entropy

Article

Deep Ensemble of Weighted Viterbi Decoders for Tail-Biting Convolutional Codes

Tomer Raviv *, Asaf Schwartz * and Yair Be'ery *

School of Electrical Engineering, Tel-Aviv University, Tel-Aviv 6997801, Israel
* Correspondence: tomerraviv95@gmail.com (T.R.); asafs@altair-semi.com (A.S.); ybeery@eng.tau.ac.il (Y.B.)

Abstract: Tail-biting convolutional codes extend the classical zero-termination convolutional codes: Both encoding schemes force the equality of start and end states, but under the tail-biting each state is a valid termination. This paper proposes a machine learning approach to improve the state-of-the-art decoding of tail-biting codes, focusing on the widely employed short length regime as in the LTE standard. This standard also includes a CRC code. First, we parameterize the circular Viterbi algorithm, a baseline decoder that exploits the circular nature of the underlying trellis. An ensemble combines multiple such weighted decoders, and each decoder specializes in decoding words from a specific region of the channel words' distribution. A region corresponds to a subset of termination states; the ensemble covers the entire states space. A non-learnable gating satisfies two goals: it filters easily decoded words and mitigates the overhead of executing multiple weighted decoders. The CRC criterion is employed to choose only a subset of experts for decoding purpose. Our method achieves FER improvement of up to 0.75 dB over the CVA in the waterfall region for multiple code lengths, adding negligible computational complexity compared to the circular Viterbi algorithm in high signal-to-noise ratios (SNRs).

Keywords: deep learning; error correcting codes; viterbi, machine learning; ensembles; tail-biting convolutional codes

check for updates

Citation: Raviv, T.; Schwartz, A.; Be'ery, Y. Deep Ensemble of Weighted Viterbi Decoders for Tail-Biting Convolutional Codes. *Entropy* **2021**, *23*, 93. https://doi.org/10.3390/e23010093

Received: 27 November 2020
Accepted: 7 January 2021
Published: 10 January 2021

1. Introduction

Wireless data traffic has grown exponentially over recent years with no foreseen saturation [1]. To keep pace with connectivity requirements, one must carefully attend available resources with respect to three essential measures: reliability, latency, and complexity. As error correction codes (ECC) are well renowned as means to boost reliability, the research of practical schemes is crucial to meet demands.

One family of ECC that has had great impact on wireless standards is the convolutional codes (CC). Specifically, tail-biting convolutional codes (TBCC) [2] were incorporated in the 4G Long-Term Evolution (LTE) standard [3], and they are also considered for 5G hybrid turbo/LDPC code-based frameworks [4].

The major practical difference between CC and TBCC lies in the termination constraint. Conventional CC encoding appends zeros bits to impose zero states; TBCC encoding requires no additional bits, avoiding the rate loss. Due to this rate loss aversion, TBCC dominates classical CC in the short-length regime: They achieve the minimum distance bound for a specified length block codes [5]. Our work focuses on improving decoding performance of short length TBCC due to their significance.

Despite having great potential, the optimality of TBCC with respect to the reliability, latency, and complexity measures is not yet guaranteed. For example, TBCC suffer from increased complexity in the maximum-likelihood decoding as the initial state is unknown. Under TBCC encoding the Viterbi algorithm (VA) [6] is not the maximum-likelihood decoder (MLD); the MLD operates by running a VA per initial state, outputting the most likely decoded codeword. Clearly, the complexity grows as the number of states.

To bridge the high complexity gap, several suboptimal and reduced complexity decoders have been proposed. Major works include the circular Viterbi algorithm (CVA) [7] and the wrap-around Viterbi algorithm (WAVA) [8]. Both methods utilize the circular nature of the trellis: They apply VA iteratively on the sequence of repeated log-likelihood ratios (LLR) values computed from the received channel word. The more repetitions employed, the lower the error rate is, yet at a cost of additional latency. Short-length codes require the additional repetitions, as indicated in [7]: "it is very important that a sufficient number of symbol times be allowed for convergence. If the number is too small. . . the path chosen in this case will be circular but will not be the maximum likelihood path."

Another common practice is to employ a list decoding scheme, for instance, the list Viterbi algorithm (LVA), along with cyclic redundancy check (CRC) code [9–11]. According to this scheme, a list of the most likely decoded codewords, rather than a single codeword, is computed in the forward pass of the VA. The minimal path metric codeword that satisfies the CRC criterion is output. Both the decrement in error rate and the increase in complexity are proportional to the list size.

The additional repetitions and list size result in complexity overhead for short TBCC decoding. To mitigate this overhead, one may take a novel approach, rooted in a data-driven field: the machine learning (ML) based decoding.

Still a growing field, ML-based decoding attempts to bridge the gap between simple analytical models and the nonlinear observable reality. Contemporary literature is split between two different model choices: model-free and model-based. Model-free works include those in [12–14], which leverage on state-of-the-art (SOTA) neural architectures with high neuronal capacity (i.e., ones that are able to implement many functions). On the other hand, under model-based approaches [15–18], a classical decoder is assigned learnable weights and trained to minimize a surrogate loss function. This approach suffers from high inductive bias due to the constrained architecture, leading to limited hypothesis space. Nonetheless, it generalizes better to longer codes than the model-free approach: Empirical simulations show that unrealistic fraction of codewords from the entire codebook must be fed to the network to achieve even moderate performance (see Figure 7 in [12]). One notable model-based method by Shlezinger et al. is the ViterbiNet [19]. This method compensates for nonlinearity in the channel with expectation-maximization clustering; an NN is utilized to approximate the marginal probability. This method holds great potential for dealing with nonlinear channels.

One recent innovation, referred to as the ensemble of decoders [20], combined the benefits of model-based approach with the list decoding scheme. This ensemble is composed of learnable decoders, each one called an expert. Each expert is responsible for decoding channel words that lie in a unique part of the input space. A low-complexity gating function is employed to uniquely map each channel word to its respective decoder. The main intuition behind this divide-and-conquer approach is that combination of multiple *diverse* members is expected to perform better than all individual basic algorithms that compose the ensemble. The paper shows this approach can achieve significant gains: Up to 1.25 dB on the benchmark.

The main contributions of this paper are the innovation of the model-based weighted circular Viterbi algorithm (WCVA) and its integration in the gated WCVAE, a designated ensemble of WCVA decoders, accompanied by a gating decoder. Next, we elaborate on the following major points.

