Deep Neural Network Media Noise Predictor Turbo-Detection System for 1-D and 2-D High-Density Magnetic Recording

Amirhossein Sayyafan[1], Ahmed Aboutaleb[1], Benjamin J. Belzer[1], Krishnamoorthy Sivakumar[1], Anthony Aguilar[1], Christopher Austin Pinkham[1], Kheong Sann Chan[2], and Ashish James[3]

[1]School of Electrical Engineering and Computer Science, Washington State University, Pullman, WA 99164-2752 USA
[2]Nanjing Institute of Technology, Nanjing 211167, China
[3]Institute for Infocomm Research (I2R), A*STAR, Singapore

This article presents a concatenated Bahl–Cocke–Jelinek–Raviv (BCJR) detector, low-density parity-check (LDPC) decoder, and deep neural network (DNN) architecture for a turbo-detection system for 1-D and 2-D magnetic recording (1DMR and TDMR). The input readings first are fed to a partial response (PR) equalizer. Two types of the equalizer are investigated: a linear filter equalizer with a 1-D/2-D PR target and a convolutional neural network (CNN) PR equalizer that is proposed in this work. The equalized inputs are passed to the BCJR to generate the log-likelihood-ratio (LLR) outputs. We input the BCJR LLRs to a CNN noise predictor to predict the signal-dependent media noise. Two different CNN interfaces with the channel decoder are evaluated for TDMR. Then, the second pass of the BCJR is provided with the estimated media noise, and it feeds its output to the LDPC decoder. The system exchanges LLRs between BCJR, LDPC, and CNN iteratively to achieve higher areal density. The simulation results are performed on a grain flipping probabilistic (GFP) model with 11.4 Teragrains per square inch (Tg/in^2). For the GFP data with 18 nm track pitch (TP) and 11 nm bit length (BL), the proposed method for TDMR achieves 27.78% areal density gain over the 1-D pattern-dependent noise prediction (PDNP). The presented BCJR-LDPC-CNN turbo-detection system obtains 3.877 Terabits per square inch (Tb/in^2) areal density for 11.4 Tg/in^2 GFP model data, which is among the highest areal densities reported to date.

Index Terms— **2-D magnetic recording (TDMR), Bahl–Cocke–Jelinek–Raviv (BCJR) detectors, convolutional neural network (CNN), CNN equalizer, deep neural network (DNN), low-density parity-check (LDPC) decoder, turbo-detectors.**

I. Introduction

THE hard disk drive (HDD) industry is facing a physical limit on the areal density of 1-D magnetic recording (1DMR) on traditional magnetic media. To increase capacity without media redesign, the 2-D magnetic recording has been introduced (TDMR) [1]. The trellis-based detection with the pattern-dependent noise prediction (PDNP) is standard practice for HDD magnetic recording [2], [3]. In typical single-track signal processing, the received samples from the read head are filtered by a linear equalizer with a 1-D partial response (PR) target $\mathbf{h}$. The effective channel model has a media noise term that models signal-dependent noise due to, e.g., magnetic grains intersected by bit boundaries. The equalizer output $\mathbf{y}$ flows into a trellis-based (Viterbi [4] or Bahl–Cocke–Jelinek–Raviv (BCJR) [5]) detector that employs a super-trellis based on the effective intersymbol interference (ISI) channel and a 1-D PDNP algorithm. The trellis detector sends soft coded bit estimates to a channel decoder, which outputs user information bit estimates. PDNP is based on an Lth-order trained autoregressive (AR) media noise model $\tilde{n}_{m_k}$

$$\tilde{n}_{m_k}(\mathbf{u}_k) = \sum_{i=1}^{L} a_i(\mathbf{u}_k) n_{m_{k-i}}(\mathbf{u}_k) + e_k(\mathbf{u}_k) \tag{1}$$

where a_i are the AR coefficients, and the model error e_k is assumed to be uncorrelated Gaussian noise that depends on the coded bit pattern vector $\mathbf{u}_k = [u_{k+\Delta}, \ldots, u_k, \ldots, u_{k-(I+L)}]$ [2], [3], [6]. In proposed generalizations to 2-D PDNP for two-reader TDMR, the trellis state cardinality becomes $4^{(\Delta+I+L)}$, where Δ is the predictor look-ahead, and I and L are the ISI channel length and predictor order. The complexity grows exponentially with $I + L$, and becomes impractical for more than two readers (e.g., [7]–[10])

The PDNP model is somewhat restrictive and may not accurately represent the media noise, especially at high storage densities; this is the *modeling problem*. To address this problem, in [11], we separate the ISI detection and media noise estimation, and also we design and train deep neural network (DNN)-based media noise predictors. Recent breakthroughs in DNNs [12], [13] have led to great success in applications such as speech recognition, image understanding, and language translation. The convolutional neural networks (CNNs) [14] have a nice performance on spatially correlated data, image recognition, and computer vision applications. DNNs [12], [13] are capable of modeling non-linear relationships between the media noise and data bits; on the other hand, the AR model employs a linear model. In this case, DNN models are much more general than AR models, and they more accurately model media noise compared with PDNP. In [11], we proposed a BCJR-DNN turbo-detector for 1DMR, without low-density parity-check (LDPC) decoding.

In the Appendix, we give a mathematical argument using a simplified model, showing that a magnetic recording system is non-linear, and the media noise is a non-linear function

Manuscript received August 4, 2020; revised October 6, 2020 and October 17, 2020; accepted November 5, 2020. Date of publication November 17, 2020; date of current version February 18, 2021. Corresponding author: B. J. Belzer (e-mail: belzer@wsu.edu).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TMAG.2020.3038419.

Digital Object Identifier 10.1109/TMAG.2020.3038419

of the data bits. We also provide some further arguments as to why the DNN should do a better job of estimating the media noise than PDNP. An insight into the *state explosion problem*, which is the impractical number of trellis states that result when more than one track is processed simultaneously, e.g., TDMR, is also presented in the Appendix.

Several recent works have investigated neural network (NN) and DNN for equalization and detection on magnetic recording channels. Luo *et al.* [15] investigate an NN equalizer (NNE) to eliminate the effect of ISI and the intertrack interference (ITI) on TDMR combined with the shingled magnetic recording (SMR). The NNE is performed with LDPC coding based on a random Voronoi grain media model. The article compares this method with the conventional 2-D linear equalizer that was followed by the *a posteriori* probability (APP) detector and a sum-product (SP) decoder.

Nishikawa *et al.* [16] propose an NN log-likelihood-ratio (LLR) modulator using an LDPC code in iterative decoding to mitigate the pattern dependent medium noise influence. The article shows that iterative decoding performance can be improved by considering LLRs of adjacent track bits as the input of the NN. A hybrid genetic algorithm (HGA) is used to select the adjacent track LLRs that affect a given decoded bit.

Qin and Zhu [17] investigate a fully connected DNN (FCDNN) detector to recover the data on high user bit density HDDs. The prominent magnetic transition jitter noise is explored in the article. The un-equalized readback signals are input to the DNN to learn the correlations between the input signals and the impact from the noise. The article finds that DNN can adapt to the ISI and is resilient against the colored magnetic noise.

In [18], an NN-based nonlinear equalizer for TDMR is investigated. The authors also present adaptation criteria to equalize the TDMR waveforms based on the cross entropy (CE) between the bits' true probabilities and detector estimations. The CE criteria obtain a lower bit error rate (BER) than the typical mean-square-error (MSE) adaptation criteria for the channel detector. The results in [18] show that the NN equalizer achieves better performance compared with the minimum MSE (MMSE) linear equalizer for both CE and MSE criteria.

Shen *et al.* [19] design DNN-based detection systems for high-density TDMR systems. The design in [19] eliminates trellis processing by replacing both the BCJR and PDNP with a single DNN that directly estimates the coded bits for TDMR. In comparison with a conventional system consisting of a 2-D BCJR and a 2-D PDNP, a 2-D linear MMSE equalizer followed by a CNN detector achieves a higher information areal density gain with lower computational costs.

In this article, we present a BCJR-LDPC-CNN turbo-detection system for 1DMR and TDMR. In [11], we designed the architecture only for 1DMR; we did not use any LDPC decoding and did not explore multiple iterations for the turbo-detection system. This article employs channel decoding with both 1DMR and TDMR cases. In addition, we also investigate using CNN equalizers and the standard linear PR equalizer, and we demonstrate the benefits of a second turbo-iteration between the detector and the channel decoder.

The main novel contributions of this work are presented as follows.

1) We employ channel decoding in the 1DMR turbo-detection system and show that the method proposed in Fig. 3 achieves a 1.60% density gain over 1-D PDNP.
2) We generalize the 1DMR architecture to TDMR. This approach achieves a density gain of 27.78% over the 1-D PDNP baseline system.
3) For TDMR, we investigate using one CNN media noise estimator for all three data tracks, versus using one CNN per data track, and show that the latter scheme gives higher areal density.
4) We explore the second iteration of turbo-detector for both 1DMR and TDMR cases. For 1DMR, the second iteration has a density gain of 2.02% over 1-D PDNP on one data set.
5) We design a CNN PR equalizer and show that it achieves 3.877 Terabits per square inch ($\mathrm{Tb/in^2}$) areal density for 11.4 Teragrains per square inch ($\mathrm{Tg/in^2}$) grain flipping probabilistic (GFP) data, which is among the highest reported areal densities to date.

II. System Model

The BCJR-LDPC-CNN turbo-detector assumes a channel model for the kth linear equalizer filter output $y(k)$ similar to that of the 1-D PDNP scheme described in Section I

$$y(k) = (\mathbf{h} * \mathbf{u})(k) + n_m(k) + n_e(k) \tag{2}$$

where $\mathbf{h}$ is the PR target, $\mathbf{u}$ are the coded bits on the track, * indicates 1-D/2-D convolution, $n_m(k)$ is media noise, and $n_e(k)$ is reader electronics additive white Gaussian noise (AWGN). Unlike PDNP, the media noise term $n_m(k)$ is not modeled as an AR process; instead, a more general model for $n_m(k)$ is learned by the CNN through off-line training.

