Applicability of single- and two-hidden-layer neural networks in decoding linear block codes

Srdan Brkic
School of Electrical Engineering
University of Belgrade
Belgrade, Serbia
srdjan.brkic@etf.rs

Predrag Ivanis
School of Electrical Engineering
University of Belgrade
Belgrade, Serbia
predrag.ivanis@etf.rs

Bane Vasic
Dept. of Electrical and Computer Eng.
University of Arizona
Tucson, Arizona
vasic@ece.arizona.edu

***Abstract*—In this paper, we analyze applicability of single- and two-hidden-layer feed-forward artificial neural networks, SLFNs and TLFNs, respectively, in decoding linear block codes. Based on the provable capability of SLFNs and TLFNs to approximate discrete functions, we discuss sizes of the network capable to perform maximum likelihood decoding. Furthermore, we propose a decoding scheme, which use artificial neural networks (ANNs) to lower the error-floors of low-density parity-check (LDPC) codes. By learning a small number of error patterns, uncorrectable with typical decoders of LDPC codes, ANN can lower the error-floor by an order of magnitude, with only marginal average complexity incense.**

***Keywords*—error-floors, neural networks, linear block codes, low-density parity-check codes, ML decoding**

I. Introduction

Artificial neural networks (ANNs) are popular mathematical concept used to solve various complex engineering problems, related to regression and classification of collected data. ANNs can be applied without substantial knowledge of the data (for example dependencies among data samples), why they are called *universal approximators*. Topological structure of feed-forward ANNs provides sufficient degrees of freedom to represent any continuous function with arbitrary precision. Early works, like Cybenko's [1], showed that even single-hidden-layer feed-forward neural networks (SLFNs) preserve the universal approximation capability. Approximating discrete functions with SLFNs can be accomplish with the number of hidden-layer neurons equal to the number of data samples of the discrete function [2]. The complexity of SLFNs limits their applicability, especially given the fact that the worst case dimension of two-hidden-layer feed-forward neural networks (TLFNs) is $O(\sqrt{M})$, where M is the number of function samples, as show by Huang [3]. Incensing the number of hidden layers potentially reduces the overcall complexity of the network, however, we are unaware of the formal proof and limit our discussion to SLFNs and TLFNs.

This work was supported by the Science Fund of the Republic of Serbia under project LIDA (no. 6462951) and the Serbian Ministry of Science under project TR32028. Bane Vasic acknowledges support of the NSF under grants CIF- 1855879, CCF-2100013, CCSS-2027844 and CCSS-2052751. Bane Vasic has disclosed an outside interest in Codelucida to the University of Arizona. Conflicts of interest resulting from this interest are being managed by The University of Arizona in accordance with its policies.

The problem of maximum likelihood (ML) decoding, is a special case of classification problem. However, decoding of an arbitrary linear block code is not the typical use case for ANN classification, given the fact that number of classes (codewords) increases exponentially with length of the code. In addition, classification error (residual frame error rate) usually needs to be 10^{-5} or lower, which is hard to achieve, when training is performed on a limited set of data. Thus, most of the relevant work is oriented into improving existing decoders, by learning (via ANN) specific decoding features.

Belief propagation (BP) decoder can be transformed into sparse neural network, with small number of learnable weights, as noticed by Nachmani *et al.* [4]. Following the same idea, Lugosch and Gross [5] showed that error-rate of min-sum decoding can be reduced if parts of the decoder are organized as neurons of a sparse neural network. In a more sophisticated approach, Xiao *et al.* [6] used recurrent quantified ANN to design finite alphabet iterative decoders applied for decoding low-density parity-check (LDPC) codes. Their work showed that the major benefit of employing ANNs is to increase convergence speed of the iterative decoding. In another research direction, Liang *et al.* [7] entangled BP decoder and convolutional ANN in a decoding loop. After predefined number of iterations BP transmits codebit decisions to the ANN, which task is to estimate channel noise and passed it back to the BP decoder for additional processing. Although proposed decoding scheme is superior, compared to the BP decoder, its complexity is high. He in [8] found another application of ANNs for decoding LDPC codes, where ANN was used to predict decoding failures and estimate bit positions for bit-flipping that will hopefully help the min-sum decoder – the main error correction algorithm in the proposed scheme. Finally, Buchberger *et al.* in [9] proposed decimation strategy (learned via ANN), which enables the min-sum decoder to perform close to ML decoding bound, for short LDPC codes.