1. **WCVA**—A parameterized CVA decoder, combining the optimality of the VA with a data-driven approach in Section 3.1. Viterbi selections in the WCVA are based on the sums of weighted path metrics and the relevant branch metrics. The magnitude of a weight reflect the contribution of the corresponding path or branch to successful decoding of a noisy word.
2. **Partition of the channel words space**—We exploit the domain knowledge regarding the TBCC problem and partition the input space to different subsets of termination states in Section 3.3; Each expert specializes on codewords that belong to a single subset.

3. **Gating function**—We reinforce the practical aspect of this scheme by introducing a low-complexity gating that acts as a filter, reducing the number of calls to each expert. The gating maps noisy words to a subgroup of experts based on the CRC checksum (see Section 3.4).

   Simulations of the proposed method on LTE-TBCC appear in Section 4.
   Please see Supplementary Materials for introduction video and python code.

2. Background

2.1. Notation

Boldface upper-case and lower-case letters refer to matrices and vectors, respectively. Probability mass functions and probability density functions are denoted with $P(\cdot)$. Subscripts refer to elements, with the ith element of the vector $\boldsymbol{x}$ symbolized as x_i, while superscripts in brackets, e.g., $\boldsymbol{x}^{(j)}$, index the jth vector in a sequence of vectors. A slice of a vector $(x_i, \dots, x_j)$ is denoted by $x_{i;j}$. At last, $(\cdot)^T$ is for the transpose operation and $\|\cdot\|$ is for the L1 norm. Throughout the paper, terms i, j, k refer to indices.

2.2. Problem Formalization

Consider the block-wise transmission scenario of CC through the additive white Gaussian noise (AWGN) channel, see Figure 1. Prior to transmission, the message word $\boldsymbol{m} \in \{0,1\}^{N_m}$ is encoded twice: By an error detection code and by an error correction code. The CRC encodes $\boldsymbol{m}$ with systematic generator matrix $\boldsymbol{G}_{CRC}$. Its parity check matrix is $\boldsymbol{H}_{CRC}$. We denote the detection codeword by $\boldsymbol{u} \in \{0,1\}^{N_u}$ and the codebook with $\mathcal{U}$. Then, the CC encodes $\boldsymbol{u}$ with generator matrix $\boldsymbol{G}_{CC}$. As a result, the codeword $\boldsymbol{c}$ is a bits sequence $\boldsymbol{c} = (\boldsymbol{c}^{(1)}, \dots, \boldsymbol{c}^{(N_u)})$ with $\boldsymbol{c}^{(i)} \in \{0,1\}^{1/R_{CC}}$ where R_{CC} denotes the rate of the CC. For brevity, we denote $V = 1/R_{CC}$; the length of the CC calculated as $N_c = N_u \cdot V$.

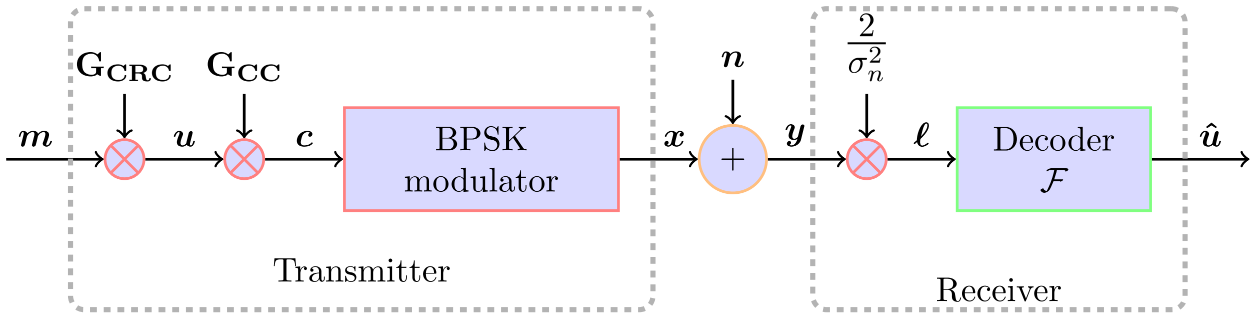


Figure 1. System diagram.

After encoding, the codeword $\boldsymbol{c}$ is BPSK-modulated ($0 \rightarrow 1$, $1 \rightarrow -1$) and $\boldsymbol{x}$ is transmitted through the channel with noise $\boldsymbol{n} \sim \mathcal{N}(\boldsymbol{0}, \sigma_n^2 \boldsymbol{I})$. At the receiver, one decodes the LLR word $\boldsymbol{\ell}$ rather than $\boldsymbol{y}$. The LLR values are approximated based on the bits i.i.d. assumption and Gaussian prior $\boldsymbol{\ell} = \frac{2}{\sigma_n^2} \cdot \boldsymbol{y}$. The decoder is represented by a function $\mathcal{F}(\cdot) : \mathbb{R}^{N_c} \rightarrow \{0,1\}^{N_u}$ that outputs the estimated detection codeword $\hat{\boldsymbol{u}}$. Our end goal is to find $\boldsymbol{u}$ that maximizes the a posteriori optimization problem:

$$\hat{\boldsymbol{u}} = \arg\max_{\boldsymbol{u} \in \mathcal{U}} P(\boldsymbol{u}|\boldsymbol{\ell}). \tag{1}$$

Note that we solve for $\boldsymbol{u}$ rather than $\boldsymbol{m}$ as bit flips in either the systematic information bits or in the CRC bits are considered as errors.

2.3. Viterbi Decoding of CC

Naive solution of Equation (1) is exponential in N_m. However, it can be simplified following Bayes:

$$\arg\max_{\boldsymbol{u}\in\mathcal{U}} P(\boldsymbol{u}|\boldsymbol{\ell}) = \arg\max_{\boldsymbol{u}\in\mathcal{U}} P(\boldsymbol{u})P(\boldsymbol{\ell}|\boldsymbol{u}) \tag{2}$$

where $P(\boldsymbol{\ell})$ is omitted, as this term is independent of $\boldsymbol{u}$.

The time complexity of the solution to Equation (2) is yet exponential, but may be further reduced to linear dependency in the memory's length by following the well known Viterbi algorithm (VA) [6]. We formulate notation for this algorithm in the following paragraphs.

Denote the memory of the CC by ν and the state space by $\mathcal{S} = \{0,\ldots,2^{\nu}-1\}$. Convolutional codes can be represented by multiple temporal transitions, each one is a function of two arguments: the input bit and the current state. The trellis diagram is one convenient way to view these temporal relations, each trellis section is called a stage. We refer to the work in [21] for a comprehensive tutorial regarding CC.

