# PR-NN: RNN-based Detection for Coded Partial-Response Channels

Simeng Zheng, *Student Member, IEEE,* Yi Liu, *Student Member, IEEE,* and Paul H. Siegel, *Life Fellow, IEEE*

***Abstract*—In this paper, we investigate the use of recurrent neural network (RNN)-based detection of magnetic recording channels with inter-symbol interference (ISI). We refer to the proposed detection method, which is intended for recording channels with partial-response equalization, as *Partial-Response Neural Network* (PR-NN). We train bi-directional gated recurrent units (bi-GRUs) to recover the ISI channel inputs from noisy channel output sequences and evaluate the network performance when applied to continuous, streaming data. The computational complexity of PR-NN during the evaluation process is comparable to that of a Viterbi detector. The recording system on which the experiments were conducted uses a rate-2/3, (1,7) runlength-limited (RLL) code with an E$^2$PR4 partial-response channel target. Experimental results with ideal PR signals show that the performance of PR-NN detection approaches that of Viterbi detection in additive white gaussian noise (AWGN). Moreover, the PR-NN detector outperforms Viterbi detection and achieves the performance of *Noise-Predictive Maximum Likelihood* (NPML) detection in additive colored noise (ACN) at different channel densities. A PR-NN detector trained with both AWGN and ACN maintains the performance observed under separate training. Similarly, when trained with ACN corresponding to two different channel densities, PR-NN maintains its performance at both densities. Experiments confirm that this robustness is consistent over a wide range of signal-to-noise ratios (SNRs). Finally, PR-NN displays robust performance when applied to a more realistic magnetic recording channel with MMSE-equalized Lorentzian signals.**

***Index Terms*—Recurrent neural network (RNN), Signal detection, Magnetic recording channels, Partial-response channels, Viterbi detection**

## I. Introduction

### *A. Background on magnetic recording*

The read/write process in longitudinal magnetic recording is often modeled as a continuous-time, linear time-invariant (LTI) system with bipolar input waveforms taking values $-1$ and $+1$ on time intervals of fixed length $T_c$. The step response often takes the form of a unimodal pulse with a finite pulsewidth at half the maximum amplitude (PW50) whose size relative to the channel input interval ($PW50/T_c$) roughly determines the extent of inter-symbol interference (ISI) arising from adjacent transitions in the input waveform. When the channel output signals are synchronously sampled at intervals of $T_c$, the sampled system is often modeled as a finite impulse-response discrete-time LTI system [38].

The authors are with the Center for Memory and Recording Research, Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA (e-mail: sizheng@ucsd.edu; yil333@ucsd.edu; psiegel@ucsd.edu).

In order to facilitate the recovery of the input signal, magnetic recording systems often use some form of partial-response (PR) equalization. The PR equalizer shapes the read-back signal in such a way that only a finite number of values are observed at sample times. For a range of linear recording densities, the sampled Lorentzian output response of the PR-equalized longitudinal recording channel is well modeled by the family of extended PR "class 4" (PR4) channels, denoted by E$^{\text{N-1}}$PR4 [37]. The impulse response of the E$^{\text{N-1}}$PR4 channel can be represented by $x_N(D) = (1-D)(1+D)^N$, where $D$ is the delay operator and $N$ is a positive integer. The channel can be treated as a linear filter with integer coefficients, whose outputs are generated by a linear finite-state machine, where the number of states is $2^{N+1}$. In practice, the selected PR step response is governed by the channel step response and the choice of $PW50/T_c$. When sampled, the PR-equalized noisy magnetic recording channel is often modeled, to first order, as a linear finite-state machine with additive, correlated noise.

The sampled PR-equalized magnetic recording channel resembles a digital communication channel, and suitable detection and coding methods from communication theory can be beneficially applied. The finite-state structure of the ISI channel is amenable to trellis-based sequence detection methods such as Viterbi detection [39], which is optimal if the additive noise is assumed to be white Gaussian noise (AWGN). The combination of PR channel equalization and Viterbi detection is referred to as PRML, an acronym for "partial-response (PR) equalization with maximum-likelihood (ML) sequence detection" [4], [18]. Channels can also use maximum a-posteriori probability (MAP) symbol detection based upon, for example, the BCJR algorithm [1], which is also optimal in AWGN. When combined with PR equalization, the resulting system is referred to as PRMAP. To account for noise correlation (colored noise) due to equalization, Noise-Predictive Maximum Likelihood (NPML) detectors embed a noise prediction/whitening process into the branch metric computation of a Viterbi detector [8]. In longitudinal recording systems, NPML detectors offer significant performance gains over PRML detectors [8], [28].

Magnetic recording systems often use both an error-correcting code and a constrained code. The general purpose of a constrained code is to improve the performance of the system by matching the characteristics of the recorded signals to those of the channel [16]. In the context of PRML-type systems, the constrained code is often used to improve timing recovery as well as to increase the distinguishability of the sampled output sequences. Several classes of such "distance-enhancing codes" have been proposed, such as runlength-limited (RLL) codes [29] and matched spectral null (MSN) codes [19].

In a coded PRML-type system, equalization plays a crucial role in determining the performance of the system. Advanced detectors must take into account the noise correlation and signal misequalization effects due to equalization [43]. The channel parameters in a magnetic hard disk storage system can vary due to several factors, including variations in temperature, head flying-height, and track-dependent rotational speed [42]. These parameter variations need to be taken into consideration in the design of the PR equalizer and they can also influence the performance of the detector. In this paper, we explore the use of machine learning methods – specifically recurrent neural networks (RNNs) – to design robust detectors for magnetic recording systems.

## B. *Machine learning for coded communication*

In recent years, machine learning has demonstrated its effectiveness in a wide range of applications, specifically in the fields of computer vision and natural language processing. The huge success of machine learning in these areas has triggered the interest of researchers to apply deep learning (DL) methods and neural networks (NNs) to channel coding problems. Nachmani *et al.* proposed *Weighted Belief Propagation* (WBP) algorithm using deep neural networks (DNNs) to decode noisy linear codewords [31]. This approach was further studied by Lian *et al.* in the context of simple scaling models with reduced complexity [23]. In [32], Satorras and Welling considered a hybrid model that combines belief propagation with an extension of graph nerual networks to factor graphs (FG-GNNs). Kim *et al.* presented the first end-to-end communication system for feedback channels designed using deep learning, with RNN models for encoding and decoding [21]. Jiang *et al.* incorporated some aspects of an iterative turbo decoder into an RNN-based end-to-end machine learning architecture that provides robust, near-optimal recovery of noisy turbo codewords, without BCJR knowledge [17]. Shlezinger *et al.* introduced ViterbiNet, a decoder that incorporates DNNs into the channel state information (CSI) estimate of the Viterbi algorithm [35]. They also invented BCJRNet, a detection approach that implements data-driven BCJR MAP symbol detector [34].

With all of these learned communication systems, the length of codewords is quite limited because the training complexity grows exponentially in the length [13]. Farsad and Goldsmith addressed this problem by creating a sliding bidirectional RNN to process a longer signal stream [9]. Bennatan *et al.* decoded codewords of an arbitrary block length by extracting the syndrome of the hard decisions and the channel output reliabilities [3]. Tandler *et al.* described a training method that gradually introduces code sequences with an increasing number of ones to limit the complexity, and used it to recover long convolutional codewords [36]. Nevertheless, during the evaluation stage, these NN-based decoders are not well suited to handling continuous streaming data, which would typically be produced by convolutional encoders and ISI channels.

## C. *Our Contribution*

In this work, we propose a novel NN architecture for detection of input-constrained PR-equalized magnetic recording channels, which we refer to as *partial-response neural network* (PR-NN). The PR-NN detector is designed for application to continuous, streaming channel outputs. The sequential processing properties of RNN cells [5], [7], including gated recurrent units (GRUs) [6] and long short-term memory (LSTM) [15], are naturally suited to the time-dependent outputs from PR channels. The primary component of our PR-NN architecture is a bi-directional gated recurrent unit (bi-GRU). (We settled on a GRU-based architecture after initial experiments indicated superior performance compared to an LSTM network. These results are not included here.)

We train and evaluate the PR-NN under various scenarios: ideal PR channel outputs with AWGN, ideal PR channel outputs with additive colored noise (ACN) generated by a minimum mean squared error (MMSE) equalizer for the Lorentzian channel, and MMSE-equalized Lorentzian outputs with corresponding equalized noise. By training the model under multiple scenarios, we show that a single PR-NN detector can be used as a substitute for multiple classical detectors. Moreover, training over a range of channel signal-to-noise ratios (SNRs) and channel densities allows the PR-NN to adapt to a variety of channel conditions. Special training and evaluation techniques make the PR-NN compatible with detection of continuous, streaming data, with no constraint on sequence length, thus overcoming a key limitation of previous NN-based decoding strategies. The continuous decoding relies on a *sliding-window* evaluation process. We also show that the computational complexity of the PR-NN detector is comparable with that of Viterbi detection.

We conduct our experiments using the $\text{E}^2\text{PR4}$ channel model, which matches the characteristics of the Lorentzian channel for high recording densities. The binary system inputs are constrained by a rate-2/3, (1,7)-RLL constrained sliding-block decodable finite-state encoder [41]. The constrained codewords are mapped into binary channel inputs by a non-return-to-zero-inverse (NRZI) precoder with system function represented by $c(D) = 1/(1 + D)$. The resulting sequence is modulated to bipolar form to represent the magnetization pattern corresponding to the recording channel input [16]. With AWGN and a reduced-state Viterbi detector that reflects the input constraint, the system serving as our benchmark achieves a 2.2dB coding gain over the uncoded $\text{E}^2\text{PR4}$ system [2].

The bit error rate (BER) performance of the PR-NN detector compares favorably to that of the classical detectors – PRML, PRMAP and NPML – in the scenarios where they are known to perform well. More importantly, the PR-NN detector exhibits a robustness not shared by the other detectors when it is jointly trained in multiple scenarios. In fact, under joint training, PR-NN essentially maintains the performance that is achieved with separate training. These results suggest that robust detection

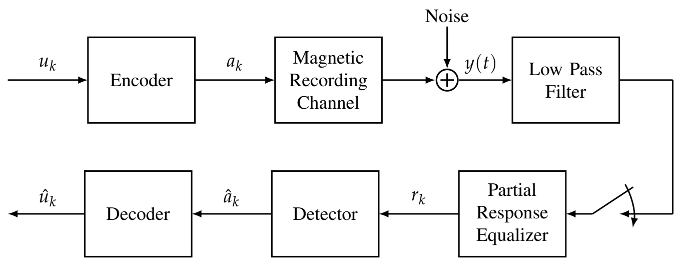


Fig. 1: System architecture.

architectures like PR-NN may hold promise for application in practical recording systems.

### D. Outline

The remainder of this paper is organized as follows. In Section II, we formulate the system architecture for the magnetic recording channel in more detail and describe the digital implementation used in our performance simulations. In Section III, we give a comprehensive discussion of the PR-NN detector, including network architecture, dataset generation, training methodology, evaluation procedure, and computational complexity. In Section IV, we apply the PR-NN detector and determine its performance under several scenarios: individually trained models for E$^2$PR4 channels with AWGN and ACN, equalized-Lorentzian channels with ACN, and various jointly trained scenarios. We then use the simulation results to assess the robustness of PR-NN detection.

## II. System Architecture and Detectors

### A. System model

A block diagram of a magnetic-disk recording system is shown in Fig. 1. User data $\{u_k\}$ ($u_k \in \{0,1\}$) is encoded using a $(d,k)$ run-length limited (RLL) code [16]. The constrained codewords are then precoded and mapped into the symbol sequence $\{a_k\}$ ($a_k \in \{-1,+1\}$). The precoder maps a binary sequence to the two-level channel input sequence. The precoding convention we use is nonreturn-to-zero-inverse (NRZI), where the precoder has system function $c(D) = 1/(1+D)$. The modulation method is binary phase shift keying (BPSK), where 0 maps to $-1$ and 1 maps to $+1$.