In this paper we examine the applicability of SLFNs and TLFNs in decoding of linear block codes in general, and specifically LDPC codes. Based on the provable learning capability of ANNs we examine the complexity of the network required to perform ML decoding on binary symmetric channel (BSC) and compare it with the trellis-based ML decoder. Furthermore, we show how ANN can be used to lower error-

floors of LDPC codes. Namely, we propose a decoding scheme in which ANN is used for post-processing, and every time a iterative LDPC decoder fails to correct channel induced errors, ANN-based decoder is turned on. Given the fact that the error-floors of LDPC codes are mostly influenced by uncorrectable error patterns with lowest weights, and that the number of such error patterns in usually not high, ANN can be trained to learn all of them and, consequently, reduce the error-floor by the order of magnitude, with only marginal increase in complexity per transmitted codeword. In contrary to [8], where the ANN learns the decoding features through excessive simulations, our approach relies on pre-computed trapping set profiles, which have been studied intensively over the past years [10]–[13]. Thus, to apply ANN-based post-processor one needs to collect specific low-weight error patterns, which is feasible for variety of LDPC codes and decoders, as recently shown by Raveendran *et al.* [10].

The rest of the paper is organized as follows. In Section II preliminaries about LDPC codes and decoders on graphs are given. Section III is dedicated to performance analysis of decoding with SLFNs, while we discuss TLFN decoders and their applicability in reducing error floors of LDPC codes in Section IV. Concluding remarks are given in Section V.

II. Preliminaries

Consider a binary linear block code (n, k), with code rate $R = k/n$, described by its parity check matrix $\mathbf{H}$. Let $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ be a valid codeword of the code, transmitted through BSC with crossover probability p. Receiver collects sequence $\mathbf{r}$, where $\Pr\{c_i = r_i\} = 1 - p$, $1 \leq i \leq n$. The ML decoder finds the closest valid codeword as

$$\widehat{\mathbf{c}} = \underset{\mathbf{c}}{\operatorname{argmax}} P(\mathbf{c}|\mathbf{r}). \tag{1}$$

ML decoding can be performed on trellis structure, constructed based on $\mathbf{H}$. States of a trellis at depth t, $0 \leq t \leq n$ are formed as follows

$$S_0 = \mathbf{0},$$
$$S_t = S_{t-1} + c_t \mathbf{h_t} = \sum_{i=1}^{t} c_i \mathbf{h_t}, \quad t = 1, 2, \ldots, n,$$

where $\mathbf{h_t}$ corresponds to the t-th column of $\mathbf{H}$. A trellis path (among total 2^k paths) corresponds to a codeword of the code. There are at most 2^{n-k} states at every trellis depth. Running Viterbi algorithm on the described trellis solves Eq. (1).