Let the sequence of states be represented by $\boldsymbol{s} \in \mathcal{S}^{N_u+1}$. Following the properties of the CC, a 1-to-1 correspondence between the codeword $\boldsymbol{u}$ and the state sequence $\boldsymbol{s}$ exists:

$$\arg\max_{\boldsymbol{u}\in\mathcal{U}} P(\boldsymbol{u})P(\boldsymbol{\ell}|\boldsymbol{u}) = \arg\max_{\boldsymbol{s}\in\mathcal{S}^{N_u+1}} P(\boldsymbol{s})P(\boldsymbol{\ell}|\boldsymbol{s}).$$

Plugging the Markov property into the previous equation leads to:

$$\begin{aligned}
&\arg\max_{\boldsymbol{s}\in\mathcal{S}^{N_u+1}} P(\boldsymbol{s})P(\boldsymbol{\ell}|\boldsymbol{s}) = \\
&\arg\max_{\boldsymbol{s}\in\mathcal{S}^{N_u+1}} \prod_{i=1}^{N_u} P(s_{i+1}|s_i)P(\boldsymbol{\ell}_{iV-V+1;iV}|s_{i+1},s_i) = \\
&\arg\max_{\boldsymbol{s}\in\mathcal{S}^{N_u+1}} \sum_{i=1}^{N_u} log(P(s_{i+1}|s_i) + log(P(\boldsymbol{\ell}_{iV-V+1;iV}|s_{i+1},s_i))
\end{aligned}$$

where the last transition is due to the monotonic nature of the log function.

Next, denote the path metric $\lambda_i = -log(P(s_{i+1}|s_i))$ and the branch metric, representing the transition over a trellis edge, as $\beta_i = -log(P(\boldsymbol{\ell}_{iV-V+1;iV}|s_{i+1},s_i))$. Then, substituting these values into the last equation:

$$\arg\max_{\boldsymbol{s}\in\mathcal{S}^{N_u+1}} P(\boldsymbol{s})P(\boldsymbol{\ell}|\boldsymbol{s}) = \arg\min_{\boldsymbol{s}\in\mathcal{S}^{N_u+1}} \sum_{i=1}^{N_u} \lambda_i + \beta_i. \tag{3}$$

Taking a dynamic programming approach, the Viterbi algorithm solves Equation (3) efficiently:

$$\lambda_i(s) = \min_{s'\in\mathcal{S}} \lambda_{i-1}(s') + \beta_i,\ s\in\mathcal{S} \tag{4}$$

starting from $i = 2$ up to $i = N_u + 1$, in an incremental fashion, with the initialization:

$$\lambda_1(s) = \begin{cases} -\lambda_{max} & \text{if } s = s_1 \\ 0 & \text{otherwise} \end{cases} \tag{5}$$

and s_1 = 0. The constant λ_{max} is called the LLR clipping parameter.

To output the decoded codeword $\hat{\boldsymbol{u}}$, one has to perform the trace-back operation $\Pi : \mathbb{R}^{N_c} \times \mathcal{S} \to \mathcal{U}$. This operation takes the LLR word along with a termination state and outputs the most likely decoded codeword: $\Pi(\boldsymbol{\ell}, s') = \hat{\boldsymbol{u}}$. Specifically, it calculates the sequence of states $\hat{\boldsymbol{s}}$ that follows the minimal $\lambda_i(s)$ values at each stage, starting from $s_{N_u+1} = s'$ backwards. Then, the sequence $\hat{\boldsymbol{s}}$ is mapped to the corresponding estimated codeword $\hat{\boldsymbol{u}}$. Under the classical zero-tail termination, $\hat{\boldsymbol{u}} = \Pi(\boldsymbol{\ell}, 0)$ is returned.

2.4. Circular Viterbi Decoding of TBCC

TBCC work under the assumption of equal start and end states. Their actual values are determined by the last ν bits. As such, the MLD with a list of size 1 is the decoded $\boldsymbol{u}$ whose matching $\lambda_{N_u+1}(s')$ value is minimal, combining the decisions from multiple VA runs, one from each state s'.

As mentioned in Section 1, the complexity of this MLD grows exponentially in the memory's length. The CVA is a suboptimal decoder that exploits the circular nature of the TBCC trellis, executing VA for a specified number of repetitions, where each new VA is initialized with the end metrics of the previous repetitions. The CVA starts and ends its run at the zero state, being error prone near the zero tails.

Explicitly, the forward pass of the CVA follows Equation (4) for $i \in \{2, \ldots, I \cdot N_u\}$ with I denoting an odd number of replications. The same initialization as in Equation (5) is used. The bits of the middle replication are the least error-prone, being farthest from the zero tails, thus returned:

$$\hat{u}_i = (\Pi(\boldsymbol{\ell}, 0))_{i+\left\lfloor \frac{I}{2} \right\rfloor \cdot N_u}$$

for $i \in \{1, \ldots, N_u\}$.

3. A Data-Driven Approach to TBCC Decoding

This section describes our novel approach to decoding: Parameterization of the CVA decoder, and its integration into an ensemble composed from specialized experts and a low-complexity gating.

3.1. Weighted Circular Viterbi Algorithm

Nachmani et al. [17] presented a weighted version of the classical Belief Propagation (BP) decoder [22]. This learnable decoder is the deep unfolding of the BP [23]. This weighted decoder outperforms the classical unweighted one by training over channel words, adjusting the weights to compensate for short cycles that are known to prevent convergence.

We follow the favorable model-based approach as well, parameterizing the branch metrics that correspond to edges of the trellis. We add another degree of freedom for each edge, assigning weights to the path metrics as well. Considering the complexity overhead, we only parameterize the middle replication. This formulation unfolds the middle replication of the CVA as a Neural Network (NN):

$$\lambda_i(s) = \min_{s' \in \mathcal{S}} w_{i,s',s} \lambda_{i-1}(s') + w_{i,\beta} \beta_i \tag{6}$$

for $\left\lfloor \frac{I}{2} \right\rfloor \cdot N_u \leq i \leq \left\lceil \frac{I}{2} \right\rceil \cdot N_u$.