We use the GFP model data to train and evaluate our system. The GFP waveforms are generated based on micro-magnetic simulations [20]. The simulated media have a grain density of 11.4 $\mathrm{Tg/in^2}$. The GFP waveforms correspond to five tracks of coded bits (± 1), denoted as tracks 0 through 4. They are written using shingled writing technology. Fig. 1 represents a capture of the GFP model readback signal. Bit regions are not rectangular, but rather curved stripes due to the relative orientation of the corner write head. The blue and red stripes represent -1 and $+1$ coded bits. Track 0 at the bottom is written first. Then, track 1 is written, overlapping part of track 0. The writing process repeats until track 4 is written. Track 4 is called the fat track since it is not followed by any more tracks and, thus, preserves the original magnetic write width (MWW), which is 75 nm. In the GFP simulations, the three central tracks have available readback signals, and the coded bits for tracks 0 and 4 are known boundary bits. The bit length (BL) of GFP data sets is equal to 11 nm.

We have four GFP data sets for the system evaluation: two data sets for the 1DMR system and the two others for the

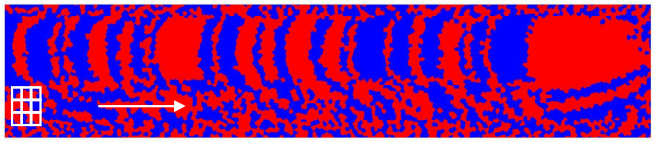

Fig. 1. Capture of the GFP model readback signal with TP of 18 nm and BL of 11 nm.

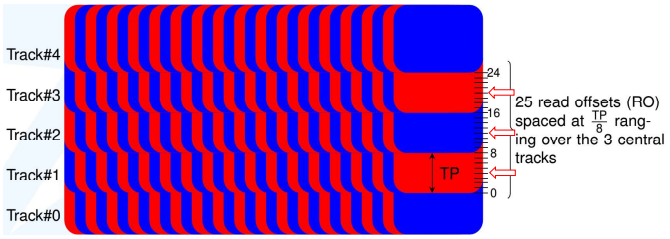

Fig. 2. Cartoon representation for the GFP model.

TDMR system. For the first data set, the track pitch (TP) (i.e., the distance between adjacent tracks) is 48 nm, and for the second, the TP is 27 nm. These two data sets are used for evaluating the 1DMR system. For the TDMR system, we use two data sets with 18 and 15 nm TPs. The number of grains per coded bit (GPB) for the 48 nm TP data set is

$$\begin{aligned}\text{GPB} &= \text{Grain density} \times \text{BL} \times \text{TP} \\ &= 11.4\ \text{Tg/in}^2 \times 11\,\text{nm} \times 48\,\text{nm} \\ &\quad \times (3.937 \times 10^{-8}\ \text{in/nm})^2 = 9.33. \end{aligned} \tag{3}$$

Similarly, for the 27, 18, and 15 nm TP data sets, we compute GPB as 5.25, 3.50, and 2.92, respectively.

As shown in Fig. 2, there are 25 read offsets spaced at TP/8 ranging over tracks 1, 2, and 3. To sample the data on these three central tracks, we use reader positions at 4, 12, and 20, approximately the center of each track. Each track in GFP data sets consists of 41 206 coded bits. The track sizes in GFP data sets are close to the sector size of 32 768 bits (4k bytes) in a typical HDD. The GFP data sets have 100 blocks (sectors) for each TP. For the 1DMR system, the readings from the center of Track #2 are used as input to the BCJR-LDPC-CNN turbo-detector and, for comparison, the baseline 1-D PDNP detector. The readings from Track #1 through #3 are inputs to the TDMR system.

The shingled writing process introduces ITI. As the MWW is fixed, a smaller TP results in greater ITI. Compared with typical commercial HDDs with TP = 48 nm, the GFP model data with lower TP used in this article suffer from rather severe ITI. We employ a PR equalizer on the GFP waveforms to reduce the effect of the ITI and the down-track ISI.

III. BCJR-LDPC-CNN Turbo-Detection System

In this system, we separate the ISI detection and media-noise prediction functions into two detectors in a turbo-equalization structure. The LLR estimates of coded bits and noise samples are exchanged iteratively between a BCJR detector, an LDPC channel decoder, and the CNN media noise predictor until BER converges to a low value. In [11], we proposed a BCJR-DNN turbo-detector for 1DMR, without LDPC decoding. In this work, we employ an LDPC channel decoder to generate the final LLRs at the end of each turbo-iteration. The areal density is determined by increasing the LDPC code rate until the decoded BER is $\leq 10^{-5}$.

A. 1DMR Turbo-Detector

In Fig. 3, the GFP simulated HDD read-head output vector $\mathbf{r}$ contains two samples per coded bit, denoted $\mathbf{r}^{(1)}$ and $\mathbf{r}^{(2)}$. These samples are on the same track and are collected by the same read head but are located at different down-track locations within a given bit; the odd samples $\mathbf{r}^{(1)}$ (the "first samples" per bit) are located near the center of each bit; and the even samples $\mathbf{r}^{(2)}$ are located at the boundary between bits. The odd samples $\mathbf{r}^{(1)}$ are first filtered by a length 15 1-D linear equalizer designed to minimize the mean squared error between the filter output $\mathbf{y}^{(1)}$ and the convolution of the coded bits $\mathbf{u}$ with the 1-D PR target mask $\mathbf{h}$. This PR equalization is done because the down-track ISI can have a span of up to about 15 bits. The filter output $\mathbf{y}^{(1)}$ is input to a trellis detector (a BCJR detector in this work), which handles only ISI equalization based on the PR target $\mathbf{h}$ and outputs a block of 41 206 coded bit LLRs. In this work, we design the PR target $\mathbf{h}$ with three taps so that the ISI channel length $I = 2$, and the BCJR detector has $M = 2^I = 4$ states and eight total branches.

In the first iteration, the BCJR LLRs $\mathbf{LLR}_{b_0}$ and $\mathbf{y}^{(1)}$, and the unfiltered even samples $\mathbf{r}^{(2)}$ are passed to the CNN to estimate the media noise $\hat{\mathbf{n}}_m$. The noise $\hat{\mathbf{n}}_m$ is fed back to the BCJR to obtain a lower BER. Next, the BCJR passes LLRs, $\mathbf{LLR}_b$, which are magnitude-limited and scaled by T_1 and W_1, respectively, to make the channel decoder converge faster. After this pre-processing, the LLRs are input to the channel decoder through the de-interleaver. The de-interleaver shuffles the coded bits in order to decorrelate them before they are input to the channel decoder. This shuffling is important because the channel decoder's SP algorithm assumes that the incoming LLRs are statistically independent.

The channel decoder can provide estimates $\mathbf{LLR}_l$ of the coded bits as *a priori* inputs to the BCJR detector to improve that detector's estimates $\mathbf{LLR}_b$ and to the CNN to improve the CNN's noise estimates $\hat{\mathbf{n}}_m$. In cases where $\mathbf{LLR}_l$ is provided to both the BCJR and the CNN, $\mathbf{LLR}_l$ would be subtracted from the BCJR's initial (intrinsic) set of LLR estimates in order to make $\mathbf{LLR}_b$ extrinsic information relative to $\mathbf{LLR}_l$ so that the CNN could make optimal use of both $\mathbf{LLR}_b$ and $\mathbf{LLR}_l$.

We use an irregular repeat accumulate (IRA) LDPC decoder as the channel decoder [21]. Henceforth, we refer to the "IRA decoder" or simply "IRA" to indicate the specific LDPC decoder employed in this article. The IRA decoder employs coset decoding since the GFP data bits are randomly distributed. The IRA encoding and decoding are done separately for each track; hence, different code rates can be used on each of the three tracks.

The channel decoder produces the extrinsic LLRs, then scales and magnitude limits them by W_2 and T_2, respectively. After limiting and scaling, they are passed through an

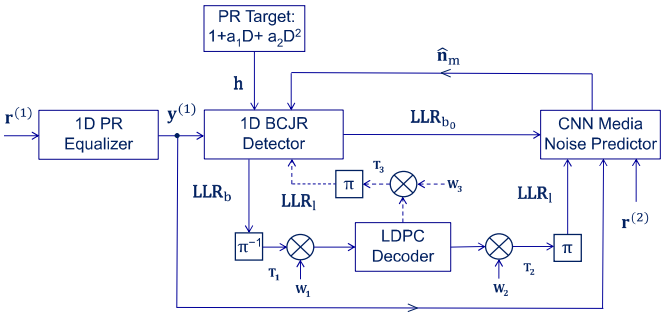

Fig. 3. Block diagram for the 1DMR turbo-detector.

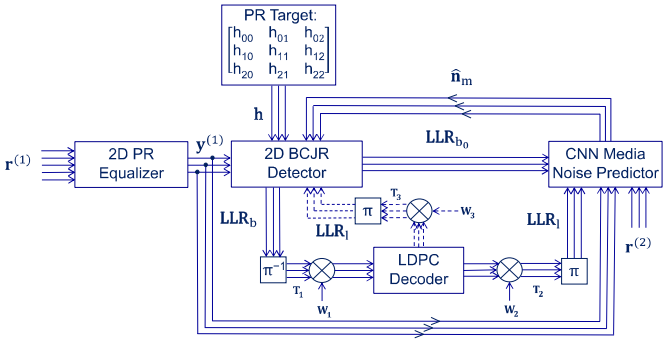

Fig. 4. Block diagram for the TDMR turbo-detector.

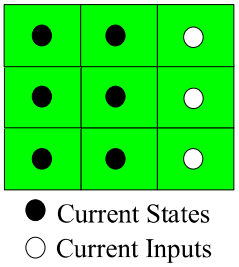

Fig. 5. State-input block for 2-D BCJR with two state bits per track.

interleaver to reorder the LLRs to be consistent with the order of $\mathbf{LLR}_b$. The final LLRs $\mathbf{LLR}_l$ are generated by passing the extrinsic LLRs through an interleaver at the end of each turbo-iteration.