Decoders of LDPC codes are commonly run on bipartite graphs, called Tanner graphs, in order to exploit sparsity of $\mathbf{H}$. A bipartite graph is $G = (V \cup C, E)$, where $V = \{v_1, v_2, \ldots, v_n\}$ is a set n of variable nodes (columns of $\mathbf{H}$), $C = \{c_1, c_2, \ldots, c_m\}$ is a set of m check nodes (rows of $\mathbf{H}$). An edge $e \in E$ connects v_i and c_j ($e = (v_i, c_j)$) iff $h_{i,j} = 1$. We denote a set of neighbours of a node $x \in (V \cup C)$ as $\mathcal{N}(x)$, which cardinality $|\mathcal{N}(x)|$ is called the degree of the node x. Specially, average degree of variable nodes is denoted by $\hat{\gamma}$. An iterative decoder $\mathcal{D}$ is defined as 5-tuple $\mathcal{D} = (\mathcal{M}, \mathcal{Y}, \Phi, \Psi, \hat{\Phi})$, where $\mathcal{M}$ and $\mathcal{Y}$ denote internal message and channel alphabets, respectably. Update functions implemented in variable and check nodes are Φ and Ψ, respectively, while $\hat{\Phi}$ is bit decision function. A decoding iteration corresponds to message exchange between all neighbouring nodes in Tanner graph, and thus, during the ℓ-th iteration a variable v sends message to its neighbour c, $\mu_{v\to c}^{(\ell)}$ and receives from the opposite direction $\nu_{c\to v}^{(\ell)}$. Messages are calculated as $\mu_{v\to c}^{(\ell)} = \Phi(\mathbf{n}^{(\ell-1)}, r_v)$ and $\nu_{c\to v}^{(\ell)} = \Psi(\mathbf{m}^{(\ell)})$, where $\mathbf{n}^{(\ell)} = \nu_{\mathcal{N}(c)\setminus v\to v}^{(\ell-1)}$ and $\mathbf{m}^{(\ell)} = \mu_{\mathcal{N}(v)\setminus c\to c}^{(\ell)}$. The decoder is run for N_{iter} iterations. By varying alphabets and update node functions, distinct decoders with various complexity-performance tradeoffs can be constructed. For example, Gallager-A/B is considered to be low a complexity solution, while offset min-sum decoder has high error correction capability.

III. Decoding linear block codes with SLFNs

A codeword $\mathbf{c}_i$ of linear block code can be projected to a line and seen as a decimal number $c_i^{(d)} \in [0, 1]$. Analogously, at channel output decimal number $r^{(d)} \in I$ in range $I = [0, 1]$ can appear. The distance between two channel outputs $r_i^{(d)}$ i $r_j^{(d)}$ satisfies $|r_i^{(d)} - r_j^{(d)}| \geq 1/2^n$. The decoding is equivalent to a classification problem in which I is divided to 2^k disjoint measurable intervals $P_1, P_2, ..., P_{2^k}$. Note that P_i may not be continuous. The decoder can be represented as a function f

$$f(x) = c_i^{(d)}, \quad \text{if} \quad x \in P_i.$$

Consider feed-forward ANN with a single hidden layer constructed to mimic $f(x)$. The neural network is a finite linear combination of the form

$$G(x) = \sum_{j=1}^{M} \alpha_j \sigma(w_j x + \theta_j),$$

where α_j, w_j and θ_j are fixed real numbers, and $\sigma(\cdot)$ is any continuous function satisfying

$$\sigma(t) = \begin{cases} 1, & \text{if } t \to \infty \\ 0, & \text{if } t \to -\infty. \end{cases}$$

Let FER_{ML} be frame error rate (FER) of the ML decoder. The following theorem shows that previously defined ANN can mimic ML decoder with negligible error.

There exists artificial neural network $G(x)$, which produce frame error rate FER_{ANN}, and under the assumption of uniformly transmitted codewords, satisfies

$$|FER_{ANN} - FER_{ML}| \leq p(1-p)^{n-1}/2^k.$$

Proof: By Cybenko's theorem [1] we know that exist network $G(x)$ and interval $D \subset I$ for which $|G(x) - f(x)| < \epsilon$ for all $x \in D$, for any arbitrary ϵ. Furthermore, Lesbegue measure of D is $m(D) \geq 1 - \epsilon$. This means that neural network can classify almost all inputs in almost all x domain. Note that decoding on BSC is less rigorous – it only requires classification on a set with discrete inputs values. For all $\epsilon < 1/2^{(k-2)}$, $G(x)$ will successfully classify every received sequence from the channel $x \in D$, since the network will

give the result closes to value (codeword) produced by the ML decoder. Given the fact that, Cybenko's theorem does not guarantee the locations of interval D, it is possible that outside D exists discrete points, which will be incorrectly classified. It follows that, under the same condition for $\epsilon < 1/2^{(k-2)}$, at most one such point exists. It is reasonable to assume that if a valid codeword remains outside of D, it presence can be detected by the simple syndrome checker, meaning that the ANN will not be used. The highest discrepancy between FER_{ANN} and FER_{ML} will be observed when outside of D remains a number closest to a valid codeword $\mathbf{c}'$, denoted by x'. Probability that channel creates such sequence is $P(\mathbf{c}')P(x'|\mathbf{c}') = p(1-p)^{(n-1)}/2^k$. Note that x' can be observed even if codewords other than $\mathbf{c}'$ are transmitted; however, in those cases ML decoder will also fail to correctly decode the codeword. □

The following theorem reveals the number of neurons that a SLFN should have, in order to learn the ML decoding.