During the write process, the two-level channel input is converted into a one-dimensional magnetization pattern on the magnetic medium in the disk. The disk spins with controlled speed, and the read and write heads effectively rotate over a track on the surface of the magnetic medium [42]. During the read process, the disk rotates beneath the read head, which senses the magnetic field induced by the magnetization pattern along the track. The read-back signal can be regarded as a linear superposition of the dipulse response corresponding to a positive pulse of width equal to a single channel bit duration at the input to the channel. Mathematically, the read-back signal $y(t)$ can be expressed as

$$y(t) = \sum_{i=-\infty}^{\infty} a_i q(t - iT_c) + \eta(t) \tag{1}$$

where $T_c$ is the channel bit spacing, and sequence $\{a_k\}$ is written on the disk at a rate of $1/T_c$. The *dipulse response* $q(t)$ is expressed as $q(t) = g(t) - g(t - T_c)$, where $g(t)$ is the unit step response of the channel. The term $\eta(t)$ represents the additive white Gaussian noise process.

The function $g(t)$ is often called the transition response of the recording system, and its characteristics are related to the specific design of the recording heads and magnetic medium. Recording systems are typically classified into two types: longitudinal [27] and perpendicular [42]. The Lorentzian model for the transition response is commonly used for longitudinal recording systems. The *tanh* function and error function approximation are widely used for perpendicular recording systems. In this work, we focus on longitudinal recording with Lorentzian transition response

$$g(t) = \frac{1}{1 + (2t/PW_{50})^2} \tag{2}$$

where $PW_{50}$ is the single parameter of the Lorentzian model and denotes the pulsewidth at 50% maximum amplitude. The recording density is characterized by the normalized density parameter $PW_{50}/T_c$.

The noisy channel output signal passes through a low pass filter (LPF). The filtered signal is sampled at the rate $1/T_c$, generating samples at times $t = kT_c$. The samples are filtered by a discrete-time equalizer which is designed to optimize detector performance. The most common scheme used for equalization and detection in longitudinal recording systems is partial-response maximum-likelihood detection (PRML). In this scheme, a *finite impulse response* (FIR) equalizer is designed to equalize the channel response to a relatively short-duration partial-response (PR) target, and the channel input sequence is recovered from the equalized signal by a maximum-likelihood detector based on the Viterbi algorithm. The family of equalizers called "Class-4" and "extended Class-4" are often used in longitudinal magnetic recording, where the choice of equalizer target is matched to the channel density [37]. The general expression for the samples of the target equalized dipulse response, expressed as a $D$-transform polynomial, takes the form

$$\begin{aligned} x(D) &= (1-D)\alpha(D) = (1-D)(1+D)^N \\ &= x_0 + x_1 D + \cdots + x_{N+1} D^{N+1} \end{aligned} \tag{3}$$

where $\alpha(D) = \alpha_0 + \alpha_1 D + \cdots + \alpha_N D^N$. When $N = 1$, the channel is called a Partial-Response Class-4 (PR4) channel. When $N \geq 2$, the channels are call Extended Partial-Response Class-4, denoted individually as E$^{\text{N-1}}$PR4.

There are many papers that address the design of the PR equalizer, such as [18], [30]. A common design objective is the MMSE equalizer, which minimizes the mean squared error of the target PR signal and the equalized channel output. Since the channel parameters may vary across and even along tracks on a magnetic disk, the equalizer can be designed to be adaptive to the channel properties. Some adaptive equalization architectures for PR channels can be found in [38].

In the longitudinal recording system, if the target PR signal is chosen to be E$^{\text{N-1}}$PR4 signal, the coefficients of the PR equalizer can be optimized to achieve an overall transfer function that reflects the head/medium characteristics and the analog LPF frequency response. If we assume an ideal LPF, the equalizer coefficients $\{z_i\}$ can be specified as

$$\begin{aligned} z_i &= \frac{1}{2\pi}\int_{-\pi}^{\pi}\frac{x(e^{-j\omega})}{Q(\omega)}e^{j\omega i}d\omega \\ &= \frac{1}{\pi^2}\sum_{\ell=0}^{N}\alpha_i\frac{(-1)^{\ell}e^{\pi PW_{50}/2}\cos(i\pi)-PW_{50}/2}{(PW_{50}/2)^2+(i-\ell)^2} \end{aligned} \tag{4}$$

where $Q(\omega)$ is the frequency response of the Lorentzian channel and $T_c = 1$ [27].

Thus the output $r_k$ of the PR equalizer in Fig. 1 consists of an ideal PR signal plus an additive distortion. Mathematically, the equalizer output $r_k$ can be written as

$$r_k = \sum_{i=0}^{N+1} x_i a_{k-i} + n_k \tag{5}$$

where $\{x_i\}$ are the coefficients of the target E$^{\text{N-1}}$PR4 channel and $\{n_k\}$ denotes the additive distortion. The distortion $n_k$ can be decomposed as

$$n_k = \sum_i z_i \eta_{k-i} + (\sum_i \tilde{q}_i a_{k-i} - \sum_{i=0}^{N+1} x_i a_{k-i}) \tag{6}$$

where the summation represents additive colored noise corresponding to the equalized samples of the low-pass filtered white noise and the expression in parentheses represents the misequalization error. Here $\{\tilde{q}_i\}$ correspond to the convolution of the PR equalizer taps with the sampled channel dipulse response.

As discussed above, given a PR target, a trellis-based Viterbi detector [39] is used to detect the data sequence from the noisy channel output sequence $\mathbf{r}$. For the Viterbi detector, the branch metric calculation is based on the squared-Euclidean distance between the noisy channel output sample and the targeted PR channel output sample labeling the particular branch. The combination of PR equalization with Viterbi detection is called PRML. The BCJR detector [1], which is based upon a MAP symbol detection algorithm, is also of interest for data recovery. For the BCJR detector, the *log-likelihood ratios* (LLRs) can be derived from forward recursion, backward recursion, and branch transition probability computations. The system combining PR equalization with BCJR detection is called PRMAP.

As shown in (6), the noise in the longitudinal recording system model is composed of colored noise and misequalization error, which cannot be modeled as AWGN. Therefore, the Viterbi detector is not an optimal sequence detector. NPML detection [8] combines a linear noise prediction/whitening filter with Viterbi detection. The coefficients of the noise predictor are designed to minimize the mean squared error (MSE) of the noise and the predictor output. Algorithmic details of these detection methods will be formulated in Section II-C.

The decoder that recovers user data estimates $\hat{u}_k$ from channel input estimates $\hat{a}_k$ is implemented by means of a sliding-block decoder (which is here assumed to incorporate a $1 + D$ post-coder operation). The decoder has memory $m$ and anticipation $a$, meaning that the current detected codeword, the previous $m$ detected codewords, and the following $a$ detected codewords are all used to determine the corresponding current user data word [24].

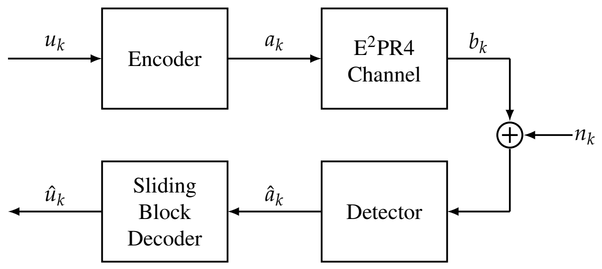


Fig. 2: Discrete model for simulation, where $n_k$ represents the additive distortion.

## B. Digital channel implementation

In the following, we formulate the digital implementation of the magnetic recording system, as shown in Fig. 1, that we will use for our simulations. The outputs of the PR equalizer can be modeled as the outputs of a binary-input, linear ISI channel with additive distortion as derived in (5) and (6). In particular, we can model the ISI channel resulting from the magnetic recording channel, the LPF, the sampler and the PR equalizer in Fig. 1 as an E$^{\text{N-1}}$PR4 channel, as shown in Fig. 2. For our experiments, we focus on the specific case of $N = 3$, namely E$^2$PR4, as this was widely used in practice. The outputs of the channel model can then be represented as

$$r_k = b_k + n_k = \sum_{i=0}^{4} x_i a_{k-i} + n_k \tag{7}$$

where $x_0 = 1$, $x_1 = 2$, $x_2 = 0$, $x_3 = -2$ and $x_4 = -1$. Here $b_k$ denotes the noiseless output from the E$^2$PR4 channel and $a_k$ denotes the channel input.

We now describe the finite-state channel representation of the input-constrained E$^2$PR4 channel. In order to improve the performance of PRML-type systems, several classes of codes with distance-enhancing properties have been proposed, such as forbidden list codes [16] and *matched spectral null* (MSN) codes [19]. In [2], Behrens and Armstrong show that $(1,\infty)$-RLL constrained codes provide a coding gain when applied to the E$^2$PR4 channel. To see this, note that a sequence satisfies the $(1,\infty)$-RLL constraint if the runs of 0s between successive 1s have length at least 1 [24]. In other words, consecutive 1s are forbidden. This means that, in the precoded $(1,\infty)$-RLL sequence, the strings 101 and 010 are prohibited. The minimum squared-Euclidean distance between channel output sequences corresponding to a closed error event (paths in the detector trellis that agree except on a finite number of branches) is 6. These events correspond to the channel inputs $+1\ -1\ +1$ and $-1\ +1\ -1$. On the other hand, for the $(1,\infty)$ input-constrained channel, with these channel inputs

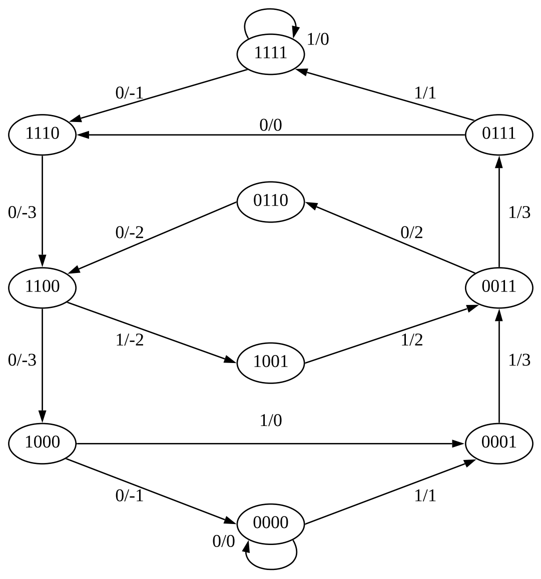


Fig. 3: $(1,\infty)$-RLL input-constrained $E^2PR4$ channel state machine.

forbidden, the minimum distance associated with a closed error event increases to 10. (A more detailed discussion of this sort of distance analysis is found in [19].) This offers the possibility of an effective 2.2 dB gain in signal-to-noise ratio (SNR), ignoring the rate loss associated with the use of the constrained code, provided that the detector trellis is modified to reflect the constraints. Note that this gain also applies to the (1,7)-RLL input-constrained channel.

Incorporating the $(1,\infty)$-RLL constraint into the Viterbi detector for the $E^2PR4$ channel not only eliminates the dominant error events, but also reduces the required number of states in the detector trellis (i.e., the number of states realized by the channel finite state machine) from 16 to 10. To see this, for convenience, we ignore the BPSK modulation used to generate the bipolar channel inputs and, by a slight abuse of notation, we let $a_k \in \{0,1\}$ denote the channel inputs. At time $k$, channel state transitions from state $\mathbf{s}_{k-1} = (a_{k-4}a_{k-3}a_{k-2}a_{k-1})$ to state $\mathbf{s}_k = (a_{k-3}a_{k-2}a_{k-1}a_k)$ with associated output $b_k$. This is represented in the channel state machine diagram by an edge from state $\mathbf{s}_{k-1}$ to state $\mathbf{s}_k$ with input/output label $a_k/b_k$. When the $(1,\infty)$-RLL constraint is applied, states $(0010)$, $(0100)$, $(0101)$, $(1010)$, $(1011)$ and $(1101)$ are eliminated, along with all their incoming and outgoing branches because they represent violations of the constraint. The resulting state machine diagram for the input-constrained $E^2PR4$ channel is shown in Fig. 3. This reduced state machine provides the structure of the trellis that can be used at each time step of the reduced-state Viterbi detector.

We selected the (1,7)-RLL constraint for our system since it has been widely used in commercial magnetic tape and hard disk recording systems. For the encoder and decoder, we use the rate-2/3 Weathers-Wolf code [41], which achieves the minimum possible number of states for any rate-2/3 (1,7)-RLL code. The code is $(0,2)$-sliding-block decodable, meaning that the decoding algorithm can be implemented by a sliding-block decoder, where the current (length-3) codeword along with the following two codewords are used to determine the corresponding (length-2) input word. The encoder and decoder structures are given in [41]. In the sliding-block decoder, a single channel bit error can affect the decoding of up to 3 input words, or 6 user bits, so the error propagation is limited.