For the second iteration, the decoder LLRs $\mathbf{LLR}_l$ are passed as inputs to the CNN instead of $\mathbf{LLR}_b$. Alternatively, the CNN can be provided with both $\mathbf{LLR}_b$ and $\mathbf{LLR}_l$, which we have found to slightly improve the performance of the CNN. However, the simulation results in this work consider replacing $\mathbf{LLR}_b$ with $\mathbf{LLR}_l$.

The dotted lines, multiplier, and interleaver in Fig. 3 indicate optional inner iterations between the BCJR and the channel decoder in order to reduce the BER and improve the quality of LLRs before passing them to the CNN. For each such inner iteration, the weight W_3 and limit threshold T_3 are applied on the channel decoder LLRs output to accelerate the BCJR convergence. More iterations between the BCJR and the channel decoder can result in a better quality of LLRs and potentially allow the use of a lower complexity CNN.

B. TDMR Turbo-Detector

We generalize the 1DMR system to the three-track TDMR system. Fig. 4 shows the BCJR-LDPC-CNN turbo-detector for TDMR with separate trellis-based ISI/ITI detection and CNN-based media-noise prediction.

The 2-D PR equalizer employs a 3×3 PR target $\mathbf{h}$ and accepts five adjacent tracks. In our data sets, the three inner tracks have unequalized readings $\mathbf{r}^{(1)}$, and the two outer tracks have only known data bits, but no readings. In practice, data bits on the outer tracks can be estimated by a relatively simple 1-D detection scheme, resulting in a non-zero BER. Previous articles by our group (e.g., [22]) have shown that outer track BERs reduce the achieved information densities on the three inner tracks by a relatively small percentage (around 7%) when outer track detection without channel decoding is performed.

The ITI affects the system design when generalizing a 1DMR system to a 2-D detector system. 2-D BCJR is a joint ISI/ITI equalizer; the state-input block has three rows because the ITI typically extends over one adjacent track on either side. In 2-D BCJR, the processing of three tracks is done simultaneously to handle the ITI from a 3×3 PR target mask since the central track is affected by the ITI from its two neighboring tracks. Fig. 5 represents the state-input block for 2-D BCJR over three tracks. The 2-D BCJR trellis detector performs ISI/ITI equalization on filtered input $\mathbf{y}^{(1)}$ and generates LLR outputs. The PR target $\mathbf{h}$ is 3×3; hence, the system has two state bits per track, and the 2-D BCJR state bits are 3×2, so the trellis has $2^{3\times 2} = 64$ states.

IV. CNN Noise Predictor Architecture and Interface

We investigate the CNN architectures for the noise predictor for 1DMR and TDMR Systems. The CNN extracts correlations among the data and exploits them to obtain the information and features. For 1DMR, we stack the data as the 2-D input images; hence, we design 2-D CNNs to predict media noise. In the TDMR detector, we stack the 2-D input images for three tracks to make a 3-D input image. Therefore, for processing the 3-D input image, we design a 3-D CNN architecture to estimate the media noise for the TDMR system.

The CNNs process their input data in a sliding block manner. For 1DMR, in the first iteration, to estimate the kth media noise sample $\hat{n}_{m_k}$, the lowest input layer of the CNN accepts a block $\mathbf{LLR}_{b_{0_k}}$ of N_i BCJR output LLRs, N_i filtered readings $\mathbf{y}_k^{(1)}$, and N_i raw second readings $\mathbf{r}_k^{(2)}$, where N_i is an odd number, and the kth noise estimate corresponds to the middle element of the N_i elements in each block; in this article, $N_i = 9$. To estimate the $(k+1)$th media noise sample, each of the input data blocks is shifted by exactly one sample into the future. For the second iteration, the CNN is provided with a block $\mathbf{LLR}_{l_k}$ of N_i channel decoder output LLRs instead of the $\mathbf{LLR}_{b_{0_k}}$ LLRs from the BCJR, and the block width N_i increases to 11.

The TDMR system processes three tracks at a time. Hence, for TDMR, each of the above-described 1-D input blocks of size $[1 \times N_i]$ becomes a $[3 \times N_i]$ input block.

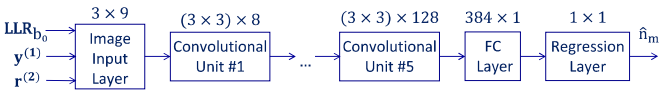

Fig. 6. 2-D CNN architecture of media noise predictor for 1DMR.

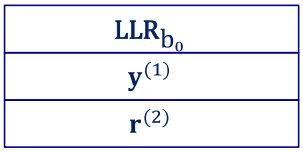

Fig. 7. 2-D input image.

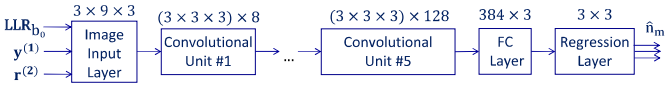

Fig. 8. 3-D CNN architecture of media noise predictor for TDMR.

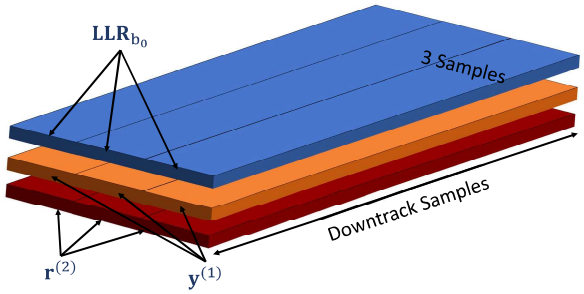

Fig. 9. 3-D input image.

A. CNN Architecture

The proposed CNNs for both 1DMR and TDMR contain 18 layers. These layers consist of one input image layer, five convolutional units, and one output layer. After normalizing the raw data received from the other blocks to have zero mean and unit variance, the system passes them to the input image layer.

Every convolutional unit includes three layers: convolutional layer, batch normalization layer, and rectified linear unit (ReLU) layer. The convolutional layer slides the filter over the input data, and the batch normalization layer normalizes the data to speed up network training and reduce sensitivity to the initial conditions (of the filter coefficients and interconnection weights) in the layers. The ReLU activation function layer assists the model to converge with greater acceleration. The output layer is a regression layer. Every convolutional layer has three properties: the filter length, the filter width, and the number of filters, which is called the number of channels. For the first turbo-iteration, the length of the input image is nine. For the second iteration, we increase the length of the input image to 11 in order to extract more information from the data. Also, we add another convolutional unit to the end of the network whose number of channels is larger than that of any of the previous convolutional units.

Every node in each fully connected (FC) layer is connected to all the nodes in the previous layer. The output of the last convolutional unit is multiplied by a weight matrix, and then, a bias vector is added to it to form the output of the FC layer. The last layer is the regression layer, which predicts the responses of the model. The regression loss function is $0.5\times$ the mean squared error between the training label media noise and the CNN prediction of the media noise.

1) CNN Noise Predictor for 1DMR: The 2-D CNN architecture for the first iteration of the 1DMR system is shown in Fig. 6. In the first iteration, the 2-D input image layer is of size 27 and includes three rows consisting of nine samples from each of the three input blocks $\mathbf{LLR}_b$, $\mathbf{y}^{(1)}$, and $\mathbf{r}^{(2)}$. Organizing the 1-D input blocks into a 2-D array in this manner induces a 2-D spatial correlation between the blocks. Fig. 7 shows the 2-D input image of CNN for 1DMR. We exploit this spatial correlation by employing trained 2-D convolutional filters on all CNN layers. In CNNs designed for 1DMR, all convolutional layers employ the filters of size $[3 \times 3]$. The number of channels in units 1 through 5 is equal to 8, 16, 32, 64, and 128, respectively, for the first iteration. In the second iteration, the length of the input increases to 11; thus, the input image layer size would be 33. Also, an additional convolutional unit with 256 channels is added at the end of the network before the FC layer.

2) CNN Noise Predictor for TDMR: For the TDMR detector, we design the 3-D CNN to predict the media noise. Fig. 8 represents the 3-D CNN architecture for the first iteration of the system. For the first iteration, each of the three input layer 2-D images contains $3 \times 9 = 27$ samples; thus, the size of the CNN's 3-D input layer is $3 \times 27 = 81$ samples. The 3-D input image contains the 2-D input images for the three tracks. By putting the three 2-D input images together, we can extract the information from the 3-D spatial correlation between the input images and the correlation between the three tracks of each block simultaneously. In Fig. 9, stacking of the 2-D input images (with three tracks each) into a 3-D input image is shown. In the designed CNNs for TDMR, the convolutional layers have the filters with the size of $[3 \times 3 \times 3]$. Similar to the 2-D CNN architecture, the number of channels through convolutional units 1 to 5 is 8, 16, 32, 64, and 128. For the second iteration, we increase the length of the input image to 11 to have 99 nodes in the 3-D input image. Also, we have another convolutional unit with 256 channels at the end of the network.

B. CNN Noise Predictor Interface With the Channel Decoder

For the TDMR system, we investigate two approaches to interface the BCJR and channel decoder with CNN. In the first approach (labeled as "1 CNN" in Table II), we use one CNN to estimate the media noise $\hat{n}_{m_j}$, $1 \leq j \leq 3$, for the three tracks. The output of the regression layer contains three bit sequences representing the predicted media noise of the three tracks. Fig. 8 shows the architecture for "1 CNN" interface with the decoder. The second approach (labeled as "3 CNNs" in Table II) predicts $\hat{n}_{m_j}$, $1 \leq j \leq 3$, using a separate CNN for each track. In this interface, the output of the regression layer of each CNN includes one bit sequence for each track.

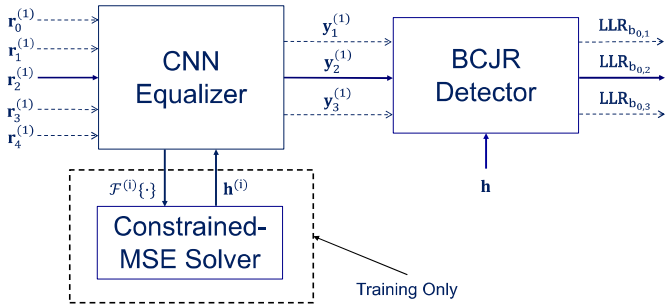

Fig. 10. Block diagram for CNN equalizer. The solid arrows indicate the data flow for 1DMR system. The dotted arrows represent the additional data flows considered for TDMR.