There exists SLFN, with $2^n - 2^k$ neurons, that can learn the ML decoding.

Proof: Proof follows directly from [2], where it was shown that every one-dimensional discrete function with M samples can be accurately represented by 2-layer neural network with exactly $2M+1$ weights and ReLu activation function in each node of the hidden layer. The number of nodes is the network is equal to M. In order to learn the ML decoding the SLFN needs to learn all $M = 2^n - 2^k$ error patterns. □

The previous theorem showed that in case of the SLFN, the number of required neurons grows linearly with the size of training sample set. If we aim to learn the ML decoder, the complexity of the network is $O(2^n)$, where n denotes length of the code. We next compare the complexity of neural network, with trellis-based decoder. Given the fact that complexity of either ANN and trellis-based decoder is proportional to the number of branches on the graph, we compare two decoding approaches in terms of total graph branches. The ML decoder requires at most $2^{(n-k)+1}n$ branches, thus, based on Theorem 2, the total number of error patterns correctable by the SLFN, E, must satisfy $E < 2^{n-k+\log n}$, if we want to achieve complexity reduction, compared to trellis-based decoder. It follows that if we want to learn the complete space of 2^n samples, it needs to be satisfied $k < \log n$. For higher code rates, we need to reduce the number of training samples.

IV. Decoding linear block codes with TLFNs

In this section we examine the possibility of employing TLFNs for the problem of decoding linear block codes. The main advantage of TLFNs is that they can be constructed with smaller total number of nodes, compared to SLFNs with the same learning capabilities. Namely, Huang [3] has shown that to learn E input samples, it is possible to construct network with one output node and L_1 and L_2 nodes in the first and the second hidden layers, respectively,

$$L_1 = \sqrt{3E} + 2\sqrt{E/3}, \quad L_2 = \sqrt{E/3}.$$

Thus, total number of branches in the TLFN is equal to

$$L_1 + L_1 \times L_2 + L_2 = \frac{4}{3}E + \sqrt{E}(\sqrt{3} + \frac{2}{\sqrt{3}}) \approx \frac{4}{3}E.$$

Compared to the SLFN, the TLFN has approximately 33% less branches. However, for majority of significant linear block codes, trellis-based structure appears to be less complex.

Nevertheless, ANNs can be useful if the number of training samples is small, which means that we can train the ANN to correct only portion of error patterns, while majority of error patterns is corrected by the other, less complex decoder. Iterative decoders $\mathcal{D}$ of LDPC codes correct majority of errors, but fails to correct specific error patterns, that correspond to trapping sets of nodes on Tanner graphs. Trapping sets determine the code performance when channel error probability is low, i.e., we say that the decoder operates in the error-floor region. A trapping set is defied as a subgraph, which contains all the errors in the received sequence. However, all the variable nodes in the trapping set may not be erroneous. Trapping sets depend on the code structure and on the decoding rules. The biggest influence on code performance have the uncorrectable error patterns with lowest weights. If we denote by t the weight of the lowest uncorrectable pattern, FER in the error-floor region can be approximated by [14]

$$FER \approx e^{\log c_t + t \log p},$$

where c_t represents the number of weight-t error patterns that cannot be corrected by $\mathcal{D}$.