## *C. Signal Detection Methods*

In the following, we review three classical signal detection methods for the magnetic recording system. Given the noisy sequence $\mathbf{r}$, the detector will output an estimate $\hat{\mathbf{a}}$ of the channel input sequence.

1) Viterbi detection: When we incorporate the $(1,\infty)$-RLL constraint into the $E^2PR4$ state machine, the 10-state graph determines the trellis structure for the Viterbi detector. The Viterbi detector maximizes the likelihood (conditional probability) $\Pr(\mathbf{r}|\mathbf{a})$ [39]. When the noise is AWGN, the branch metric is the squared Euclidean distance. Specifically, the branch metric at time $kT_c$ from state $\mathbf{s}_j$ to state $\mathbf{s}_m$ takes the form
$$\lambda_k(\mathbf{s}_j,\mathbf{s}_m) = [r_k - (a_k(\mathbf{s}_m) + \sum_{i=1}^{4} x_i a_{k-i}(\mathbf{s}_j))]^2 \qquad (8)$$
where $a_k(\mathbf{s}_m)$, $a_{k-1}(\mathbf{s}_j)$, $a_{k-2}(\mathbf{s}_j)$, $a_{k-3}(\mathbf{s}_j)$ and $a_{k-4}(\mathbf{s}_j)$ are the BPSK input values determined by hypothesized state transition $\mathbf{s}_j \to \mathbf{s}_m$.
2) BCJR detection: The BCJR detection algorithm maximizes the a-posteriori probability $\Pr(\mathbf{a}|\mathbf{r})$. The complete derivation can be found in [1]. In order to reduce the computational complexity, we make use of a modified algorithm, called *max-log-map* detection, that uses the approximation $\ln \sum_j e^{a_j} \approx \max_j a_j$.
3) NPML detection: As mentioned above, the additive distortion in a realistic longitudinal recording system cannot be considered to be simply AWGN. A better approximation takes into account the noise coloration introduced by the equalizer as well as the misequalization error. In the presence of such noise, the Viterbi detector using squared Euclidean distance metric will not provide optimal detection and the system performance will be degraded [28]. NPML detection introduces a noise prediction process into the branch computation of the Viterbi detector that significantly improves the system performance [8]. An estimate of the current noise sample, $\hat{n}_k$ is formed from previous $N_p$ noise samples, and then subtracted from $r_k$. The coefficients of the $N_p$-tap noise predictor $\{p_i\}$ are chosen to minimize the mean squared error between the noise $n_k$ and the estimate $\hat{n}_k$,
$$\mathbb{E}[|n_k - \hat{n}_k|^2] = \mathbb{E}[|n_k - \sum_{i=1}^{N_p} n_{k-i}p_i|^2], \qquad (9)$$
where $n_k$ takes the form in (6). The derivation of the MMSE predictor coefficients can be found in [8].

The implementation of the NPML detector requires the use of tentative decisions from the survivor path memory associated with each state of the Viterbi detector. Mathematically, the branch metric at time $kT_c$ from state $\mathbf{s}_j$ to state $\mathbf{s}_m$ takes the form

$$\lambda_k(\mathbf{s}_j, \mathbf{s}_m) = [r_k - \sum_{i=1}^{N_p}(r_{k-i} - (\sum_{l=0}^{4} x_i \hat{a}_{k-i-l}(\mathbf{s}_j)))p_i - (a_k(\mathbf{s}_m) + \sum_{i=1}^{4} x_i a_{k-i}(\mathbf{s}_j))]^2 \tag{10}$$

where the terms $\hat{a}_{k-i}(\mathbf{s}_j)$, $\hat{a}_{k-i-1}(\mathbf{s}_j)$, $\hat{a}_{k-i-2}(\mathbf{s}_j)$, $\hat{a}_{k-i-3}(\mathbf{s}_j)$ and $\hat{a}_{k-i-4}(\mathbf{s}_j)$ represent past decisions taken from the survivor path history associated with state $\mathbf{s}_j$, and $a_k(\mathbf{s}_m)$, $a_{k-1}(\mathbf{s}_j)$, $a_{k-2}(\mathbf{s}_j)$, $a_{k-3}(\mathbf{s}_j)$ and $a_{k-4}(\mathbf{s}_j)$ are determined by the hypothesized state transition $\mathbf{s}_j \to \mathbf{s}_m$.

## D. Detector implementation details

In practice, Viterbi detectors can retain only a finite path memory and must make an output decision after some fixed delay whether or not all survivor paths have merged. A common practice is to determine the trellis state with the minimum survivor path metric, and then to trace back along the path to the initial branch, whose label is then used to generate the estimated input/output symbol (or word, in the case of a convolutional code). Various estimates for a suitable traceback length $L_{tb}$ for convolutional codes have been proposed, based upon random coding analysis [12], code sequence properties [14], and experimentation [25], [26]. A reasonable rule of thumb for a rate-$r$ code with memory $\nu$ is

$$L_{tb} \approx A\frac{\nu}{1-r} \tag{11}$$

where $A$ is between 2 and 3. For rate $r = 1/2$ codes, this agrees with the often cited estimate $L_{tb} \approx A\nu$ with $A$ between 4 and 6. Similar methods have been used to estimate $L_{tb}$ for Viterbi detection of ISI channels, and a reasonable choice for the traceback length, which we use to guide our experiments, is $L_{tb} \approx 5\nu$.

Rather than decoding one symbol at each iteration of the survivor metric update procedure in the Viterbi detector, we will make use of a sliding-window approach. In the sliding-window decoder, successive blocks of a specified "evaluation length" $L_{eval}$ are estimated, as illustrated schematically in Fig. 4. The survivor metric computation starts at time 0, and the survivor update procedure is performed continuously as the first $L_{eval} + L_{overlap}$ received symbols arrive, where $L_{overlap} \geq L_{tb}$. The detector then traces back along the survivor path corresponding to the state with the smallest survivor metric. The symbol estimates along the first $L_{eval}$ branches can be considered to be fairly reliable and the detector outputs these values. The first $L_{eval}$ branches of the survivor paths are then truncated, and the detector proceeds to extend the remaining portion of the survivor paths for another $L_{eval}$ steps, up to time $2L_{eval} + L_{overlap}$. The detector then traces back along the minimum metric survivor path and outputs the symbol estimates along the first $L_{eval}$ branches, corresponding to symbols at time $L_{eval} + 1$ through $2L_{eval}$. This process is then repeated. In each successive block from time $mL_{eval}$ to time $(m+1)L_{eval} + L_{overlap}$, we refer to the first $L_{eval}$ steps as the evaluation part and the final $L_{overlap}$ steps as the overlap part. The final $L_{overlap}$ symbols can be treated as dummy symbols, or a termination sequence of dummy symbols can be used to force the detector to a known state, enhancing the reliability of the last group of estimated symbols. We adopt the latter termination approach in our simulations.

The sliding-window approach is easily adapted to NPML detection. For the BCJR detector, a conceptually similar sliding-window approach can be implemented using a forward state metric processor and a pair of backward state metric processors [40].

The length of the evaluation block, $L_{eval}$, can be chosen to be any size greater than or equal to one symbol, with the lower limit corresponding to conventional symbol-by-symbol Viterbi decoding. Larger sizes increase the required storage for survivor paths and the delay until the first symbol is decoded. However, the use of longer survivor paths should increase the likelihood of survivor path merging, thereby improving reliability of decoding. In Section III, where we adopt a similar block streaming decoding architecture, the sizes of $L_{eval}$ and $L_{overlap}$ have an exponential effect on the size of the training dataset. This plays a role in the choice of these parameters.

**Remark 1.** For the digital implementation of the longitudinal recording channel, we assume that the channel bit spacing $T_c = 1$. The discrete channel model in Fig. 2 uses a 41-tap model of the Lorentzian channel $\{g_i\}$ and a 21-tap PR equalizer $\{z_i\}$. The NPML detector is implemented using 4-tap noise predictor, 8-tap noise predictor, and 16-tap noise predictors. In the sliding-window evaluation process, the evaluation length $L_{eval}$ is 10 and the overlapping length $L_{overlap}$ is 20. □

## III. PR-NN: RNN-based Detection

In this section, we present PR-NN (partial response - neural network), an RNN-based detection method for coded partial-response channels. We discuss the details of network architecture, dataset generation, training methodology, evaluation procedure, and computational complexity.

The main idea of PR-NN is to replace the classical detectors with a robust RNN-based detector. The motivation of our approach comes from the GRU-based decoder for noisy convolutional codewords in [36]. Although our paper focuses on the application of PR-NN to the coded E$^2$PR4 channel for longitudinal magnetic recording channels, we believe the proposed approach can be easily adapted to other practical magnetic recording systems.

## A. Neural Network Architecture

Thanks to the rapid development of deep learning, many neural network architectures have emerged and shown their power

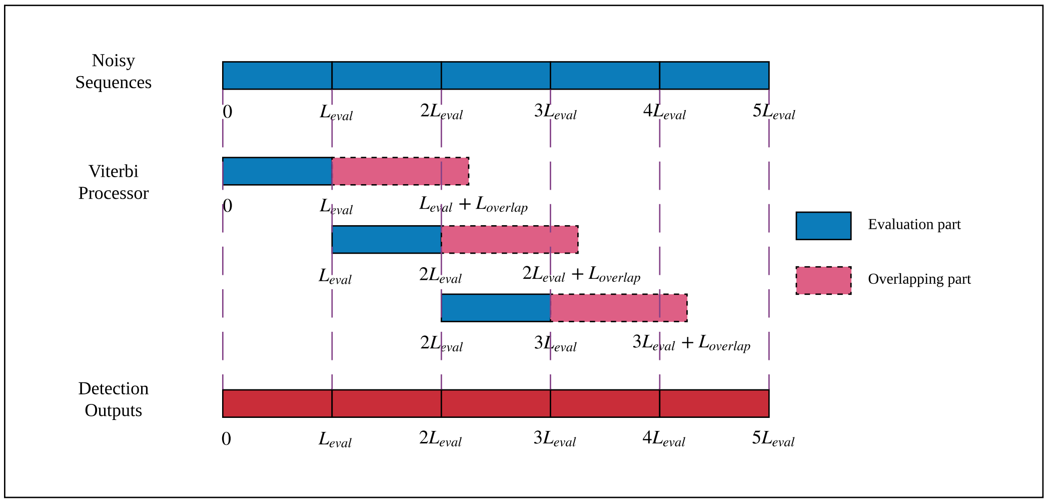


Fig. 4: Sliding-window evaluation process for Viterbi detection.

in a variety of application domains. RNNs, in particular, have been adopted in several scenarios involving time-sequential data, making an RNN-based architecture a natural candidate for processing signals produced by a magnetic recording channel with inter-symbol interference and possibly correlated additive noise.

In practice, we exploit more sophisticated recurrent hidden units that implement a gating mechanism [5], [7], such as long short-term memory (LSTM) units [15] and gated recurrent units (GRUs) [6]. Comparable performance has been found in networks using GRU and LSTM [7]. Our aim in this work is to explore the potential of RNN-based signal detection in channels with ISI, rather than to compare the performance of different RNN units, so we will only consider networks based on GRU cells.