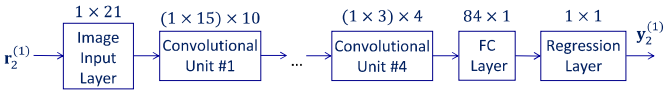

Fig. 11. CNN PR equalizer architecture for 1DMR.

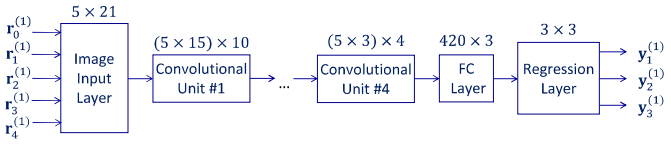

Fig. 12. CNN PR equalizer architecture for TDMR.

By employing three CNNs, we expand the search space to improve the estimation of the media noise.

V. CNN PR Equalizer Design

We investigate the design of CNN-based PR equalizers to be used as the 1-D and 2-D PR equalizers in Figs. 3 and 4, instead of the standard linear PR equalizers. The CNN equalizer architecture is represented in Fig. 10.

A. Nonlinear CNN Equalization System

The 1-D/2-D linear PR equalizer minimizes the MSE between ideal PR signals and the actual output of the equalizer. The ideal PR signals $\mathbf{y}_{id}^{(1)}$ are based on the ideal convolutional model and are given by

$$\mathbf{y}_{id}^{(1)}(k) = (\mathbf{h} * \mathbf{u})(k) \tag{4}$$

where * indicates the 1-D/2-D convolution, $\mathbf{h}$ is target mask of the appropriate length that is adapted during training, and $\mathbf{u}$ are the bit sequences for the track(s).

By considering the equalization of a signal

$$\mathbf{r}^{(1)} = \mathbf{h} * \mathbf{u} \tag{5}$$

applying a linear equalizer $\mathbf{f}$ to $\mathbf{r}^{(1)}$ gives

$$\mathbf{y}_{\text{linear}}^{(1)} = \mathbf{f} * \mathbf{r}^{(1)}. \tag{6}$$

If we apply a non-linear equalizer $\mathcal{F}$ to $\mathbf{r}^{(1)}$, we obtain that the equalized signal is given by

$$\mathbf{y}_{\text{non-linear}}^{(1)} = \mathcal{F}\{\mathbf{r}^{(1)}\}. \tag{7}$$

For the CNN equalizer, the mapping $\mathcal{F}$ is the overall effect of the learned target and ReLU functions at each layer of the CNN.

Fig. 10 shows the non-linear CNN equalizer followed by a trellis detector. During training, the CNN equalizer iterates with a constrained MSE solver to adjust the PR target masks. Using stochastic gradient descent (SGD) on mini-batches of length N, the equalizer CNN minimizes the average MSE J_{MSE} between its output and the ideal PR waveforms ($\mathbf{y}_{id,2}^{(1)}$ for 1DMR and $\mathbf{y}_{id,1}^{(1)}$, $\mathbf{y}_{id,2}^{(1)}$, and $\mathbf{y}_{id,3}^{(1)}$ for TDMR).

The input to the CNN equalizer consists of the readings $\mathbf{r}^{(1)}$ obtained over a sliding window. The outputs are the equalized signals to be fed to the BCJR detector. Training the CNN equalizer is performed as follows [23]. Given a fixed target, the CNN equalizer is trained to minimize J_{MSE}. After the CNN equalizer converges, a constrained MSE solver accepts the fixed CNN equalizer output. Given such an output, the solver minimizes the constrained MSE given by

$$\min_{\mathbf{h}} J_{\text{MSE}} \tag{8a}$$

$$\text{s. t. } c_{\min} \leq \|\mathbf{h}\|_2^2 \leq c_{\max} \tag{8b}$$

$$\mathbf{a}^T\mathbf{h} = 1 \tag{8c}$$

where $c_{\min}$ and $c_{\max}$ are lower and upper bounds on the energy of the target $\mathbf{h}$. A monic constraint is imposed, which sets the central element of the target to one, i.e., $\mathbf{a} = [0, \ldots, 0, 0, 1, 0, 0, \ldots, 0]$. The constrained MSE solver finds a new MSE-optimal target by adjusting the target coefficients. The new target is used to generate ideal PR signals according to (4). Then, the CNN equalizer is retrained on the new ideal PR signal. Iterating between the CNN equalizer and the constrained MSE solver continues until no more significant reductions in J_{MSE} are achieved.

B. CNN PR Equalizer Architecture

The length of the input image is 21. The size of the input image is [1 × 21] for 1DMR since only $\mathbf{r}_2^{(1)}$ is provided for the CNN. For TDMR, the input data are $\mathbf{r}_0^{(1)}, \ldots, \mathbf{r}_4^{(1)}$; hence, the input image has the size of [5 × 21] for TDMR.

The proposed equalizer employs a CNN with four convolutional units. Fig. 11 shows the architecture for the 1DMR CNN PR equalizer. For the 1DMR system, the convolutional layers 1 through 4 have sizes [1 × 15], [1 × 7], [1 × 3], and [1 × 3]. The number of channels for these four convolutional layers is 10, 8, 6, and 4, respectively. For the first two convolutional units, there is another layer before ReLU called the "dropout" layer that randomly sets input elements to zero with a given probability. The probability of dropping for these two layers is equal to 0.1.

The TDMR CNN PR architecture is presented in Fig. 12. In the CNN equalizer for TDMR, the four convolutional layers have size of [5 × 15], [5 × 7], [5 × 3], and [5 × 3]. The convolutional units contain 10, 8, 6, and 4 channels, respectively. The first two convolutional units contain dropout layers with probability 0.1, which is the same as the 1DMR case.

TABLE I
AREAL DENSITY COMPARISON FOR 1DMR

Detectors	Track Pitch (nm)	Raw Channel BER	Areal Density ($\mathrm{Tb/in^2}$)	User Bits per Grain	Code Rate	Decoded BER
1D PDNP, Lin-Eq	48	0.0661	1.186	0.1041	0.9710	0 (9.38e-7)
CNN 1 pass, Lin-Eq	48	0.0661	1.205	0.1057	0.9865	7.64e-6 (1.07e-5)
CNN 2 passes, Lin-Eq	48	0.0661	**1.210**	0.1062	0.9905	9.59e-7 (2.48e-6)
CNN 1 pass, Lin-Eq	27	0.1032	2.018	0.1770	0.9294	0 (8.99e-7)
CNN 1 pass, CNN-Eq	27	0.1032	**2.028**	0.1779	0.9341	0 (9.04e-7)

TABLE II
AREAL DENSITY COMPARISON FOR TDMR

Detectors	Track Pitch (nm)	Raw Channel BER	Areal Density ($\mathrm{Tb/in^2}$)	User Bits per Grain	Code Rate	Decoded BER
1D PDNP 1 pass, Lin-Eq	18	0.1641	2.482	0.2177	0.7600	0 (1.21e-6)
1D PDNP 2 passes, Lin-Eq	18	0.1641	2.531	0.2220	0.7750	0 (1.21e-6)
2D PDNP 1 pass, Lin-Eq	18	0.1641	2.230	0.1957	0.6830	0 (6.84e-6)
1 CNN 1 pass, Lin-Eq	18	0.1637	3.228	0.2833	0.9883	0 (4.02e-6)
1 CNN 1 pass, CNN-Eq	18	0.1637	3.231	0.2836	0.9892	0 (4.02e-6)
1 CNN 1 pass, Lin-Eq, SNR 20 dB	18	0.1637	3.225	0.2830	0.9874	0 (4.01e-6)
1 CNN 2 passes, Lin-Eq, SNR 20 dB	18	0.1637	3.228	0.2833	0.9883	0 (4.02e-6)
3 CNNs 1 pass, Lin-Eq	18	0.1637	**3.234**	0.2838	0.9901	0 (4.02e-6)
3 CNNs 2 passes, Lin-Eq	18	0.1637	**3.234**	0.2838	0.9901	0 (4.02e-6)
1 CNN 1 pass, Lin-Eq	15	0.2058	3.873	0.3385	0.9883	0 (4.02e-6)
1 CNN 2 passes, Lin-Eq	15	0.2058	**3.877**	0.3388	0.9892	0 (4.02e-6)
1 CNN 1 pass, CNN-Eq	15	0.2058	**3.877**	0.3388	0.9892	0 (4.02e-6)
1 CNN 1 pass, Lin-Eq, SNR 20 dB	15	0.2058	3.869	0.3382	0.9874	0 (4.01e-6)
1 CNN 2 passes, Lin-Eq, SNR 20 dB	15	0.2058	3.873	0.3385	0.9883	0 (4.02e-6)
3 CNNs 1 pass, Lin-Eq	15	0.2058	**3.877**	0.3388	0.9892	0 (4.02e-6)
3 CNNs 2 passes, Lin-Eq	15	0.2058	**3.877**	0.3388	0.9892	0 (4.02e-6)

VI. SIMULATION RESULTS

This section presents the simulation results for the BCJR-LDPC-CNN turbo-detector on 1DMR and TDMR channels. The simulations are performed on GFP data sets with TPs 48 and 27 nm for 1DMR and with TPs 18 and 15 nm for TDMR. We provide the results for both linear and CNN-based PR equalization before the turbo-detector. Also, the simulation results of the two different CNN interfaces with the channel decoder are presented for the TDMR system. In Tables I and II, the simulations in which one iteration between the BCJR, IRA, and CNN is performed are labeled as "1 pass" and the other simulations that have two iterations executed are labeled as "2 passes."

The CNN media noise predictor is provided with the LLR probabilities. The estimated bit and the estimation's reliability can be determined by the LLR probabilities. In [11], it is experimentally shown that CNN provided with LLR probabilities performs better compared with when it uses the signed LLR values. This might be due to the non-linear scale inherent in the LLRs.

