The Trainable BCJR and Its Applications

The Trainable BCJR and Its Applications

Martina Magnaldi, *Student Member, IEEE*, Guido Montorsi, *Fellow, IEEE*

***Abstract*—We propose a trainable version of the Additive Bahl-Cocke-Jelinek-Raviv (A-BCJR) algorithm by interpreting it as a Recurrent Neural Network (RNN). By leveraging the smoothness and differentiability of the max* operator, the core component of the A-BCJR forward and backward recursions, we derive a backpropagation algorithm that enables end-to-end training from the network's output. The resulting model, referred to as T-BCJR, comprises a linear layer that computes edge metrics from state and input metrics, followed by a nonlinear max* layer that marginalizes these metrics back to the state and output domains. We further derive the corresponding delta backpropagation recursions, which exhibit a structural symmetry with the original forward-backward predictive steps. Unlike prior approaches that only replaced memoryless metric units in BCJR or Viterbi algorithms with Neural Networks (NNs), T-BCJR enables full trainability of the entire recursive structure. We derive a trainable detector, termed T-Detector, by connecting the T-BCJR to the channel output via an additional convolutional linear layer that emulates a channel shortening filter. The T-Detector can be trained to operate as a modulation- and channel-agnostic detector, capable of performing channel shortening, Maximum Likelihood (ML) sequence detection, and the computation of symbol-level or bit-level Log-Likelihoods (LLs) for soft-input decoding. Moreover, it is compatible with iterative receiver schemes involving outer channel decoders. The predictive step of the T-Detector maintains the same Digital Signal Processing (DSP) complexity as a conventional model-based detector, with the added benefit of being trainable from a cost function defined on the generated LLs. Experimental results demonstrate that the T-Detector can be trained to match the performance of the corresponding model-based detector for static and slow-time varying channels with appropriate pilot densities.**

***Index Terms*—Additive SISO, BCJR algorithm, model-based neural networks, Maximum Likelihood equalization, channel shortening**

I. Introduction

DEEP LEARNING (DL) and machine learning architectures have gained significant popularity over the past decades due to their remarkable performance in various fields, including image and speech recognition, as well as diverse domains such as biology and medicine [1]. This growing success has also driven interest in applying DL to communication systems across different layers of the Open Systems Interconnection (OSI) stack [2]. In [3], a survey of the initial deployment of DL on the physical layer signal processing of communication systems is presented. The initial applications of DL in physical layer signal processing focused on replacing traditional Digital Signal Processing (DSP) blocks with off-the-shelf Neural Networks (NNs), which were originally developed for different tasks. In [4], the authors use reinforcement learning-optimized NN autoencoders to remove the need for channel gradients, while in [5], a deep Convolutional NN (CNN) is presented that fully executes the 5G receiver pipeline by treating the receiver's task as a supervised learning problem. In this context, Recurrent NNs (RNNs) have gained considerable attention for their ability to handle sequential inputs, like data transmissions, by retaining past information in hidden states [6]. In [7], for example, the authors use a sliding bidirectional RNN for training a detector without knowledge of the underlying channel model. It is widely recognized that NNs offer flexibility by being data-driven and channel model agnostic, and adaptability across various channel conditions [2], [8], [9]. This comes in contrast to model-based classical receivers that rely on specific models and unrealistic assumptions [2], [10]. To name a few, ideal quantization of channel outputs, absence of nonlinear effects, knowledge of spectral density properties of additive noise, IQ imbalance, and knowledge of the channel's autocorrelation function. Consequently, conventional detection algorithms, such as Viterbi [11] and BCJR (Bahl-Cocke-Jelinek-Raviv) [12], face significant challenges in the absence of precise channel knowledge and accurate Channel State Information (CSI), which NNs do not require. However, NN architectures demand extensive training due to their "black box" nature [13]. As a consequence, these initial attempts have raised two other main criticisms. Firstly, the proposed NNs have much higher DSP complexity than classical models [14]. Processing complexity directly relates to component cost, power consumption, and/or maximum throughput, so its practicality is called into question. Secondly, the resulting network lacks of interpretability [2]. To overcome these issues, recent research has introduced "model-based" NNs for digital receivers, replicating traditional DSP blocks while retaining flexibility [8], [14], [15]. These networks are trainable from data and adaptable to various channel models, with processing complexity similar to classical receivers. An excellent survey of model-based DL approaches is presented in [8], while [14] discusses key challenges faced by these emerging networks. In this line of research, several model-based DL methods were developed. We can mention [16]–[20], which integrated NNs for CSI estimation into the Viterbi, BCJR, interference cancellation [21] algorithm and Kalman filter estimator. In [22], BCJRNet is adapted to handle imperfect channel knowledge and memory assumptions. In [23] is proposed a set of data augmentation techniques tailored to digital communication data, while [24] presents a predictive online meta-learning algorithm for DNN-based receivers, allowing fast adaptation to time-varying channels using minimal pilots and re-encoding successfully decoded words.

This work was supported by the European Union - Next Generation EU under the Italian National Recovery and Resilience Plan (NRRP), Mission 4, Component 2, Investment 1.3, CUP E13C22001870001, partnership on "Telecommunications of the Future" (PE00000001 - program "RESTART").
Authors' addresses: Martina Magnaldi and Guido Montorsi are with the Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Torino, Italy, E-mail: (martina.magnaldi@polito.it; guido.montorsi@polito.it).

Within the same area of study, in a previous conference paper [25], we proposed using the T-BCJR, a new trainable version of the *additive* BCJR (A-BCJR) [26], whose structure reflects an underlying trellis diagram. As a trainable architecture, the T-BCJR operates in two distinct modes: a predictive phase and a training phase. The predictive phase corresponds to the conventional BCJR algorithm, whereas the training phase introduces an additional mechanism that allows the model to update its parameters using known reference symbols. As a consequence, the T-BCJR has the added benefit of being fully trainable from the output, which takes the form of bits or symbol Log-Likelihoods (LLs). For these reasons, it differs from other proposals like [16], [18], [19]. This is accomplished by exploiting the smoothness and the differentiability of the max* operator (log-sum of exponentials), the core operator in the A-BCJR's forward and backward recursions. The T-BCJR consists of a linear layer to form the edge metrics from the state and input metrics, followed by a nonlinear max* layer to marginalize the edge metrics back to the state and output spaces. It is defined by four trainable matrices describing the additive BCJR equations. We derived the forward and backward recursions for delta propagation to train the matrices' weights from the output cost function.

This work aims to provide a complete and self-contained exposition of the novel approach initially introduced in [25]. To this end, we begin with a detailed description of the T-BCJR, a representation of the A-BCJR as a RNN. Building on this foundation, we then introduce the central contribution of our study: a fully trainable T-Detector architecture for inter-symbol interference (ISI) channels that incorporates the T-BCJR as its core component. Preliminary descriptions and specific application examples of the T-Detector have been described in previous conference papers [27], [28]. The T-Detector is based on the channel shortening approach [29], [30], using a convolutional linear layer to mimic the presence of the shortening filter before the T-BCJR. The purpose of this layer is to transform the received signal in such a way that a BCJR trellis detector with reduced memory can achieve the maximum information rate possible, while operating under the constraint of a fixed trellis memory. The T-Detector can be trained to implement a channel and modulation agnostic detector, able to execute all the digital baseband processing from analog-to-digital (A/D) converter to the delivery of LL for the following soft-input decoder. The DSP complexity of the T-Detector matches that of the corresponding classical receiver, while offering the added advantage of being trainable directly from the cost function defined on the output LLs, a characteristic inherited from the T-BCJR. The resulting framework significantly extends all prior contributions by providing a unified formulation of the T-Detector for ISI channels.

In the result section, we consider a classical single-carrier system affected by ISI. We study both static and slowly time-varying channel conditions. Our analysis begins with the static scenario, where we introduce a comparison among our T-Detector, the traditional model-based BCJR receiver and the BCJRNet proposed in [18] using the Bit Error Rate (BER) metric to show the effectiveness of our work with respect to the state-of-the-art. We then extend the analysis to two additional statistical channel models: the first characterized by a strong but short ISI, while the second corresponds to a more realistic scenario, represented by the TDL-A channel defined in the 5G NR standard. In these cases, the Mutual Information (MI) is used as the performance metric. Results demonstrate that, when the memory of the T-BCJR is larger than zero, the T-Detector achieves substantial improvement over Minimum Mean Square Error (MMSE), approaching the performance of the Maximum Likelihood Sequence Estimation (MLSE) receiver or the optimal shortening detector proposed in [29]. In contrast, with no memory, indicating no trellis processing, it behaves as MMSE but with the crucial difference that it embeds the LLs computation block. Next, we evaluate the detector on the same ISI channel under time-varying conditions by introducing a training phase with interspersed pilot symbols. We investigate its performance across different Doppler frequencies and pilot densities. This preliminary analysis has also been reported in [27]. Finally, we perform a complexity analysis expressed both in Big-O notation, $\mathcal{O}(\cdot)$, and in terms of floating-point operations (FLOPs).

The remainder of the paper is organized as follows. Section II introduces the T-BCJR, whose structure is inherited from the additive version of the classic BCJR. Subsection II-A, details the predictive step, while Subsection II-B derives the fundamental recursions used for training via δ-backpropagation. Special cases and generalizations are discussed in Subsection II-C, while Subsection II-D highlights similarities and differences between T-BCJR and BiLSTM. In Section III the T-Detector is presented, which integrates the T-BCJR as its central component. Subsection III-A describes how to train the full detector using an external cost function, Subsection III-B clarifies the distinction between trainable and fixed matrices in the context of the assumed ISI channel and Subsection III-C shows the fundamental differences between T-Detector and classical ASG-MMSE. Section IV reports simulation results demonstrating the effectiveness of the T-Detector across various statistical channels, along with an analysis of its training time. A discussion on the extension to slowly time-varying channels is provided in Section V. In Section VI a detailed complexity analysis is performed. Finally, Section VII concludes the paper and outlines directions for future research.

A. Notations

Throughout the paper, we use boldface letters only for denoting matrices, e.g., $\mathbf{A}$. We use lower- or upper-case letters to represent column vectors, e.g., a, and add round brackets to denote specific elements of vectors or matrices. We use the letter T in superscript to indicate the transpose, e.g., $\mathbf{A}^T$ or a^T.

II. The T-BCJR

We follow the notations in [26], [31], which provide a general description of the BCJR algorithm in both additive and multiplicative versions. The A-BCJR described in [26] and depicted in Fig. 1 is a four-port device (two input and two output ports), accepting as input the sequence of LLs on the input and output symbols $\pi^I_{u/c}$ of a Finite State Machine (FSM)

Fig. 1. The Soft-Input Soft-Output (SISO) A-BCJR module.

and providing as output the sequence of extrinsic LLs $\pi_{u/c}^O$, an updated version of the two. According to [26], the FSM is specified by a single trellis section. For this purpose, we define:

- a set of states $\mathcal{S}$ of cardinality $|\mathcal{S}|$,
- a set of input symbols $\mathcal{U}$ of cardinality $|\mathcal{U}|$,
- a set of output symbols $\mathcal{C}$ of cardinality $|\mathcal{C}|$.

