Parallel Deep Learning Detection Network in the MIMO Channel

Xianglan Jin, *Member, IEEE*, and Hyoung-Nam Kim, *Member, IEEE*

***Abstract*— For deep learning detection networks in the multiple-input-multiple-output (MIMO) channel, deepening the network does not significantly improve performance beyond a certain number of layers. In this letter, we propose a parallel detection network (PDN) that consists of several deep learning detection networks in parallel without connection. By designing a specific loss function and reducing similarity between detection networks, the PDN obtains a considerable diversity effect. The performance of the PDN improves significantly as the number of parallel detection networks increases in time-varying MIMO channels. This is superior to the existing deep learning detection networks, in both performance and complexity.**

***Index Terms*— Deep learning, detection, MIMO.**

I. Introduction

In COMMUNICATION systems, multiple-input-multiple-output (MIMO) systems that send multiple data streams simultaneously have become very popular due to the requirement for high data rates. The computational challenges in detecting the multiplexed signals have led to some linear detectors such as zero-forcing and minimum mean square error [1], and several sub-optimal detectors [2] such as the sphere decoding (SD) [3], [4], and the semidefinite relaxation (SDR) [5].

On the other hand, detection methods with data-driven approach have recently begun to be studied [6]–[10]. In [6], a trainable projected gradient detector was proposed for massive overloaded MIMO channels. By incorporating deep learning into the orthogonal approximate message passing (OAMP), the authors in [7] proposed a model-driven detection called OAMP-Net. In addition, a deep learning MIMO detection network (DetNet) was proposed in [8], [9]. The DetNet applies a deep unfolding approach that transforms a computationally intractable probabilistic model into a deep neural network by unfolding iterative process [11]. In [10], the authors proposed a multilevel MIMO detection by applying the similar deep learning network structure in [8] and introducing a multi-plateau sigmoid function. The above works demonstrate the possibility of signal detection in time-varying MIMO channels using deep learning techniques.

Manuscript received September 11, 2019; revised October 19, 2019; accepted October 24, 2019. Date of publication October 29, 2019; date of current version January 8, 2020. This research has been supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Education (NRF-2017R1D1A1A09000565). The associate editor coordinating the review of this letter and approving it for publication was S. Sorour. *(Corresponding author: Hyoung-Nam Kim.)*

X. Jin was with the Department of Electronics Engineering, Pusan National University, Busan 46241, South Korea. She is now with the Division of Electronic Engineering, Jeonbuk National University, Jeonju 54896, South Korea (e-mail: jinxl77@gmail.com).

H.-N. Kim is with the Department of Electronics Engineering, Pusan National University, Busan 46241, South Korea (e-mail: hnkim@pusan.ac.kr).

Digital Object Identifier 10.1109/LCOMM.2019.2950201

The DetNet shows reasonable performance compared to the existing sub-optimal detectors when the number of the receiving antennas is sufficiently large; otherwise, the performance is quite far form them. Deepening the network does not significantly improve performance beyond a certain number of layers. To solve this problem, we propose a new deep learning detection network called parallel detection network (PDN) which divides a single deep detection network into multiple parts and arranges them in parallel. By training the multiple parallel neural networks simultaneously and selecting the one that maximizes the likelihood function, a diversity effect is obtained and so the error performance is improved. In the PDN, similarity between detection networks degrades the diversity effect. In order to avoid the similarity as much as possible, we apply a specific loss function in training. With this design, the performance of the PDN improves significantly as the number of parallel networks increases, and is much better than the performance of the existing deep learning detections in the same total number of detection layers. Simulation results demonstrate the superiority of the proposed PDN.

Throughout the letter, we use the following notations. The superscript $(\cdot)^T$ denotes the transpose of a matrix; $\mathrm{tr}(\cdot)$ denotes the trace of a matrix; $\mathrm{Re}\{\cdot\}$ and $\mathrm{Im}\{\cdot\}$ mean the real and imaginary parts of a complex number, respectively; $\mathrm{E}[\cdot]$ denotes the expectation with respect to the random variables in the argument. $\mathbb{C}^{n\times m}$ denotes a set of $n \times m$ complex matrices; $[\cdot]_{ij}$ means the element in ith row and jth column of a matrix; $[\cdot]_{k:l}$ means a vector consisting of the elements from the kth row to the lth row in the original vector; ∇ denotes the gradient with respect to the parameters in the argument; $\mathbf{A} \sim \mathcal{CN}(0, \sigma_{\mathbf{A}}^2\mathbf{I})$ denotes that the elements of $\mathbf{A}$ are independent and identically distributed (i.i.d.) circularly symmetric complex Gaussian random variables with zero mean and variance $\sigma_{\mathbf{A}}^2$. We omit the size of the identity matrix $\mathbf{I}$ since it can be revealed from the size of the matrix $\mathbf{A}$.