Our goal is to calculate parameters $\{w_{i,s,s'}, w_{i,\beta}\}$ that achieve termination states equal to the ground-truth start and end states. The exact equality criterion is non-differentiable; thus, we minimize the multi-class cross entropy loss, acting as a surrogate loss [24]:

$$\mathcal{L}(\boldsymbol{s}, \boldsymbol{\lambda}) = -\log \sigma(\lambda^l(s_{N_u+1}))$$

where $\lambda^l(\cdot) = \lambda_{\left\lceil \frac{I}{2} \right\rceil \cdot N_u}(\cdot)$ stands for the last learnable layer, and σ being the softmax function:

$$\sigma(\lambda^l(s)) = \frac{e^{\lambda^l(s)}}{\sum\limits_{s' \in \mathcal{S}} e^{\lambda^l(s')}}$$

This specific choice encourages the equality of the end states in the mid-replication to their ground-truth values. Note that the gradients back-propagate through the non-differentiable min criterion in Equation (6) as in the maximum pooling operation: They only affect the state that achieved the minimum metric.

One fallacy of this approach is the similar importance for all edges, contrary to the BP, where not all edges are created equal (e.g., ones that participate in many short cycles). All edges are of the same importance as derived from the problem's symmetry: Due to the unknown initial state, each state is equally likely.

This indicates that training this architecture may leave the weights as they are, or at worst even lead to divergence. To fully exploit the performance gained by the adjustment of the weights, one must first break the symmetry. We alter the uniform prior over the termination states by assigning only a subset of the termination states to a single decoder. We further elaborate on this proposition below.

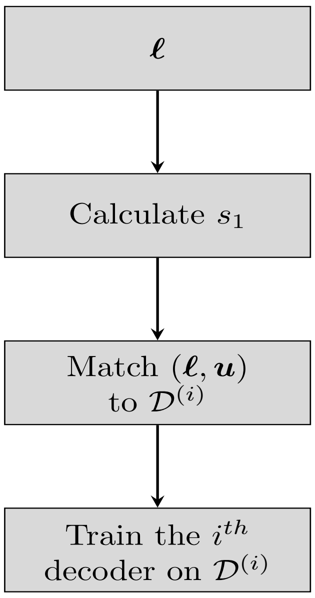
3.2. Ensembles in Decoding

Ensembles [25,26] shine in data-driven applications: They exploit independence between the base models to enhance accuracy. The expressive power of the ensemble surpasses that of a single model. Thus, whereas a single model may fail to capture high-dimensional and nonlinear relations in the dataset, a combination of such models may succeed.

Nonetheless, ensembles also encompass computational complexity which is linear in the number of base learners, being unrealistic for practical considerations. To reduce complexity, our previous work [20] suggests to employ a low complexity gating decoder. This decoder allows one to uniquely map each input word to a single most fitting decoder, keeping the overall computation complexity low. We further elaborate on the gated ensemble, referred to as gated WCVAE, in Section 3.3 and gating in Section 3.4.

3.3. Specialized-Experts Ensemble

The WCVAE is an ensemble comprised of WCVA experts, each one specialized on words from a specific subset of termination states. We begin by discussing the forming of the experts in training, see Figure 2 for the relevant flowchart.



$\boldsymbol{\ell}$
Calculate s_1
Match $(\boldsymbol{\ell}, \boldsymbol{u})$ to $\mathcal{D}^{(i)}$
Train the i^{th} decoder on $\mathcal{D}^{(i)}$

Figure 2. Training flowchart.

3.4. Gating

Let the number of trainable WCVA decoders in the WCVAE be α, with each decoder possessing I repetitions. To form the experts, we first simulate many message words randomly, each message word is encoded and transmitted through the channel. The initial state of a transmitted word $\boldsymbol{u}$, known in training, is denoted by s_1 as before. Then, we add the tuple $(\boldsymbol{\ell}, \boldsymbol{u})$ to the dataset representing the subset of states that include state s_1:

$$\mathcal{D}^{(i)} = \{(\boldsymbol{\ell}, \boldsymbol{u}) : \frac{2^\nu}{\alpha} \cdot (i-1) \leq s_1 \leq \frac{2^\nu}{\alpha} \cdot i - 1\} \tag{7}$$

with $i \in \{1, \ldots, \alpha\}$ and ν is yet the memory of the CC. Each dataset accumulates a high number of words by the above procedure.

All the WCVA decoders are trained as in the guidelines of Section 3.1, with one exception: The ith decoder is trained with the corresponding $\mathcal{D}^{(i)}$. Subsequently, α specialized experts are formed, each one specializes on decoding words affiliated to a specific subset of termination states. Each codeword has equal probability to appear, thus the distribution over the termination states is uniform. We further elaborate on the intuition to this particular division of close-by states in Section 4.4.

One common practice is to separate TBCC decoding into an initial state estimation followed by decoding. For example, Fedorenko et al. [27] run a soft-input soft-output (SISO) decoder prior to LVA decoding. This prerun determines the most reliable starting state.

Similarly, our work presents a gating decoder which acts as a coarse state estimation. The gating is composed of two parts: a single forward pass of the CVA and a multiple trace-backs phase. We only employ the gating in the evaluation phase; Check Figure 3 for the complete flow.

First, a forward pass of a CVA is executed on the input word $\boldsymbol{\ell}$, as in Equation (4). As all states are equiprobable, the initialization is chosen as $\lambda_1(s) = 0, \forall s \in \mathcal{S}$ instead of Equation (5). After calculating $\lambda_i(s)$ for every state and stage, the trace-back $\Pi(\cdot)$ runs α times, each time starting from a different state. The starting states are spread uniformly over $\mathcal{S}$, with the decoded words given as