The set of edges $\mathcal{E}$, of cardinality $|\mathcal{E}| = |\mathcal{U}| \times |\mathcal{S}|$, is the Cartesian product of the set of states and input symbols. A trellis section is described by providing four functions $s^S(e)$, $u(e)$, $c(e)$, $s^E(e)$, which associate with each edge e the corresponding starting state, input symbol, output symbol, and ending state.

The equations of the A-BCJR that accept and provide LL $\pi_{u/c}^{IO}$ on input/output symbols read:

$$\alpha^+(s) = \overset{*}{\underset{e:\ s^E(e)=s}{\max}} \{\alpha(s^S(e)) + \pi_c^I(c(e)) + \pi_u^I(u(e))\}\ , \quad (1)$$

$$\beta^-(s) = \overset{*}{\underset{e:\ s^S(e)=s}{\max}} \{\beta(s^E(e)) + \pi_c^I(c(e)) + \pi_u^I(u(e))\}\ , \quad (2)$$

$$\pi_u^O(u) = \overset{*}{\underset{e:\ u(e)=u}{\max}} \{\alpha(s^S(e)) + \pi_c^I(c(e)) + \beta(s^E(e))\}\ , \quad (3)$$

$$\pi_c^O(c) = \overset{*}{\underset{e:\ c(e)=c}{\max}} \{\alpha(s^S(e)) + \pi_u^I(u(e)) + \beta(s^E(e))\}\ , \quad (4)$$

where $\alpha(s)$ is the LL of the state s given the past observations, and $\beta(s)$ is the LL of the state s given the future observations. In Eqs. 1 and 2, the time index is omitted for notational simplicity. The state metrics at the next and previous trellis steps are denoted by $\alpha^+(s)$ and $\beta^-(s)$, respectively. In the following sections, we first demonstrate how the A-BCJR equations can be represented as the output of a RNN, and subsequently derive the training recursions required to update its weights from the delivered outputs.

A. Representation of the A-BCJR as a RNN

The A-BCJR algorithm, as defined in Eqs. 1-4, can be interpreted as an RNN due to its structure, which integrates both the linear and nonlinear operations characteristic of NNs.

To highlight this correspondence, we start by representing the metrics involved in the A-BCJR equations as the result of a **linear** operation on the vectors $\alpha,\ \beta,\ \pi_c^I,\ \pi_u^I$:

$$\epsilon_\alpha(e) \triangleq \alpha(s^S(e)) \to \epsilon_\alpha = \mathbf{W}_S\alpha, \quad (5)$$

$$\epsilon_\beta(e) \triangleq \beta(s^E(e)) \to \epsilon_\beta = \mathbf{W}_E\beta, \quad (6)$$

$$\epsilon_i(e) \triangleq \pi_c^I(c(e)) \to \epsilon_i = \mathbf{W}_C\pi_c^I, \quad (7)$$

$$\epsilon_i'(e) \triangleq \pi_u^I(u(e)) \to \epsilon_i' = \mathbf{W}_U\pi_u^I, \quad (8)$$

where $\mathbf{W}_S$, $\mathbf{W}_E$ are $|\mathcal{E}| \times |\mathcal{S}|$ matrices with one-hot rows that embed the structure of "label-free" trellis associated to the $s^S(\cdot)$ and $s^E(\cdot)$ functions, respectively. The matrices $\mathbf{W}_C$ and $\mathbf{W}_U$ embed the structure of the labeling functions $c(\cdot)$ and $u(\cdot)$, associating the input and output symbols to each edge. With this notation, Eqs. 1-4 can be rewritten as:

$$\alpha^+(s) = \overset{*}{\underset{e:\ s^E(e)=s}{\max}} \{\epsilon_f(e)\}\ , \quad (9)$$

$$\beta^-(s) = \overset{*}{\underset{e:\ s^S(e)=s}{\max}} \{\epsilon_b(e)\}\ , \quad (10)$$

$$\pi_u^O(u) = \overset{*}{\underset{e:\ u(e)=u}{\max}} \{\epsilon_o'(e)\}\ , \quad (11)$$

$$\pi_c^O(c) = \overset{*}{\underset{e:\ c(e)=c}{\max}} \{\epsilon_o(e)\}\ , \quad (12)$$

where we define the edge metrics as $\epsilon_f = \epsilon_\alpha + \epsilon_i + \epsilon_i'$, $\epsilon_b = \epsilon_\beta + \epsilon_i + \epsilon_i'$, $\epsilon_o = \epsilon_\alpha + \epsilon_\beta + \epsilon_i'$, $\epsilon_o' = \epsilon_\alpha + \epsilon_\beta + \epsilon_i$ and the max* as:

$$\overset{*}{\max_i} x(i) = \ln \sum_i \exp(x(i)). \quad (13)$$

The results of the A-BCJR equations are obtained by applying the smooth **nonlinear** max* operator over proper subsets of edge metrics. These subsets are determined by the same four functions that define the trellis section.

We then introduce the four matrices of log-weights, $\mathbf{w}_x$ for $x \in \{S, E, U, C\}$, where the elements of $\mathbf{W}_x$ are related to the corresponding elements of $\mathbf{w}_x$ in the following way:

$$\mathbf{W}_x(i,j) = \frac{e^{\mathbf{w}_x(i,j)}}{\sum_j e^{\mathbf{w}_x(i,j)}}. \quad (14)$$

Since the log-weights can take values of either 0 or $-\infty$, they can be used to remove the subset constraints on the max* operator, allowing us to express:

$$\alpha^+(s) = \overset{*}{\underset{e}{\max}}\{\mathbf{w}_E(e,s) + \epsilon_f(e)\}\ , \quad (15)$$

$$\beta^-(s) = \overset{*}{\underset{e}{\max}}\{\mathbf{w}_S(e,s) + \epsilon_b(e)\}\ , \quad (16)$$

$$\pi_u^O(u) = \overset{*}{\underset{e}{\max}}\{\mathbf{w}_U(e,u) + \epsilon_o'(e)\}\ , \quad (17)$$

$$\pi_c^O(c) = \overset{*}{\underset{e}{\max}}\{\mathbf{w}_C(e,c) + \epsilon_o(e)\}\ . \quad (18)$$

In fact, all edges for which $\mathbf{w}(e,s) = -\infty$ are excluded from the output. We now introduce a *matrix* notation for the $\max^*$ operator, defined as:

$$\mathbf{C} = \mathbf{A} \boxplus \mathbf{B} : \mathbf{C}(i,j) = \overset{*}{\underset{k}{\max}}\{\mathbf{A}(i,k) + \mathbf{B}(k,j)\}. \quad (19)$$

Using this notation, the A-BCJR recursions and output computations can be expressed in a concise vector form as follows:

$$\alpha^+ = \mathbf{w}_E^T \boxplus \epsilon_f, \quad (20)$$

$$\beta^- = \mathbf{w}_S^T \boxplus \epsilon_b, \quad (21)$$

$$\pi_u^O = \mathbf{w}_U^T \boxplus \epsilon_o', \quad (22)$$

$$\pi_c^O = \mathbf{w}_C^T \boxplus \epsilon_o. \quad (23)$$

The final structure of the A-BCJR is reported in Fig. 2. We used red lines to highlight the anticausal recursion associated with the backward recursion results.

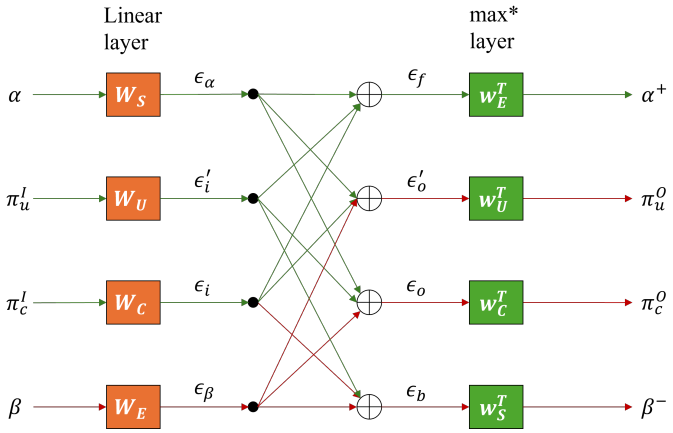

Fig. 2. The A-BCJR predictive step.

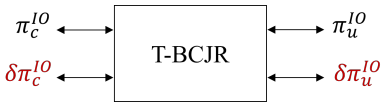

Fig. 3. The T-BCJR. Two additional IO ports for its training $\delta\pi_{u,c}^{IO}$ are available.

B. The training step of the T-BCJR with the δ backpropagation

After introducing the four matrices that define the A-BCJR structure, we now present the backpropagation recursions used for their training.

To enable integration into a trainable network, the T-BCJR (see Fig. 3) is endowed with two additional input-output ports that accept and provide the sequence of δ to be minimized. The input sequences δ^I may originate from external trainable blocks or be directly generated by defining an appropriate cost function on the delivered LLs. Similarly, the output δ^O can be used to train other external blocks. This will be further discussed in the next section, where we will insert the T-BCJR in a more complex structure to construct a trainable detector for ISI channels.

From the moment the system becomes trainable, all the defined matrices that constitute it can be trainable. This means that the initial weights assume random values and matrices lose their sparse nature. However, if the system is properly defined and adequately trained, the rows of these matrices tend to return to a one-hot representation.

The block diagram illustrating the recursions for δ backpropagation in T-BCJR is shown in Fig. 4. This diagram is derived from Fig. 2 by applying classical backpropagation rules commonly used in NN literature: the direction of arrows are reversed, addition nodes are replaced with repetition nodes, and vice versa.

The δ backpropagation through the max* operator can be obtained by a straightforward application of the chain rule.

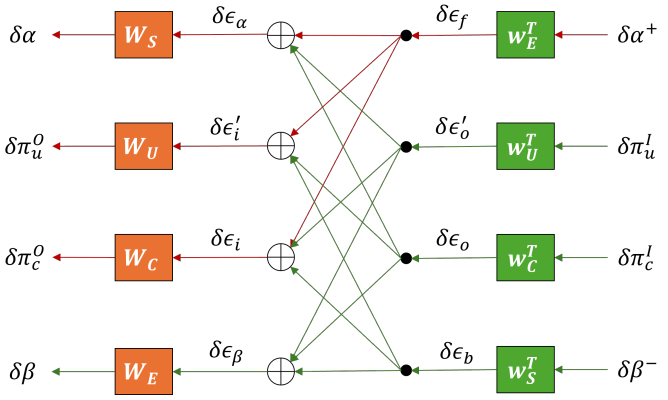

Fig. 4. δ backpropagation in the T-BCJR (training step).