In the first turbo-iteration, the BCJR initially assumes that the media noise is zero and computes an initial set of output LLRs $\mathbf{LLR}_{b_0}$. In the second turbo-iteration, the LLRs $\mathbf{LLR}_l$ for a code rate greater than the highest code rate in the first iteration are passed to the CNN. In an alternative scheme for the second iteration, the CNN can be trained with both $\mathbf{LLR}_l$ and $\mathbf{LLR}_b$. In this case, the LLRs $\mathbf{LLR}_b$ are considered as an additional input of the CNN, and the training process is performed. Simulations show that the CNN performance is slightly better with the additional input; however, we report the simulation results for the second iteration by replacing the LLRs $\mathbf{LLR}_b$ with $\mathbf{LLR}_l$.

A. Data Sets

We use two GFP waveform data sets, both with 11 nm BL. Each block in each data set has $5 \times N_b$ input bits, where $N_b = 41\,206$. The central three tracks for each data set have two readings per bit, i.e., $3 \times 2N_b$ readings per block. The simulation results assume that the two boundary tracks in GFP data sets have perfect knowledge of the coded bits. In the

realistic case, the bits on the top and bottom boundary tracks are unknown. In [22], it is shown that, by considering the unknown boundary bits, the areal density decreases <7%.

For the 1DMR system, the first set has TP = 48 nm and GPB = 9.33; the second has TP = 27 nm and GPB = 5.25. The 1-D detectors considered in this article use only the central track in the GFP waveforms for training and testing. To be consistent with [11], for the 1-D detector with linear PR equalizer, we use 16 GFP blocks as the training data set to train the CNNs. Another (distinct set of) 84 blocks are used as the test data set to generate (by simulation) the BER. For the 1-D turbo-detector with CNN PR equalizer, we used 50, 10, and 30 GFP blocks as the training, validation, and test data sets, respectively. To train the 1-D PDNP parameters on the TP 48 nm data set as the comparison baseline, 20 training blocks are used, and the testing is performed on 80 blocks.

The two other GFP data sets with 11 nm BL are used for the TDMR system. The TP and GFP for the first data set are 18 nm and 3.50, and for the second data set, they are 15 nm and 2.92, respectively. For the 2-D BCJR-LDPC-CNN detector with linear PR equalizer, we use 80 blocks for the training and 20 blocks for the testing phase. The training, validation, and test data sets for the TDMR turbo-detector using the CNN PR equalizer have 50, 5, and 20 GFP blocks, respectively. We also compare the performance of our TDMR turbo-detection system with the 2-D PDNP design of [10], using the same parameters described in [19]. The 2-D PDNP employs 40 and 60 blocks for training and testing, respectively.

B. Simulation Parameters

The iterations between the BCJR, IRA, and CNN are implemented. In this article, the weight W_1 and magnitude limit threshold T_1 for the BCJR output LLRs are set to 0.1 and 100 for 1DMR, and for TDMR, they are set to 0.5 and 10, respectively. After scaling and limiting LLRs $\mathbf{LLR}_b$, they are passed to the de-interleaver to shuffle the code bits. The minimum internal IRA code iteration is set to 200, and the maximum iteration is considered as 400. W_2 and T_2 are considered to scale and limit the decoder output LLRs $\mathbf{LLR}_l$ to pass through the interleaver to CNN. In the proposed system, the parameters W_2 and T_2 equal to 0.5 and 60 for 1DMR and 1 and 10 for TDMR. We report the results in terms of user bits per grain (U/G) and areal density. (U/G) can be computed as U/G = achieved-code-rate/GPB, where the achieved-code-rate is the highest code rate after puncturing that achieves a final decoded BER equal to or less than 10^{-5}. The GPB values for each data set are listed in Section II. The areal density is obtained by

$$\text{Areal density} = \text{U/G} \times \text{Grain-density}. \tag{9}$$

In order to design IRA codes with a higher code rate, the puncturing scheme in [22] is used to simulate puncturing bits written to an HDD.

Tables I and II summarize the results for the 1DMR and TDMR detectors, respectively. The raw channel BER for 1DMR is reported for the detected bits on the input readings of the central single-track, i.e., track 2. For TDMR, the raw channel BER is the average detected errors on the three central tracks for tracks 1, 2, and 3. A 95% confidence upper bound on the BER is provided in parentheses in the last columns of Tables I and II. In the case of non-zero errors, the BER upper bound is computed as

$$I_p(x+1, N_{\text{tcb}} - x) = \gamma \tag{10}$$

where I_p indicates the beta distribution with parameters $x+1$ and $N_{\text{tcb}} - x$ for γ-quantile, x is the number of errors, N_{tcb} is the total number of transmitted coded bits, and γ is the confidence threshold that we set to 0.95. In the case of zero error count, the BER upper bound is obtained as $3/N_{\text{tcb}}$ [24].

C. Discussion of Simulation Results

Simulation results for three-tap monic PR target mask for 1DMR and three-input-three-output monic mask for TDMR designed using the method described in [25] are included.

Table I summarizes the results for the TP 48 nm and TP 27 nm data sets for 1DMR. The table compares the U/G and areal density performance of the proposed 1-D BCJR-LDPC-CNN detector with 1-D PDNP BCJR. The 1-D PDNP uses the same PR target $\mathbf{h}$ as the BCJR-LDPC-CNN method with the linear PR equalizer uses but has 128 trellis states, corresponding to $I = 2$, $L = 4$, and $\Delta = 1$. The pattern vector length of $I + 1 + L + \Delta = 8$ bits of 1-D PDNP is comparable to the channel input $\mathbf{y}^{(1)}$ length of 9 sample bits for the CNN. LLRs are exchanged between the 1-D PDNP and the IRA decoder iteratively in a turbo-architecture. For the 1DMR waveforms, the 1-D PDNP is performed only on TP 48 nm data set as the comparison baseline. The simulation results on TP 48 nm data set are shown in the first three rows of Table I. For 1-D PDNP, the second iteration does not improve the areal density; hence, we only report its first iteration. The first turbo-iteration of BCJR-LDPC-CNN detector with linear PR equalizer labeled as "CNN 1 pass, Lin-Eq" achieves 1.60% density gain over 1-D PDNP. The second iteration of this detector labeled as "CNN 2 passes, Lin-Eq" has density gain of 2.02% over 1-D PDNP. The last two rows of Table I present the results of the TP 27 nm data set for 1DMR waveforms.

Table II presents the simulation results for 1-D PDNP, 2-D PDNP, and BCJR-LDPC-CNN detectors with the TDMR system on the TP 18 and 15 nm data sets. The 1-D PDNP, 2-D PDNP, and BCJR-LDPC-CNN detectors process one, two, and three tracks, respectively, so that the BCJR-LDPC-CNN system gives a throughput increase of 3× and 1.5× over 1-D and 2-D PDNP, respectively. Table II shows the PDNP results only for the TP 18 nm data set, which are reported from [19]. The two-track 2-D PDNP looks at 2×3 bit patterns and has 64 states. The best PDNP performance belongs to 1-D PDNP with two turbo-iterations labeled as "1-D PDNP 2 passes, Lin-Eq," which we consider as the baseline. The 1-D PDNP looks at a pattern with seven down-track bits. On the other hand, the 2-D PDNP is applied on the three bits on each of the two tracks [10] to keep a reasonable number of trellis states; hence, the 1-D PDNP could outperform the 2-D PDNP.

Two different PR equalizers are investigated for the TDMR system. Tables I and II show the simulation results for the

linear filter equalizer with a 1-D/2-D PR target labeled as "Lin-Eq" and the proposed CNN equalizer labeled as "CNN-Eq." In Table II, for the 18 nm data set, the one-CNN architecture using linear PR equalizer with one iteration (labeled "1 CNN 1 pass, Lin-Eq") achieves a density gain of 27.54% over the "1-D PDNP 2 passes," which is the baseline. For the three-CNN architecture with linear PR equalizer where only one iteration is performed (labeled "3 CNNs 1 pass, Lin-Eq"), the BCJR-LDPC-DNN detector has a density gain of 27.78% compared with the baseline. By using the CNN PR equalizer, the one-CNN architecture with one iteration (labeled as "1 CNN 1 pass, CNN-Eq") could outperform the baseline with a 27.66% density gain.

Comparing the CNN PR equalizer with the similar case with the linear PR equalizer for TDMR system, by 1 pass of the one-CNN architecture, the CNN PR equalizer improves from 3.228 to 3.231 Tb/in^2 (0.09% improvement) for the TP 18 nm data set and from 3.873 to 3.877 Tb/in^2 (0.10% density gain) for the TP 15 data set. To the best of our knowledge, 3.877 Tb/in^2 is the highest density reported on the GFP model data with 11.4 Tg/in^2. For a 1DMR detector with one iteration, using the CNN PR equalizer gets 0.50% density gain for the TP 27 nm data set.

Since designing IRA codes with different rates is difficult, we first design a few IRA codes with some specific code rates. To determine the maximum code rate of a new system, based on the BER, we choose one of the designed IRA codes as the base code (the rate of this code is called base rate); by a puncturing process [22] over the base code, we can achieve a code with a higher rate. For Table I, the base rate of the IRA code is 0.8956 for all the simulations. In Table II, the base rate of the IRA code for the two first rows that show the 1-D PDNP on the TP 18 nm data set is 0.7507, and for the third row that shows the 2-D PDNP on the TP 18 nm data set, it is 0.6506. The base rate code for the three-CNN architecture on the TP 18 nm data set is 0.9287, and for the rest of the TDMR detectors, it is 0.8956.

The second iteration between the BCJR, IRA, and noise-prediction CNN is investigated. In this iteration, the CNN is provided with IRA's LLRs $\mathbf{LLR_l}$ (as well as $\mathbf{y}_k^{(1)}$ and $\mathbf{r}_k^{(2)}$) to estimate the media noise more accurately in comparison to the first iteration. CNN's noise prediction is fed back to the BCJR and IRA to derive the new LLRs. In some cases, the second iteration could enhance the areal density compared with the first one. By using the linear PR equalizer, for the TP 48 nm data set with 1DMR waveforms and the TP 15 nm data set with TDMR waveforms, the second iteration of one-CNN architecture achieves 0.41% and 0.10% density gains, respectively, over the first iteration. However, the three-CNN architecture could not improve the areal density by using the second iteration, perhaps because the improved performance of three CNNs in the first iteration has left less room for improvement in the second iteration.

