comparison, we will also include the counterpart of the average Frame Error Rate (FER) given by 1-FER for the given E_b/N_0 indicating the average probability of error-free decoding. This can be used as SNR-based classifier, because 1-FER equals a classifier that forecasts $\hat{\ell}_t = 1$ for all packets.

B. How to train?

Due to the natural imbalances in the number of decodable and non-decodable packets for different SNR, we will introduce a bias into the training dataset and our forecast performance varies. Due to space limitations we cannot show a full analysis of different training datasets here but we want to emphasize that we are aware of this issue and tested various datasets and biases and found a training over a range of different SNR to deliver the best performance tradeoff between false forceast accuracy P_{FF} and false probabilities. The range of SNR to train over is code dependent and has to be chosen carefully to cover a good amount of decodable and non decodable packets.

C. Performance evaluation for different NNs

In this section we evaluate different NN-FoCs for the (7, 4) Hamming code and the BP decoder with different structures, in order to judge, how many weights and layers are needed for the forceasting NN to deliver good performance. The NN-FoC configurations follow a general form: the input layer has the width of the codeword length n and the output layer consists of a single neuron with a sigmoid activation. We always use ReLU activation functions for the hidden layers of the NN-FoCs.

We investigate different number of hidden layers and vary the width of these layers and evaluate the performance. The configurations can be seen in table I.

Hidden Layer	width of layer	# of weights	Name
4	2000, 1000, 500, 200	2617901	NN-FoC 1
4	50, 50, 50, 20	9091	NN-FoC 2
2	50, 20	1441	NN-FoC 3

TABLE I
TESTED NN CONFIGURATIONS

We also show multiple instances of NN-FoC 1 which are trained and evaluated per SNR. Using a NN-FoC 1 per SNR is impractical but serves as a benchmark here. NN-FoC 1 has a large amount of trainable weights but only 4 hidden layers. The lowest complexity NN-FoC 3, has 2 hidden layers and a very low amount of trainable parameters. In Fig. 3 we show P_{FF} in log scale for various NN-FoC versus the SNR. As expected, the per SNR NN-FoCs show the best performance, but are nearly indistinguishable to NN-FoC 3. The lower the complexity of the NN becomes, the higher the false forecast accuracy becomes. The accuracy drop seen, is on the other hand, very low considering the amount of computational complexity saved. All NN-FoCs outperform the basic 1-FER classifier.

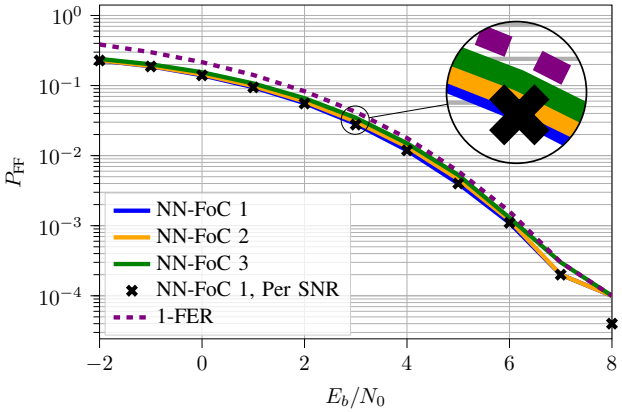

Fig. 3. False forecast accuracy P_{FF} versus SNR for different NN-FoC and the (7, 4) Hamming code

We further evaluate the false probabilites in Fig. 4 and 5. We also show the performance of the NN-FoC in combination with the ML decoder. For a detailed comparison of BP and ML decoding we refer to subsection IV-D.