After some algebra, one can obtain:

$$\delta\epsilon_f = e^{\mathbf{z}_E}\cdot\delta\alpha^+, \quad \delta\mathbf{w}_E(e,s) = e^{\mathbf{z}_E(e,s)}\delta\alpha^+(s), \tag{24}$$
$$\delta\epsilon_o' = e^{\mathbf{z}_U}\cdot\delta\pi_u^I, \quad \delta\mathbf{w}_U(e,u) = e^{\mathbf{z}_U(e,u)}\delta\pi_u^I(u), \tag{25}$$
$$\delta\epsilon_o = e^{\mathbf{z}_C}\cdot\delta\pi_c^I, \quad \delta\mathbf{w}_C(e,c) = e^{\mathbf{z}_C(e,c)}\delta\pi_c^I(c), \tag{26}$$
$$\delta\epsilon_b = e^{\mathbf{z}_S}\cdot\delta\beta^-, \quad \delta\mathbf{w}_S(e,s) = e^{\mathbf{z}_S(e,s)}\delta\beta^-(s), \tag{27}$$

where the exponents $\mathbf{z}(\cdot)$ of the exponential matrices are:

$$\mathbf{z}_E(e,s) = \mathbf{w}_E(e,s) + \epsilon_f(e) - \alpha^+(s), \tag{28}$$
$$\mathbf{z}_U(e,u) = \mathbf{w}_U(e,u) + \epsilon_o'(e) - \pi_u^I(u), \tag{29}$$
$$\mathbf{z}_C(e,c) = \mathbf{w}_C(e,c) + \epsilon_o(e) - \pi_c^I(c), \tag{30}$$
$$\mathbf{z}_S(e,s) = \mathbf{w}_S(e,s) + \epsilon_b(e) - \beta^-(s). \tag{31}$$

In backpropagation, sum nodes in the predictive step become repetition nodes and vice-versa, so that:

$$\delta\epsilon_\alpha = \delta\epsilon_f + \delta\epsilon_o + \delta\epsilon_o', \tag{32}$$
$$\delta\epsilon_i' = \delta\epsilon_f + \delta\epsilon_o + \delta\epsilon_b, \tag{33}$$
$$\delta\epsilon_i = \delta\epsilon_f + \delta\epsilon_o' + \delta\epsilon_b, \tag{34}$$
$$\delta\epsilon_\beta = \delta\epsilon_b + \delta\epsilon_o + \delta\epsilon_o'. \tag{35}$$

Finally, backpropagation through the linear layer follows the standard rule of using the transpose matrix:

$$\delta\alpha = \mathbf{W}_S^T\cdot\delta\epsilon_\alpha, \quad \delta\mathbf{W}_S(e,s) = \delta\epsilon_\alpha(e)\alpha(s), \tag{36}$$
$$\delta\pi_u^O = \mathbf{W}_U^T\cdot\delta\epsilon_i', \quad \delta\mathbf{W}_U(e,u) = \delta\epsilon_i'(e)\pi_u^I(u), \tag{37}$$
$$\delta\pi_c^O = \mathbf{W}_C^T\cdot\delta\epsilon_i, \quad \delta\mathbf{W}_C(e,c) = \delta\epsilon_i(e)\pi_c^I(c), \tag{38}$$
$$\delta\beta = \mathbf{W}_E^T\cdot\delta\epsilon_\beta, \quad \delta\mathbf{W}_E(e,s) = \delta\epsilon_\beta(e)\beta(s). \tag{39}$$

Moreover, since rows of $\mathbf{W}_x$ ($x\in\{S,U,C,E\}$) are related to those of $\mathbf{w}_x$ with Eq. 14, we can further merge the $\delta\mathbf{W}_x$ contributions into $\delta\mathbf{w}_x$. After some manipulations, one can find:

$$\delta'\mathbf{w}_x(e,s) = \mathbf{W}_x(e,s) \left(\delta\mathbf{W}_x(e,s) - \sum_{s'}\mathbf{W}_x(e,s')\delta\mathbf{W}_x(e,s')\right). \tag{40}$$

Notice from Fig. 4 that, during training, backpropagation generates recursions in the opposite time direction of the corresponding α and β recursions in the predictive step of

A-BCJR (Fig. 2). Consequently, the propagation of $\delta\alpha$ is now anticausal (red lines), whereas the propagation of $\delta\beta$ is causal. To efficiently address the anticausality problem, the detector's predictive and training phases can be performed with the sliding window with a grouped decision approach. Input streams are processed periodically in blocks of N trellis steps, resulting in a latency of $N+D$. The anticausal recursions, β in the predictive phase and $\delta\alpha$ in the training phase, are initialized with uniform metrics. To ensure reliable initialization, D initial backward steps are performed before the subsequent N backward steps, resulting in a total of $N + D$ backward steps.

C. Special cases and generalizations

The T-BCJR, with the predictive steps in Eqs. 20-23 and the associated training steps Eqs. 24-27 and Eqs. 36-39 described in the previous section, admits several important special cases and generalizations, listed in the following:

1) Some input LLs may not be available, and/or some output LLs may not be required. In these cases, the corresponding input ports are set to zero, and the associated output ports are omitted from evaluation.
2) Similarly, if certain inputs δ^I are unavailable, and/or some output δ^O are not required, the corresponding input ports are set to zero, and the associated output ports are not computed.
3) Some of the functions defining the trellis section may be fixed, i.e. not trainable. Consequently, the associated weight matrices have a fixed structure with one-hot rows. As an example, if an edge e is represented by the pair of starting state and input symbol $e = (s, u)$, then the mappings $u(e)$ and $s^S(e)$ reduce to fixed projections of e onto the input and state spaces, respectively.
   When a weight matrix is fixed with one-hot rows, its sparse nature considerably simplifies both the prediction and training. For instance, if $\mathbf{W}_S$ is fixed, the training step at the max* layer in Eq. 27 reduces to:
$$\delta\epsilon_b(e) = e^{\epsilon_b(e)-\beta^-(s^S(e))}\delta\beta^-(s^S(e)), \quad \delta\mathbf{w}_S = 0, \tag{41}$$
   while at the linear layer Eq. 36 it becomes:
$$\delta\alpha(s) = \sum_{e:s^S(e)=s} \delta\epsilon_\alpha(e), \qquad \delta\mathbf{W}_S = 0. \tag{42}$$
   Finally, even in the extreme case in which all weight matrices are fixed, the training step of the T-BCJR remains necessary to provide the required δ^O to the external blocks in an end-to-end trainable network.
4) Input and/or output LLs may be replaced by alternative representations. Consequently, the input and output matrices $\mathbf{W}_C$ and $\mathbf{W}_U$ may include a linear mapping corresponding to this change in representation. For example, if the input u consists of an n-tuple of bits, the LL defined over u, of size 2^n, can be replaced by n vectors of size 2, each representing the bit LL.

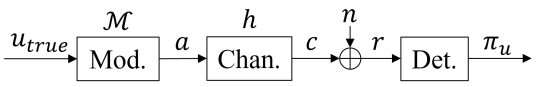

Fig. 5. Block diagram of the system model with an ISI channel.

D. T-BCJR vs BiLSTM

As an RNN, the T-BCJR is similar to the well-known Bidirectional Long Short-Term Memory (BiLSTM), widely adopted in NN literature and successfully used in many applications. Like all RNNs, both architectures rely on forward and backward recursions for prediction and training. Indeed, the forget, input, and output gates of BiLSTM are closely related to the matrices $\mathbf{W}_S$, $\mathbf{W}_U$, $\mathbf{W}_C$, and $\mathbf{W}_E$ of the T-BCJR. However, the T-BCJR is simpler and specifically tailored to process LLs, with the nonlinearity bounded to the $\max^*$ operator. Furthermore, it inherits from the A-BCJR the bidirectional nature of the δ ports. The computed δ^O are extrinsic, allowing the T-BCJR to be embedded within *iterative trainable* networks, a concept that deserves further investigation. Overall, the T-BCJR offers a natural and interpretable bridge between general-purpose RNNs and maximum a posteriori (MAP) sequence processing based on inference, as performed by the BCJR algorithm.

III. The T-Detector. A trainable detector for the ISI channel.

In this section we present the focus of our work: the use of the T-BCJR to implement a trainable T-Detector for ISI channels based on channel shortening approach [29][1].

Fig. 5 depicts the system model for ISI channels. We include a modulator characterized by the modulation alphabet $\mathcal{M}$. The ISI channel is represented by a tapped delay line (TDL) model, defined by the channel impulse response (CIR) vector h and additive white Gaussian noise n. The discrete-time baseband model of the received signal can be expressed as:
$$r_k = \sum_{l=0}^{L_C} h_l a_{k-l} + n_k. \tag{43}$$
Here, L_C denotes the memory length of the channel, a_k are the transmitted symbol at time k, h_l represents the complex channel coefficient of l^{th} path, and $n_k \sim \mathcal{CN}(0, \sigma^2)$ denotes the additive complex Gaussian noise. The resulting received sequence r is then processed by a detector, which generates soft information π_u on the system input u_{true}.

Fig. 6 illustrates a classical detector architecture based on channel shortening. This structure consists of a preliminary channel shortening filter of length L_F, followed by a BCJR trellis processor with memory L_T. The "branch metric (BM) computation" block computes the LL π_c^I metrics for the BCJR using the output z of the shortening filter as:
$$\pi^I_{c,k}(e) = \frac{1}{\sigma^2}\left[\Re(z_k a_k^*) - \frac{\gamma_0\|a_k\|^2}{2} - \Re\left(a_k^* \sum_{i=1}^{L_T} a_{k-i}\gamma_i\right)\right] \tag{44}$$

[1]In this reference, the author derives the structure assuming a MIMO channel model. The interpretation in a ISI setting is straightforward.

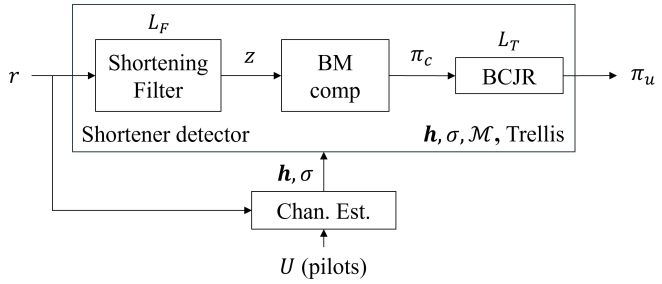

Fig. 6. The detector based on channel shortening. An external channel estimation block provides the required CSI.