In designing the CNN noise predictor, we face some challenges, such as *overfitting* and *underfitting* problems. The overfitting occurs when CNN is too closely fitted to the training data set, which makes it difficult to generalize and do predictions for the new data. It might happen due to the complicated topology or a high number of hyperparameters of the CNN. The underfitting makes the CNN inflexible in learning from the training data set due to an over-simplified model with too few features or it is regularized too much. To avoid such problems, we need to tune the CNN hyperparameters. The hyperparameters are generally the properties of NNs that are determined by the designer. The CNN hyperparameters include the topology of CNN, the number of layers, the number of channels, the sizes of the input image and the filters in each layer, the activation function, and the optimization algorithm. We use the SGD as the optimization algorithm to tune the hyperparameters of the CNN noise predictor and train it. SGD computes the gradients and updates the weights by using some subsets of the training blocks called "mini-batches." To tune the hyperparameters of the CNN noise predictor using the CNN PR equalizer, the validation data set results are used. The validation data set results can be used to tune the model regularization parameter and adjust the learning rate schedule as well [13, Ch. 7]. No hyperparameters are tuned based on test data set results in order to have a realistic performance during implementation.

The overfitting (underfitting) can be recognized by the lower (higher) root MSE (RMSE) of the training data set than RMSE of the validation data set, respectively, during the training process. Fig. 13 shows the CNN learning curves (i.e., RMSE versus the number of training iterations) from the validation and training data sets for the CNN noise predictor architectures with CNN PR equalizer. The training process for these architectures contains 803 900 iterations; one-fifth of them are represented in Fig. 13. After tuning the hyperparameters, the learning curves for the validation and training data sets merge together after some point, which means that the model prevents the overfitting and underfitting problems. Also, the RMSE values for the three-CNN architecture are lower than one-CNN architecture in this figure, which confirms the idea of expanding the search space by using one CNN per track.

The GFP data contain no read-head electronic AWGN, i.e., $n_e(k) = 0$ in (2). The rows without label "SNR 20 dB" report the results for this case. The rows that include the label "SNR 20 dB" report the results when non-zero AWGN $n_e(k)$ at an SNR of 20 dB is added to both sample sequences $\mathbf{r}_k^{(1)}$ and $\mathbf{r}_k^{(2)}$. The SNR is computed as

$$\text{SNR} = 10\log_{10}\left(\frac{1}{\sigma_e^2}\mathbb{E}\left[(r_k^{(1)})^2 + (r_k^{(2)})^2\right]\right) \tag{11}$$

where σ_e^2 indicates the AWGN variance. This SNR is computed based on all the code bits of each track. For the 20 dB SNR AWGN results, the second iteration could improve the detector's performance compared with the first iteration for the TDMR system.

We explore the areal density of the proposed system for lower SNRs. The system settings for lower SNR simulations are optimized for 20 dB SNR. The purpose of these simulations is to find out how robust the system optimized for 20 dB SNR is to lower SNRs. In Fig. 14, the curve of areal density of the one-CNN with linear PR equalizer for different values

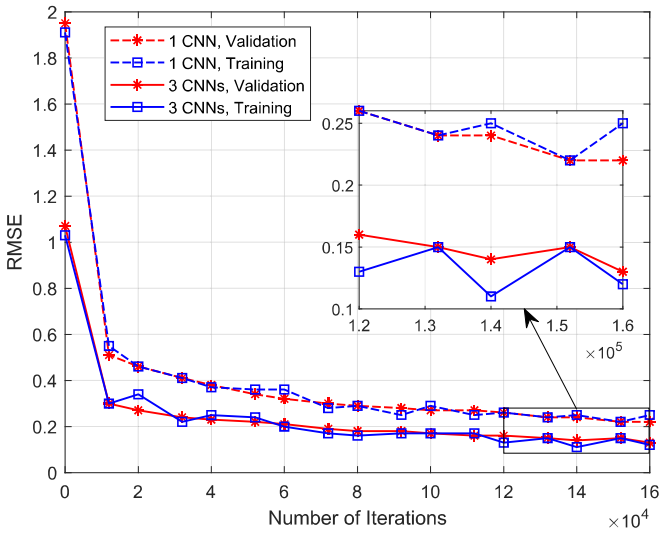

Fig. 13. Learning curves for CNN noise predictor in the architectures using the CNN PR equalizer on the TP 18 nm data set. For the three-CNN architecture, the learning curves are plotted for the CNN corresponding to the central track. An iteration refers to an instance of SGD based on a gradient estimate derived from a mini-batch.

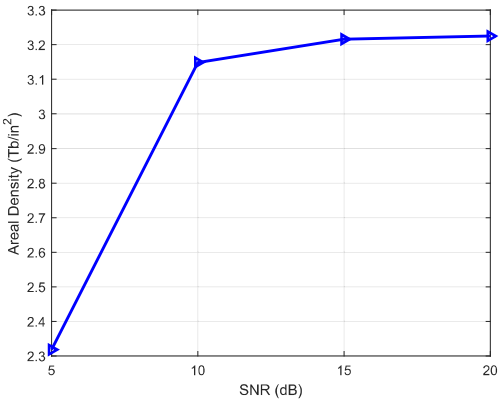

Fig. 14. Areal density versus SNR curve for one-CNN architecture with linear PR equalizer for the TP 18 nm data set. At 0 dB SNR, the average BER over three central tracks is 0.23, and the areal density is not competitive since very low code rates would be required.

of SNR is represented. By decreasing the SNR, the areal density of the system goes down, such that the areal density with AWGN 5 dB SNR is 2.318 Tg/in^2. For 0 dB SNR, the average BER for the second pass of 2-D BCJR over the three central tracks is equal to 0.23; this BER would require an impractically low code rate to achieve a BER $\leq 10^{-5}$ after channel decoding, resulting in a very low achieved density. Thus, the performance of the system breaks at some point between 5 and 0 dB SNR, which is the lower limit of SNR for the system.

D. Computational Complexity Comparison

The computational complexity (per bit) figures of TDMR detectors are shown in Table III. The complexities for the equalizer and the detector are considered; however, the channel coding complexity is not included. The numbers for the second row of this table, that is, for 2-D PDNP, are reported from [26]. The reported numbers for the complexity of the proposed BCJR-LDPC-CNN turbo-detectors are based on the first iteration. The required computations of all BCJR-LDPC-CNN detectors are much higher than the 1-D PDNP with 2 passes. It is due to the higher number of branches per state for 2-D BCJR performed in the BCJR-LDPC-CNN detector and the 3-D CNN trained on the three tracks. The three-CNN architecture requires roughly 3× computations of the one-CNN architecture.

TABLE III
DETECTOR COMPLEXITY FOR THE TDMR

Detectors	mul/div	add/sub	exp/log
1D PDNP	32,532	31,251	514
2D PDNP	4,481	4,216	257
1 CNN, Lin-Eq	782,007	740,468	343
1 CNN, CNN-Eq	831,610	784,966	343
3 CNNs, Lin-Eq	2,110,816	1,991,662	343
3 CNNs, CNN-Eq	2,159,649	2,035,387	343

The complexity for 1-D PDNP is reported for the two passes implementation, which is considered as the baseline. However, the numbers for 2-D PDNP are reported for one iteration. Comparing the complexity of 1-D and 2-D PDNP, 1-D PDNP has more complexity compared with 2-D PDNP since the number of the states of 1-D PDNP is twice the number of states for 2-D PDNP, and also the BCJR algorithm's complexity grows as the square of the number of states.

We point out that the computations in Table III of [11] should be multiplied by factors of roughly 27 and 28 for the second and third rows, respectively, which are associated with the complexity of BCJR-DNN architectures. Also, by considering a more efficient 1-D PDNP implementation, the reported numbers in the first row of this table need to be multiplied by a factor of 0.12 and 0.38 for the first and second columns, respectively. The reported short run-time in [11] is due to the efficient implementation of MATLAB's matrix multiplication, which accelerates the required processing for the DNN. The reported computations in this table do not include the complexity of the equalizer.

Since the complexity of the proposed method in this article is higher than the complexity of the baseline, we have implemented some other reduced-complexity CNN architectures to investigate the tradeoff between the performance and complexity of our method. In Table IV, some of the CNN architectures that have lower complexity (compared with the architecture of the proposed method) are presented. The factor of increase in complexity and the density gain of these architectures over the baseline are shown in Table IV. The CNN architectures are designed based on a one-CNN architecture with the linear PR target. The simulations are run on the TP 18 nm data set. In the future work, we will consider the further study of complexity/performance tradeoffs, and we will investigate lower complexity implementation architectures.

TABLE IV
TRADEOFF BETWEEN THE PERFORMANCE AND COMPLEXITY OF ONE-CNN DETECTOR WITH LINEAR PR EQUALIZER IN COMPARISON TO 1D PDNP WITH TWO PASSES FOR TDMR. THE CNN ARCHITECTURE SHOWS THE NUMBER OF CHANNELS FOR THE FIVE CONVOLUTIONAL LAYERS. THE FACTOR OF COMPLEXITY AND DENSITY GAIN OVER THE BASELINE IS REPORTED. THE COLUMN WITH TITLE "**WITH $\mathbf{r}^{(2)}$**" DETERMINES THE PRESENCE AND ABSENCE OF $\mathbf{r}^{(2)}$ (LABELED AS "Y" AND "N," RESPECTIVELY) IN THE CNN INPUTS

CNN Architecture	With $\mathbf{r}^{(2)}$	mul/div Factor	add/sub Factor	Density Gain
8-8-16-16-16	N	4.96×	4.82×	1.02×
	Y	6.66×	6.66×	1.14×
8-16-8-16-128	N	5.28×	5.17×	1.21×
	Y	7.00×	7.02×	1.26×
8-16-16-16-128	N	5.97×	5.74×	1.22×
	Y	8.61×	8.59×	1.26×
8-16-16-32-128	N	7.23×	6.88×	1.25×
	Y	11.07×	11.02×	1.28×
8-16-32-64-128	N	13.20×	12.01×	1.27×
	Y	24.04×	23.69×	1.28×

To have an insight into the effect of the inputs on the CNN design, we also train the different CNN architectures by removing $\mathbf{r}^{(2)}$ from the inputs. In Table IV, the complexity and density gain of the CNN architectures without providing $\mathbf{r}^{(2)}$ are shown. In this case, the size of the image input layer and the filters in Fig. 8 would change to $[3 \times 9 \times 2]$ and $[3 \times 3 \times 2]$, respectively. Deleting $\mathbf{r}^{(2)}$ for the inputs reduces the CNN architecture complexity, and as a result, its density gain decreases as well. Due to the increased complexity of the 2-D BCJR over the 1-D BCJR used in the 1-D PDNP baseline, the complexity of the proposed method without considering the CNN is around 3.61× (based on the number of multiplications and divisions) the complexity of the baseline. Due to this fact, we are not able to achieve the same complexity as the baseline by changing the CNN architecture.