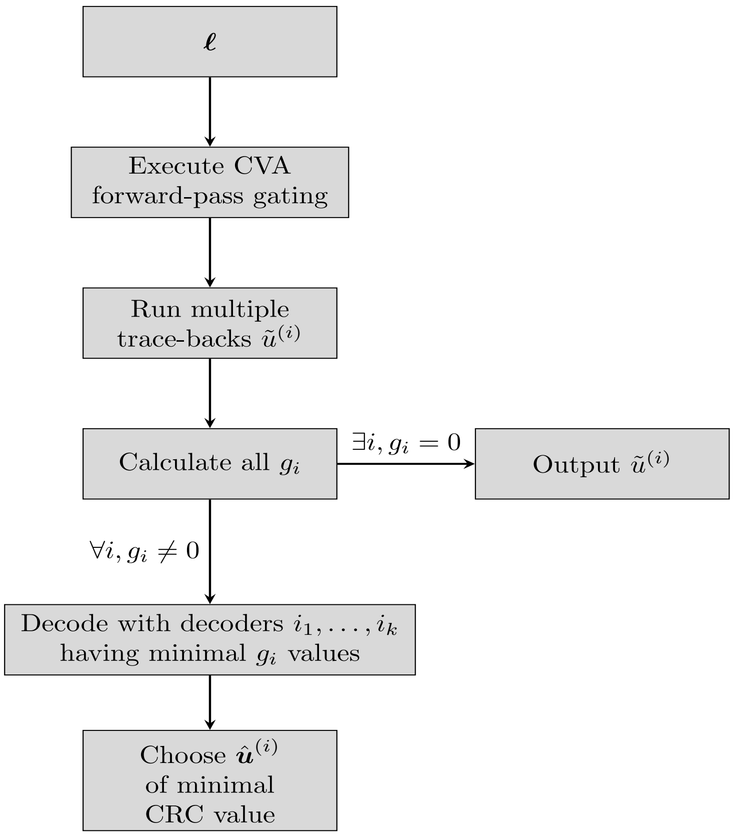

$\boldsymbol{\ell}$
Execute CVA forward-pass gating
Run multiple trace-backs $\tilde{u}^{(i)}$
Calculate all g_i
$\exists i, g_i = 0$
Output $\tilde{u}^{(i)}$
$\forall i, g_i \neq 0$
Decode with decoders $i_1, \ldots, i_k$ having minimal g_i values
Choose $\hat{\boldsymbol{u}}^{(i)}$ of minimal CRC value

Figure 3. Evaluation flowchart.

$$\tilde{\boldsymbol{u}}^{(i)} = \Pi(\boldsymbol{\ell}, \frac{2^\nu}{\alpha} \cdot (i - \frac{1}{2})), 1 \leq i \leq \alpha. \tag{8}$$

Notice that the trace-back is a cheap operation, compared to the forward pass [7]. Next, the value of the CRC syndrome is calculated for each trace-back:

$$g_i = \|\tilde{\boldsymbol{u}}^{(i)} \boldsymbol{H}_{CRC}^T\| \tag{9}$$

with $g_i = 0$ indicating that no error has occurred (or is detectable). In case that a single g_i is zero, the corresponding decoded word $\tilde{\boldsymbol{u}}^{(i)}$ is output. If more than one g_i is zero, the decoded word is chosen randomly among all candidates. Only if no i exists such that $g_i = 0$, the word $\boldsymbol{\ell}$ continues for additional decoding at the ensemble.

Note that each computed value $g_i \neq 0$ is correlative with the ascription of the word $\boldsymbol{\ell}$ to the termination state $\frac{2^\nu}{\alpha} \cdot (i - \frac{1}{2})$. Though the ground-truth termination state may not necessarily have the minimum g_i, this minimal value still hints that the ground-truth state is close by. As a result, decoding with the decoder corresponding to the minimal g_i is satisfactory. If multiple $g_{i_1}, \ldots, g_{i_k}$ share the same minimal value then decoders $i_1, \ldots, i_k$ decode the word, choosing the output $\hat{\boldsymbol{u}}^{(i)}$ of minimal CRC value among the candidates.

4. Results

4.1. Performance and Complexity Comparisons

The WCVAE was simulated with CRC codes and TBCC that are in accordance with the LTE standard. Note that while LTE employs QPSK modulation, we used BPSK for simplicity. A code of specific length is denoted with $(N_c,\ N_u,\ N_m)$, referring to the code's length, detection codeword's length, and message's length, respectively. A summary of relevant code parameters appears in Table 1.

Table 1. Code parameters

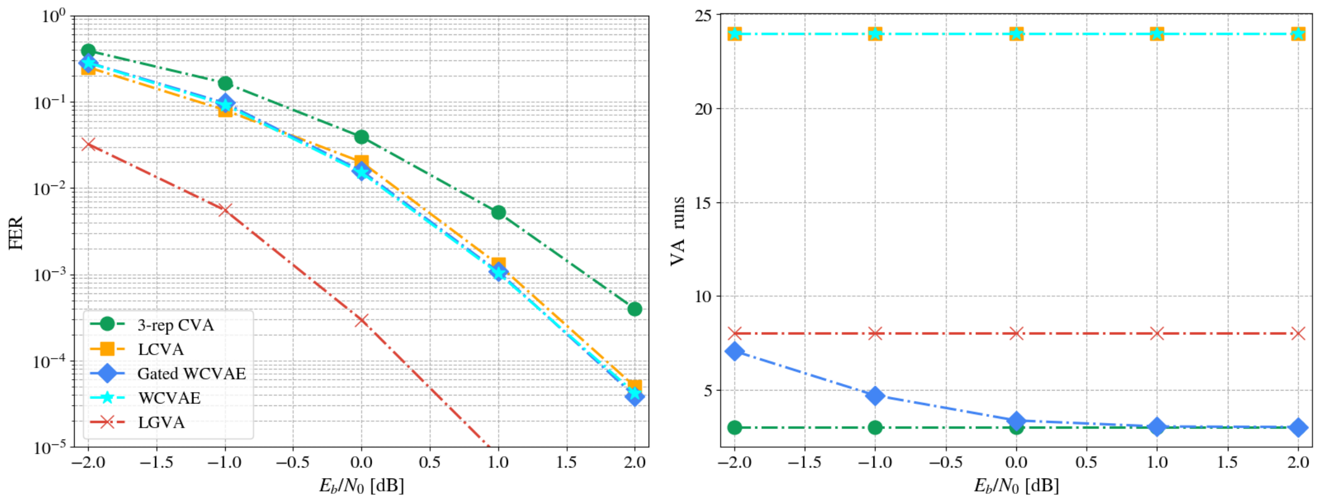
Symbol	Definition	Value
ν	CC memory size	6
-	CC polynomials	$(133, 171, 165)$
R_{CC}	CC rate	1/3
-	CRC length	16

We compared both the gated WCVAE and the WCVAE to the next common baselines:

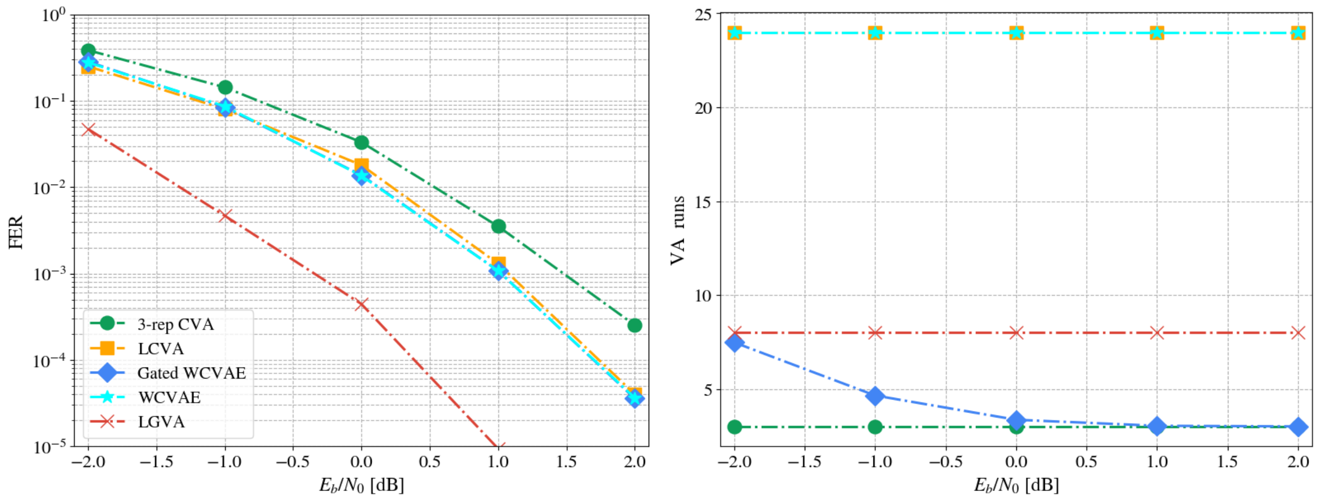
1. **3-repetition CVA**—a fixed-repetitions CVA [7].
2. **List circular Viterbi algorithm (LCVA)**—an LVA that runs CVA instead of a VA; All other details are as explained in Section 1.
3. **List genie VA (LGVA)**—an LVA decoder with list of size α, that runs from a known ground-truth state; The optimal decoded codeword is chosen by the CRC criterion. The FER of the gated and non-gated WCVAE are lower bounded by this genie-empowered decoder.

All Monte Carlo experiments ran on a validation dataset composed of signal-to-noise ratio (SNR) values in the range of -2 dB to 2 dB with a step of 1 dB. Simulations at each point continued until at least 500 errors were accumulated. The number of decoders was set to $\alpha = 8$. As words are drawn from the channels arbitrarily, the notion of "epoch" which refers to the number of full transitions over the training dataset is ill defined: We instead provide the number of training mini-batches. All decoders, i.e., the gating and the experts, were executed with $I = 3$ repetitions. The overall hyperparameters for the ensemble training are depicted in Table 2.

Figure 4 presents the results for the two different lengths: Both in error rate and computational complexity (measured in VA runs). The method achieves FER gains of up to 0.75 dB and 0.625 dB gain over the CVA in the waterfall region, for the lengths 13 and 15, respectively. Our method also surpasses the LCVA by a small margin. Considering the complexity of the scheme, the number of VA runs decreases as a function of the SNR and converges with the 3-repetition CVA in high SNR values. As the trace-back has negligible complexity compared to the forward pass of the VA [7], one may claim that the computational complexities at evaluation are similar.



(**a**) TBCC $(87, 29, 13)$



(**b**) TBCC $(93, 31, 15)$

Figure 4. FERand complexity plots for decoding LTE-TBCC.

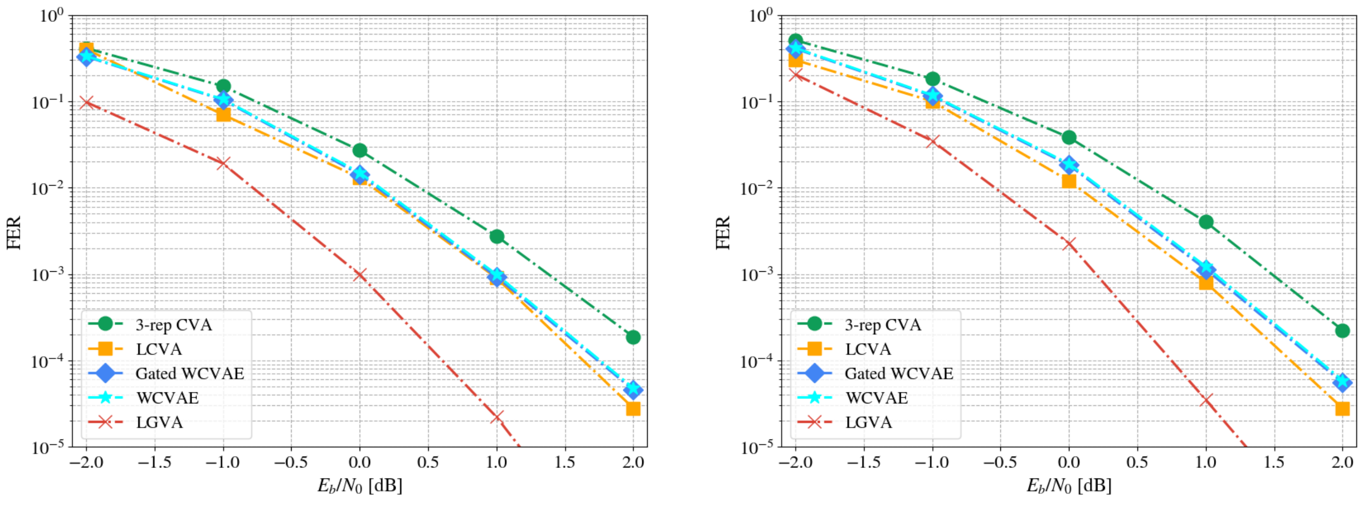
Table 2. Hyper-parameters of the ensemble

Symbol	Definition	Value
α	Ensemble size	8
I	Repetitions per decoder	3
λ_{max}	LLR clipping	20
lr	Learning rate	10^{-3}
-	Optimizer	RMSPROP
-	Loss	Cross Entropy
-	Training SNR range [dB]	(-2)–0
-	Mini-batch size	450
-	Number of mini-batches	50

4.2. Generalization to Longer Lengths

As mentioned in Section 1, one benefit of the model-based approach is the capability to easily generalize to longer codes. This benefit should apply to our proposed method; To test this notion, we trained and evaluated the WCVAE on the same code but over two longer lengths. All other codes parameters and training hyperparameters are exactly as in Tables 1 and 2.