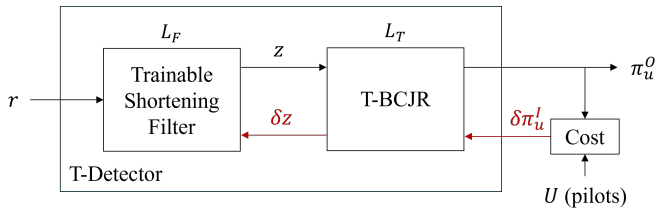

Fig. 7. The T-Detector.

where σ^2 is the noise variance, a_k is the hypothesis on current symbol at time k, $s_k = (a_{k-1}, ..., a_{k-L_T})$ is a vector of length L_T defining the state metrics associated with the trellis states at time index k. The coefficients $\gamma = (\gamma_0, \gamma_1, ..., \gamma_{L_T})$ depend on the shortening filter and are computed as described in [29]. By tuning the shortening filter memory L_F and the trellis memory L_T, this structure enables a trade-off between complexity and performance. The shortening filter reshapes the effective channel impulse response (CIR) to fit within the target memory, thereby simplifying subsequent trellis-based equalization while preserving the relevant information. When $L_T = 0$, meaning no trellis is used, the detector reduces to a MMSE filter followed by LL computation. Conversely, when L_T equals the channel memory, the detector becomes the optimal, but more complex, ML detector. In [29], the optimal shortening filter coefficients and the corresponding BCJR metrics are derived as functions of the channel impulse response $\boldsymbol{h}$, the noise variance σ^2, and the modulation alphabet $\mathcal{M}$. In practical scenarios, an additional channel estimation module is required to estimate $\boldsymbol{h}$ and σ^2, from which the optimal filter coefficients and metrics are computed. To eliminate the need for this explicit channel estimation block, we propose the trainable T-Detector, as illustrated in Fig. 7. We replace the traditional channel shortening filter with a trainable convolutional linear layer that mimics its function. The discrete sequence of IQ samples r feeds the adaptive shortening filter, which computes the sequence of observations z according to:

$$z_k = \mathbf{W}_{\text{sh}} \cdot r'_k, \tag{45}$$

where r'_k is the vector collecting the last L_F samples of the input sequence r, $r'_k \triangleq (r_k, r_{k-1}, \ldots, r_{k-L_F+1})^T$. $\mathbf{W}_{\text{sh}}$ is a trainable complex weight vector defining an additional convolutional linear layer in the T-Detector. The BCJR is replaced by the T-BCJR, which is discussed in detail in Section II. The "BM comp" block is now integrated into the T-BCJR, which takes as input the output z of the shortening filter (one of the special cases detailed under item 4 in Section II-C) and directly computes the edge metrics ϵ_i using the trainable matrix $\mathbf{W}_C$ with an additional bias $\mathbf{b}_C$. Indeed, one can observe that Eq. 44 can be represented as:

$$\epsilon_{i,k} = \mathbf{W}_C z_k + \mathbf{b}_C, \tag{46}$$

with suitable $\mathbf{W}_C$ and $\mathbf{b}_C$. For the sake of clarity, in NNs complex operations are mapped into linear operation with 2D vectors. As an example: $\Re(z_k a_k^*) = \Re(z_k) \cdot \Re(a_k) + \Im(z_k) \cdot \Im(a_k)$. The matrix $\mathbf{W}_C$ has then two columns and $|\mathcal{E}|$ rows.

The T-Detector is configured by providing the length of the shortening filter L_F, the modulation cardinality $|\mathcal{M}|$, the memory of the trellis processor L_T, and the type of delivered LLs π_u^O. The LL π_u^O may be the input to a soft-input decoder, such as a turbo decoder or an LDPC decoder. The LLs π_u^O can be defined directly on the constellation set (1 vector of size $|\mathcal{M}|$), or on the bits that label the constellation set (m vectors of size 2). This second option is more suitable in bit-interleaved coded modulation (BICM) schemes where the outer decoder processes bit LL.

A. Training the T-Detector

In the proposed T-Detector, the $\delta\pi_u^I$ required for training are obtained by applying a cost function placed directly after the detector. Specifically, we adopt the instantaneous conditional entropy of the output LLs π_u^O delivered by the detector as the cost function, assuming uniformly distributed inputs:

$$C(\pi_u^O) = \max_x{}^* \pi_u^O(x) - \pi_u^O(u_{true}), \tag{47}$$

where u_{true} is the "ground truth", i.e. the transmitted input bit or index of the constellation symbol. By taking the gradient of the cost function, we obtain the $\delta\pi_u^I$ as:

$$\delta\pi_u^I = \nabla C(\pi_u^O) = \text{AMAX}^*(\pi_u^O) - 1_{u_{true}}, \tag{48}$$

where $1_{u_{true}}$ denotes the one-hot column vector with a one in position u_{true} and $\text{AMAX}^*(x)$ is defined as $\left(\frac{e^{x(i)}}{\sum_j e^{x(j)}}\right)_{i=1}^N$. These gradients $\delta\pi_u^I$ serve as input to the T-BCJR, which computes the corresponding output δz as described in Section II-B. The resulting gradients are then backpropagated to update the coefficients of the adaptive shortening filter $\mathbf{W}_{\text{sh}}$ according to:

$$\delta\mathbf{W}_{\text{sh}} = \delta z_k \cdot r'_k. \tag{49}$$

The T-Detector results to be fully trainable from its output LLs. This fundamental feature is inherited from the T-BCJR.

B. Tailoring the T-BCJR for the T-Detector

In the context of a channel with ISI, we assume the BCJR "label-free" trellis is known. Specifically, we assume the channel follows the state evolution of a Finite Impulse Response (FIR) filter, where the next state is obtained by shifting the oldest hypothesis out of the register and inserting the current symbol a_k:

$$s_k = (a_{k-1}, \ldots, a_{k-L_T}) \rightarrow s_{k+1} = (a_k, \ldots, a_{k-L_T+1}).$$

TABLE I
$\mathbf{W}_S$, $\mathbf{W}_E$, $\mathbf{W}_U^S$, $\mathbf{W}_U^b$ MATRICES FOR $M = 4$ AND $L_T = 1$.

e	$\mathbf{W}_S$	$\mathbf{W}_E$	$\mathbf{W}_U^S$	$\mathbf{W}_U^b$
0	1000	1000	1000	1010
1	1000	0100	0100	1001
2	1000	0010	0010	0110
3	1000	0001	0001	0101
4	0100	1000	1000	1010
5	0100	0100	0100	1001
6	0100	0010	0010	0110
7	0100	0001	0001	0101
8	0010	1000	1000	1010
9	0010	0100	0100	1001
10	0010	0010	0010	0110
11	0010	0001	0001	0101
12	0001	1000	1000	1010
13	0001	0100	0100	1001
14	0001	0010	0010	0110
15	0001	0001	0001	0101

This means that matrices $\mathbf{W}_S$ and $\mathbf{W}_E$ are fixed with one-hot rows. The number of trellis states is defined as $|\mathcal{S}| = 2^{m \cdot L_T}$ where m is the modulation efficiency and L_T is the trellis memory. The number of input symbols $\mathcal{U}$ is the cardinality of the modulation set $|\mathcal{M}| = 2^m$, and, as a consequence, the number of trellis edges is $|\mathcal{E}| = 2^{m \cdot (L_T+1)}$. Moreover, the output symbol corresponds to the least significant symbol of the final state, meaning the rows of the output matrix $\mathbf{W}_U$ are also fixed. Since the output LLs can be either binary or non-binary, the structure of the matrix $\mathbf{W}_U$ varies accordingly. When the required outputs are symbol LLs, the rows of $\mathbf{W}_U$ have lengths equal to the constellation cardinality $|\mathcal{M}|$. The set of $|\mathcal{M}|$ rows of $\mathbf{W}_U$ corresponding to edges having the same starting state is initialized with the one-hot vectors where 1s are placed in all possible $|\mathcal{M}|$ positions. When the required outputs are binary LLs, the rows of $\mathbf{W}_U$ have length $2 \cdot m$. The set of $|\mathcal{M}|$ rows of $\mathbf{W}_U$ corresponding to edges having the same starting state is initialized with the vectors where 1s are placed in odd or even positions according to all binary possibilities.

In Table I, we present, as an example, the values of the fixed matrices $\mathbf{W}_S$, $\mathbf{W}_E$, $\mathbf{W}_U^S$ (when outputs are symbols LLs) and $\mathbf{W}_U^b$ (when outputs are binary LLs) when $|\mathcal{M}| = 4$ and $L_T = 1$ (4 states and 16 edges). The trainable matrices of the T-Detector are $\mathbf{W}_{\text{sh}}$, and $\mathbf{W}_C$ along with their biases $\mathbf{b}_C$. The knowledge of the constellation set and binary labeling of the constellation points is not required, making the resulting detector agnostic to both the channel and modulation set.

For clarity, Table II summarizes all the matrices defining the proposed system, including their dimensions and physical interpretations. Moreover, in the context of ISI channels where a "label-free" trellis is assumed to be known, the table highlights which parameters are fixed and which are trainable.

C. T-Detector vs ASG-MMSE

The proposed T-Detector can be interpreted as a generalization of the classical adaptive stochastic gradient (ASG)-MMSE, differing in two fundamental aspects.

First, the training procedure jointly optimizes both the linear filtering and the T-BCJR module. This is achieved by leveraging the NN's capability to efficiently propagate gradients through nonlinearities. As a result, gradient-based learning can be applied to nonlinear devices and various cost functions, as long as the nonlinearities are differentiable.

TABLE II
SUMMARY TABLE FOR SYSTEM PARAMETERS.

	Dimension	Physical meaning	BCJR "label-free" trellis
$\mathbf{W}_S$	$\lvert\mathcal{E}\rvert \times \lvert\mathcal{S}\rvert$	Associates each edge with its starting state state	Fixed, one-hot (trellis structure)
$\mathbf{W}_E$	$\lvert\mathcal{E}\rvert \times \lvert\mathcal{S}\rvert$	Associates each edge with its ending state state	Fixed, one-hot (trellis structure)
$\mathbf{W}_U^S$	$\lvert\mathcal{E}\rvert \times \lvert\mathcal{M}\rvert$	Associates each edge with its corresponding symbol label (symbol-level mapping)	Fixed, structure for symbol LL
$\mathbf{W}_U^b$	$\lvert\mathcal{E}\rvert \times 2 \cdot m$	Associates each edge with its corresponding bit-level representation (bit-level mapping)	Fixed, structure for binary LL
$\mathbf{W}_C$	$\lvert\mathcal{E}\rvert \times 2$	Computes edge metrics from channel observations	Trainable, learn the branch metric
$\mathbf{b}_C$	$\lvert\mathcal{E}\rvert \times 1$	Bias term for edge metric computation	Trainable, learn the branch metric
$\mathbf{W}_{\text{sh}}$	$2 \times L_F$	Linear channel-shortening filter applied to the received signal	Trainable, learn filter coefficients

Second, rather than minimizing the mean squared error (MSE) of the transmitted constellation symbols, the T-Detector is trained by minimizing the conditional entropy of the output LLs. This choice makes the T-Detector independent of both the modulation constellation and the associated binary labeling, while also improving robustness to nonlinear impairments, such as those introduced by power amplifiers or A/D converters. Furthermore, the SNR estimation required for accurate LL computation in ASG-MMSE is implicitly learned through the optimization of the weights $\mathbf{W}_C$ and biases $\mathbf{b}_C$. From an information-theoretic perspective, we argue that conditional entropy constitutes a more appropriate objective for detection than the MSE criterion. Indeed, its first- and second-order statistics are known to provide bounds on the frame error rate (FER) performance of the outer code. Consequently, minimizing conditional entropy directly aligns the detector optimization with improved end-to-end FER performance, thereby reinforcing the efficacy of the proposed approach.