VII. CONCLUSION

This article presents the combined BCJR-LDPC-CNN architecture of the turbo-detection system on the single-track and parallel multi-track HDDs. The ISI/ITI detection and the media noise estimation are separated, and iterative decoding with the LDPC channel decoder is implemented. Employing the CNNs as the media noise estimator in the turbo-detector makes a probabilistic prediction model that can outperform PDNP-based detectors. In this work, two CNN interfaces with the LDPC decoder are investigated for the multi-track TDMR system. A CNN PR equalizer is presented, which performs better in comparison with a comparable linear PR equalizer. All the proposed architectures have significant areal density gains over the standard 1-D and 2-D PDNP used as comparison baselines. Also, it is shown that additional areal density gains can be obtained by performing more iterations in the turbo-detection system. The achieved areal density of 3.877 Tb/in^2 for the TP 15 nm data is among the highest published figures for magnetic media with grain densities of 11.4 Tg/in^2.

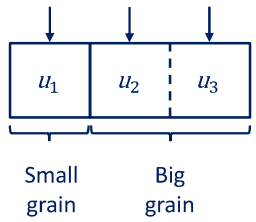

Fig. 15. Grain Model for three bits and two grains. The left and right grains are the small and big grains, respectively. The arrows on the top of the grains represent the locations of the read head when the bits are read.

APPENDIX
MATHEMATICAL INSIGHTS ON THE WRITE–READ MODEL

To mathematically show the superiority of the DNN media noise model over the PDNP media noise model, first, we claim that the media noise $\mathbf{n}_m$ is a non-linear function of the data bits in a typical magnetic recording write–read model. To argue this, we consider a 1-D version of the four-grain rectangular model in [27]. In particular, the 1-D track in the magnetic medium is made up of two types of rectangular grains, designated as grain "A" and grain "DE" in [27, Fig. 2]. Fig. 15 shows three bits corresponding to two grains. The left and right grains are the small (type "A") and big (type "DE") grains, respectively. In the grain model, generally, the bits are small grain-sized, and the large grains have a size of around 2× small grains. The big grains occur with probability p_b and the small ones with probability $1 - p_b$. The polarity of the grains is ± 1. The last part of the big grain written determines the polarity of the entire grain since the last part magnetizes the previous part that is written first. This model is called the "erasure model," and the overwritten bits are "erasure bits." For instance, the polarity of big grain in Fig. 15 is determined by u_3 that is the last part of this grain. For the small grains, the polarity is determined based on the corresponding bit.

In the erasure model, a code word, such as $\mathbf{c}$, is written into the magnetic grains. After writing, the true polarity corresponding to code word $\mathbf{c}$ would be $\mathbf{u}_t$. However, due to the grain–bit interactions in big grains, the erasure bits would equal to $\mathbf{u}_e$, which are not necessarily equal to $\mathbf{u}_t$. For the erasure model, we use $\mathbf{u}_t$ instead of $\mathbf{u}$ in (2). Also, we consider $n_e(k)$ as a part of media noise in (2) since the term $n_m(k)$ is the dominant term of the noise. For the erasure model, the equalizer output can be written as follows:

$$\mathbf{y} = \mathbf{h} * \mathbf{u}_e. \tag{12}$$

Therefore, to obtain the media noise for the kth trellis stage in erasure model, we derive the following equations:

$$n_m(k) = y(k) - (\mathbf{h} * \mathbf{u}_t)(k) \tag{13a}$$
$$= (\mathbf{h} * \mathbf{u}_e)(k) - (\mathbf{h} * \mathbf{u}_t)(k) \tag{13b}$$
$$= (\mathbf{h} * (\mathbf{u}_e - \mathbf{u}_t))(k). \tag{13c}$$

Now, with a counterexample, we show that the media noise is a non-linear function of the input data bits. We assume a general PR target $\mathbf{h}$ for the system. The PR target is some impulse response function that can be completely arbitrary for the current discussion. Let us suppose that the two code words $\mathbf{c}_1 = [1, 0, 1]$ and $\mathbf{c}_2 = [0, 1, 1]$ are written to the magnetic grains in Fig. 15. The true corresponding polarities for the first and second code words are $\mathbf{u}_{t,1} = [1, -1, 1]$ and $\mathbf{u}_{t,2} = [-1, 1, 1]$, respectively. Based on the erasure model in Fig. 15, the erasure bits associated with the first and second code words are $\mathbf{u}_{e,1} = [1, 1, 1]$ and $\mathbf{u}_{e,2} = [-1, 1, 1]$. From (13c), the media noises for the code words are $\mathbf{n}_{m,1} = \mathbf{h} * [0, 2, 0]$ and $\mathbf{n}_{m,2} = \mathbf{h} * [0, 0, 0]$. Therefore, the superposition of the media noise for $\mathbf{c}_1$ and $\mathbf{c}_2$ is equal to $\mathbf{n}_{m,1} + \mathbf{n}_{m,2} = \mathbf{h} * [0, 2, 0]$. To show the non-linearity of the system, we consider the superposition of the code words, $\mathbf{c}_1$ and $\mathbf{c}_2$, under the GF(2) field, that is, the code word $\mathbf{c}_s = [1, 1, 0]$. The true polarities and the erasure bits of $\mathbf{c}_s$ would be $\mathbf{u}_{t,s} = [1, 1, -1]$ and $\mathbf{u}_{e,s} = [1, -1, -1]$. Thus, the corresponding media noise of $\mathbf{c}_s$ is equal to $\mathbf{n}_{m,s} = \mathbf{h} * [0, -2, 0]$. To have a linear system, the equation $\mathbf{n}_{m,s} = \mathbf{n}_{m,1} + \mathbf{n}_{m,2}$ should be true. The only impulse response that satisfies this equation is $\mathbf{h} = 0$; however, this impulse response is not true generally. Hence, we have a contradiction here, and we can conclude that the system in 1DMR case is non-linear.

Even if we assume that the write process is completely linear over the real line $\mathbb{R}$, and we take the GF(2) binary addition out of the picture, we claim that our grain channel is still non-linear. Now, we make the assumption that we start at the write fields where we have already done the level shifting, and we assume that the writer is linear. We assume that the grains can store magnetization equal to any finite real number. Here, we provide a counterexample to prove our claim. As in the previous counterexample, we suppose a general target $\mathbf{h}$ for the system. After level shifting, let us consider two code word polarities: $\mathbf{u}_{t,1} = [1, 1, 1]$ and $\mathbf{u}_{t,2} = [1, 1, -1]$. Thus, the first and second corresponding erasure bits will be $\mathbf{u}_{e,1} = [1, 1, 1]$ and $\mathbf{u}_{e,2} = [1, -1, -1]$ according to the erasure model. The media noises for the first and second code words are equal to $\mathbf{n}_{m,1} = \mathbf{h} * [0, 0, 0]$ and $\mathbf{n}_{m,2} = \mathbf{h} * [0, -2, 0]$ based on (13c). In this case, the superposition of these two media noise values equals $\mathbf{n}_{m,1} + \mathbf{n}_{m,2} = \mathbf{h} * [0, -2, 0]$. Now, to argue that the system is non-linear, we find that the superposition of $\mathbf{u}_{t,1}$ and $\mathbf{u}_{t,2}$ is equal to $\mathbf{u}_{t,s} = [2, 2, 0]$. Therefore, the associated erasure bits will be $\mathbf{u}_{e,s} = [2, 2, 2]$ since writing a 0 on the last half of the big grain will not affect its magnetization. Hence, the corresponding media noise is $\mathbf{n}_{m,s} = \mathbf{h} * [0, 0, 2]$, which is not equal to $\mathbf{n}_{m,1} + \mathbf{n}_{m,2} = \mathbf{h} * [0, -2, 0]$. By this argument, we showed that, even under a linear writer assumption, the channel still is non-linear due to the grain–bit interactions.

The AR media noise model of PDNP is based on (1). For the kth trellis stage, the optimal (in the MMSE sense) AR coefficients a_i are unknown, and we can find them for the coded bit pattern vector $\mathbf{u} = [u_{k+\Delta}, \ldots, u_k, \ldots, u_{k-(I+L)}]$ from the statistics of $n_k(\mathbf{u})$. We multiply both sides of (1) by $\mathbf{n}(\mathbf{u}) \triangleq [n_{k-1}(\mathbf{u}), n_{k-2}(\mathbf{u}), \ldots, n_{k-L}(\mathbf{u})]$ and then take the expectation. Since $e_k(\mathbf{u})$ is statistically independent of $\mathbf{n}_k(\mathbf{u})$, we can say that $\mathbb{E}[e_k(\mathbf{u})\mathbf{n}_k(\mathbf{u})] = 0$; therefore, by simplifying,

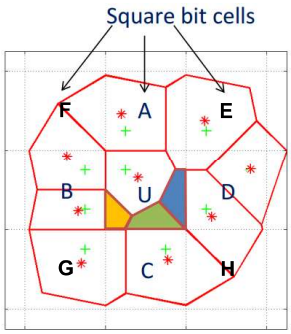

Fig. 16. Generated Voronoi grains (outlined in solid red) for 3 × 3 channel coded bits (indicated by black dotted lines), with approximately one coded bit per grain. The "+" and "*" indicate the grain nuclei and centroids, respectively. The grain–bit interactions are represented as the portions of central coded bit U affected by vertically and horizontally adjacent bits: B (orange), C (green), and D (blue).