II. System Model

We consider a MIMO fading channel with N_t antennas at the transmitter and N_r antennas at the receiver. Then the received signal $\mathbf{y}^C \in \mathbb{C}^{N_r}$ is written as

$$\mathbf{y}^C = \mathbf{H}^C\mathbf{x}^C + \mathbf{z}^C \tag{1}$$

where $\mathbf{x}^C = [x_1^C \;\, x_2^C \;\, \ldots \;\, x_{N_t}^C]^T \in \mathbb{C}^{N_t}$, $\mathbf{H}^C \in \mathbb{C}^{N_r\times N_t}$ is the channel coefficient matrix of the MIMO fading channel, and the $N_r \times 1$ noise vector $\mathbf{z}^C$ is distributed as $\mathcal{CN}(0, \sigma^2\mathbf{I})$.

To simplify the expressions, we convert the complex system model to an equivalent real system. Let $\mathbf{x} = [x_1 \;\, x_2 \;\, \ldots \;\, x_{2N_t}]^T = \begin{bmatrix}\mathrm{Re}\{\mathbf{x}^C\}\\ \mathrm{Im}\{\mathbf{x}^C\}\end{bmatrix}$, $\mathbf{y} = \begin{bmatrix}\mathrm{Re}\{\mathbf{y}^C\}\\ \mathrm{Im}\{\mathbf{y}^C\}\end{bmatrix}$, $\mathbf{z} = \begin{bmatrix}\mathrm{Re}\{\mathbf{z}^C\}\\ \mathrm{Im}\{\mathbf{z}^C\}\end{bmatrix}$, and $\mathbf{H} = \begin{bmatrix}\mathrm{Re}\{\mathbf{H}^C\} & -\mathrm{Im}\{\mathbf{H}^C\}\\ \mathrm{Im}\{\mathbf{H}^C\} & \mathrm{Re}\{\mathbf{H}^C\}\end{bmatrix}$. Then the

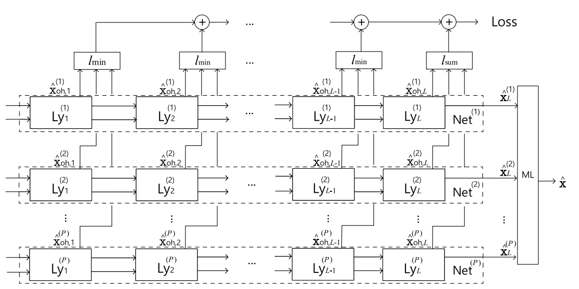

Fig. 1. The parallel detection network in the MIMO channel.

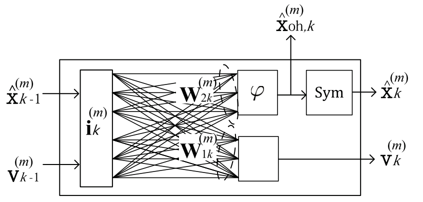

Fig. 2. A single detection layer $\text{Ly}^{(m)}_k$ in the parallel detection network.

equivalent real system model is written as

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{z}. \tag{2}$$

We assume $|\mathcal{S}| = M$, $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$, and $s_m = 2m - M - 1$ for $m = 1, \ldots, M$. x_n is randomly chosen from the symbol set $\mathcal{S}$ with equal probability, thus we have $\text{E}[x_n^2] = \frac{\sum_{i=1}^M |s_i|^2}{M}$. Since the symbols $x_1, x_2, \ldots, x_{2N_t}$ are i.i.d., the signal-to-noise ratio (SNR) is written as

$$\text{SNR} = \frac{\text{E}\|\mathbf{H}\mathbf{x}\|^2}{\text{E}\|\mathbf{z}\|^2} = \frac{\text{E}\|\mathbf{H}^C\|^2}{N_t N_r} \cdot \rho$$

where $\rho = \frac{2N_t \sum_{i=1}^M |s_i|^2}{M\sigma^2}$.

In this MIMO channel, we want to detect $\mathbf{x}$ in the maximum likelihood (ML) sense written as

$$\hat{\mathbf{x}} = \arg \min_{\mathbf{x} \in \mathcal{S}^{2N_t}} \left\|\mathbf{y} - \mathbf{H}\mathbf{x}\right\|^2. \tag{3}$$

This is known to be nondeteministic polynomial-time (NP)-hard. In the next section, we propose a parallel deep learning detection network to solve the problem approximately.

III. Parallel Detection Network

A. Structure of the Parallel Detection Network

The parallel detection network (PDN) includes P detection networks in parallel as shown in Fig. 1. The mth detection network (inside the dashed box in Fig. 1), $\text{Net}^{(m)}$, consists of L detection layers $\text{Ly}^{(m)}_1, \text{Ly}^{(m)}_2, \ldots, \text{Ly}^{(m)}_L$, and the P detection networks are not connected each other and they are trained simultaneously by minimizing a specific total loss.