Numerous methods are proposed to determine the numbers t and c_t, and usually $c_t \ll \binom{n}{t}$. It follows that c_t error patterns can be corrected by the ANN, with relatively small number of nodes. Thus, we propose decoding approach in which the sequence received from the channel is decoded by $\mathcal{D}$, and in a case of decoding failure the output of $\mathcal{D}$ is sent to the ANN, previously trained on a set of uncorrectable error patterns $\mathcal{B}_t$. The cardinality of the training set depends on the code structure and decoder type. In some cases iterative decoding converges to a fixed trapping set, i.e., after some iteration all messages exchanged by nodes in the Tanner graph do not change. For example, when Margulis code is coupled with Gallager-B decoder, all weight-4 uncorrectable error patterns will cause the decoder to be stuck in a fixed trapping set. For such cases, to eliminate lowest-weight error patterns, the cardinality of the training set is $|\mathcal{B}_t| = c_t$. For other cases in is possible that decoding process oscillate between erroneous states. This means that to correct an error pattern, ANN needs to learn all the possible outcomes of $\mathcal{D}$ that correspond to that error pattern. Thus, the cardinality of the training set is bigger than c_t, and can be determine after careful enumeration. All in all, after ANN-based decoding FER is reduced to $\exp(\log c_{t+1} + (t+1)\log p)$.

It should be emphasized that the average number of channel errors per a codeword is pn, which is usually much larger than t. This means that weight-t error patterns are rare, and do not directly influence the code performance. However, during the decoding error patterns with larger weights are reduced

TABLE I: Complexity increase for different LDPC codes.

Code	Decoder	n	R	t	c_t	$\lvert\mathcal{B}_t\rvert$	c_{t+1}	ψ for $FER = 10^{-5}$
C_1-QC [10]	Offset min-sum	18432	0.903	4	19232	19232	15712	0.0015%
C_2-Tanner [11]	Gallager-A/B	155	0.4	3	155	310	456	0.007%
C_3-MacKay [12]	Gallager-A/B	1008	0.5	3	179	193	1215	0.00008%
C_4-MacKay [12]	Gallager-A/B	816	0.5	3	173	212	1372	0.00008%
C_5-Margulis [12]	Gallager-A/B	2640	0.5	4	1320	1320	11088	0.0001%
C_6-QC [12]	Gallager-A/B	900	0.5	3	50	100	675	0.000012%
C_7-QC [13]	Gallager-A	200	0.5	3	1434	2868	N/A	0.0015%

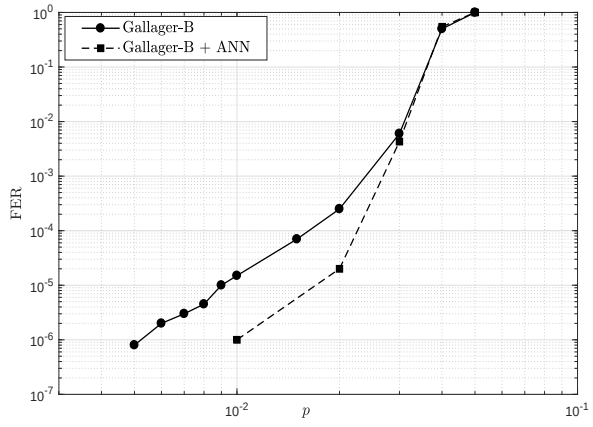

Fig. 1: Performance of ANN-based decoder (C_5 code).

to weight-t error patterns. This is the reason why we do not send sequence received from the channel to the ANN decoder, rather the output produced by $\mathcal{D}$.

Proposed serial concatenation of $\mathcal{D}$ and ANN decoder, dramatically increase the worst case complexity of the decoding. However, we argue that the average complexity, determined by the average number of graph branches, is only marginally increased. The average number of used graph branches can be calculated as follows

$$N_{b,avg} = N_{iter}\bar{\gamma}n + FER_T \times \Big[\frac{4}{3}|\mathcal{B}_t| + \sqrt{|\mathcal{B}_t|}(\sqrt{3} + \frac{2}{\sqrt{3}})\Big],$$

where the first term represents the contribution of $\mathcal{D}$, and the second the average complexity of the ANN decoder, while FER_T denotes FER without employment of the ANN. We can also define the relative complexity increase ψ as