IV. RESULTS STATIC ISI CHANNEL

In this section, we provide simulation results that show how the T-Detector can learn to perform as an optimal shortening detector [29] in statistical channels affected by static ISI. In static channel conditions, only an initial acquisition phase is required to train our T-Detector. In this phase, only pilot signals are transmitted to feed the backpropagation process and to enable the system to adapt to the channel conditions. Since the channel is assumed to remain constant, once the convergence to its maximum performance is reached, no further retraining is necessary and the training overhead becomes zero.

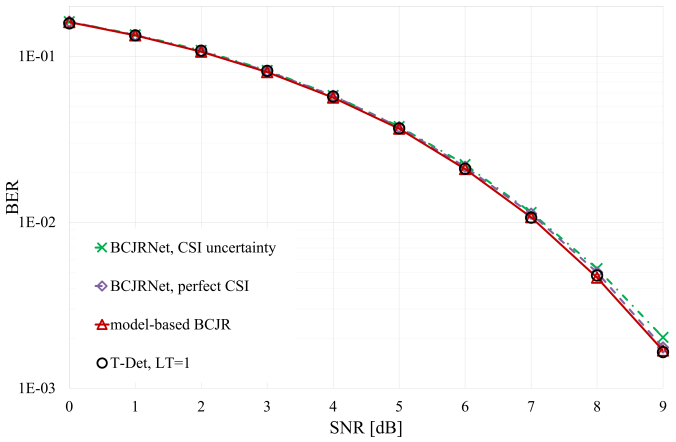

Fig. 8. BER vs SNR for the two-tap ISI channel. Comparison among the traditional model-based BCJR, the T-Detector with $L_T = 1$ and the BCJRNet [18].

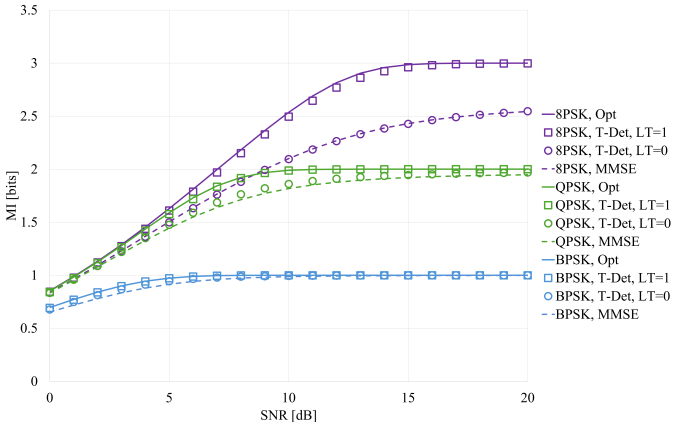

Fig. 9. MI vs SNR for the two-taps ISI channel with PDP [0, 0] dB. Comparison of MMSE, optimal ML detector, and T-Detector with $L_T = 0$ and $L_T = 1$. Results are averaged over 100 snapshots.

A. Comparison with the BCJRNet

We begin by presenting a comparison among the traditional model-based BCJR, our T-Detector and the BCJRNet model proposed in [18], considered as the state-of-the-art in terms of model-based NN. The performance of BCJRNet is achieved by learning the underlying factor graph of a finite-memory channel[2].

For our numerical evaluation, we adopt the same scenario proposed by the authors in [18]. The channel impulse response has an exponentially decaying profile as $h_l^{(\gamma)} = e^{-\gamma \cdot l}$ for $\gamma > 0$ and $l = 0, 1$. We consider i.i.d. binary phase shift keying inputs, i.e., $a \in \{-1, 1\}$. Both BCJRNet and our T-Detector are trained for each SNR value in the range [0, 9] dB. The BER results are averaged over 20 different channel impulse responses, with γ varying uniformly in the range [0.1, 2]. The shortening filter length L_F has been set to one as $L_C = L_T = 1$. The numerically computed BER values are depicted in Fig. 8. Performance evaluations indicate that the T-Detector exactly replicates the performance of the optimal detector (red solid curve with triangle marks) when $L_T = 1$ (black circle marks). Our T-Detector matches also the performance of the BCJRNet, both with perfect CSI (purple dotted line with diamond marks) and CSI uncertainty (green line with cross marks). These results validate the effectiveness of our method. Notably, our detector operates without requiring CSI, as it is both channel and modulation agnostic. It equals the DSP complexity of the corresponding classical receiver and is trainable using the cost function defined on the delivered LLs.

For this comparison, performance is evaluated in terms of BER, a standard metric in the literature, which enables direct comparison with state-of-the-art detectors. However, while BER is commonly used to evaluate the performance of channel decoders, it may not fully capture the effectiveness of detectors that provide messages in form of LLs to subsequent soft-input capacity-achieving decoders. In these cases, average MI, normalized to the modulation efficiency, serves as a more precise indicator of the maximum code rate to ensure reliable transmission. To support this argument, performances of a complete system with an outer decoder is presented at the end of this section (Fig. 14). Therefore, in the following results, performance is evaluated in terms of MI, measured on the delivered symbol or bit LLs. The MI is calculated from the conditional entropy defined in Eq. 47 as:

$$\text{MI} = m - \frac{1}{\ln(2)} \cdot \mathbb{E}\{C(\pi_u^O)\} \qquad [\text{bits}]. \qquad (50)$$

B. Two-taps channel model

To further test our detector, we consider two additional statistical channels. The first one is a two-taps ISI channel. The channel's power delay profile (PDP) is constant, with both taps having the same average power of [0, 0] dB. It means that the channel is characterized by a short but strong ISI and its memory is $L_C = 1$. We assume a unitary energy for each channel snapshot.

In Fig. 9 we report the average MI measured on the LL at the output of the T-Detector over 100 channel realizations across an SNR range of [0, 20] dB, for BPSK, QPSK, and 8PSK constellations (depicted in blue, green, and violet, respectively), evaluated using four different detectors. Solid lines show the performance of the optimal ML detector based on BCJR with perfect CSI knowledge. The input filter is matched to the channel impulse response, and the memory of trellis processing coincides with the channel memory $L_T = L_C = 1$. Square and circle (discrete) markers represent the performance of the T-Detector with $L_F = 4$, and $L_T = 0$ (circles) or $L_T = 1$ (squares). The parameters of the T-Detector are set equal to the simulations presented in Section IV-A. As before, $\mathbf{W}_{\text{sh}}$, $\mathbf{W}_C$ and $\mathbf{b}_C$ are the trainable matrices. The T-Detector is unaware of the channel or constellation set but knows the constellation cardinality $|\mathcal{M}|$. The dashed lines report the performances of the MMSE-based detector. The output LLs of the MMSE-based detector are obtained by computing the distance of the MMSE filter output z_k from all constellation points $m(u)$, assumed to be known at the

[2]https://github.com/nirshlezinger1/LearnedFactorGraphs.

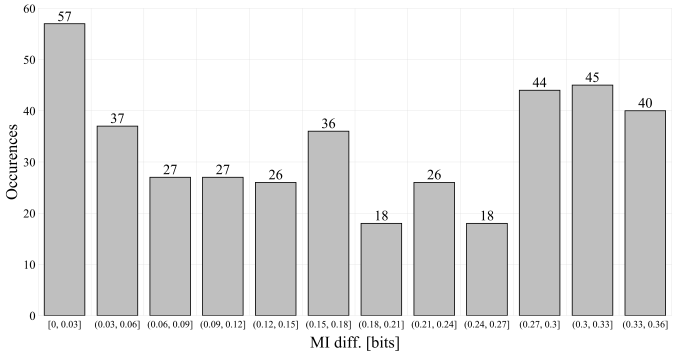

Fig. 10. Histogram of MI differences between T-Detector with $L_T = 1$ and MMSE. Histogram based on 400 snapshots from two-tap ISI channel model, with both taps having equal average power of [0, 0] dB. The SNR is fixed to 7 dB. Normalized channel energy for each snapshot.

receiver. The distance is then normalized with the estimated residual MSE of the noise plus the ISI term σ_z^2:

$$\pi_k(u) = -\frac{1}{2\sigma_z^2}|z_k - m(u)|^2, \quad u \in [0, ..., M-1]. \qquad (51)$$

Performances show that the T-Detector exactly replicates the performance of the MMSE detector when $L_T = 0$ and that of the optimal MLSE detector when $L_T = 1$ for all the modulations.

To further elaborate on the analysis, Fig. 10 presents a histogram of the MI differences between our T-Detector at $\mathrm{SNR} = 7$ dB with $L_T = 1$ and the MMSE detector with QPSK modulation for a set of 400 channels randomly obtained from the two-taps ISI channel model. The plot exhibits a nearly uniform distribution in the range [0 - 0.36] bits. Given the two-tap ISI channel model with a constant PDP, the power in each tap can vary significantly between different snapshots, which explains this observed behaviour. In one extreme, the power is equally distributed across the two taps, while in the other, it is concentrated in one tap, making the power in the other tap negligible. In the former scenario, the difference is larger, reflecting the advantage in using ML detection. In the latter case, the difference is minimal, as the MMSE achieves a nearly ideal performance.