In detail, the kth detection layer in the mth detection network, $\text{Ly}^{(m)}_k$, is depicted in Fig. 2 which includes one input sublayer

$$\mathbf{i}^{(m)}_k = \begin{bmatrix} \hat{\mathbf{x}}^{(m)}_{k-1} - \alpha^{(m)}_{1k}\mathbf{H}^T\mathbf{H}\hat{\mathbf{x}}^{(m)}_{k-1} + \alpha^{(m)}_{2k}\mathbf{H}^T\mathbf{y} \\ \text{Diag}(\mathbf{H}^T\mathbf{H})\hat{\mathbf{x}}^{(m)}_{k-1} \\ \mathbf{v}^{(m)}_{k-1} \end{bmatrix} \tag{4}$$

and one output sublayer

$$\begin{aligned} \mathbf{v}^{(m)}_k &= \mathbf{W}^{(m)}_{1k}\mathbf{i}^{(m)}_k \\ \hat{\mathbf{x}}^{(m)}_{\text{oh},k} &= \varphi\big(\mathbf{W}^{(m)}_{2k}\mathbf{i}^{(m)}_k\big) & (5) \\ \hat{\mathbf{x}}^{(m)}_k &= \text{Sym}\big(\hat{\mathbf{x}}^{(m)}_{\text{oh},k}\big). & (6) \end{aligned}$$

We explain the above two sublayers in detail.

The first line of the input $\mathbf{i}^{(m)}_k$ is from the idea of the projected gradient descent method applied in [9]:

$$\begin{aligned} \hat{\mathbf{x}}_k &= \phi\Big(\hat{\mathbf{x}}_{k-1} - \delta_k \nabla \big\|\mathbf{y} - \mathbf{H}\mathbf{x}\big\|^2\Big|_{\mathbf{x}=\hat{\mathbf{x}}_{k-1}}\Big) \\ &= \phi\Big(\hat{\mathbf{x}}_{k-1} - \delta'_k\mathbf{H}^T\mathbf{H}\hat{\mathbf{x}}_{k-1} + \delta'_k\mathbf{H}^T\mathbf{y}\Big) \end{aligned} \tag{7}$$

where $\phi(\cdot)$ is an element-wise nonlinear projection operator, e.g., $\phi(\mathbf{x}) = [\text{sgn}(x_1), \ldots, \text{sgn}(x_{2N_t})]^T$ for $\mathbf{x} \in \mathcal{S}^{2N_t} = \{+1, -1\}^{2N_t}$, $\hat{\mathbf{x}}_{k-1}$ is the estimate in the $(k-1)$th iteration, $\delta'_k = 2\delta_k$ is a step size in the kth iteration, and $\hat{\mathbf{x}}_k$ is the estimate in the kth iteration. In (4), we use trainable scalar step sizes $\alpha^{(m)}_{1k}$ and $\alpha^{(m)}_{2k}$ instead of the fixed one, δ'_k, to improve the performance. Furthermore, we add $\text{Diag}(\mathbf{H}^T\mathbf{H})\hat{\mathbf{x}}^{(m)}_{k-1}$ as another input in $\mathbf{i}^{(m)}_k$, where $\text{Diag}(\mathbf{H}^T\mathbf{H}) = \text{diag}([\mathbf{H}^T\mathbf{H}]_{1,1}, \ldots, [\mathbf{H}^T\mathbf{H}]_{2N_t,2N_t})$ and $[\mathbf{H}^T\mathbf{H}]_{n,n}$ is the maximum achievable channel power of the nth symbol without interference. This provides information about the best achievable performance for the machine. Moreover, the auxiliary vectors $\mathbf{v}^{(m)}_{k-1}$ and $\mathbf{v}^{(m)}_k$ are added in the input and the output to speed up the training convergence. Setting the size of $\mathbf{v}^{(m)}_{k-1}$ as $2N_t$, we have the input $\mathbf{i}^{(m)}_k$ with the size of $6N_t$.