A GRU schematic is shown in Fig. 5a. The calculations in a GRU cell can be formulated as follows

$$\begin{aligned}
\boldsymbol{\gamma}_t &= \sigma(\mathbf{W}_{\alpha\gamma} \cdot \boldsymbol{\alpha}_t + \mathbf{b}_{\alpha\gamma} + \mathbf{W}_{h\gamma} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{h\gamma}) \\
\boldsymbol{\mu}_t &= \sigma(\mathbf{W}_{\alpha\mu} \cdot \boldsymbol{\alpha}_t + \mathbf{b}_{\alpha\mu} + \mathbf{W}_{h\mu} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{h\mu}) \\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_{\alpha\tilde{h}} \cdot \boldsymbol{\alpha}_t + \mathbf{b}_{\alpha\tilde{h}} + \boldsymbol{\gamma}_t \odot (\mathbf{W}_{h\tilde{h}} \cdot \mathbf{h}_{t-1} + \mathbf{b}_{h\tilde{h}})) \\
\mathbf{h}_t &= (\mathbf{1} - \boldsymbol{\mu}_t) \odot \tilde{\mathbf{h}}_t + \boldsymbol{\mu}_t \odot \mathbf{h}_{t-1}
\end{aligned} \tag{12}$$

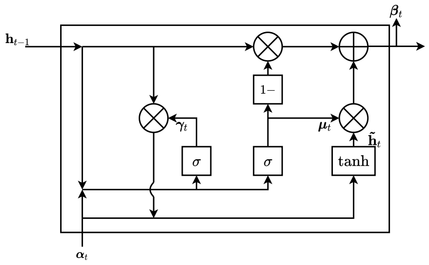
where we use standard notation for weight matrices $\mathbf{W}$, biases $\mathbf{b}$, and activation function $\sigma$. In the calculations, the input at time step $t$ is $\boldsymbol{\alpha}_t \in \mathbb{R}^{m_{in}}$, representing $m_{in}$ features. The hidden state at time step $t-1$ is $\mathbf{h}_{t-1} \in \mathbb{R}^{m_h}$. The output at time step $t$ is $\boldsymbol{\beta}_t \in \mathbb{R}^{m_{out}}$. The hidden state at time step $t$ is $\mathbf{h}_t \in \mathbb{R}^{m_h}$. For a GRU cell, the hidden state $\mathbf{h}_t$ is the same as the output $\boldsymbol{\beta}_t$. The reset, update, and new gates are represented by $\boldsymbol{\gamma}_t \in \mathbb{R}^{m_h}$, $\boldsymbol{\mu}_t \in \mathbb{R}^{m_h}$, and $\tilde{\mathbf{h}}_t \in \mathbb{R}^{m_h}$, respectively. The Hadamard product is denoted by $\odot$.

We adopt a bi-directional GRU (bi-GRU) architecture [33], which can be understood as two separate GRU networks, one operating in the forward direction and the other operating in the backward direction. As illustrated in Fig. 5b, in the forward (resp. backward) direction, the forward (resp. backward) GRU component of the bi-GRU at time step $t$ takes the hidden state $\mathbf{h}^f_{t-1}$ from time step $t-1$ (resp. the hidden state $\mathbf{h}^b_{t+1}$ from time step $t+1$) and produces the hidden state $\mathbf{h}^f_t$ for the next GRU cell at time step $t+1$ (resp. the hidden state $\mathbf{h}^b_t$ for the next GRU cell at time step $t-1$). The forward and backward outputs of the bi-GRU cells are concatenated at each time step.

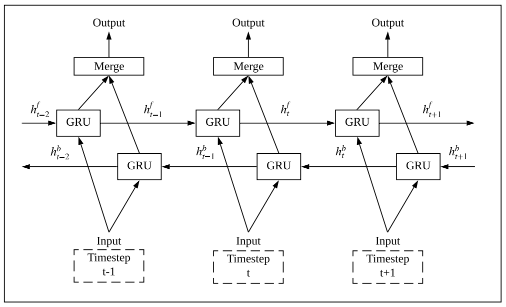
**Remark 2.** Our implementation of the GRU will incorporate multi-layer bi-GRU cells as illustrated in Fig. 5c. We suppose that the total number of time steps in each layer is $T_r$. The bi-GRU cell operates with simultaneous forward and backward passes at each time step $t$ $(1 \le t \le T_r)$. The input of the $i$-th layer $(i \ge 2)$ is the hidden state of the previous layer. (We do not dropout any features from the GRU layer outputs.) We set the default initial hidden states of bi-GRU cells (in both directions) to $\mathbf{0}$; specifically, $\mathbf{h}^f_0 = \mathbf{0}$ and $\mathbf{h}^b_{T_r+1} = \mathbf{0}$. □

Referring to the coded E$^2$PR4 state machine in Fig 3, we see a conceptual similarity between the forward pass of the bi-GRU and the operation of the Viterbi detector, with [current state/input] and [next state/output] corresponding to [previous hidden state/input] and [next hidden state/output], respectively. Similarly, the forward/backward passes of the bi-GRU bear a conceptual resemblance to the foward/backward passes of the BCJR detector.

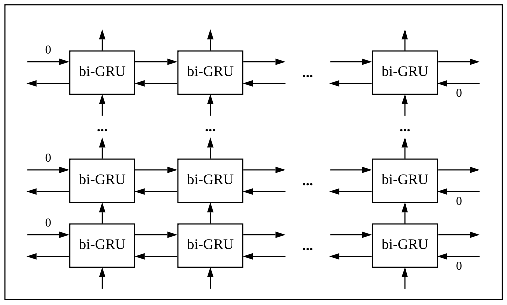
In order to design an RNN-based detector for coded PR channels, we will have to train the network with noisy channel output sequences. The number of coded channel output sequences grows exponentially in the length of the channel input (approximately $2^{Rn}$, where $n$ is the length of the user input sequence and $R$ is the constrained code rate). This suggests the use of a block-oriented network architecture, with a limited block size. In order for the RNN-based detector to process continuous streaming channel outputs, we adopt a sliding-window approach, similar to the sliding-window implementations of the Viterbi detector and BCJR detector presented in Section II-C, both of which process overlapping blocks.



(a) Structure of one GRU cell.



(b) Structure of unfolded bi-directional GRUs.



(c) Model of multi-layer bi-directional GRUs.

Fig. 5: Model of bi-directional GRU network.

Referring to Fig. 4, the idea is for the bi-GRU network to serve as the detection module for each block of length $L_{eval}$, using as the network input a length-$(L_{eval}+L_{overlap})$ block of the recording channel output. However, when we feed a sequence into the multi-layer bi-GRU cells, the sequence needs to respect the default network initialization conditions in Remark 2, i.e., the initial hidden state for the forward direction and the backward direction should be $\mathbf{0}$. We assume that the default initial hidden state $\mathbf{0}$ corresponds to the state $(0000)$ in the coded E$^2$PR4 state machine. This poses a problem, because there is no guarantee that the block of inputs to the network correspond to a state sequence in the PR channel state machine that starts and ends in state $(0000)$. To compensate for this, we propose a *zero compensation* approach, in which we append suitable starting and ending dummy values before and after each block to force sequences to start and end at state $(0000)$. The exact rules of the zero compensation approach are provided in the next subsection.

The input sequence to the bi-GRUs is therefore composed of four parts: starting dummy values, evaluation part, overlapping part, and ending dummy values. The respective lengths of these parts are denoted $L_{start}$, $L_{eval}$, $L_{overlap}$, and $L_{end}$. The resulting total number of time steps in the bi-GRUs is $T_r = L_{start} + L_{eval} + L_{overlap} + L_{end}$.

Now we specify the network components of the PR-NN detector. There are three kinds of layers: dense layers $\mathcal{D}(\mathbf{x})$, multi-layer bi-GRU cells $\mathcal{R}(\mathbf{x}, \mathbf{H}^f_t, \mathbf{H}^b_t)$, and the sigmoid layer $\mathcal{S}(\mathbf{x})$. In this network, GRU cells are the key components. For time step $t$ $(1 \le t \le T_r)$, the details of three kinds of layers are listed below.

1) Dense layer $\mathcal{D}(\mathbf{x})$: given an input vector $\mathbf{x} \in \mathbb{R}^{m_{din}}$, a dense layer defines the following operation
$$\mathbf{y} = \mathcal{D}(\mathbf{x}) = \mathbf{W}_d \cdot \mathbf{x} + \mathbf{b}_d \tag{13}$$
where $\mathbf{y} \in \mathbb{R}^{m_{dout}}$ is the output of the dense layer.
2) Multi-layer bi-GRU cells $\mathcal{R}(\mathbf{x}, \mathbf{H}^f_t, \mathbf{H}^b_t)$: The number of layers is denoted as $N_r$. The input vector is $\mathbf{x} \in \mathbb{R}^{m_{rin}}$, the forward (resp. backward) hidden state set is $\mathbf{H}^f_t = \{\mathbf{h}^{f1}_t, \mathbf{h}^{f2}_t, \cdots, \mathbf{h}^{fN_r}_t\}$ (resp. $\mathbf{H}^b_t = \{\mathbf{h}^{b1}_t, \mathbf{h}^{b2}_t, \cdots, \mathbf{h}^{bN_r}_t\}$), where $\mathbf{h}^*_t \in \mathbb{R}^{m_{rh}}$ is the hidden state vector and $t$ corresponds to the current time step. The output vector at time step $t$ is $\mathbf{y} \in \mathbb{R}^{m_{rout}}$. The mathematical expression for the network operation is
$$\mathbf{y} = \mathcal{R}(\mathbf{x}, \mathbf{H}^f_t, \mathbf{H}^b_t) \tag{14}$$
(The calculations of the GRU cell were presented in (12) and the structure of the multi-layer bi-GRU cells was formulated in Remark 2.)
3) Sigmoid layer $\mathcal{S}(\mathbf{x})$: given an input vector $\mathbf{x} \in \mathbb{R}^{m_{sin}}$, the output $\mathbf{y} \in \mathbb{R}^{m_{sout}}$ of the sigmoid layer is
$$\mathbf{y} = \mathcal{S}(\mathbf{x}) : y_i = \frac{1}{1+e^{x_i}}. \tag{15}$$

The PR-NN contains the dense layers (Dense1 layer $\mathcal{D}_1(\mathbf{x})$ and Dense2 layer $\mathcal{D}_2(\mathbf{x})$), the multi-layer bi-GRU cells $(\mathcal{R}(\mathbf{x}, \mathbf{H}^f_t, \mathbf{H}^b_t, t))$, and the Sigmoid layer $(\mathcal{S}(\mathbf{x}))$. The input to the PR-NN is derived from a length-$T_r$ noisy channel output sequence, $\mathbf{r} = \{r_1, r_2, \cdots, r_{T_r}\}$, where the total number of time steps in PR-NN is also $T_r$. To reflect the memory of the E$^2$PR4 channel, we transform the sequence $\mathbf{r}$ into the network input vector $\underline{\mathbf{r}}' = \{\mathbf{r}^{(1)}, \mathbf{r}^{(2)}, \cdots, \mathbf{r}^{(T_r)}\}$, where $\mathbf{r}^{(k)} = \{r_{k-4}, r_{k-3}, r_{k-2}, r_{k-1}, r_k\}$.

The network architecture is shown in Fig. 6. Taking into account the starting and ending dummy values, we discard the outputs of multi-layer bi-GRUs for the time steps $1 \le k \le L_{start}$ and $T_r - L_{end} + 1 \le k \le T_r$. The output vector is $\mathbf{y} = \{y_{L_{start}+1}, y_{L_{start}+2}, \cdots, y_{T_r - L_{end}}\} \in \mathbb{R}^{(L_{start}+L_{eval})}$ where, for time step $L_{start}+1 \le k \le T_r - L_{end}$, the output is given by
$$y_k = \mathcal{S}(\mathcal{D}_2(\mathcal{R}(\mathcal{D}_1(\mathbf{r}^{(k)}), \mathbf{H}^f_k, \mathbf{H}^b_k))). \tag{16}$$
Here we use time index $k$, rather than $t$, to be consistent with the indexing in the symbol sequences generated by the magnetic recording channel.