C. TDL-A model

We finally test our detector also in the TDL-A channel specified in the 5GNR standard. This channel is characterized by many interferers spread in time, in contrast to the previous one. Delays and power levels are defined according to [32]. We consider a delay spread of 1 μs, yielding the following PDP for symbol rate of 1 Mbaud:

$$\begin{aligned} k = 0, &\qquad \mathrm{PD} = -5.6\,\mathrm{dB}, \\ k = 1, &\qquad \mathrm{PD} = -8.8\,\mathrm{dB}, \\ k = 2, &\qquad \mathrm{PD} = -12.9\,\mathrm{dB}, \\ k = 3, &\qquad \mathrm{PD} = -16.4\,\mathrm{dB}, \\ k = 4, &\qquad \mathrm{PD} = -19.5\,\mathrm{dB}, \\ k = 5, &\qquad \mathrm{PD} = -20.6\,\mathrm{dB}, \\ k = 10, &\qquad \mathrm{PD} = -38.1\,\mathrm{dB}. \end{aligned}$$

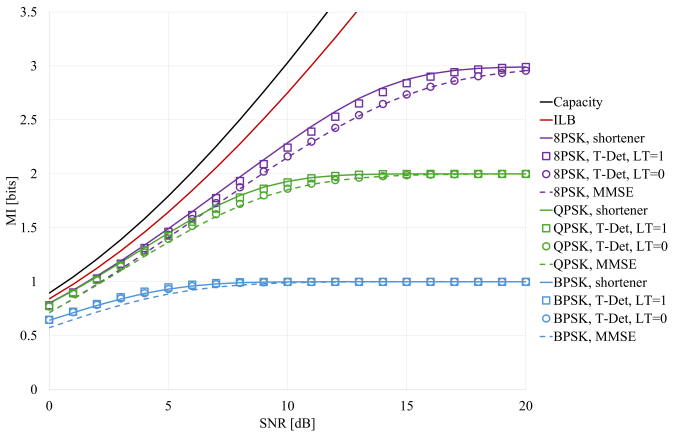

Fig. 11. MI vs SNR for the static TDL-A channel. Comparison among MMSE, shortener detector [29] and T-Detector with $L_T = 0$ and $L_T = 1$. As reference the capacity and the I_{LB} are reported. Results are averaged over 100 snapshots, with TDL-A tap powers defined according to [32].

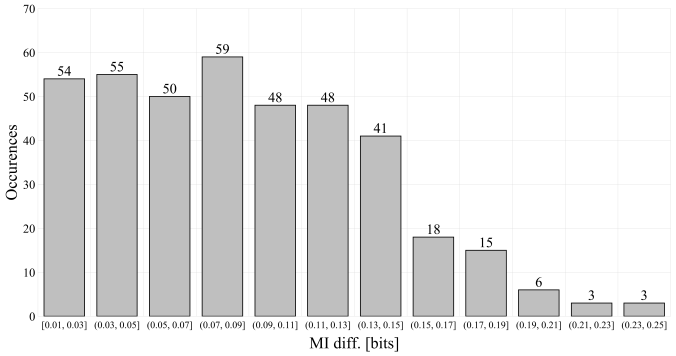

Fig. 12. Histogram of MI differences between T-Detector with $L_T = 1$ and MMSE. Histogram based on 400 snapshots from TDL-A channel model. The SNR is fixed to 7 dB. Power profile is kept fixed for each snapshot.

Also for this channel, we report the average MI over 100 channel realizations. The corresponding performance results are presented in Fig. 11. In this case, the optimal BCJR detector is not considered due to its prohibitive computational complexity, as the channel has a larger memory of $L_C = 10$. Instead, the performance of the shortener detector proposed in [29], described in Section III and depicted in Fig. 6, is reported as the optimum reference in solid lines, with the same parameter choices of the T-Detector, the shortening filter length L_F and the trellis memory L_T. The shortening filter length is increased to $L_F = 50$ to better match the channel characteristics. When $L_T = 1$, the T-Detector matches the performance of the optimum model-based correspondent and the performance gap with the MMSE detector is similar to that observed in Fig. 9. As references, we also inserted the Shannon capacity results (black curve) and the achievable information rate (lower bound, I_{LB}) of the reduced complexity detector, optimized in [29] for Gaussian inputs (red curve). In Fig. 12, we present a histogram of MI differences between T-Detector with $L_T = 1$ and MMSE detector, evaluated over 400 TDL-A channel snapshots at an SNR of 7 dB. The histogram exhibits a more concentrated distribution, with a mean MI gain of 0.08 bits and improvements reaching up to 0.25 bits.

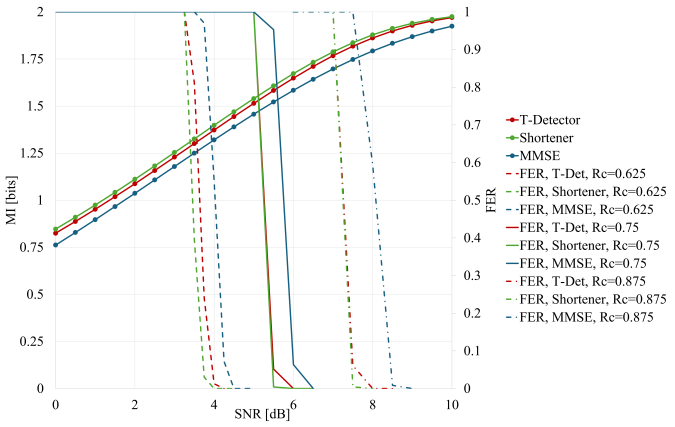

Fig. 13. MI (solid-marked lines) and FER performance of rate $R_c \in \{0.625, 0.75, 0.875\}$ 5GNR LDPC decoder with MMSE, T-Detector ($L_T = 1$) and shortener detector [29], for QPSK modulation. MI measured at the output of three detectors. Layered optimal BP decoder, 25 iterations.

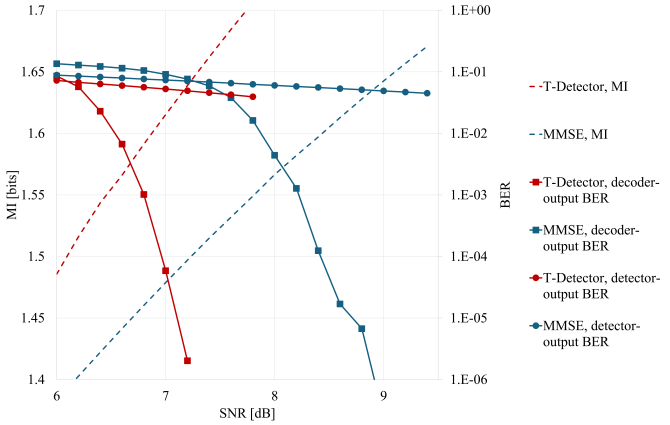

Fig. 14. BER performance of rate $R_c = 3/4$ $(8000, 6000)$ 5GNR LDPC decoder with MMSE and T-Detector ($L_T = 1$) and QPSK. The dashed lines report the MI measured at the output of two detectors. Layered optimal BP decoder, 25 iterations.

D. MI validation as performance metric for detector evaluation

To validate the relevance of MI as performance metric for detector evaluation, we propose two representative results. The first result is shown in Fig. 13, which reports the performance obtained from a single snapshot using three detectors: MMSE (blue), T-Detector with $L_T = 1$ (red) and shortener detector (green). The MI measured at the detector output is plotted on the primary axis, while the FER, evaluated after decoding, is reported on the secondary axis. The system employs QPSK modulation over the TDL-A channel model and 5GNR LDPC codecs. The purpose of this figure is to highlight the consistency between the decoding performance and the predictions based on the MI metric. Three code rates are considered. On the left, for $R_c = 0.625$ (8000, 5000), when the MI reaches the transmitter SE ($\mathrm{SE_{TX}}$) of $\mathrm{SE_{TX}} = m \cdot R_c = 2 \cdot 0.625 = 1.25$, the dashed FER curves drop to zero. Since the MMSE detector achieves lower MI values than the other two detectors, the corresponding FER curve exhibits its waterfall region at a higher SNR. In the central part of the figure, for $R_c = 0.75$ (8000, 6000), the drop in the solid FER curves occurs when $\mathrm{SE_{TX}} = 1.5$. Finally, on the right, for $R_c = 0.875$ (8000, 7000), the dot-dashed FER curves show a sharp decrease when $\mathrm{SE_{TX}} = 1.75$. Overall, the results confirm that the MI at the detector output accurately predicts the decoding threshold for the different coding rates and detection schemes.

The second experiment, reported in Fig. 14, focuses on the case $R_c = 3/4$ (8000, 6000), considering the MMSE detector (blue curves) and the proposed T-Detector ($L_T = 1$, red curves). Here, the BER at both the detector output (circle markers) and the decoder output (square markers) is shown on the secondary axis, while the dashed curves represent the MI measured at the detector output (primary axis). The results further support the predictive value of the MI metric. The SNR at which the BER after decoding begins to decrease corresponds closely to the point where the measured MI exceeds $\mathrm{SE_{TX}} = 1.5$. Conversely, assessing BER metric at the detector output is less relevant in applications where the detector is followed by an outer decoder. For instance, at the SNR of 7 dB, the MMSE detector and the T-Detector exhibit BERs of 7.37×10^{-2} and 5.26×10^{-2} (circle markers), respectively. Although this difference appears moderate at the detector stage, it translates into an SNR gain of approximately 1.5 dB at the output of the LDPC decoder.

E. Training/Acquisition time

In all simulations detailed above, the T-Detector employs a sliding window with grouped decision scheduling. The initialization window is set to $D = 10$, with $N = 40$. Coefficient updates are executed with a periodicity of 40. Following an optimization phase, the updating step (learning rate) for the backward propagation was established at 0.01. For all evaluations, the trainable weight matrices ($\mathbf{W}_{\mathrm{sh}}$, $\mathbf{W}_C$, $\mathbf{b}_C$) are initialized using independent zero-mean Gaussian variables with a standard deviation of 0.2, while the structural matrices $\mathbf{W}_S$, $\mathbf{W}_E$, $\mathbf{w}_U$ are defined according to the label-free trellis structure. To better reflect practical operational constraints, the trained T-Detector coefficients at a specific SNR serve as the initialization for the subsequent SNR level. This approach ensures the T-Detector does not restart the learning process at each SNR point. In practice, this mirrors a scenario where an initial "cold-start" training phase (which can be conducted offline) is followed by rapid fine-tuning via a limited number of pilot symbols, significantly reducing the effective training overhead.

Tables III and IV quantify the time required to train the T-Detector from scratch for the two static channel models previously introduced. This analysis spans various configurations of shortening filter lengths L_F and SNR values. Here, training (or acquisition) time is defined as the total number of symbols required for the output LLs to reach a MI equivalent to a predefined fraction of the maximum achievable value. These tables specifically characterize the convergence behavior during initial acquisition under a fixed channel realization, representing a worst-case initialization scenario. Table III details the results for the two-tap ISI channel model, while

TABLE III
ACQUISITION TIME FOR 90 AND 95% OF THE MAXIMUM MI. QPSK, TWO-TAPS ISI CHANNEL.