In the output sublayer, we multiply $2N_t \times 6N_t$ weight matrix $\mathbf{W}^{(m)}_{1k}$, and $2MN_t \times 6N_t$ weight matrix $\mathbf{W}^{(m)}_{2k}$ and the input $\mathbf{i}^{(m)}_k$, apply the softmax function $\varphi(z_1, \ldots, z_M) = \Big[\frac{e^{z_1}}{\sum_{j=1}^M e^{z_j}}, \ldots, \frac{e^{z_M}}{\sum_{j=1}^M e^{z_j}}\Big]^T$, and then obtain the soft estimators, $\hat{\mathbf{x}}^{(m)}_{\text{oh},k}$, which are supposed to be compared with the one-hot encoding of the transmitted signals, $\mathbf{x}_{\text{oh}}$, i.e., the one-hot encoding for $x_n = s_i$ is $[\mathbf{x}_{\text{oh}}]_{((n-1)M+1):nM} = \mathbf{e}_i$, where $[\mathbf{e}_i]_j = 0$ if $j \neq i$ and $[\mathbf{e}_i]_j = 1$ if $j = i$ for $i \in \{1, \ldots, M\}$. The output of the nth softmax function, $[\hat{\mathbf{x}}^{(m)}_{\text{oh},k}]_{((n-1)M+1):nM}$ in (5), represents $\text{Pr}(x_n = s_i|\mathbf{y})$, $i = 1, \ldots, M$, satisfying $\sum_{i=1}^M \text{Pr}(x_n = s_i|\mathbf{y}) = 1$. Taking a conditional expectation on x_n, we have an estimator

$$\hat{x}_n = \text{E}[x_n|\mathbf{y}] = \sum_{i=1}^M s_i \text{Pr}(x_n = s_i|\mathbf{y}), \tag{8}$$

which is expressed as the nth output of $\text{Sym}(\cdot)$, i.e., $[\hat{\mathbf{x}}^{(m)}_k]_n = [\text{Sym}(\hat{\mathbf{x}}^{(m)}_{\text{oh},k})]_n = \sum_{i=1}^M s_i[\hat{\mathbf{x}}^{(m)}_{\text{oh},k}]_{(n-1)M+i}$ in (6). Then the $2N_t \times 1$ estimated symbol vector $\hat{\mathbf{x}}^{(m)}_k$ which is the output of $\text{Sym}(\cdot)$ enters the next detection layer together with $\mathbf{v}^{(m)}_k$.

In the last layers, we have $[\hat{\mathbf{x}}^{(m)}_L]_n = s_{\hat{i}}$ where $\hat{i} = \arg\max_{1\leq i\leq M}[\hat{\mathbf{x}}^{(m)}_{\text{oh},k}]_{(n-1)M+i}$ which means $\hat{i} = \arg\max_{1\leq i\leq M} \text{Pr}(x_n = s_i|\mathbf{y})$. Finally, we compare all the detected symbols obtained from P detection networks and choose the one that maximizes the likelihood function (the process of ML in Fig. 1):

$$\hat{\mathbf{x}} = \arg \max_{\mathbf{x}\in\{\hat{\mathbf{x}}_L^{(1)},\ldots,\hat{\mathbf{x}}_L^{(P)}\}} p(\mathbf{y}|\mathbf{x}) = \arg \min_{\mathbf{x}\in\{\hat{\mathbf{x}}_L^{(1)},\ldots,\hat{\mathbf{x}}_L^{(P)}\}} \|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2. \tag{9}$$

To further improve the performance, we adopt the residual learning [12], [9], i.e., setting a weighted average of the previous output and the current output as a new current output.

B. Loss Function

In the PDN, a set of parameters that needs to be optimized is written as

$$\theta = \{\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(P)}\},$$

where $\theta^{(m)} = \{\alpha_{1k}^{(m)}, \alpha_{2k}^{(m)}, \mathbf{W}_{1k}^{(m)}, \mathbf{W}_{2k}^{(m)},\ k = 1,\ldots,L\}$, $m = 1,\ldots,P$. Since the parallel detection networks have the same structure and use the same training data, it is likely to obtain similar results from multiple networks, $\text{Net}^{(1)}, \text{Net}^{(2)}, \ldots, \text{Net}^{(P)}$. This degrades the diversity effect. Hence, we apply a specific loss function as

$$\text{Loss} = \sum_{k=1}^{L-1} l_{\min}\big(\hat{\mathbf{x}}_{\text{oh},k}^{(1)}, \ldots, \hat{\mathbf{x}}_{\text{oh},k}^{(P)}, \mathbf{x}_{\text{oh}}\big) + l_{\text{sum}}\big(\hat{\mathbf{x}}_{\text{oh},L}^{(1)}, \ldots, \hat{\mathbf{x}}_{\text{oh},L}^{(P)}, \mathbf{x}_{\text{oh}}\big), \tag{10}$$

where

$$l_{\min}\big(\hat{\mathbf{x}}_{\text{oh},k}^{(1)}, \ldots, \hat{\mathbf{x}}_{\text{oh},k}^{(P)}, \mathbf{x}_{\text{oh}}\big) = \min_{m\in\{1,2,\ldots,P\}} \|\hat{\mathbf{x}}_{\text{oh},k}^{(m)} - \mathbf{x}_{\text{oh}}\|^2$$
$$l_{\text{sum}}\big(\hat{\mathbf{x}}_{\text{oh},L}^{(1)}, \ldots, \hat{\mathbf{x}}_{\text{oh},L}^{(P)}, \mathbf{x}_{\text{oh}}\big) = \sum_{m=1}^{P} \|\hat{\mathbf{x}}_{\text{oh},L}^{(m)} - \mathbf{x}_{\text{oh}}\|^2.$$

To illustrate how it works, we provide a toy example and explain the reasons.