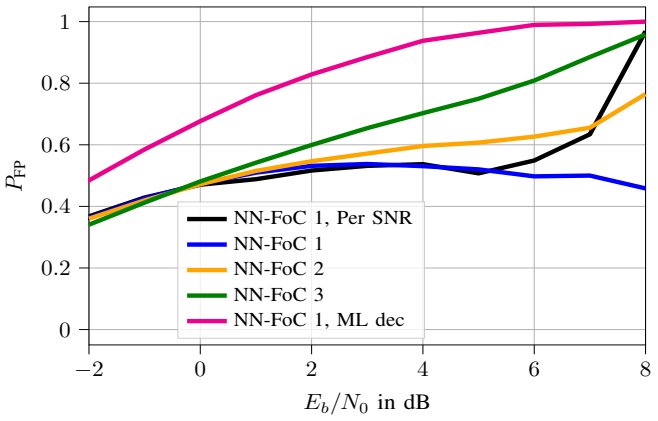

Fig. 4. False positive probabilities versus SNR for different complexity NNs and the (7, 4) Hamming code

NN-FoC 1 shows the best performance here, especially considering the false positive probabilites P_{FP}. The lower the complexity of the NN-FoC becomes, the higher the false pos-

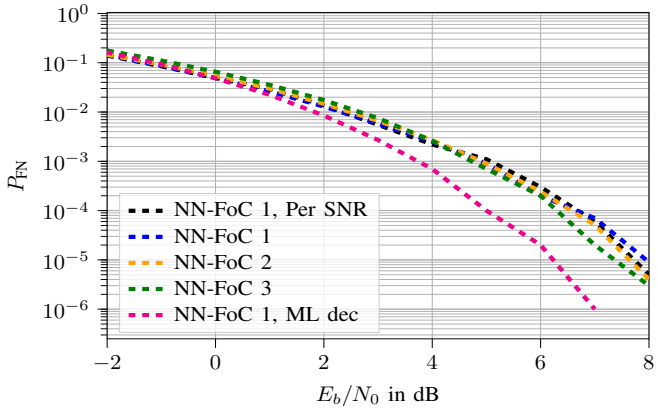

Fig. 5. False negative probabilities versus SNR for different complexity NNs and the (7, 4) Hamming code

itive probabilites raise. Furthermore, for our benchmark NN-FoC 1 per SNR, the false positives show good performance for low SNR but worsens for high SNR due to the imbalance in the training dataset, the NN-FoC tend to forecast decodable.

On the other hand, the false negative probabilites are nearly indistinguishable for all NN-FoC and tend to 0 for high SNR. This is an expected behaviour as the number of decodable packets increase for high SNR. They are, however, likely to differ for very low SNR as the number of non decodable packets increases.

In general, it is not worth spending so many more weights for the shown gains.

D. Maximum-Likelihood Decoding

In order to verify the results and not solely rely on the BP decoder for our conclusions, we provide evaluations for the ML decoder for the (7, 4) Hamming to gain insights for an optimal decoder. Therefore, we train the NN-FoC 1 for each decoder separately and evaluate the performance afterwards. In Fig. 6 we show the false forecast accuracy P_{FF} versus the SNR and observe that the general behaviour of the NN-FoC matches with the NN-FoC of the BP decoder.

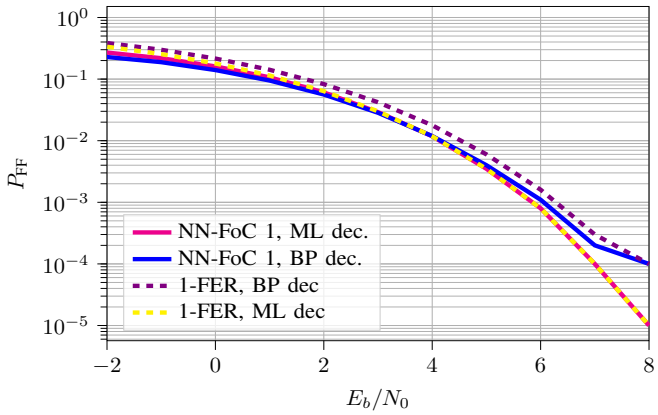

Fig. 6. False forecast accuracy P_{FF} versus SNR of the NN-FoC for the ML and BP decoder for the (7, 4) Hamming code

The obtained results indicate that forecasting for an optimal decoder decreases the potential gains the NN-FoC can achieve, as the ML decoder has further error correcting capabilities which BP cannot offer. We further see in Fig. 5 and 4 the false probabilites versus the SNR. On the other hand, the false negative probabilites are lower for the optimal decoder. This behaviour is expected as the number of decodable packets is higher for the optimal decoder per SNR. The false positive probabilities P_{FP} rise higher than for the BP decoding. The suboptimalities of the BP seem to lead to a better performing NN-FoC. The general behaviour of the NN-FoC however, remains the same.