	SNR = 0 dB			SNR = 5 dB			SNR = 10 dB		
	90%	**95%**	**MI**	**90%**	**95%**	**MI**	**90%**	**95%**	**MI**
$L_T = 0$									
$L_F = 2$	3080	3080	0.71	1600	2840	1.11	1720	2880	1.32
$L_F = 5$	3080	64320	0.78	1600	6880	1.31	1960	3000	1.69
$L_F = 10$	3000	64400	0.77	2520	7360	1.33	2440	3800	1.77
$L_T = 1$									
$L_F = 2$	9160	64240	0.78	13400	49600	1.46	12520	22720	1.97
$L_F = 5$	7760	64280	0.78	11880	46880	1.48	10520	28840	1.97
$L_F = 10$	7000	12200	0.74	13360	16840	1.46	7480	14800	1.98

TABLE IV
ACQUISITION TIME FOR 90 AND 95% OF THE MAXIMUM MI. QPSK, TDL-A CHANNEL.

	SNR = 0 dB			SNR = 5 dB			SNR = 10 dB		
	90%	**95%**	**MI**	**90%**	**95%**	**MI**	**90%**	**95%**	**MI**
$L_T = 0$									
$L_F = 2$	2400	2680	0.59	2520	3280	0.96	2720	3640	1.16
$L_F = 5$	2680	3000	0.69	2920	3360	1.17	3920	4680	1.51
$L_F = 10$	3480	4120	0.72	3920	4920	1.28	4920	6840	1.71
$L_T = 1$									
$L_F = 2$	9080	9080	0.66	11160	12880	1.13	16160	26880	1.51
$L_F = 5$	4120	7000	0.69	8920	8920	1.25	16760	29880	1.73
$L_F = 10$	6720	7360	0.74	10960	13680	1.36	16720	23000	1.90

Table IV provides the results for the TDL-A channel model, both evaluated using QPSK modulation.

As expected, training is faster for $L_T = 0$. Increasing the shortening filter length L_F generally leads to longer training times and improved performance. An exception is observed for $L_T = 1$ in the two-tap ISI channel case, where $L_F = 2$ corresponds to the optimal ML detector. Regarding the SNR, no clear trend can be observed. Further analysis is required to better understand this behavior.

V. RESULTS SLOWLY-TIME VARYING ISI CHANNELS

Throughout the first sections of this work, we have considered a static scenario, where only an initial acquisition phase is required to train the T-Detector. In time-varying scenarios, training can occur continuously, with an initial acquisition phase followed by a tracking phase in which pilot symbols are periodically interspersed with payload data, allowing the T-Detector's weights to slowly adapt in response to channel variations rather than being re-trained from scratch for each burst.

A. Results for the slowly time-varying two-taps channel model

Here we present some results of our T-Detector in the presence of a slow time-varying channel as evaluated also in [27]. The two taps coefficients defined in Section IV-B are substituted by two equal power Gaussian processes with Jake's spectrum [33] and a given Doppler frequency. We analyze the performance of the T-Detector considering different pilot densities, pilot fields and Doppler frequencies. The pilot density (overhead) is defined as the number of pilots sent with respect to the entire transmission that includes both pilots and data while the pilot field as the length of each pilot transmission. Given the initialization parameters, we choose the pilot field to be equal to 20 symbols to keep the periodicity small. For each experiment we simulate time series of 10 millions samples. Performances for the channel with strong ISI are shown in Fig. 15.

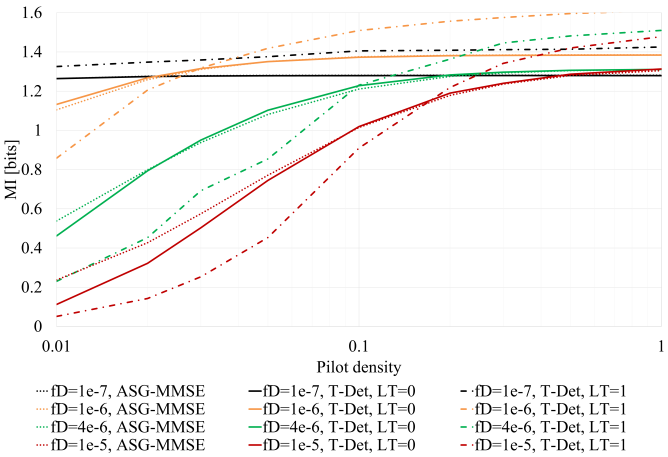

Fig. 15. Mutual Information vs pilot density. Strong ISI channel, SNR = 7 dB, $f_D \in \{10^{-7}, 10^{-6}, 4 \times 10^{-6}, 10^{-5}\}$. Comparison of ASG-MMSE and T-Detector with $L_T = 0, 1$.

Fig. 15 shows the MI of the delivered LLs as a function of the pilot density (logarithmic scale) for QPSK at SNR = 7 dB. This SNR is selected based on the static-channel results in Fig. 9, where the performance gap between the ASG-MMSE receiver and the T-Detector with $L_T = 1$ is most pronounced. Four normalized Doppler frequencies, $f_D \in \{10^{-7}, 10^{-6}, 4 \times 10^{-6}, 10^{-5}\}$, and three detector configurations are considered. The dotted curves correspond to the ASG-MMSE receiver, while the solid and dash-dotted curves represent the T-Detector with $L_T = 0$ and $L_T = 1$, respectively. For very slow channel variations ($f_D = 10^{-7}$), the T-Detector with $L_T = 0$ achieves the same MI as the ASG-MMSE receiver, whereas increasing the trellis memory to $L_T = 1$ yields higher MI across all pilot densities. As the normalized Doppler frequency increases, e.g., $f_D = 10^{-5}$, the ASG-MMSE receiver outperforms the T-Detector at low pilot densities, since the latter requires sufficient pilot support to adapt its coefficients, particularly for $L_T > 0$. With increasing pilot overhead, the T-Detector with $L_T = 0$ approaches ASG-MMSE performance, while the $L_T = 1$ configuration eventually surpasses it. Although absolute MI values differ across realizations due to finite-length simulations, the overall trend indicates that higher Doppler frequencies require a larger pilot density for the T-Detector with $L_T = 1$ to outperform the ASG-MMSE receiver.

VI. COMPLEXITY

To provide a comprehensive assessment, we perform a complexity analysis expressed both in Big-O notation, $\mathcal{O}(\cdot)$, and in terms of FLOPs. The analysis considers the normalized computational complexity per transmitted symbol.

The parameters used for the complexity evaluation, which have already been introduced throughout the paper, are summarized below for completeness, together with the FLOP cost associated with each basic operation:

TABLE V
TRAINABLE SHORTENING FILTER COMPLEXITY

	FLOP	$\mathcal{O}(\cdot)$
Prediction	$4 \cdot L_F \cdot (1 + c_m)$	$\mathcal{O}(L_F)$
Training	$4 \cdot L_F \cdot c_m$	$\mathcal{O}(L_F)$
Update	$4 \cdot L_F \cdot (1 + 2 \cdot c_m)$	$\mathcal{O}(L_F)$

- m: modulation order (bits per symbol);
- L_T: trellis memory;
- $S = |\mathcal{S}| = 2^{m \cdot L_T}$: number of trellis states;
- $E = |\mathcal{E}| = 2^{m \cdot (L_T+1)}$: number of trellis edges;
- $C = |\mathcal{C}|$: cardinality of output symbol alphabet;
- $U = |\mathcal{U}|$: size of the LL vector for input symbols, i.e. 2^m for symbol-level and $2 \cdot m$ for bit level LLs;
- L_F: shortening filter memory;
- sum and max operations: 1 FLOP each;
- max* operation: $(1 + c_L)$ FLOPs, where c_L denotes the lookup-table (LUT) access cost;
- multiplications: c_m FLOPs;
- exponential and logarithmic operations (nonlinear): c_{nl} FLOPs.

We recall that the training procedure is activated exclusively when pilot symbols are available. When pilots are absent, the training phase is not executed and no training-related computations are performed.

Table V reports the computational complexity of the trainable shortening filter. It can be observed that all processing stages (prediction, training, and update) depend on the filter memory length L_F.

For clarity, the complexity analysis of the T-BCJR algorithm is divided into multiple tables. We first analyze the prediction (Table VI) and training (Table VII) steps for the general system described in Section II, where all matrices are full and no simplifications are applied. Table VI includes the branch metric computation (BMC), the forward (F) and backward (B) recursions, and the output computational unit (OCU), which together account for all trellis computations. These steps coincide with those of the conventional model-based BCJR algorithm. Table VII reports the training complexity. In addition to the operations performed during prediction, an extra component is considered: the computation of gradients for updating the trainable matrices $\mathbf{W}_x$, with $x \in S, U, C, E$, and the corresponding parameter update operations. While the training phase involves additional nonlinear operations in terms of FLOPs compared to prediction, the Big-$\mathcal{O}$ analysis shows the same asymptotic complexity. As a result, when pilot symbols are present and training is performed, the complexity of the T-BCJR is approximately twice that of the corresponding model-based detector. In contrast, when no training is carried out, the computational complexity is identical to that of the model-based approach.

In the context of channels affected by ISI, where the proposed T-Detector is applied, we assume that the BCJR label-free trellis is known. As a consequence, the simplifications described in Section III-B can be applied. In this setting, the matrices $\mathbf{W}_S$, $\mathbf{W}_E$, and $\mathbf{W}_U$ become sparse and fixed. Tables VIII and IX present the complexity analysis of the T-BCJR with a label-free trellis. As expected, the overall number of operations is significantly reduced, and the resulting asymptotic complexity depends only on the number of trellis edges E.

As a final result, Table X reports the computational complexity of the cost function as a function of the output LL provided by the T-BCJR, namely binary (bit-level) or non-binary (symbol-level). The gradients necessary for training are computed only when pilots are present.