$$\psi = (N_{b,avg}/(N_{iter}\bar{\gamma}n) - 1) \times 100\%.$$

We next examine complexity increase for different codes and $\mathcal{D}$ decoders. This is illustrated in Table I. At this point we do not provide a detail description of our example codes, but refer to referent works for more details. The average complexity of the proposed decoding approach is negligibly higher, if we aim to correct low-weight error patterns. On the other hand, adding ANN reduces the error rate for an order of magnitude, in the error-floor region. For example, for code C_1 and $p = 0.01$, eliminating $c_t = 4$ error patterns lowers error rate from $FER \approx 1.9 \times 10^{-4}$ to $FER \approx 1.4 \times 10^{-5}$.

To further illustrate the benefit of ANN-enhanced decoders, we conducted Monte Carlo simulation and evaluate the performance of C_5-Margulis code. The results of the simulation are presented in Fig. 1. We observed that applying ANN to correct residual weight-4 error patterns reduces the error floor of Margulis code by the order of magnitude.

V. Conclusion

In this paper we examined possibility of using TLFNs and SLFNs in decoding linear block codes. We showed that current knowledge of TLFNs and SLFNs cannot guarantee performance of ML decoding with significantly lower complexity, than trellis-based decoding. On the other hand, TLFNs can be efficient post-processing block in decoding LDPC codes and lower error-floors with small complexity penalty. Future work will involve more elaborate examination of employing ANNs in decoding practically significant LDPC codes and possibility of involving deep neural networks into decoding process.

References

[1] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control Signals Systems*, vol. 2, pp. 303–314, 1989.

[2] C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals, "Understanding deep learning requires rethinking generalization," *https://arxiv.org/abs/1611.03530*, Feb. 2017.

[3] G. Huang, "Learning capability and storage capacity of two-hidden-layer feedforward networks," *IEEE Trans. Neural. Net.*, vol. 14, no. 2, pp. 274–281, Mar 2003.

[4] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *in Proc. 54th Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2016.

[5] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2017.

[6] X. Xiao, B. Vasic, R. Tandon, and S. Lin, "Designing finite alphabet iterative decoders of ldpc codes via recurrent quantized neural networks," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 3963–3974, July 2020.

[7] F. Liang, C. Shen, and F. Wu, "An iterative BP-CNN architecture for channel decoding," *IEEE Journal of Select. Topics in Sig. Process.*, vol. 12, no. 1, pp. 144–159, Jan. 2018.

[8] J. He, "A deep learning-aided post-processing scheme to lower the error floor of LDPC codes," in *Proc. 2020 IEEE 20th Inter. Conf. on Commun. Technology (ICCT)*, Oct. 2020.

[9] A. Buchberger, C. Hager, H. Pfistery, L. Schmalenz, and A. Amat, "Learned decimation for neural belief propagation decoders," in *Proc. ICASSP 2021 - 2021 IEEE Inter. Conf. on Acoustics, Speech and Sig. Process. (ICASSP)*, June 2021.

[10] N. Raveendran, D. Declercq, and B. Vasic, "A sub-graph expansion-contraction method for error floor computation," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 3984–3995, July 2020.

[11] M. Karimi and A. Banihashemi, "On characterization of elementary trapping sets of variable-regular LDPC codes," *IEEE Trans. Inform. Theory*, vol. 60, no. 9, pp. 5188–5203, Sep. 2014.

[12] S. Chilappagari, S. Sankaranarayanan, and B. Vasic, "Error floors of LDPC codes on the binary symmetric channel," in *in Proc. 2006 IEEE Inter. Conf. Commun.*, June, 2006.

[13] H. Xiao and A. Banihashemi, "A sub-graph expansion-contraction method for error floor computation," *IEEE Trans. Commun.*, vol. 68, no. 7, pp. 3984–3995, July 2020.

[14] S. C. M. Ivkovic and B. Vasic, "Eliminating trapping sets in low-density parity check codes by using tanner graph covers," *IEEE Trans. Inform. Theory*, vol. 54, no. 8, pp. 3763–3768, Aug. 2008.