E. Performance Evaluation for a short LDPC

For the (32, 16) LDPC code we use the NN-FoC 1 from table I. Please note that the number of weights is slightly increased as the input dimension of the NN-FoC is now $n = 32$ instead of 7. In Fig. 7 we show the forecast accuracy and the false probabilities versus the SNR for the (32, 16) LDPC code.

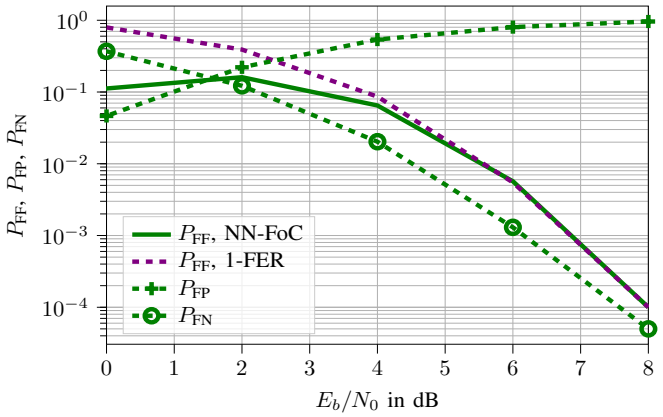

Fig. 7. False forecast accuracy P_{FF} and false probabilites versus E_b/N_0 for the (32, 16) LDPC code

We observe that the NN-FoC shows lower false forecast accuracy P_{FF} in comparison to the 1-FER classifier, especially for low SNR. At 0 dB the false forecast accuracy actually decreases again since more packets are non decodable than decodable. In comparison to the 1-FER classifier, the NN-FoC shows respectable gains. For example at an SNR of 2dB the P_{FF} of the NN-FoC is at 10^{-1} and the 1-FER is at $4 \cdot 10^{-1}$.

Looking at the false probabilities, we notice that the false positives highly increase the higher the SNR and vice versa for the false negatives. This behavior indicates that the NN-FoC tends to always give decodable forecasts at high SNR and not decodable at low SNR, which is an undesired property that needs further investigations.

F. Receiver Operating Characteristic

To further analyse the performance, a Receiver Operating Characteristic (ROC) analysis is performed [14]. The ROC is generally used to evaluate the performance of binary classifiers. An ROC curve shows the true positives P_{TP} versus

the false positives P_{FP}. To get different values for these probabilities, the NN-FoC 1 is used for inference at different SNR. In Fig. 8, the ROC curves for the (7, 4) Hamming and (32, 16) LDPC code are shown. Furthermore the performance for the ML decoder of the (7, 4) Hamming code and the worst case performance, which is just guessing, is shown. As an example, the 1-FER classifier always has a $P_{\text{TP}} = 1$ and $P_{\text{FP}} = 1$ as it always classifies a packet as decodable.

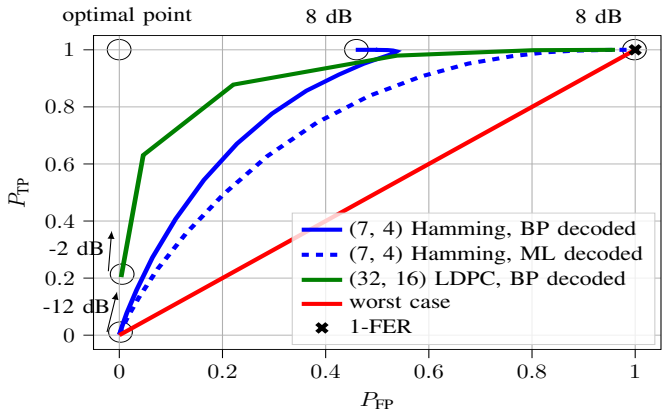

Fig. 8. ROC curve for the NN-FoC for the Hamming and LDPC code

Everything above the red line (worst case) is a performance gain in comparison to a random guesser. On the top left we show the optimal point of a classifier, i.e. 100% true posivites, while no false positives occur. We notice that all NN-FoC classifier are capable of providing substantial gains in comparison to a random guesser. The NN-FoC performance of the LDPC code performs better than for the Hamming code, which is an indication that the performance is dependent on the codeword length n.