**Remark 3.** In our experiments, for the network architecture, we chose $L_{start} = L_{end} = 5$, $L_{eval} = 10$. The length of

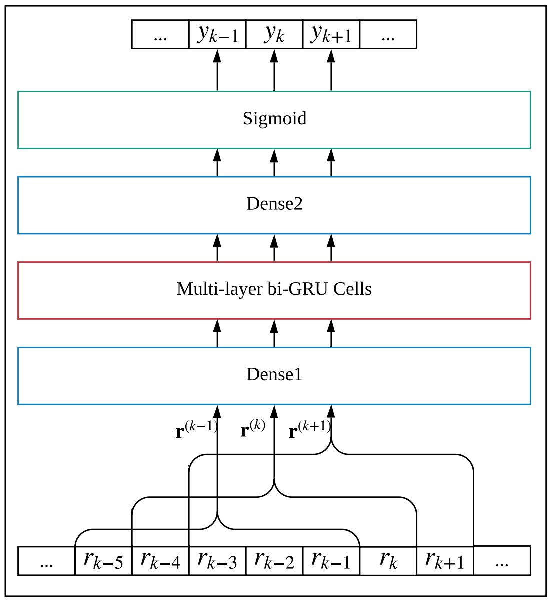


Fig. 6: Network architecture for proposed PR-NN.

the overlapping part is $L_{overlap} = 20$, which is the same as the truncation depth in the Viterbi detector. Thereby, the total number of time steps in PR-NN is $T_r = 40$. The parameters $L_{eval}$ and $L_{overlap}$ are also used in the simulations for the classical detection methods. For Dense1 layer, the number of input features is 5 and the number of output features is 5. For the multi-layer bi-GRU cells, the number of layers is $N_r = 4$ and the number of features in a hidden state is 50. For Dense2 layer, the number of input features is 100 and the number of output features is 1. □

## B. Data Acquisition

The PR-NN detector recovers PR channel inputs from noisy PR channel outputs. The training dataset of noisy channel outputs is created as follows. First, the coded E$^2$PR4 channel state machine in Fig. 3 is used to generate a length-($L_{eval} + L_{overlap}$) channel input sequence from an arbitrarily chosen initial state, and then suitable distortion is added. In our experiments, we consider three kinds of noise generated from the longitudinal recording system: AWGN, ACN generated by the MMSE PR equalizer for the Lorentzian channel, and total distortion noise $n_k$ generated according to (6). In order to train the PR-NN to adapt to different SNRs, the noise in the training set also reflects a range of SNRs.

We use a *zero compensation* rule to ensure the network inputs satisfy the initial settings of bi-GRU cells, which require the corresponding starting and ending states of the PR channel state machine to be $(0000)$. A string of $L_{start} = 5$ starting dummy values is prepended to the noisy channel output sequence according to the initial state, as indicated in Table I. As will be described in Section III-D, PR-NN uses a sliding-window evaluation process, in which the starting state for each truncated input block is determined by the symbols in the previously recovered evaluation block. The starting dummy

| State | Starting dummy values | Ending dummy values |
|---|---|---|
| $(0000)$ | $\{0,0,0,0,0\}$ | $\{0,0,0,0,0\}$ |
| $(0001)$ | $\{0,0,0,0,1\}$ | $\{3,2,-2,-3,-1\}$ |
| $(0011)$ | $\{0,0,0,1,3\}$ | $\{2,-2,-3,-1,0\}$ |
| $(0110)$ | $\{0,0,1,3,2\}$ | $\{-2,-3,-1,0,0\}$ |
| $(0111)$ | $\{0,0,1,3,3\}$ | $\{0,-3,-3,-1,0\}$ |
| $(1000)$ | $\{1,3,2,-2,-3\}$ | $\{-1,0,0,0,0\}$ |
| $(1001)$ | $\{1,3,2,-2,-2\}$ | $\{2,2,-2,-3,-1\}$ |
| $(1100)$ | $\{0,1,3,2,-2\}$ | $\{-3,-1,0,0,0\}$ |
| $(1110)$ | $\{0,1,3,3,0\}$ | $\{-3,-3,-1,0,0\}$ |
| $(1111)$ | $\{0,1,3,3,1\}$ | $\{-1,-3,-3,-1,0\}$ |
| Unknown | $\{0,0,0,0,0\}$ | $\{0,0,0,0,0\}$ |

TABLE I: Starting and ending dummy values for each state in the coded E$^2$PR4 state machine. “Unknown” means unknown starting or ending state for the sequence.

values are then determined by the zero compensation rule, and since the starting state is assumed to be correct, we do not add noise to the starting dummy values. The final state of the path used to generate the input sequence determines a path to state $(0000)$ and a corresponding string of $L_{end} = 5$ ending dummy values, also shown in Table I. However, since the ending state will be unknown during evalution, we add noise to these ending dummy values in the training sequence to represent noisy outputs corresponding to an unknown ending state sequence. The training label for this training sequence is the length-($L_{eval} + L_{overlap}$) channel input sequence.

During evalution, length-($L_{eval} + L_{overlap}$) truncated blocks of a continuous streaming noisy channlel output sequence will be applied to the PR-NN detector. The evaluation labels are the corresponding detected PR-channel input sequences. In order to process the truncated blocks, a string of starting dummy values of length $L_{start}$ is prepended, using the starting state derived from the previously recovered evaluation block and Table I. An all-zero string of length $L_{end}$ is appended to the block, reflecting the fact that the final state of the truncated block is unknown.

Example 1 illustrates the use of Table I in the generation of dummy values.

**Example 1.** If the starting state is $(1001)$, a path which forces the sequence from $(0000)$ to $(1001)$ is

$$(0000)\to(0001)\to(0011)\to(0110)\to(1100)\to(1001).$$

Thus the corresponding starting dummy values for state $(1001)$ are $\{1,3,2,-2,-2\}$. If the ending state is $(1001)$, a path which drives the sequence from $(1001)$ to $(0000)$ is

$$(1001)\to(0011)\to(0110)\to(1100)\to(1000)\to(0000).$$

Thus the corresponding ending dummy values for state $(1001)$ are $\{2,2,-2,-3,-1\}$. During training, noise will then be added to these values. □

## C. Training Methodology

The training dataset is fed into the network and we compare the outputs from the network with the training labels. The comparison metric is the loss function. The loss function between an output vector $\mathbf{y}$ and its corresponding channel input $\mathbf{a}$ can be defined as

$$\mathcal{L} = \sum_{k=L_{start}+1}^{L_{start}+L_{overlap}} -a_k \cdot \log(y_k) - (1-a_k) \cdot \log(1-y_k) \tag{17}$$

To address the complexity of training the nework with the entire channel output space, we adopted the *a-priori ramp-up* training method in [36]. Instead of beginning the training with independent, uniform user data $u_k \sim \text{Bern}(0.5)$, we start training with user data $u_k \sim \text{Bern}(p)$ ($p < 0.5$) and gradually increase $p$ to 0.5. In our case, the training data $r_k$ is generated from the precoded constrained symbols $a_k$, and $a_k$ is determined by the user data $u_k$. The biasing probability $p(\text{ep})$ in $u_k \sim \text{Bern}(p)$ is a function of the epoch number ep, defined as

$$p(\text{ep}) = \begin{cases} 0.1 + 0.01 \cdot \lfloor \text{ep}/\#\text{step} \rfloor, & \text{ep} \leq 40 \cdot \#\text{step} \\ 0.5, & \text{ep} > 40 \cdot \#\text{step} \end{cases} \tag{18}$$

where #step is the number of epochs between increments in the probability. After every #step epochs, the probability $p(\text{ep})$ will increase by 0.01 until 0.5 is reached.

## D. Evaluation Process

The outputs $y_k$ of the network are real values in the range $[0,1]$. These are converted to binary detector outputs $\hat{a}_k$ using an indicator function: $\hat{a}_k = \mathbb{1}_{\{x>0.5\}}(y_k)$.

The sliding-window concept used in the evaluation process is shown in Fig. 7. We assume the state machine has initial state $(0000)$. When the first block of $L_{eval} + L_{overlap}$ symbols are received, the PR - NN detector prepends to the block the starting dummy values corresponding to state $(0000)$ and appends the ending dummy values corresponding to the unknown state. The resulting sequence is processed by the network, producing the detector outputs for the first length-$L_{eval}$ block. The last 4 bits of the recovered block determine the starting state for the next detection stage. When an additional $L_{eval}$ output symbols are received, the sliding window shifts by $L_{eval}$ positions and begins processing the next length-$(L_{eval} + L_{overlap})$ truncated block. The starting dummy values for this block depend on the previously recovered starting state, and the ending dummy values correspond to the unknown ending state. The resulting length-$L_r$ block is then processed by the network. This procedure continues until the entire stream of noisy channel outputs is processed.

For a noisy channel output sequence of length $L$, the evaluation metric is the bit error rate (BER) between the input $\mathbf{a}$ and the detection result $\hat{\mathbf{a}}$, defined as

$$\text{BER} = \frac{1}{L} \sum_{k=1}^{L} \mathbb{1}_{\hat{a}_k \neq a_k}(\hat{a}_k) \tag{19}$$

**Remark 4.** For the training set, the noise is generated for SNR values (in dB) in the set $\mathbb{S} = \{8.5, 9.0, 9.5, 10.0, 10.5\}$ (The SNR definition can be found in the Appendix.) Throughout the training, Adam optimizer [22] was used with learning rate $10^{-3}$. The value #step used in the a-priori ramp-up training was set to 50. □

## E. Computational Complexity Analysis

In this section, we compare the computational complexity of the Viterbi dectector and PR-NN detector. We denote the length of noisy channel output sequence by $L$ and the number of states in the input-constrained channel trellis by $N_{st}$. Note that $N_{st} \leq 2^v$.

We first analyze the asymptotic computational complexity. For a Viterbi detector, the computational complexity can be estimated as $O(N_{st}^2 L) = O(2^{2\nu} L)$.

We now consider the complexity of PR-NN. Noting that the complexity of RNN is asymptotically linear in the number of time steps, we let $\mathcal{T}_{NN}$ denote the computational complexity of one time step in the network. For the evaluation process, the processing of each truncated block of length $L_{eval} + L_{overlap}$ requires $(L_{start} + L_{eval} + L_{overlap} + L_{end})$ steps, where $L_{start} = L_{end} = L_{dummy}$ and $L_{overlap}$ has the form $A\nu$, for some constants $L_{dummy}$ and $A$. Thus, the associated complexity is nominally $C = (2L_{dummy} + L_{eval} + A\nu)\mathcal{T}_{NN}$. Since the multi-layer bi-GRU outputs corresponding to dummy values can be ignored by the Dense2 layer, the complexity is actually $C' \leq C$. Processing the entire network input stream involves approximately $L/L_{eval}$ blocks, so the overall complexity is approximately $LC'/L_{eval} = O(\nu L)$.

Thus, in the asymptotic view as the stream length $L$ becomes large, the complexity of PR-NN compares favorably to that of Viterbi detection in PRML. Analogous complexity analysis for PRMAP and NPML detection leads to similar conclusions.

To more precisely evaluate the constants in the "order of" complexity estimate, we analyze in more detail the complexity of operations in the Viterbi and PR-NN detectors. For a Viterbi detector, a typical implementation consists of a pipeline of three units: a branch metric unit (BMU), an add-compare-select unit (ACSU), and a survivor-memory-unit (SMU) [10]. The complexity of the add-compare-select unit for the PR Viterbi decoder with $N_{st}$ states is approximately $N_{st}$ adders, where $N_{st}/2$ adders are required for the addition of branch metrics and the other $N_{st}/2$ make up the complexity of the $N_{st}/2$ two-level compares [11]. If we denote the complexity of this pipeline at each time step as $\mathcal{T}_0$, the overall complexity can be estimated as $\mathcal{T}_0 L$. The MAP decoder can be implemented with no more than 4 times that of a Viterbi decoder [40].

We now discuss the complexity of PR-NN. The number of multiplication operations and the number of addition operations in a combined matrix-vector product and vector addition operation such as $\mathbf{W} \cdot \boldsymbol{x} + \mathbf{b}$ are both equal to the number of elements in the matrix. We let $\mathcal{T}_1$ denote the complexity of a pair of multiplication and addition operations, and assume that the complexity of the activation and indicator functions can be ignored. Denoting the numbers of weights in Dense1 layer, multi-layer bi-GRU cells (one time step) and Dense2 layer as

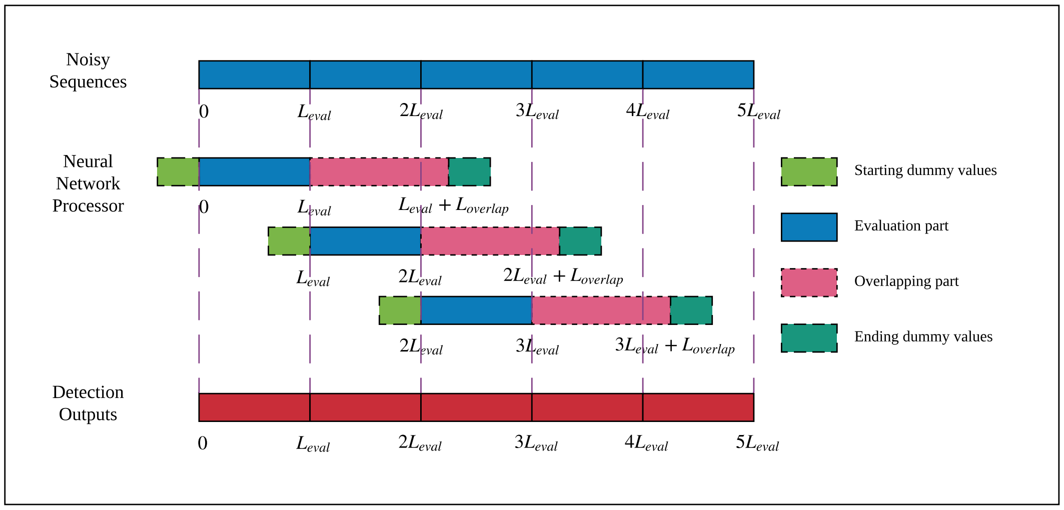


Fig. 7: Sliding-window evaluation process for PR-NN detector.