Example 1: Consider a PDN with $P = 2$ and $L = 10$. Let $\hat{m}_k = \arg \min_{m\in\{1,2\}} \|\hat{\mathbf{x}}_{\text{oh},k}^{(m)} - \mathbf{x}_{\text{oh}}\|^2$. In jth iteration, we have $\hat{m}_k = 1$ for $k = 1,3,7,9$ and $\hat{m}_k = 2$ for $k = 2,4,5,6,8$ corresponding to a training data set and the parameter set θ_{j-1} given from the previous iteration. Taking the gradient descent method as an example, the update in jth iteration is written as

$$\theta_j = \theta_{j-1} - \eta \nabla_\theta \text{Loss}\big|_{\theta=\theta_{j-1}} \tag{11}$$

where η is a step size, and

$$\begin{aligned}
&\nabla_\theta \text{Loss}\big|_{\theta=\theta_{j-1}} \\
&= \nabla_\theta \Big(\sum_{k=1}^{9} l_{\min}\big(\hat{\mathbf{x}}_{\text{oh},k}^{(1)}, \hat{\mathbf{x}}_{\text{oh},k}^{(2)}, \mathbf{x}_{\text{oh}}\big) \\
&\quad + l_{\text{sum}}\big(\hat{\mathbf{x}}_{\text{oh},10}^{(1)}, \hat{\mathbf{x}}_{\text{oh},10}^{(2)}, \mathbf{x}_{\text{oh}}\big)\Big)\Big|_{\theta=\theta_{j-1}} \\
&= \begin{bmatrix} \nabla_{\theta^{(1)}} \Big(\sum_{k\in\{1,3,7,9,10\}} \|\hat{\mathbf{x}}_{\text{oh},k}^{(1)} - \mathbf{x}_{\text{oh}}\|^2\Big)\Big|_{\theta^{(1)}=\theta_{j-1}^{(1)}} \\ \nabla_{\theta^{(2)}} \Big(\sum_{k\in\{2,4,5,6,8,10\}} \|\hat{\mathbf{x}}_{\text{oh},k}^{(2)} - \mathbf{x}_{\text{oh}}\|^2\Big)\Big|_{\theta^{(2)}=\theta_{j-1}^{(2)}} \end{bmatrix}.
\end{aligned} \tag{12}$$

We observe that the update to the parameter set $\theta^{(1)}$ (corresponding to $\text{Net}^{(1)}$) is related to losses from the layers $\text{Ly}_k^{(1)}$, $k = 1,3,7,9,10$, while the parameter set $\theta^{(2)}$ is updated by reflecting losses from the layers $\text{Ly}_k^{(2)}$, $k = 2,4,5,6,8,10$. Through this layer puncturing, the similarity between two networks is reduced.

Note that if the training data set is changed, the layer indices applied in the updates for $\theta^{(1)}$ and $\theta^{(2)}$ will also be changed although they do not overlap each other except for the last layer.

Based on Example 1, we explain the reasons for the performance improvement by applying the loss function as follows.

- $l_{\min}$ makes the minimum loss smaller and the total performance is improved by (9).
- On the other hand, by applying $l_{\min}$, not all the L layers are considered in the update to optimize a single detection network $\text{Net}^{(m)}$ (see (12)). Instead, each of the detection networks is trained by applying losses from parts of layers with partitioned indices. This reduces similarities between detection networks, $\text{Net}^{(1)}, \ldots, \text{Net}^{(P)}$, and results in a considerable diversity effect applying (9).
- Besides, any of the last detection layers, $\text{Ly}_L^{(m)}, m = 1,2,\ldots,P$, cannot be punctured and must attend training since the comparison in (9) is done with the last layers.

C. Training Detail

The training is implemented on the TensorFlow frameworks [13] by applying the Adam optimizer, a variation of the stochastic gradient descent method [14].

The goal of deep learning detectors is to train the detection network off-line and apply it in time-varying channel environments directly. To accomplish the purpose and reduce training time, we apply the following three methods in the training.

- Training the PDN by using samples of uniformly distributed signal $\mathbf{x}$ and channel matrices $\mathbf{H}^C \sim \mathcal{CN}(0, \mathbf{I})$.[1]
- Training the network under uniformly distributed SNR in a reasonable range so that the detector can be applied to various SNRs.
- Training the network using normalized input, i.e., applying $\frac{\mathbf{H}^T\mathbf{H}}{\frac{1}{N_t}\text{tr}(\mathbf{H}^T\mathbf{H})}$, $\frac{\mathbf{H}^T\mathbf{y}}{\frac{1}{N_t}\text{tr}(\mathbf{H}^T\mathbf{H})}$, and $\frac{\text{Diag}(\mathbf{H}^T\mathbf{H})}{\frac{1}{N_t}\text{tr}(\mathbf{H}^T\mathbf{H})}$ in (4) to speed up convergence.