By comparing the NN-FoC for ML decoding of the Hamming code and the BP decoded trained NN-FoC it is again noticeable that the classifier for the BP performs better. An interesting point is the high SNR regime of the BP decoded NN-FoC. A clear cut is observable, where the classifier decreases the false positives and increases the true positives. This behaviour needs further investigation.

V. Throughput Analyis

In this section we demonstrate the impact of an early ARQ feedback on the throughput as a function of the processing latency at the receiver by using the NN-FoC in combination with a BP decoder. To this end, the state diagram of the ARQ scheme with early feedback is derived. The processing time of the decoder T_{Dec} is defined as a multiple of the time duration T_{B} of a packet and $\kappa = T_{\text{Dec}}/T_{\text{B}}$

The state diagram is shown in Fig. 9. This way of analysis is well known from noisy feedback ARQ evaluations [1].

We use Selective-Repeat (SR) as our ARQ scheme. The variables a,b,... describe the state transistions. We introduce a placeholder D that sets the transitions in relation to the block time T_{B}. The process for each packet starts with

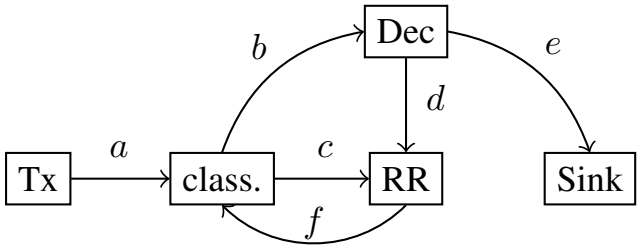

Fig. 9. State diagaram for the proposed classifier in an ARQ environment

the initial transmission requiring the time $a = D^1$. The NN-FoC forecasts the decoder success and asks for a retransmission (NACK) with probability $c = P_{\text{F}}$ (block RR) or starts the decoding process with probability $b = 1 - P_{\text{F}}$ and feedbacks ACK. In the DEC block the iterative decoder is executed (requiring D^κ time instances) and by CRC check the decoder success is determined. With probability P_{fp} the decoder is unable to decode the packet that was forecasted to be decodable. Thus, a NACK is transmitted with parameter $d = P_{\text{fp}}D^k$. On the other hand, with P_{tp} the deocder is able to decode the packet leading to the parameter $e = P_{\text{tp}}D^k$. The Retransmission Request (RR) block actually repeats the block based on an NACK and requires $f = D^1$ time instances. Please note that we assume, that the NN-FoC has no delay but NN-FoCs with delay are currently under investigation. With the help of the derived state diagramm, we can add up the different times needed relative to the decoding latency κ. We first take a look at the transfer function H:

$$\begin{aligned} H = \frac{\text{Sink}}{\text{Tx}} = \frac{eba}{1 - cf - bdf} = \\ = \frac{P_{\text{tp}}D^\kappa(1 - P_{\text{F}})D}{1 - P_{\text{F}}D - (1 - P_{\text{F}})P_{\text{fp}}D^\kappa D} \end{aligned} \tag{8}$$

Calculating the change of the transfer function H wrt. to the placeholder D at $D = 1$ yields an expression for the average time T_{avg} per block time T_{B}:

$$\begin{aligned} \frac{T_{\text{avg}}}{T_{\text{B}}} = \left.\frac{\partial H}{\partial D}\right|_{D=1} = \\ = \frac{P_{\text{tp}}(1 - P_{\text{F}})\left((\kappa + 1)(1 - P_{\text{F}}) + P_{\text{F}}\right)}{(1 - P_{\text{F}} - P_{\text{fp}}(1 - P_{\text{F}}))^2}. \end{aligned} \tag{9}$$