$n_{D1}$, $n_G$ and $n_{D2}$, respectively, we see that $\mathcal{T}_{NN} = (n_{D1} + n_G + n_{D2})\mathcal{T}_1$. Because the dummy parts are ignored by Dense2 layer, the complexity is actually $C' = C - 2L_{dummy}n_{D2}\mathcal{T}_1$. Taking into account the sliding-window process, the overall complexity of PR-NN is $LC'/L_{eval}$.

The following example uses the E$^2$PR4 channel to illustrate the calculation of the operational complexity of Viterbi and PR-NN detectors.

**Example 2.** For the coded E$^2$PRML detector, the trellis at each time step has 10 states in Fig. 3. At each time step, the BMU will compute 16 branch metrics, the ACSU will compute 10 path metrics, and the SMU will store the decisions made by the ACSU. The complexity of the add-compare-select operation is about 10 adders.

For the PR-NN, based on the parameters in Remark. 3, the number of weights (including biases) in the network layers are $n_{D1} = 30$, $n_G = 153.9k$ and $n_{D2} = 101$. Thus, $C' = 40\mathcal{T}_{NN} - 1010\mathcal{T}_1$. The overall complexity of PR-NN is approximately $615k \cdot \mathcal{T}_1 L$. □

## IV. Experimental Results

In this section, we present our experimental results for PR-NN detection of the coded E$^2$PR4 channel. We consider three scenarios. First, we train the network separately on ideal PR signals with AWGN or ACN generated by a PR equalizer. Then we use joint training on different combinations of AWGN and ACN, as well as on a combination of ACN corresponding to different channel densities. Finally, we train the network with "realistic" equalized Lorentzian channel signals and distortions that include colored noise and misequalization errors. Together, these experiments shed light on the robustness of PR-NN detection. We note that under each scenario, with the a-priori ramp-up approach, PR-NN training converges after $(40 \cdot \#\text{step})$ epochs based on monitoring the loss function.

### *A. Experimental Setup*

In the first scenario, we train PR-NN with only one kind of noise, i.e., AWGN or ACN. For ACN, the colored noise is generated by applying the MMSE PR equalizer to AWGN samples. According to (4), the ACN is affected by the channel density parameter $PW_{50}/T_c$, so we use two different, representative values of $PW_{50}/T_c$ in our experiments. In the training process, the batch size for each SNR in $\mathbb{S}$ is 30.

In the second scenario, we assess PR-NN robustness by training a single network to adapt to two different types of noise: (a) AWGN and ACN at $PW_{50}/T_c = 2.54$, or (b) ACN at $PW_{50}/T_c = 2.54$ and at $PW_{50}/T_c = 2.88$. The training batch size settings for case (a) are shown in lines 1, 2, and 3 and for case (b) in line 4 of Table II.

In the third scenario, the channel outputs represent a more realistic, MMSE-equalized Lorentzian channel with misequalization errors and ACN. To assess robustness to different channel densities, we train the PR-NN with separate datasets at $PW_{50}/T_c = 2.54$ or $PW_{50}/T_c = 2.88$, and then jointly with a dataset combining the two densities. Batch sizes are shown in lines 5, 6, and 7 of Table II.

### *B. Scenario 1: Individual Training Experiments*

In scenario 1, PR-NN is only trained with one noise, i.e., AWGN, ACN ($PW_{50}/T_c = 2.54$), ACN($PW_{50}/T_c = 2.88$), where the batch size is 30 for each SNR in $\mathbb{S}$. The evaluation results over the corresponding channels are described below.

**Experiment 1.1**: We trained the PR-NN with AWGN. As shown in the BER plot in Fig. 8, coded E$^2$PRML achieves the expected 2.2dB gain over uncoded E$^2$PRML with Viterbi detection [2]. In both cases, E$^2$PRMAP, implemented by *max-log-map* approximation, performs essentially the same as E$^2$PRML. (In the subsequent experiments, because E$^2$PRMAP shows similar performance with E$^2$PRML, we only present the simulation results of E$^2$PRML.) The user data BER of the coded E$^2$PRML channel suffers a loss of about 0.9dB due to error propagation of the sliding-block decoder.

| SNR | 8.5dB | 9.0dB | 9.5dB | 10.0dB | 10.5dB | 8.5dB | 9.0dB | 9.5dB | 10.0dB | 10.5dB |
|---|---|---|---|---|---|---|---|---|---|---|
| Noise type | White noise | | | | | Colored noise ($PW_{50}/T_c = 2.54$) | | | | |
| Experiment 2.1 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| Experiment 2.2 | 40 | 40 | 40 | 40 | 40 | 20 | 20 | 20 | 20 | 20 |
| Experiment 2.3 | 50 | 50 | 50 | 50 | 50 | 10 | 10 | 10 | 10 | 10 |
| Noise type | Colored noise ($PW_{50}/T_c = 2.54$) | | | | | Colored noise ($PW_{50}/T_c = 2.88$) | | | | |
| Experiment 2.4 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |
| Noise type | "Realistic" system ($PW_{50}/T_c = 2.54$) | | | | | "Realistic" system ($PW_{50}/T_c = 2.88$) | | | | |
| Experiment 3.1 | 30 | 30 | 30 | 30 | 30 | - | - | - | - | - |
| Experiment 3.2 | - | - | - | - | - | 30 | 30 | 30 | 30 | 30 |
| Experiment 3.3 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 | 30 |

TABLE II: Batch size settings for the training datasets and evaluation cases in each experiment of the three scenarios.

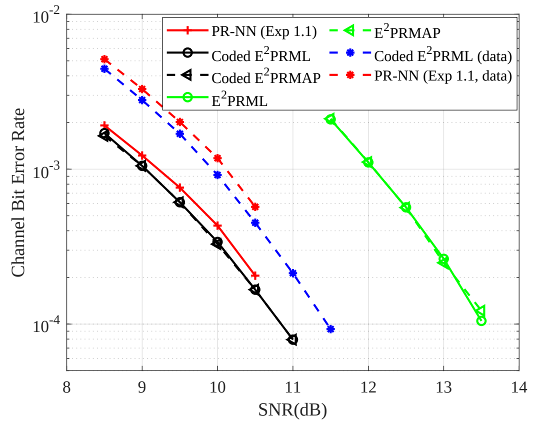
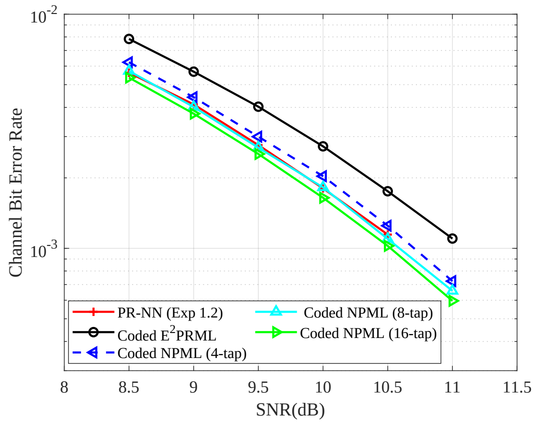
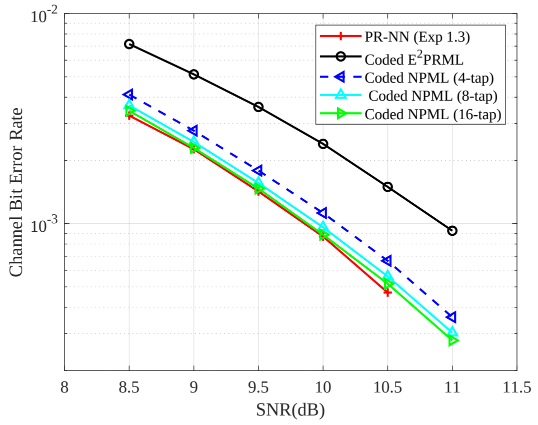


Fig. 8: Scenario 1: Individual training with AWGN.



(a) $PW_{50}/T_c = 2.54$



(b) $PW_{50}/T_c = 2.88$

Fig. 9: Scenario 1: Individual training with ACN (for two channel densities).

We see that the PR-NN achieves performance within 0.1dB of the optimal Viterbi detector. There is a similar gap in performance for the user data BER. (In subsequent experiments, we present only the BER results at the detector output, not the user data results.)

At all SNRs, the histograms of error positions observed within an evaluation block of length $L_{eval} = 10$ for the PR-NN detector and the Viterbi detector are approximately uniform, indicating that the overlapping part is providing "reliable" state information for the evaluation part.

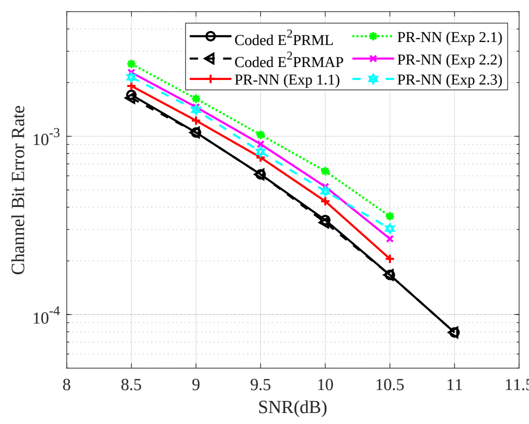
**Experiment 1.2**: The PR-NN detector is trained and evaluated with ACN ($PW_{50}/T_c = 2.54$). Referring to Fig. 9a, we see that the performance of coded E$^2$PRML is degraded in colored noise. The NPML detectors with 4-tap, 8-tap, and 16-tap predictors realize gains of 0.4dB, 0.5dB, and 0.6dB gain, respectively, over coded E$^2$PRML. Note that for these experiments, the NPML detector designs assume no misequalization error. The PR-NN detector has very similar performance to the 8-tap NPML detector.

**Experiment 1.3**: The PR-NN detector is trained and tested with ACN ($PW_{50}/T_c = 2.88$). Referring to Fig. 9b, we see that, in this case, the NPML detectors with 4-tap, 8-tap, and 16-tap predictors realize gains of 1.3dB, 1.5dB, and 1.55dB, respectively, over coded E$^2$PRML. The PR-NN performance is very close to that of the 16-tap NPML detector. In fact, at SNR=10.5dB, the BER achieved by PR-NN is even slightly better.

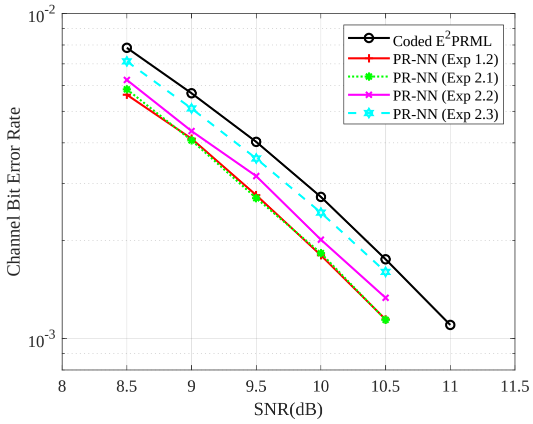
## C. Scenario 2: Joint Training Experiments

In scenario 2, we train a single PR-NN using a dataset that combines noisy outputs representing different recording channels, and then evaluate its performance on both channels. Four different situations are considered.