In the training phase, $N_{\text{batch}} = 1000$ data samples of $\mathbf{x}, \mathbf{y}$, and $\mathbf{H}$ are generated according to their relationship in each iteration, and $N_{\text{iteration}} = 50000$ iterations are implemented to P detection networks with L detection layers. We initialize $\mathbf{v}_0 = \mathbf{1}$ and $\hat{\mathbf{x}}_0 = \mathbf{0}$. In the jth iteration, we compute the bit error rate (BER), P_b, and compare it to a minimum error $P_{b-\min}$. If $P_b < P_{b-\min}$, the parameters in θ_j are saved to θ^* and $P_{b-\min}$ is set to P_b. After $N_{\text{iteration}}$ iterations, we have the final parameter set θ^*.

Once θ^* is determined, the transmitted signal in the time-varying MIMO fading channel can be detected in real time.

[1]The reason for choosing the channel model is that Rayleigh fading channel is a reasonable channel model in urban environments.

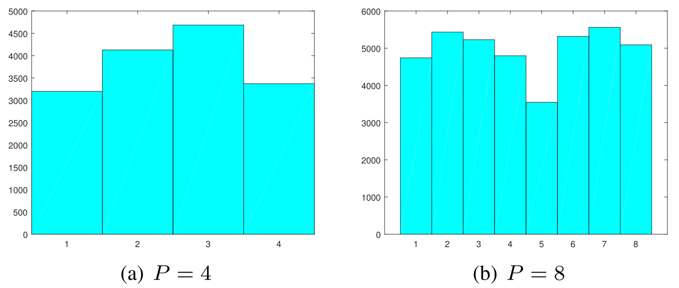

Fig. 3. Frequency of the detection network indices for the PDN with $L = 20$ in the MIMO Rayleigh fading channel with $N_r = N_t = 20$ in $\rho = 12$ dB.

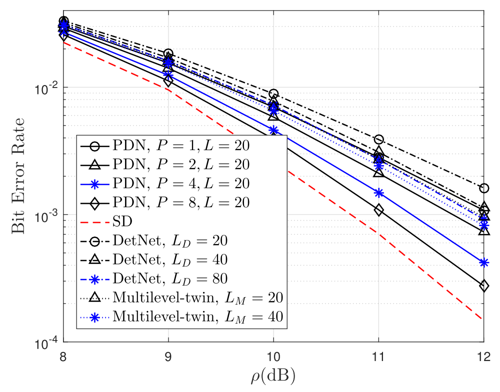

Fig. 4. BER comparison of deep learning detectors using QPSK over the MIMO Rayleigh fading channel with $N_r = 20, N_t = 20$.

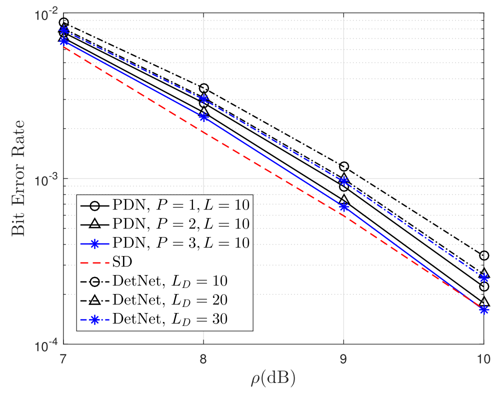

Fig. 5. BER comparison of deep learning detectors using QPSK over the MIMO Rayleigh fading channel with $N_r = 30, N_t = 20$.

IV. Simulation Results

In the previous section, we propose a parallel detection network which improves the drawback of the DetNet [9]. First, we clarify the improvements of the proposed PDN compared to the DetNet [9].

i) The PDN has an additional input of $\mathrm{Diag}(\mathbf{H}^T\mathbf{H})\hat{\mathbf{x}}_{k-1}^{(m)}$ in each detection layer which denotes the signals with the maximum achievable channel powers. This improves performance by leading the machine to the best performance signals. Moreover, the PDN has one input sublayer and one output sublayer and no hidden sublayers in each detection layer unlike the DetNet. To simply compare their complexity, we apply the naive calculation method, i.e., the complexity is $O(nmp)$ for the multiplication of matrices of $n \times m$ and $m \times p$, and $O(n^3)$ for the $n \times n$ matrix inversion. Let $\beta = \frac{N_r}{N_t}$. The calculations of $\mathbf{H}^T\mathbf{H}$ and $\mathbf{H}^T\mathbf{y}$ that are the common parts for the two detections require the complexity of $O\big(8\beta \times N_t^3\big)$ and $O\big(4\beta \times N_t^2\big)$, respectively. Except for these, the PDN with P detection networks and L layers requires the complexity of $O\big([12M + 16]PL \times N_t^2\big)$, and the DetNet with L_D layers has the complexity of $O\big([4M^2 + 12M + 4]L_D \times N_t^2\big)$. When $L_D = PL$, the PDN is less complicated than the DetNet for any $M \geq 2$.

ii) Ultimately, the PDN divides a single deep detection network into multiple parts and arranges them in parallel without connection. By designing a specific loss function, a diversity effect is obtained from multiple parallel detection networks.