Therefore, the effiency η is finally given by

$$\eta = R_c\frac{T_{\text{B}}}{T_{\text{avg}}} = R_c\frac{(1 - P_{\text{F}} - P_{\text{fp}}(1 - P_{\text{F}}))^2}{P_{\text{tp}}(1 - P_{\text{F}})\left((\kappa + 1)(1 - P_{\text{F}}) + P_{\text{F}}\right)}. \tag{10}$$

As shown, the efficiency is dependent on the true positive probability P_{TP}, false positive probability P_{FP} and the decoder latency expressed in terms of κ. We analogously derive the effiencies without a forecast, i.e. standard ARQ, and a genie classifier which perfectly forecasts all packets.

In Fig. 10 the efficiencies η for values of $\kappa = 1, 2, 3$ for the (7, 4) Hamming code are shown. As expected, the genie

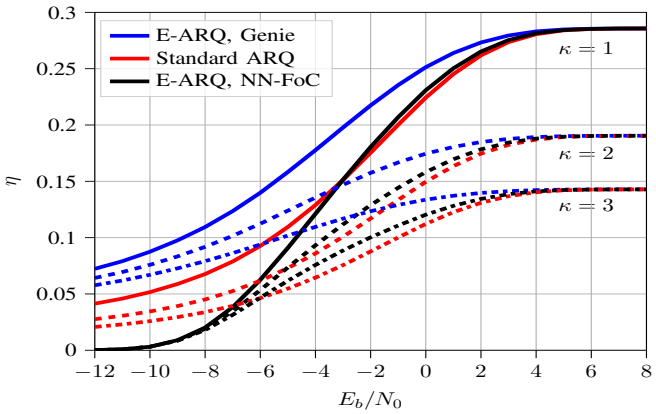

Fig. 10. Effiency η for different decoder delays κ for (7, 4) Hamming code in comparison for a schematic with no classifier, a genie classifier and the actual trained NN-FoC.

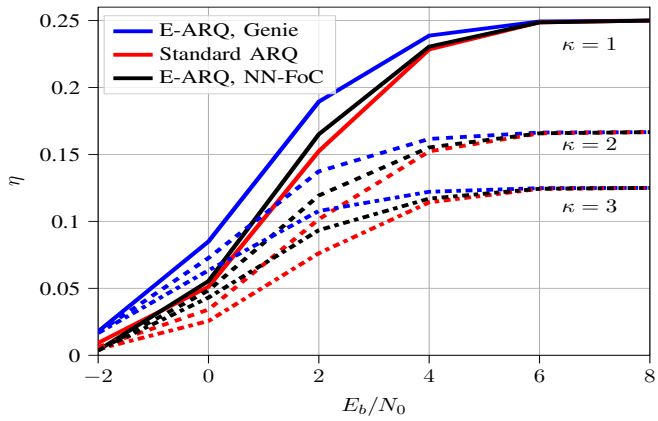

Fig. 11. Effiency η for different decoder delays κ for (32, 16) LDPC code in comparison for a schematic with no classifying NN, a genie classifier and the actual trained NN-FoC.

classifier has the best performance. For low SNR the NN-FoC with E-ARQ actually performs worse than the standard ARQ scheme, as it has high amount of false positive forecasts.On the other hand, for higher SNR the NN-FoC shows performance in between the Genie classifier and the standard ARQ scheme. The higher κ, the more performance can be gained by making a forecast as the circumvented decoder runs save more time. For example for $\kappa = 1$ at an SNR of 0 dB the performance gap of the standard ARQ to the NN-FoC is 0.005 and for $\kappa = 3$ the gap is 0.02. The efficiency saturates at $\eta = \frac{R_c}{\kappa+1}$ and, thus, the maximum efficiency highly depends on the decoder latency κ.

In Fig. 11 we show the same efficiency analysis for the (32, 16) LDPC code. Due to the low false positive rate for low SNR, the gains are larger in comparison to the Hamming coded scheme. Thus the early ARQ scheme with NN-FoC outperforms the standard ARQ in nearly the whole range of investigated SNRs.