**Experiments 2.1, 2.2, 2.3**: First, the PR-NN detector is trained with both AWGN and ACN ($PW_{50}/T_c = 2.54$). Experiments 2.1, 2.2, and 2.3 use different relative batch sizes for AWGN and ACN samples within the training set, as shown in Table II. The simulation results are summarized in Fig. 10.

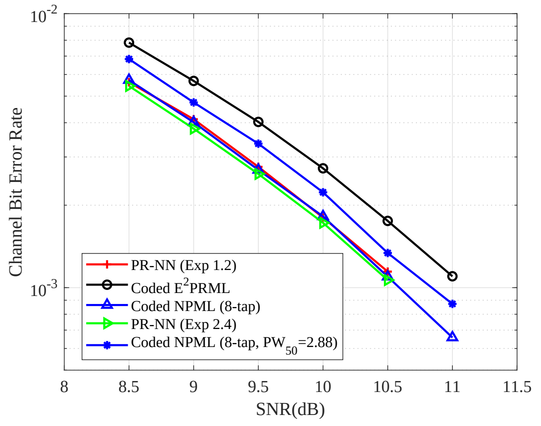


(a) AWGN

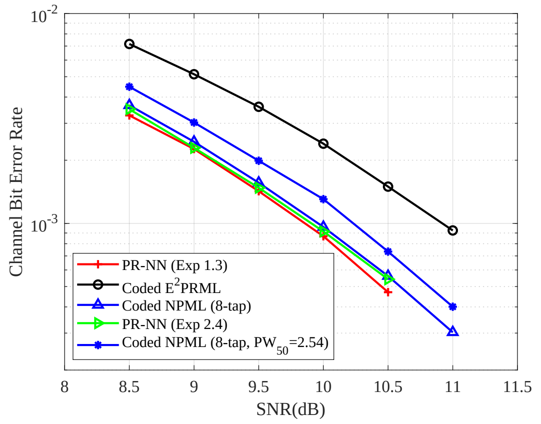


(b) ACN ($PW_{50}/T_c = 2.54$)

Fig. 10: Scenario 2: Joint training with AWGN and ACN ($PW_{50}/T_c = 2.54$).



(a) $PW_{50}/T_c = 2.54$



(b) $PW_{50}/T_c = 2.88$

Fig. 11: Scenario 2: Joint training with ACN ($PW_{50}/T_c = 2.54$) and ACN ($PW_{50}/T_c = 2.88$).

The solid red curve in Fig. 10a represents the best performance in AWGN achieved by the network individually trained with AWGN (Experiment 1.1). When trained with the combined datasets, the jointly trained PR-NNs suffer losses of 0.4dB, 0.2dB, and 0.1dB, respectively, with respect to the network individually trained with ACN (Experiment 1.2), with the larger relative batch size for AWGN giving the best performance. On the other hand, as shown in Fig. 10b, when we evaluate the jointly-trained PR-NN under ACN ($PW_{50}/T_c = 2.54$), the performance losses are 0dB, 0.1dB, and 0.4dB, respectively, with increasing relative AWGN batch size.

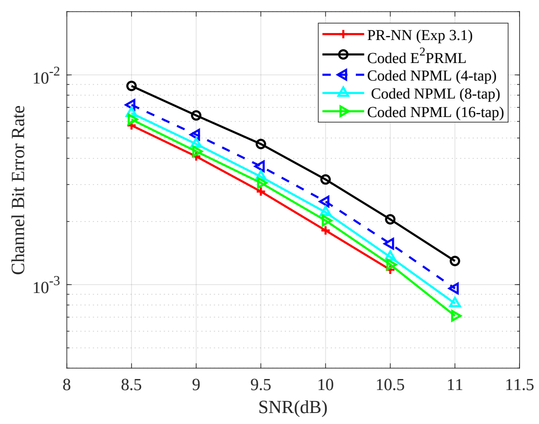
These results suggest that the best compromise in terms of robustness of performance is offered by the jointly trained PR-NN in Experiment 2.2, with losses of only 0.2dB in AWGN and 0.1dB in ACN.

**Experiment 2.4**: In this experiment, we explore the robustness of a jointly-trained PR-NN detector at different channel densities ($PW_{50}/T_c = 2.54$ and $PW_{50}/T_c = 2.88$). The training batch sizes are shown in Table II. The simulation results in Fig. 11a show that the resulting PR-NN detector matches the performance of the individually-trained network in Experiment 1.2. Moreover, as seen in Fig. 11b, the performance is only slightly worse than that of the network trained in Experiment 1.3. Thus, the jointly-trained network appears to offer adaptivity to different recording densities, which can arise from system variations in temperature and head flying height, or at different disk radii.

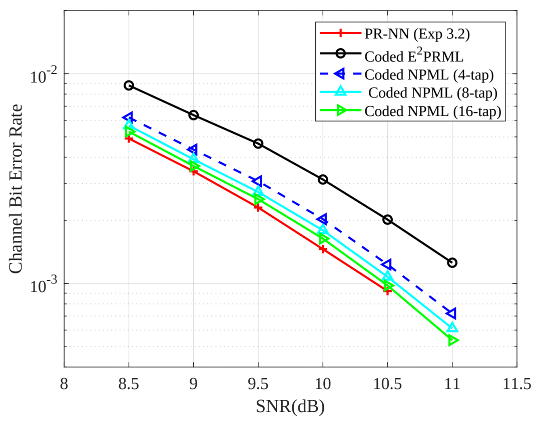
Interestingly, the NPML detectors do not exhibit the same sort of robustness as the PR-NN detector. Fig. 11a shows that the NPML detector optimized for $PW_{50}/T_c = 2.88$ experiences a performance loss of 0.2dB with respect to the NPML detector properly optimized for $PW_{50}/T_c = 2.54$. Similarly, we see in Fig. 11b that the NPML detector designed for $PW_{50}/T_c = 2.54$ incurs a penalty of 0.3dB compared to the NPML detector designed for $PW_{50}/T_c = 2.88$.

### *D. Scenario 3: "Realistic" Equalized Lorentzian Channel*

In a "realistic" recording system modeled as an equalized Lorentzian channel, the signal distortions include both colored noise and misequalization errors, as shown in (6). In this third set of experiments, we first compare the performance of NPML detection and PR-NN detection for such a system at two channel densities. We then assess the robustness of a PR-NN detector trained jointly for use at both densities. Note that in

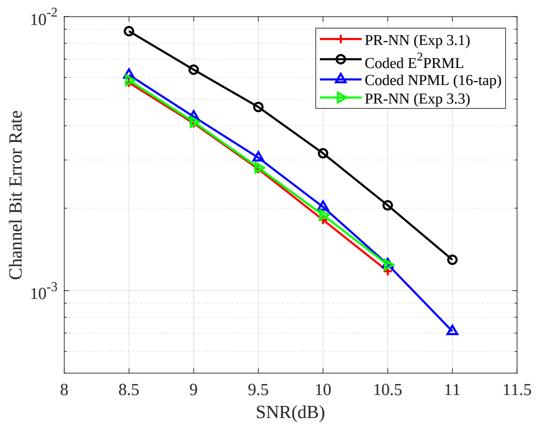


(a) $PW_{50}/T_c = 2.54$

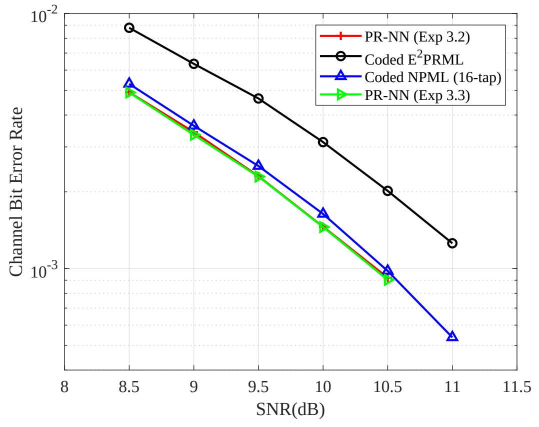


(b) $PW_{50}/T_c = 2.88$

Fig. 12: Scenario 3: Individual training with "realistic" datasets at $PW_{50}/T_c = 2.54$ and $PW_{50}/T_c = 2.88$.



(a) $PW_{50}/T_c = 2.54$



(b) $PW_{50}/T_c = 2.88$

Fig. 13: Scenario 3: Joint training with "realistic" datasets for both $PW_{50}/T_c = 2.54$ and $PW_{50}/T_c = 2.88$.

these experiments, the NPML detector designs take into account both colored noise and misequalization errors.

**Experiment 3.1**: The simulation results for a PR-NN detectors trained individually at $PW_{50}/T_c = 2.54$ are shown in Fig. 12a. The NPML detectors with 4-tap, 8-tap, and 16-tap predictors have gains of 0.3dB, 0.4dB and 0.45dB, respectively, over coded E$^2$PRML. The PR-NN detector trained with the "realistic" channel dataset achieves even slightly better performance than the 16-tap NPML detector.

**Experiment 3.2**: In Fig 12b, we consider the channel with density $PW_{50} = 2.88$. Here the gains of the NPML detectors with 4-tap, 8-tap, and 16-tap predictors are 0.5dB, 0.6dB, and 0.7dB, respectively, over coded E$^2$PRML. The PR-NN detector trained with the corresponding dataset surpasses that of the 16-tap NPML detector.

**Experiment 3.3**: As in Experiment 2.4, we explore the adaptability of PR-NN detection to changes in recording density. The results obtained after training with a combined dataset of "realistic" equalized Lorentzian channel outputs for $PW_{50} = 2.54$ and $PW_{50} = 2.88$ are shown in Fig. 12. In Fig. 12a, corresponding to $PW_{50} = 2.54$ , we see that the jointly-trained network essentially matches the performance of the individually trained network (Experiment 3.1), surpassing the 16-tap NPML detector. Similarly, Fig. 12b shows that the jointly-trained PR-NN detector preserves the performance of the network individually trained at $PW_{50} = 2.88$ (Experiment 3.2). The robustness of PR-NN detection in this more realistic channel setting is thus confirmed.

## E. *Experimental Analysis*

Our results from Scenario 1 demonstrate that the PR-NN detection architecture can achieve performance close to Viterbi detection and NPML detection on coded E$^2$PR4 channels in AWGN and ACN, respectively, over a range of SNRs. In Scenario 2, we saw that a PR-NN detector jointly trained for AWGN and ACN shows greater tolerance to ACN than the Viterbi detector, and retains comparable performance in AWGN. When jointly trained in ACN corresponding to equalizers for two different channel densities, the PR-NN detector again exhibits robust performance over a range of SNRs. Finally, when evaluated on a more realistic equalized Lorentzian channel model with both ACN and misequalization errors, the PR-NN detectors designed individually for two channel densities surpass the performance of 16-tap NPML detectors over a range of SNRs. The jointly-trained PR-NN detector maintains the performance of the individually-trained networks at both densities, displaying a robustness that the NPML detectors fail to offer.

The near-optimal performance and robustness of PR-NN can be explained as follows: 1) the RNN-based structure of PR-NN, which exploits the time-sequential connections between cells, reflects the nature of the signal generated by the ISI channel; 2) non-linear functions included in the RNN help PR-NN to model the effects of a variety of noise sources and distortions; and 3) the sliding-window evaluation process helps the block-wise PR-NN architecture to detect the noisy outputs in streaming fashion, in analogy to the classical detection methods used in magnetic recording channels.

## V. Conclusion

In this paper, we first formulated a magnetic recording channel model and reviewed three classical detectors. Then, we proposed PR-NN, an RNN-based detection approach for coded partial-response channel models. The PR-NN detector processes the noisy outputs of the equalized recording channel in a block-streaming fashion, with computational complexity comparable to that of classical sequence detectors. Simulation results confirm the attractive performance of PR-NN when compared to classical detection algorithms and, moreover, demonstrate a robustness to different noise characteristics and channel densities that classical methods can not provide.