TABLE VI
T-BCJR PREDICTION COMPLEXITY

	FLOP	$\mathcal{O}(\cdot)$
BMC	$E \cdot (C + U) \cdot (1 + c_m)$	$\mathcal{O}(E \cdot (C + U))$
F/B	$E \cdot (1 + S \cdot (2 + c_m + c_L))$	$\mathcal{O}(E \cdot S)$
OCU (U)	$E \cdot (2 + U \cdot (1 + c_L))$	$\mathcal{O}(E \cdot U)$
OCU (C)	$E \cdot (2 + C \cdot (1 + c_L))$	$\mathcal{O}(E \cdot C)$

TABLE VII
T-BCJR TRAINING COMPLEXITY

	FLOP	$\mathcal{O}(\cdot)$
BMC	$E \cdot (C + U) \cdot (3 + c_m + c_{nl})$	$\mathcal{O}(E \cdot (C + U))$
F/B	$E \cdot (1 + S \cdot (5 + 2 \cdot c_m + c_{nl}))$	$\mathcal{O}(E \cdot S)$
OCU (U)	$E \cdot (2 + U \cdot (1 + c_m))$	$\mathcal{O}(E \cdot U)$
OCU (C)	$E \cdot (2 + C \cdot (1 + c_m))$	$\mathcal{O}(E \cdot C)$
Matrices updating	$E \cdot (2 \cdot S + C + U) \cdot$ $\cdot (4 + 5 \cdot c_m + c_{nl} + (2 \cdot S + C + U) \cdot (1 + c_m))$	$\mathcal{O}(E \cdot (S + C + U))$

VII. CONCLUSIONS

We have derived a fully trainable version of the additive SISO A-BCJR algorithm, named T-BCJR. It consists of a linear layer for constructing the edge metrics from the input and state metric, and a nonlinear max* layer to marginalize them back to the state and output space. The T-BCJR is described with four matrices and possibly their biases, which capture the trellis structure, branch metrics construction, and output LL computation. We derived the backward delta propagation equations to train the T-BCJR. Unlike previous approaches, the

TABLE VIII
T-BCJR "LABEL FREE" TRELLIS PREDICTION COMPLEXITY

	FLOP	$\mathcal{O}(\cdot)$
BMC	$E \cdot (3 + 2 \cdot c_m)$	$\mathcal{O}(E)$
F/B	$E \cdot (2 + c_L)$	$\mathcal{O}(E)$
OCU	$E \cdot (3 + c_L)$ if symbol-level	$\mathcal{O}(E)$
	$E \cdot (2 + m \cdot (1 + c_L))$ if bit-level	$\mathcal{O}(E \cdot m)$

TABLE IX
T-BCJR "LABEL FREE" TRELLIS TRAINING COMPLEXITY

	FLOP	$\mathcal{O}(\cdot)$
BMC	$E \cdot (2 + c_{nl})$ if symbol-level	$\mathcal{O}(E)$
	$E \cdot m \cdot (3 + c_{nl})$ if bit-level	$\mathcal{O}(E \cdot m)$
F/B	$E \cdot (2 + c_m + c_{nl})$	$\mathcal{O}(E)$
OCU	$E \cdot (4 + 2 \cdot c_m)$	$\mathcal{O}(E)$
Matrices updating	$E \cdot (3 + 8 \cdot c_m)$	$\mathcal{O}(E)$

TABLE X
T-BCJR COST FUNCTION COMPLEXITY

	FLOP	$\mathcal{O}(\cdot)$
COST, bit-level	$m \cdot (2 \cdot (1 + c_L) + c_m + c_{nl} + 1)$	$\mathcal{O}(m)$
δ, bit-level	$m \cdot (2 \cdot (c_m + c_{nl}) + 1$	$\mathcal{O}(m)$
COST, symbol-level	$2^m \cdot (1 + c_L) + (1 + c_m + c_{nl})$	$\mathcal{O}(2^m)$
δ, symbol-level	$2^m \cdot (1 + c_m + c_{nl})$	$\mathcal{O}(2^m)$

T-BCJR can totally replace the additive BCJR with the added feature of being trainable from its output and, consequently, with the flexibility required to face unknown channels or to recover from possible mismatches. The T-BCJR can be fully or only partially trained if the label-free trellis structure is known. By adding an input convolutional linear layer to the T-BCJR, we then introduced a trainable T-Detector with a structure inherited from the channel-shortening approach in order to deal with ISI channels. The T-Detector is trainable directly from its output LLs, which can be defined either on symbols or on bits. We showed in different scenarios that the T-Detector can rapidly learn to perform as a classical adaptive MMSE linear filter but also as an adaptive nonlinear detector based on the channel shortening approach. On top of this flexibility, being modulation agnostic, the detector is also robust to constellation distortion and other possible nonlinear impairments introduced by the channel.

The T-Detector can be easily inserted in a turbo equalization scheme where ground truth for the cost function evaluation (Eq. 47) is provided by an external decoder instead of pilots [28].

Although we presented the detector's performance in a classical scenario with single carrier systems affected by ISI, its employment can be easily adapted in all scenarios where channel shortening is effective. For example, systems where digital interference from multiple resources is present in the observation model, or equivalently, where the data symbol is affecting multiple observations. These scenarios include MIMO and multiuser systems and systems using orthogonal or non-orthogonal allocation strategies like OFDM, faster-than-Nyquist, or Orthogonal Time Frequency Space (OTFS) systems.

The time adaptability of the T-Detector shares the same pros and cons as the ASG-MMSE. The adaptation of its processing to the varying conditions of the channel follows the dynamic of the stochastic gradient algorithm, which relies on the periodic insertion of pilots to feed the backpropagation. Then it requires a careful optimization of the updating step in backpropagation with a necessary trade-off between tracking capability and accuracy. This approach is efficient when the channel coherence time is much larger than the symbol interval. Preliminary investigations on this topic are available in [27] and presented here for completeness. When this condition does not hold, alternative model-based RNN structures should be considered, where channel estimation is jointly considered in the predictive step.

REFERENCES

[1] S. Dong, P. Wang, and K. Abbas, "A Survey on Deep Learning and its Applications," *Computer Science Review*, vol. 40, p. 100379, 2021.

[2] L. Dai, R. Jiao, F. Adachi, H. V. Poor, and L. Hanzo, "Deep Learning for Wireless Communications: An Emerging Interdisciplinary Paradigm," *IEEE Wireless Communications*, vol. 27, no. 4, pp. 133–139, 2020.

[3] T. O'Shea and J. Hoydis, "An Introduction to Deep Learning for the Physical Layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, 2017.

[4] F. A. Aoudia and J. Hoydis, "End-to-End Learning of Communications Systems Without a Channel Model," in *2018 52nd Asilomar Conference on Signals, Systems, and Computers*, 2018, pp. 298–303.

[5] M. Honkala, D. Korpi, and J. M. Huttunen, "DeepRx: Fully convolutional deep learning receiver," *IEEE Transactions on Wireless Communications*, vol. 20, no. 6, pp. 3925–3940, 2021.

[6] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[7] N. Farsad and A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Transactions on Signal Processing*, vol. 66, no. 21, pp. 5663–5678, 2018.

[8] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-Based Deep Learning," *Proceedings of the IEEE*, vol. 111, no. 5, pp. 465–499, 2023.

[9] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends® in Finance*, vol. 2, no. 1, pp. 1–127, 2009.

[10] Z. Qin, H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep learning in physical layer communications," *IEEE Wireless Communications*, vol. 26, no. 2, pp. 93–99, 2019.

[11] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Transactions on Information Theory*, vol. 13, no. 2, pp. 260–269, 1967.

[12] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate (Corresp.)," *IEEE Transactions on Information Theory*, vol. 20, no. 2, pp. 284–287, 1974.

[13] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Processing Magazine*, vol. 38, no. 2, pp. 18–44, 2021.

[14] T. Raviv, S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger, "Adaptive and Flexible Model-Based AI for Deep Receivers in Dynamic Channels," *IEEE Wireless Communications*, 2024.

[15] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li, and Z. Xu, "Model-driven deep learning for physical layer communications," *IEEE Wireless Communications*, vol. 26, no. 5, pp. 77–83, 2019.

[16] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: A deep learning based Viterbi algorithm for symbol detection," *IEEE Transactions on Wireless Communications*, vol. 19, no. 5, pp. 3319–3331, 2020.

[17] N. Shlezinger, R. Fu, and Y. C. Eldar, "DeepSIC: Deep Soft Interference Cancellation for Multiuser MIMO Detection," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1349–1362, 2021.

[18] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Data-Driven Factor Graphs for Deep Symbol Detection," in *2020 IEEE International Symposium on Information Theory (ISIT)*, 2020, pp. 2682–2687.

[19] J. Yang, Q. Du, and Y. Jiang, "Neural Network-Assisted Receiver Design via Learning Trellis Diagram Online," *IEEE Transactions on Communications*, vol. 70, no. 12, pp. 8075–8085, 2022.

[20] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. G. van Sloun, and Y. C. Eldar, "KalmanNet: Neural Network Aided Kalman Filtering for Partially Known Dynamics," *IEEE Transactions on Signal Processing*, vol. 70, pp. 1532–1547, 2022.

[21] J. Andrews, "Interference cancellation for cellular systems: a contemporary overview," *IEEE Wireless Communications*, vol. 12, no. 2, pp. 19–29, 2005.

[22] C.-H. Chen, B. Karanov, W. van Houtum, W. Yan, A. Young, and A. Alvarado, "On the robustness of deep learning-aided symbol detectors to varying conditions and imperfect channel knowledge," in *2024 IEEE Wireless Communications and Networking Conference (WCNC)*. IEEE, 2024, pp. 1–6.

[23] T. Raviv and N. Shlezinger, "Data augmentation for deep receivers," *IEEE Transactions on Wireless Communications*, vol. 22, no. 11, pp. 8259–8274, 2023.

[24] T. Raviv, S. Park, O. Simeone, Y. C. Eldar, and N. Shlezinger, "Online meta-learning for hybrid model-based deep receivers," *IEEE Transactions on Wireless Communications*, vol. 22, no. 10, pp. 6415–6431, 2023.

[25] G. Montorsi and B. Ripani, "RNN BCJR: a fully trainable version of the additive BCJR algorithm," in *ICC 2023-IEEE International Conference on Communications*. IEEE, 2023, pp. 3696–3701.
[26] S. Benedetto, G. Montorsi, D. Divsalar, and F. Pollara, "Soft-input soft-output modules for the construction and distributed iterative decoding of code networks," *European transactions on telecommunications*, vol. 9, no. 2, pp. 155–172, 1998.
[27] M. Magnaldi and G. Montorsi, "The RNN BCJR Detector in Time-Varying Channels," in *2025 IEEE Wireless Communications and Networking Conference (WCNC)*, 2025, pp. 1–6.
[28] ——, "Adaptive T-Detector exploiting Outer SISO Decoder in Time-Selective Channels," in *2025 International Symposium on Topics in Coding (ISTC)*, 2025.
[29] F. Rusek and A. Prlja, "Optimal channel shortening for MIMO and ISI channels," *IEEE transactions on wireless communications*, vol. 11, no. 2, pp. 810–818, 2011.
[30] D. D. Falconer and F. Magee Jr, "Adaptive channel memory truncation for maximum likelihood sequence estimation," *Bell System Technical Journal*, vol. 52, no. 9, pp. 1541–1562, 1973.
[31] S. Benedetto, D. Divsalar, G. Montorsi, and F. Pollara, "A Soft-Input Soft-Output APP module for iterative decoding of concatenated codes," *IEEE Communications Letters*, vol. 1, no. 1, pp. 22–24, 1997.
[32] 3GPP, "Study on channel model for frequencies from 0.5 to 100 GHz," 2018.
[33] M. J. Gans, "A power-spectral theory of propagation in the mobile-radio environment," *IEEE Transactions on Vehicular Technology*, vol. 21, no. 1, pp. 27–38, 1972.