The NN-FoC hence improves the overall system efficiency. The gap to the Genie classifier is large, but this is a solely theoretical curve as a genie classifier is not possible. This paper should hence be seen as a first attempt to apply NNs in this scenario and is by no means the final conclusion.

VI. Summary and Future Work

In this paper we propose the NN-FoC that is capable of classifying received packets wrt. to their decodability without actually running the decoder by applying an offline trained NN. We have shown that low complex NN-FoC can be used and we are able to improve the efficiency of classical ARQ schemes.

As an outlook, the extension for codes with longer codeword length and the best parameterization of the NN are currently being investigated, as well as the decoupling from the codewords. On top, a feature extraction to reduce the input dimension n of the NN-FoC is considered, as well as using parts of the decoder (e.g. output after first BP iteration) to further enhance the performance of the NN-FoC. Furthermore a practical scenario shall be considered, e.g. a specific 5G working point, to achieve further performance gains for the NN-FoC.

References

[1] S. B. Wicker, *Error Control Systems for Digital Communication and Storage*. Prentice-Hall, Inc., 1994.

[2] G. Berardinelli, S. R. Khosravirad, K. I. Pedersen, F. Frederiksen, and P. Mogensen, "Enabling Early HARQ Feedback in 5G Networks," in *IEEE 83rd Vehicular Technology Conference (VTC Spring), Nanjing, China*, May 2016.

[3] B. Goektepe, S. Faehse, L. Thiele, T. Schierl, and C. Hellge, "Subcode-Based Early HARQ for 5G," in *IEEE International Conference on Communications Workshops (ICC Workshops), Kansas City, MO, USA*, May 2018.

[4] P. Rost and A. Prasad, "Opportunistic Hybrid ARQ—Enabler of Centralized-RAN Over Nonideal Backhaul," *IEEE Wireless Communications Letters*, vol. 3, no. 5, pp. 481–484, Dec. 2014.

[5] N. Strodthoff, B. Göktepe, T. Schierl, C. Hellge, and W. Samek, "Enhanced Machine Learning Techniques for Early HARQ Feedback Prediction in 5G," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 11, pp. 2573–2587, Jan. 2019.

[6] N. Strodthoff, B. Göktepe, T. Schierl, W. Samek, and C. Hellge, "Machine Learning for Early HARQ Feedback Prediction in 5G," in *IEEE Globecom Workshops (GC Wkshps), Abu Dhabi, United Arab Emirates*, Dec. 2018.

[7] A. Elkelesh, S. Cammerer, and S. ten Brink, "Reducing Polar Decoding Latency by Neural Network-Based On-the-Fly Decoder Selection," in *2020 IEEE Workshop on Signal Processing Systems (SiPS), Coimbra, Portugal*, Oct. 2020, pp. 1–2.

[8] J. Pearl, "Reverend Bayes on Inference Engines: A Distributed Hierarchical Approach," in *Proceedings of the Second AAAI Conference on Artificial Intelligence*, 1982, p. 133–136.

[9] O. Simeone, "A Very Brief Introduction to Machine Learning With Applications to Communication Systems," *IEEE Trans. Cogn. Comm. & Networking*, vol. 4, no. 4, pp. 648–664, Nov. 2018.

[10] F. Chollet *et al.*, "Keras," https://keras.io, 2015.

[11] M. Abadi *et al.*, "TensorFlow: Large-scale machine learning on heterogeneous systems," 2015, software available from tensorflow.org.

[12] M. Helmling, S. Scholl, F. Gensheimer, T. Dietz, K. Kraft, S. Ruzika, and N. Wehn, "Database of Channel Codes and ML Simulation Results," www.uni-kl.de/channel-codes.

[13] D. P. Kingma and J. Ba, "Adam: A Method for Stochastic Optimization," in *3rd International Conference on Learning Representations, (ICLR), San Diego, CA, USA*, May 2015.

[14] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognit. Lett.*, vol. 27, pp. 861–874, December 2006.