## Appendix

We consider three noise scenarios in our experiments: AWGN, ACN and a “realistic” system with equalized noise and misequalization error. The general expression for the noise is given in (6). The signal-to-noise (SNR) ratio is defined in each scenario as follows:

1) AWGN: An ideal channel is assumed and the additive noise term $n_k$ is simply AWGN. The SNR is defined as $SNR_1 = 10\log_{10}(E/\sigma^2)$, where $\sigma^2$ is the variance of the Gaussian random variable $n_k$. Here $E$ is a constant related to the output-voltage amplitude in the recording channel. According to [20], we utilize the matched-filter bound (MFB) as $E$, and we define MFB as the the distance
$$d_{MFB}^2 = ||x(D)||^2. \tag{20}$$
In the E$^2$PR4 channel, $d_{MFB}^2 = 10$.
2) ACN: Misequalization error is ignored and the noise term is $n_k = \sum_i z_i \eta_{k-i}$, where $\{z_i\}$ represent the normalized coefficients of the PR equalizer and $\eta_k$ is AWGN. The SNR is defined as $SNR_2 = 10\log_{10}(E/\sigma^2)$, where $E$ is the MFB and $\sigma^2$ is the variance of $\eta_k$.
3) “Realistic” system: There is misequalization error and colored noise due to equalization. The noise term $n_k$ is given by (6). The SNR is defined as $SNR_3 = 10\log_{10}(E/\sigma^2)$, where $E$ is the MFB and $\sigma^2$ is the variance of $\eta_k$.

## Acknowledgment

The authors would like to acknowledge helpful discussions with Joseph Soriaga. They would also like to thank the reviewers for their many helpful suggestions.

## References

[1] L. R. Bahl, J. Cocke, F. Jelinek, and J. Raviv, “Optimal decoding of linear codes for minimizing symbol error rate,” *IEEE Trans. Inf. Theory*, vol. 20, no. 2, pp. 284-287, Mar. 1974.

[2] R. T. Behrens and A. J. Armstrong, “An advanced read/write channel for magnetic disk storage,” in *Proc. 26th Asdomar Conf. on Signals, Systems, and Computers*, Pacific Grove, CA, USA, Oct. 1992, pp. 956-960.

[3] A. Bennatan, Y. Choukroun, and P. Kisilev, “Deep learning for decoding of linear codes - A syndrome-based approach,” in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Vail, CO, USA, June. 2018.

[4] R. D. Cideciyan, F. Dolivo, R. Hermann, W. Hirt, and W. Schott, “A PRML system for digital magnetic recording,” *IEEE J. Select. Areas Commun.*, vol. 10, pp. 38-56, Jan. 1992.

[5] K. Cho, B. V. Merrienboer, C. Gulcehre, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using RNN encoder-decoder for statistical machine translation,” in *Proceedings of the Empirical Methods in Natural Language Processing (EMNLP)*, Doha, Qatar, Oct. 2014.

[6] K. Cho, B. V. Merrienboer, D. Bahdanau, and Y. Bengio, “On the properties of neural machine translation: Encoder-decoder approaches,” *arXiv:1409.1259*, Sep. 2014.

[7] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv:1406.1078*, Dec. 2014.

[8] J. D. Coker, E. Eleftheriou, R. L. Galbraith, and W. Hirt, “Noise-predictive maximum likelihood (NPML) detection,” *IEEE Trans. Magn.*, vol. 34, no. 1, pp.110-117, Jan. 1998.

[9] N. Farsad and A. J. Goldsmith, “Neural network detection of data sequences in communication systems,” *IEEE Trans. Signal Process.*, vol. 66, pp. 5663-5678, Nov. 2018.

[10] G. P. Fettweis and H. Meyr, “High-speed parallel Viterbi decoding: algorithm and VLSI-architecture,” *IEEE Commun. Magazine*, vol.29, no.5, pp. 46-55, 1991.

[11] G. P. Fettweis, R. Karabed, P. H. Siegel, and H. K. Thapar, “Method and means for detecting partial response waveforms using a modified dynamic programming heuristic,” U.S. Patent 5 430 744, Jul. 1995.

[12] G. D. Forney Jr, “Convolutional codes II. Maximum-likelihood decoding,” *Information and Control*, vol.25, no.3, pp. 222-266, 1974.

[13] T. Gruber, S. Cammerer, J. Hoydis, and S. T. Brink, “On deep learning-based channel decoding,” in *Proc. IEEE 51st Annu. Conf. Inf. Sci. Syst. (CISS)*, Baltimore, MD, USA, Mar. 2017, pp. 1-6.

[14] F. Hemmati and D. J. Costello, “Truncation error probability in Viterbi decoding,” *IEEE Trans. Commun.*, vol. 25, no. 5, pp. 530-532, May. 1977.

[15] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, Dec, 1997.

[16] K. A. S. Immink, P. H. Siegel, and J. K. Wolf, “Codes for digital recorders,” *IEEE Trans. Inf. Theory*, vol. 44, no. 6, pp. 2260-2299, Oct. 1998.

[17] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, “DeepTurbo: Deep turbo decoder,” in *IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Cannes, France, July 2019, pp. 1-5.

[18] R. Karabed and N. Nazari, “Analysis of error sequences for PRML and EPRML signaling performed over Lorentzian channel,” in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM)*, London, U.K., Nov. 1996, pp. 368-373.

[19] R. Karabed and P. H. Siegel, “Matched spectral-null codes for partial-response channels,” *IEEE Trans. Inf. Theory*, vol. 37, no. 3, pp. 818-855, May. 1991.

[20] R. Karabed, P. H. Siegel, and E. Soljanin, “Constrained coding for binary channels with high intersymbol interference,” *IEEE Trans. Inf. Theory*, vol. 45, no. 6, pp. 1777-1797, Sep. 1999.

[21] H. Kim, Y. Jiang, S. Kannan, S. Oh, and P. Viswanath, “Deepcode: Feedback codes via deep learning,” in *Advances in Neural Information Processing Systems (NeurIPS)*, Montréal, Canada, Dec. 2018, pp. 9436-9446.

[22] D. P. Kingma and J. L. Bai, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Representation (ICLR)*, San Diego, CA, USA, Dec. 2015.

[23] M. Lian, F. Carpi, C. Häger, and H. D. Pfister, "Learned belief-propagation decoding with simple scaling and SNR adaptation," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Paris, France, June, 2019.

[24] B. H. Marcus, R. M. Roth, and P. H. Siegel, "An introduction to coding for constrained systems," Oct. 2001. [Online]. Available: https://www.math.ubc.ca/~marcus/Handbook/.

[25] R. J. McEliece and I. M. Onyszchuk, "Truncation effects in Viterbi decoding," in *Proc. IEEE Military Communications Conference (MILCOM)'89*, Boston, MA, USA, Oct. 1989.

[26] B. E. Moision, "A truncation depth rule of thumb for convolutional codes," in *Proc. Information Theory and Applications Workshop (ITA)*, San Diego, CA, USA, Jan. 2008, pp. 555-557.

[27] B. E. Moision, "Constrained coding and detection for magnetic recording channels," Ph.D. Thesis, University of California, San Diego, La Jolla, CA, 2000.

[28] B. E. Moision and P. H. Siegel, "Distance enhancing constraints for noise predictive maximum likelihood detectors," in *Proc. IEEE Global Telecommunications Conf. (GLOBECOM)*, Sydney, Australia, Nov. 1998, pp. 2730-2735.

[29] B. E. Moision, P. H. Siegel, and E. Soljauin, "Distance-enhancing codes for digital recording," *IEEE Trans. Inf. Theory*, vol. 34, no. 1, pp. 69-74, Jan, 1998.

[30] J. Moon and W. Zeng, "Equalizer for maximum likelihood detectors," *IEEE Trans. Magn.*, vol.31, no.2, pp.1083-1088, Mar. 1995.

[31] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Proc. 54th Annu. Allerton Conf. Commun., Control, Comput.*, Monticello, IL, USA, Sep. 2016, pp. 341-346.

[32] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," *arXiv:2003.01998*, Mar. 2020.

[33] M. Schuster and K.K. Paliwal, "Bidirectional recurrent neural networks," *Signal Processing, IEEE Trans. on*, vol. 45, no. 11, pp. 2673-2681, Nov. 1997.

[34] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Data-driven factor graphs for deep symbol detection," in *Proc. IEEE Int. Symp. Information Theory (ISIT)*, Los Angeles, CA, USA, June, 2020.

[35] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: Symbol detection using a deep learning based Viterbi algorithm," in *IEEE 20th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Cannes, France, July 2019, pp. 3319-3331.

[36] D. Tandler, S. Dörner, S. Cammerer, and S. T. Brink, "On recurrent neural networks for sequence-based processing in communications," *arXiv:1905.09983v3*, Nov. 2019.

[37] H. K. Thapar and A. M. Patel, "A class of partial response systems for increasing storage density in magnetic recording," *IEEE Trans. Magn.*, vol. 23, no. 5, pp.3666-3668, Sep. 1987.

[38] B. Vasic and E. Kurtas, *Coding and Signal Processing for Magnetic Recording Systems*. Boca Raton, FL: CRC, 2005.

[39] A. J. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. 13, no. 2, pp. 260-269, Apr. 1967.

[40] A. J. Viterbi, "An intuitive justification and a simplified implementation of the MAP decoder for convolutional codes," *IEEE J. Sel. Areas Commun.*, vol. 16, no. 2, pp. 260-264, Feb. 1998.

[41] A. D. Weathers and J. K. Wolf, "A new rate 2/3 sliding block code for the (1,7) runlength constraint with the minimal number of encoder states," *IEEE Trans. Inf. Theory*, vol. 37, no. 3, pp. 908-913, May. 1991.

[42] Z. Wu, "Channel modeling, signal processing and coding for perpendicular magnetic recording," Ph.D. Thesis, University of California, San Diego, La Jolla, CA, 2009.

[43] N. Zheng, J. Li, S. Dahandeh, and T. Zhang, "Self-directed equalization for magnetic recording channels with multi-sensor read head," *IEEE Trans. Magn.*, vol. 52, no. 1, Jan. 2016.

**Simeng Zheng** (S'20) received the B.E degree in Electronic and Information Engineering from Beihang University, Beijing, China, in 2018 and the M.S. degree in Electrical and Computer Engineering from University of California, San Diego, CA, USA, in 2020. He is currently working toward the Ph.D. degree with the department of Electrical and Computer Engineering, University of California, San Diego, where he is also affiliated with the Center for Memory and Recording Research.

**Yi Liu** (S'16) received the B.S. degree in physics from Peking University, Beijing, China, in 2014, and the Ph.D. degree in electrical engineering from the University of California San Diego, CA, USA, in 2020. From 2015 to 2020, he was associated with the Center for Memory and Recording Research. He is currently with Apple Inc.

**Paul H. Siegel** (M'82–SM'90–F'97–LF'19) received the S.B. and Ph.D. degrees in mathematics from the Massachusetts Institute of Technology, Cambridge, MA, USA, in 1975 and 1979, respectively. He held a Chaim Weizmann Postdoctoral Fellowship with the Courant Institute, New York University, New York, NY, USA. He was with the IBM Research Division, San Jose, CA, USA, from 1980 to 1995. He joined the faculty at University of California, San Diego, CA, USA, in 1995, where he is currently a Distinguished Professor of Electrical and Computer Engineering with the Jacobs School of Engineering. He is affiliated with the Center for Memory and Recording Research where he holds an Endowed Chair and served as Director from 2000 to 2011. His research interests include information theory and communications, particularly coding and modulation techniques, with applications to digital data storage and transmission. He is a Member of the National Academy of Engineering. He was a Member of the Board of Governors of the IEEE Information Theory Society from 1991 to 1996 and from 2009 to 2014. He was the 2015 Padovani Lecturer of the IEEE Information Theory Society. He was a recipient of the 2007 Best Paper Award in Signal Processing and Coding for Data Storage from the Data Storage Technical Committee of the IEEE Communications Society. He was the co-recipient of the 1992 IEEE Information Theory Society Paper Award and the 1993 IEEE Communications Society Leonard G. Abraham Prize Paper Award. He served as a Co-Guest Editor of the 1991 Special Issue on Coding for Storage Devices of the IEEE Transactions on Information Theory. He served as an Associate Editor of Coding Techniques of the IEEE Transactions on Information Theory from 1992 to 1995, and as the Editor-in-Chief from 2001 to 2004. He was also a Co-Guest Editor of the 2001 two-part issue on The Turbo Principle: From Theory to Practice and the 2016 issue on Recent Advances in Capacity Approaching Codes of the IEEE Journal on Selected Areas in Communications.