Figure 5 depicts the performance over two longer codes. Notice that the gain is around the 0.6 dB in FER, similarly to the one on the two shorter codes. This empirically shows the generalization of the novel method to longer lengths. The training process remains as simple as before, even as the length increases; There is no need to enforce a curriculum based ramp-up method for convergence as in [14].



(**a**) TBCC $(138, 46, 30)$ (**b**) TBCC $(198, 66, 50)$

Figure 5. Generalization to longer lengths.

4.3. Training Analysis

We provide further insights to the benefits of training by studying the performance of the trained specialized decoders versus their non-trained counterparts. We fixed the code to TBCC $(87, 29, 13)$ and the SNR to 0 dB. Figure 6 depicts the FER as function of the termination states, each subplot shows two decoders: The classical CVA and the trained WCVA. The ith classical CVA had 3 repetitions, as before, and ran trace-back from state $\frac{2^\nu}{\alpha} \cdot (i-1)$. The trained WCVA is the ith decoder of the WCVAE, responsible for decoding states $\{\frac{2^\nu}{\alpha} \cdot (i-1), \dots, \frac{2^\nu}{\alpha} \cdot i - 1\}$. It ran trace-back from the same state. At each point, codewords of the given state, and **only this state**, were simulated until 250 accumulated errors.

One may observe that the CVA has peak performance at the trace-back state, yet at all other states it performs poorly. On the other hand, the WCVA decoders manage a trade-off: They sacrifice performance over the trace-back state, compensating for this loss by achieving lower error at other states. To conclude this part, note the specialized decoders indeed specialize at decoding words with a termination state included in their respective subset of termination states.

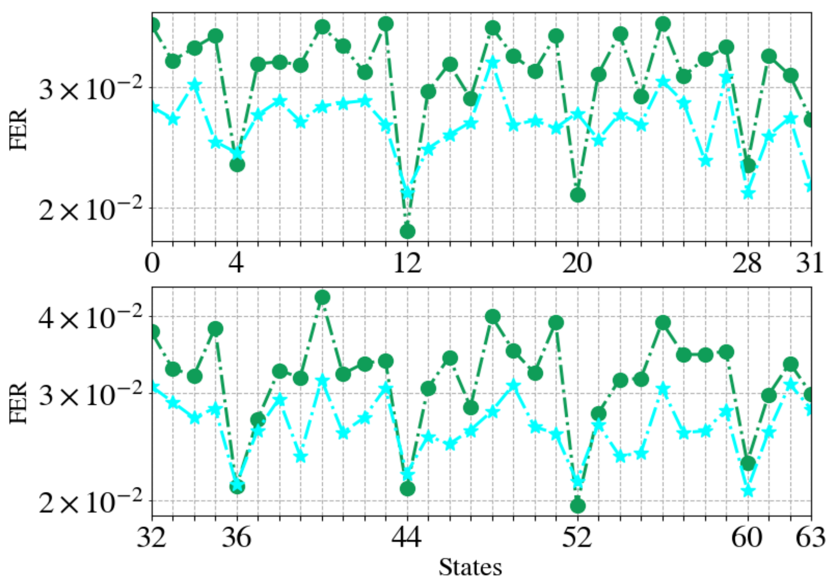


Figure 6. Training analysis. Legend: (●) CVA, (★) WCVA.

4.4. Ensemble Size Evaluation

In Figure 7, we inspect the performance of our method over different ensemble sizes, $\alpha \in \{4, 8, 16, 32\}$. An ensemble with α decoders is denoted by α-WCVAE. We fixed the code to TBCC $(87, 29, 13)$ and simulated each ensemble with 500 accumulated errors (per point). All other parameters and hyperparameters are the same as in Tables 1 and 2.

The figure implies that increasing the number of decoders by a factor of two results in around 0.1 dB FER gain. This simulation empirically validates an intuitive assumption: Our ensemble is more diverse as its size increases, i.e., it successfully captures a larger portion of the input space.

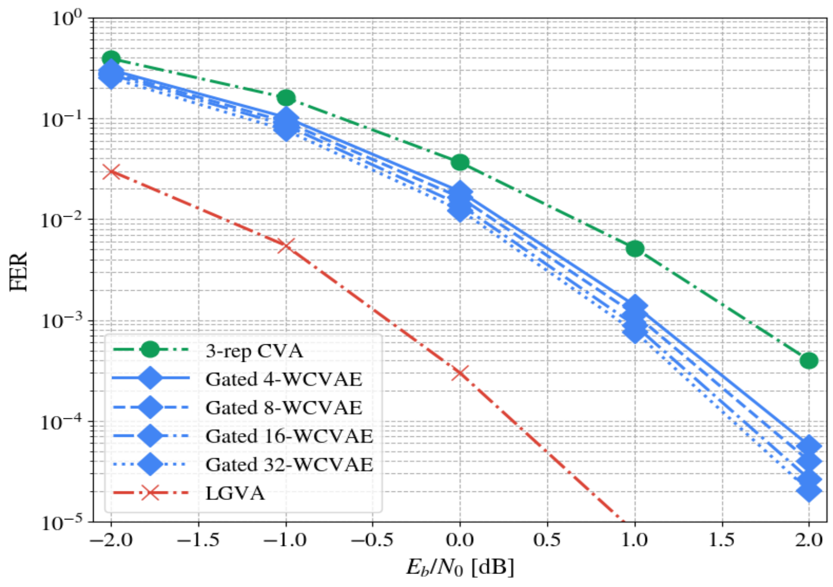


Figure 7. The effect of different ensemble sizes on performance.

5. Discussion

This work follows the model-based approach and applies it for TBCC decoding, starting with the parameterization of the common CVA decoder. Its parameterization relies on domain knowledge to effectively exploit the decoder: A classical low-complexity CVA acts as a gating decoder, filtering easy to decode channel words and directing harder ones to fitting experts; Each expert is specialized in decoding words that belong to a specific subset of termination states. This solution improves the overall performance, compared to a single decoder, as well as reduces the complexity in a data-driven fashion.

Future directions are to extend the ensemble approach to more use cases, such as different codes and various learnable decoders. For example, an ensemble tailored for polar codes, with a CRC-based gating, is one idea we intend to explore. This scenario is indeed practical, as 5G standard incorporates polar codes accompanied by CRC codes. Another direction is the theoretical study and analysis of the input space, e.g., the regions of the pseudo-codewords [5] and tailbits errors [7]. This could direct the training of the learnable decoder to surpass current results.