To show the diversity effect directly, we compare frequencies of the detection network indices of $1, \ldots, P$ when all P detection networks have different detection results.[2] Fig. 3 shows the frequencies of the detection network indices of the PDN with $L = 20$ for the quadrature phase shift keying (QPSK) modulation in the MIMO Rayleigh fading channel with $N_r = N_t = 20$ in $\rho = 12$ dB. From the figures, we can find that the occurrences of the indices are relatively evenly distributed from 1 to P for both cases of $P = 4$ and $P = 8$. This shows that the parallel structure of the PDN brings a diversity effect.

[2]If any two networks have the same detection results, the PDN will always choose the network with the lower index. Thus, if we look at the frequency of network indices without any conditions, the first index will of course appear the most and then other indices will appear sequentially.

Now we are ready to compare the performance. Among various sub-optimal detectors, we compare the proposed detection algorithm and the DetNet to the SD in [4] for the case of $N_t = 20$ and $N_t = 30$. Since the SD cannot be completed in a reasonable time for larger size of N_t, we apply another kind of deep learning detection, OAMP-Net [7], for reference in the case of $N_t = 64$. Moreover, there is one more deep learning detector, the multilevel MIMO detection and its twin structure that trains two multilevel MIMO detections separately and applies the better one [10]. Since the paper [10] did not suggest the multi-plateau sigmoid function for the case of QPSK ($M = 2$ case) and the loss function, we compare the performance for $N_r = N_t = 20$ by appropriately selecting a multi-plateau sigmoid function and a loss function. For the deep learning detectors, we apply the same batch size of $N_{\text{batch}} = 1000$ in each iteration and $N_{\text{iteration}} = 50000$ iterations in the training.

Figs. 4, 5, 6, and 7 show the BERs of the PDN and DetNet using QPSK in the MIMO Rayleigh fading channel for the cases of $N_r = N_t = 20$, $N_r = 30, N_t = 20$, $N_r = N_t = 30$, and $N_r = N_t = 64$, respectively, where the DetNet applies L_D detection layers, and the multilevel-twin uses L_M layers in each multilevel MIMO detector and so has $2L_M$ equivalent layers. For the PDN, the performance improvement achieved by increasing the number of the serial detection layers is less than the performance improvement achieved by increasing P when the total number of detection layers exceeds L. Thus we set $L = 20$ in the cases of $N_r = N_t = 20$ and $N_r = N_t = 30$, and $L = 10$ in the cases of $N_r = 30, N_t = 20$ and $N_r = N_t = 64$.

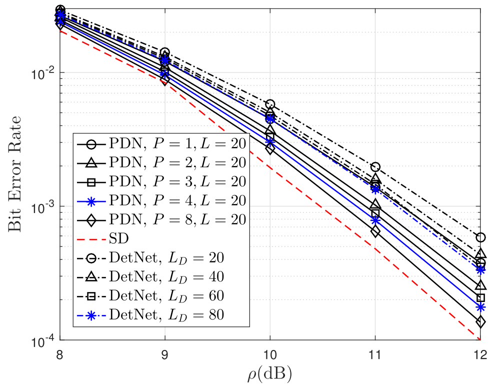

Fig. 6. BER comparison of deep learning detectors using QPSK over the MIMO Rayleigh fading channel with $N_r = 30, N_t = 30$.

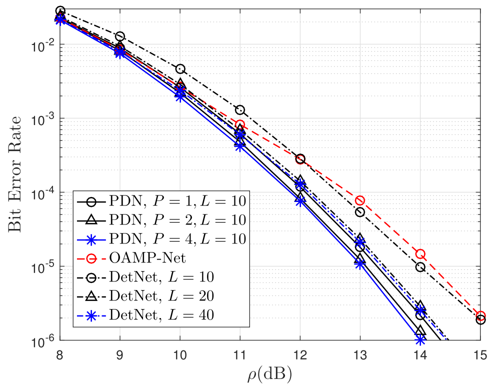

Fig. 7. BER comparison of deep learning detectors using QPSK over the MIMO Rayleigh fading channel with $N_r = 64, N_t = 64$.