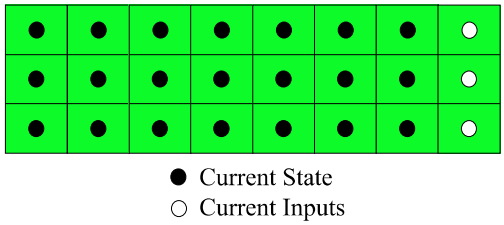

Fig. 17. State-input block for the 2-D trellis-based detector with seven state bits per track on three tracks.

we have

$$\mathbf{a}(\mathbf{u})^T = \mathbb{E}[n_k(\mathbf{u})\mathbf{n}(\mathbf{u})]R^{-1}(\mathbf{u}) \tag{14}$$

where $\mathbf{a}(\mathbf{u}) = [a_1(\mathbf{u}), a_2(\mathbf{u}), \ldots, a_L(\mathbf{u})]$ and $R^{-1}(\mathbf{u}) = \mathbb{E}[\mathbf{n}(\mathbf{u})\mathbf{n}(\mathbf{u})^T]^{-1}$ [3]. In (14), the cross correlation between the media noise and the past media noise, $\mathbb{E}[n_k(\mathbf{u})\mathbf{n}(\mathbf{u})]$, is quadratic in terms of the data bits $\mathbf{u}$. The auto-correlation matrix R is also quadratic in the data bits; therefore, by multiplying these terms in (14), the AR coefficients, $\mathbf{a}(\mathbf{u})$, are quartic functions of the data bits at best.

As we argued, the magnetic media noise itself is a non-linear function of the data bits $\mathbf{u}$. A DNN can model media noise of any order of the input data bits, not restricted to quartic order. Also, because of the hidden layers in the DNN, it is a universal function estimator of the media noise [28]. In [29], it is shown that the networks with enough convolutional layers using ReLU activation functions are universal function estimators as well. Due to these superiorities, the DNN model is more general and accurate than the AR model.

The non-linearity of the system is due to the grain–bit interactions that we have discussed for 1DMR in Fig. 15. Grain–bit interactions also occur in TDMR. Fig. 16 shows a simplified model for TDMR grain–bit interactions, where a more realistic Voronoi grain model is assumed [22]. By generalizing the 1-D system to a 2-D system, we can argue (as we have in the previous paragraph for 1DMR) that the 2-D system is non-linear, and the media noise is a non-linear function of the data bits for TDMR as well.

The trellis-based detection algorithms suffer from the state explosion problem. The 1-D PDNP with two passes uses

seven state bits per track in the baseline; thus, it has $2^7 = 128$ states. Generalizing the trellis-based detection system to 2-D with the same number of state bits per track results in an impractically large number of states. Fig. 17 shows the state-input block for seven state bits per track on three tracks. In this case, the number of states would be $2^{3\times7} > 2$ million states. To avoid this problem, the proposed BCJR-LDPC-CNN method employs a 2-D BCJR with two state bits per track, which has $2^{3\times2} = 64$ states. By utilizing 2-D BCJR with a lower number of state bits per track, the complexity of the method reduces dramatically. However, the complexity of 2-D BCJR with 64 states is still higher than the 1-D PDNP baseline. Due to this fact, the complexity of the proposed system without considering the CNN complexity is around $3.61\times$ the baseline system's complexity. This is the reason why the detector architectures in Section VI-D could not achieve the same complexity as the baseline.

ACKNOWLEDGMENT

This work was supported in part by the United States National Science Foundation under Grant CCF-1817083 and in part by a gift from the Advanced Storage Research Consortium.

REFERENCES

[1] R. Wood, M. Williams, A. Kavcic, and J. Miles, "The feasibility of magnetic recording at 10 terabits per square inch on conventional media," *IEEE Trans. Magn.*, vol. 45, no. 2, pp. 917–923, Feb. 2009.

[2] J. Moon and J. Park, "Pattern-dependent noise prediction in signal-dependent noise," *IEEE J. Sel. Areas Commun.*, vol. 19, no. 4, pp. 730–743, Apr. 2001.

[3] E. M. Kurtas, J. Park, X. Yang, W. Radich, and A. Kavcic, "Detection methods for data-dependent noise in storage channels," in *Coding and Signal Processing for Magnetic Recording Systems*, B. Vasic and E. M. Kurtas. Boca Raton, FL, USA: CRC Press, Nov. 2004.

[4] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding*. New York, NY, USA: McGraw-Hill, Dec. 1979.

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

[6] A. Kavcic and J. M. F. Moura, "The Viterbi algorithm and Markov noise memory," *IEEE Trans. Inf. Theory*, vol. 46, no. 1, pp. 291–301, Jan. 2000.

[7] J. Yao, E. Hwang, B. V. K. Vijaya Kumar, and G. Mathew, "Two-track joint detection for two-dimensional magnetic recording (TDMR)," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Jun. 2015, pp. 418–424.

[8] Y. Wang and B. V. K. V. Kumar, "Multi-track joint detection for shingled magnetic recording on bit patterned media with 2-D sectors," *IEEE Trans. Magn.*, vol. 52, no. 7, pp. 1–7, Jul. 2016.

[9] Y. Wang and B. V. K. Vijaya Kumar, "Micromagnetics-based analysis of multi-track detection with simplified 2-D write precompensation on shingled magnetic recording," *IEEE Trans. Magn.*, vol. 52, no. 9, pp. 1–11, Sep. 2016.

[10] S. Shi and J. R. Barry, "Multitrack detection with 2D pattern-dependent noise prediction," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2018, pp. 1–6.

[11] A. Sayyafan, B. J. Belzer, K. Sivakumar, J. Shen, K. S. Chan, and A. James, "Deep neural network based media noise predictors for use in high-density magnetic recording turbo-detectors," *IEEE Trans. Magn.*, vol. 55, no. 12, pp. 1–6, Dec. 2019.

[12] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006. [Online]. Available: http://www.cs.toronto.edu/~fritz/absps/ncfast.pdf

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015.

[14] Y. Lecun and Y. Bengio, "Convolutional networks for images, speech, and time-series," in *The Handbook of Brain Theory and Neural Networks*, M. Arbib, Ed. Cambridge, MA, USA: MIT Press, Jan. 1995.

[15] K. Luo *et al.*, "A study on block-based neural network equalization in TDMR system with LDPC coding," *IEEE Trans. Magn.*, vol. 55, no. 11, pp. 1–5, Nov. 2019.

[16] M. Nishikawa, Y. Nakamura, Y. Kanai, H. Osawa, and Y. Okamoto, "A study on iterative decoding with LLR modulator by neural network using adjacent track information in SMR system," *IEEE Trans. Magn.*, vol. 55, no. 12, pp. 1–5, Dec. 2019.

[17] Y. Qin and J.-G. Zhu, "Deep neural network: Data detection channel for hard disk drives by learning," *IEEE Trans. Magn.*, vol. 56, no. 2, pp. 1–8, Feb. 2020.

[18] J. Shen and N. Nangare, "Nonlinear equalization for TDMR channels using neural networks," in *Proc. 54th Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2020, pp. 1–6.

[19] J. Shen, A. Aboutaleb, K. Sivakumar, B. J. Belzer, K. S. Chan, and A. James, "Deep neural network a posteriori probability detector for two-dimensional magnetic recording," *IEEE Trans. Magn.*, vol. 56, no. 6, pp. 1–12, Jun. 2020.

[20] K. S. Chan *et al.*, "Channel models and detectors for two-dimensional magnetic recording," *IEEE Trans. Magn.*, vol. 46, no. 3, pp. 804–811, Mar. 2010.

[21] H. Jin, A. Khandekar, and R. McEliece, "Irregular repeat-accumulate codes," in *Proc. 2nd Int. Symp. Turbo Codes Rel. Topics*, Brest, France, Sep. 2000, pp. 1–8.

[22] X. Sun *et al.*, "ISI/ITI turbo equalizer for TDMR using trained local area influence probabilistic model," *IEEE Trans. Magn.*, vol. 55, no. 4, pp. 1–15, Apr. 2019.

[23] A. Aboutaleb *et al.*, "Deep neural network-based detection and partial response equalization for multilayer magnetic recording," *IEEE Trans. Magn.*, early access, pp. 1–12, Nov. 2020.

[24] F. Scholz. (2008). *Confidence Bounds and Intervals for Parameters Relating to the Binomial, Negative Binomial, Poisson and Hypergeometric Distributions With Applications to Rare Events*. [Online]. Available: http://www.stat.washington.edu/people/fritz/DATAFILES/ConfidenceBounds.pdf

[25] C. K. Matcha and S. G. Srinivasa, "Generalized partial response equalization and data-dependent noise predictive signal detection over media models for TDMR," *IEEE Trans. Magn.*, vol. 51, no. 10, pp. 1–15, Oct. 2015.

[26] J. Shen, B. J. Belzer, K. Sivakumar, K. S. Chan, and A. James, "Convolutional neural network based symbol detector for two-dimensional magnetic recording," *IEEE Trans. Magn.*, early access, pp. 1–5, Nov. 2020.

[27] M. Carosino *et al.*, "Iterative detection and decoding for TDMR with 2-D intersymbol interference using the four-rectangular-grain model," *IEEE Trans. Magn.*, vol. 51, no. 7, pp. 1–12, Jul. 2015.

[28] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals, Syst.*, vol. 2, no. 4, pp. 303–314, Dec. 1989, doi: 10.1007/BF02551274.

[29] A. Heinecke, J. Ho, and W.-L. Hwang, "Refinement and universal approximation via sparsely connected ReLU convolution nets," *IEEE Signal Process. Lett.*, vol. 27, pp. 1175–1179, Jun. 2020.