Supplementary Materials: An introduction video and python code are available at https://www.youtube.com/watch?v=nrP61KiG8fE and https://github.com/tomerraviv95/TailBitingCC, respectively.

Author Contributions: Conceptualization, A.S.; Formal analysis, T.R.; Project administration, Y.B.; Software, T.R. and A.S.; Supervision, Y.B.; Validation, A.S.; Writing—original draft, T.R. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data simulation is found in the github repository.

Acknowledgments: The authors are indebted to Nir Raviv for insightful discussions. We also thank the anonymous reviewers for their helpful feedback.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Cisco, V. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2017–2022. Cisco White Paper. Available online: https://davidellis.ca/wp-content/uploads/2019/12/cisco-vni-mobile-data-traffic-feb-2019.pdf (accessed on 6 January 2021).
2. Ma, H.; Wolf, J. On tail biting convolutional codes. *IEEE Trans. Commun.* **1986**, *34*, 104–111. [CrossRef]
3. LTE. Evolved Universal Terrestrial Radio Access (E-UTRA); Multiplexing and Channel Coding (3GPP TS 36.212 Version 13.10.0 Release 13). Available online: https://www.etsi.org/deliver/etsi_ts/136200_136299/136212/13.10.00_60/ts_136212v131000p.pdf (accessed on 6 January 2021).
4. Maunder, R.G. A Vision for 5G Channel Coding. Available online: https://eprints.soton.ac.uk/401809/ (accessed on 6 January 2021).
5. Stahl, P.; Anderson, J.B.; Johannesson, R. Optimal and near-optimal encoders for short and moderate-length tail-biting trellises. *IEEE Trans. Inf. Theory* **1999**, *45*, 2562–2571. [CrossRef]
6. Viterbi, A. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inf. Theory* **1967**, *13*, 260–269. [CrossRef]
7. Cox, R.V.; Sundberg, C.E.W. An efficient adaptive circular Viterbi algorithm for decoding generalized tailbiting convolutional codes. *IEEE Trans. Veh. Technol.* **1994**, *43*, 57–68. [CrossRef]
8. Shao, R.Y.; Lin, S.; Fossorier, M.P. Two decoding algorithms for tailbiting codes. *IEEE Trans. Commun.* **2003**, *51*, 1658–1665. [CrossRef]
9. Seshadri, N.; Sundberg, C.E. List Viterbi decoding algorithms with applications. *IEEE Trans. Commun.* **1994**, *42*, 313–323. [CrossRef]
10. Chen, B.; Sundberg, C.E. List Viterbi algorithms for continuous transmission. *IEEE Trans. Commun.* **2001**, *49*, 784–792. [CrossRef]
11. Kim, J.W.; Tak, J.W.; Kwak, H.Y.; No, J.S. A new list decoding algorithm for short-length TBCCs with CRC. *IEEE Access* **2018**, *6*, 35105–35111. [CrossRef]
12. Gruber, T.; Cammerer, S.; Hoydis, J.; ten Brink, S. On deep learning-based channel decoding. In Proceedings of the 51st Annual Conference on Information Sciences and Systems (CISS), Baltimore, MD, USA, 22–24 March 2017.
13. Kim, H.; Jiang, Y.; Rana, R.; Kannan, S.; Oh, S.; Viswanath, P. Communication algorithms via deep learning. *arXiv* **2018**, arXiv:1805.09317.
14. Tandler, D.; Dörner, S.; Cammerer, S.; Brink, S.T. On Recurrent Neural Networks for Sequence-based Processing in Communications. *arXiv* **2019**, arXiv:1905.09983.

15. Nachmani, E.; Be'ery, Y.; Burshtein, D. Learning to decode linear codes using deep learning. In Proceedings of the Annual Allerton Conference on Communication, Control, and Computing (Allerton), Monticello, IL, USA, 27–30 September 2016.
16. Nachmani, E.; Marciano, E.; Burshtein, D.; Be'ery, Y. RNN decoding of linear block codes. *arXiv* **2017**, arXiv:1702.07560.
17. Nachmani, E.; Marciano, E.; Lugosch, L.; Gross, W.J.; Burshtein, D.; Be'ery, Y. Deep learning methods for improved decoding of linear codes. *IEEE J. Sel. Top. Signal Process.* **2018**, *12*, 119–131. [CrossRef]
18. Be'ery, I.; Raviv, N.; Raviv, T.; Be'ery, Y. Active deep decoding of linear codes. *IEEE Trans. Commun.* **2019**, *68*, 728–736. [CrossRef]
19. Shlezinger, N.; Farsad, N.; Eldar, Y.C.; Goldsmith, A.J. ViterbiNet: A deep learning based Viterbi algorithm for symbol detection. *IEEE Trans. Wirel. Commun.* **2020**, *19*, 3319–3331. [CrossRef]
20. Raviv, T.; Raviv, N.; Be'ery, Y. Data-Driven Ensembles for Deep and Hard-Decision Hybrid Decoding. In Proceedings of the IEEE International Symposium on Information Theory (ISIT), Angeles, CA, USA, 21–26 June 2020; pp. 321–326. [CrossRef]
21. Moon, T.K. *Error Correction Coding: Mathematical Methods and Algorithms*; John Wiley & Sons: Hoboken, NJ, USA, 2005.
22. Pearl, J. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*; Elsevier: Hoboken, NJ, USA, 2014.
23. Hershey, J.R.; Roux, J.L.; Weninger, F. Deep unfolding: Model-based inspiration of novel deep architectures. *arXiv* **2014**, arXiv:1409.2574.
24. Goodfellow, I.; Bengio, Y.; Courville, A. *Deep Learning*; MIT Press: Cambridge, MA, USA, 2016.
25. Rokach, L. *Pattern Classification Using Ensemble Methods*; World Scientific: Singapore, 2010; Volume 75.
26. L. Rokach. Ensemble-based classifiers. *Artif. Intell. Rev.* **2010**, *33*, 1–39. [CrossRef]
27. Fedorenko, S.V.; Trefilov, M.; Wei, Y. Improved list decoding of tail-biting convolutional codes. In Proceedings of the 2014 XIV International Symposium on Problems of Redundancy in Information and Control Systems, St. Petersburg, Russia, 1–5 June 2014; pp. 35–38.