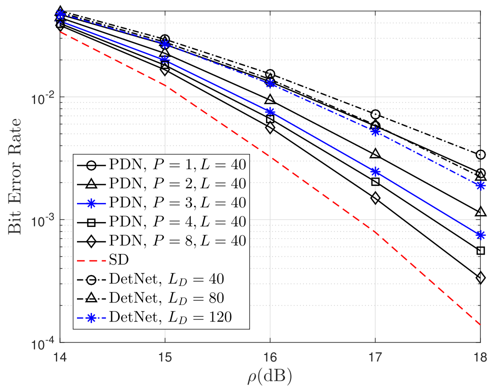

Fig. 8. BER comparison of deep learning detectors using 16QAM over the MIMO Rayleigh fading channel with $N_r = 25, N_t = 20$.

From the curves, one can observe as follows.

- The DetNet performs quite well compared to the SD for the case of $N_r = 30, N_t = 20$, but not for the cases of $N_r = N_t = 20$ and $N_r = N_t = 30$. This illustrates the motivation of this letter.
- Comparing the PDN and the DetNet for $L_D = PL$ (with the same markers in figures), the PDN obtains much better performance than the DetNet.
- The performance of the PDN overcomes that of the DetNet even in a single detection network ($P = 1$). This means that the input $\text{Diag}(\mathbf{H}^T\mathbf{H})\hat{\mathbf{x}}_{k-1}^{(m)}$ in (4) definitely helps improve performance and demonstrates the related explanation in i).
- The PDN greatly improves performance as P increases, while the DetNet does not as L_D increases for all cases of N_r, N_t. This demonstrates the diversity effect of the proposed PDN.
- With the same equivalent number of detection layers, the PDN achieves much better performance than the multilevel-twin detection for the case of $N_r = N_t = 20$ and the OAMP-Net for the case of $N_r = N_t = 64$.

Fig. 8 shows the performance using 16 quadrature amplitude modulation (QAM) in the MIMO Rayleigh fading channel with $N_r = 25, N_t = 20$. The results are similar to the case of QPSK.

V. Conclusion

In this letter, we proposed a parallel detection network (PDN) that consists of several deep learning detection networks in parallel without connection. By designing a specific loss function, the performance of the PDN improves significantly as the number of parallel networks increases in the time-varying MIMO fading channel. This overcame the problem of the DetNet, where increasing the number of detection layers does not significantly improve performance.

References

[1] R. Lupas and S. Verdu, "Linear multiuser detectors for synchronous code-division multiple-access channels," *IEEE Trans. Inf. Theory*, vol. 35, no. 1, pp. 123–136, Jan. 1989.

[2] S. Yang and L. Hanzo, "Fifty years of MIMO detection: The road to large-scale MIMOs," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 4, pp. 1941–1988, 4th Quart., 2015.

[3] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, "Closest point search in lattices," *IEEE Trans. Inf. Theory*, vol. 48, no. 8, pp. 2201–2214, Aug. 2002.

[4] A. Ghasemmehdi and E. Agrell, "Faster recursions in sphere decoding," *IEEE Trans. Inf. Theory*, vol. 57, no. 6, pp. 3530–3536, Jun. 2011.

[5] W.-K. Ma, T. N. Davidson, K. M. Wong, Z.-Q. Luo, and P.-C. Ching, "Quasi-maximum-likelihood multiuser detection using semi-definite relaxation with application to synchronous CDMA," *IEEE Trans. Signal Process.*, vol. 50, no. 4, pp. 912–922, Apr. 2002.

[6] S. Takabe, M. Imanishi, T. Wadayama, and K. Hayashi, "Trainable projected gradient detector for massive overloaded MIMO channels: Data-driven tuning approach," *IEEE Access*, vol. 7, pp. 93326–93338, 2019.

[7] H. He, C.-K. Wen, S. Jin and G. Y. Li, "A model-driven deep learning network for MIMO detection," in *Proc. IEEE GlobalSIP Conf.*, Anaheim, CA, USA, Nov. 2018, pp. 584–588.

[8] N. Samuel, T. Diskin and A. Wiesel, "Deep MIMO detection," in *Proc. IEEE 18th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Sapporo, Japan, Jul. 2017, pp. 1–5.

[9] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Trans. Signal Process.*, vol. 67, no. 10, pp. 2554–2564, May 2019.

[10] V. Corlay, J. J. Boutros, P. Ciblat, and L. Brunel, "Multilevel MIMO detection with deep learning," in *Proc. IEEE Asilomar Conf. Signals, Syst., Comput.*, Pacific Grove, CA, USA, 2018, pp. 1805–1809.

[11] J. R. Hershey, J. L. Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," 2014, *arXiv:1409.2574*. [Online]. Available: https://arxiv.org/abs/1409.2574.

[12] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[13] M. Abadi *et al.*, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," 2016, *arXiv:1603.04467*. [Online]. Available: http://arxiv.org/abs/1603.04467

[14] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: https://arxiv.org/abs/1412.6980.