Learnable MIMO Detection Networks Based on Inexact ADMM

Minsik Kim and Daeyoung Park, *Member, IEEE*

***Abstract*— In this article, we present a new iterative MIMO detection algorithm based on inexact alternating direction method of multipliers. Each iteration is considered as a neural network layer with learnable parameters, which are optimized by the stochastic gradient descent algorithm with a training data set of the received vectors and the ground truth transmitted signals. Numerical results show that the proposed algorithm outperforms the existing learnable detection network and it achieves near-optimal performance close to the sphere decoder in the case of a large number of receive antennas.**

***Index Terms*— MIMO detection, alternating direction method of multipliers, neural networks.**

I. Introduction

MULTI-ANTENNA communication plays a very important role to achieve high spectral efficiency using the limited radio resources. Multiple input and multiple output (MIMO) systems increase the degrees of freedom so that we can transmit multiple data streams at the same time and the same frequency [1]. In cellular systems, the base station can simultaneously transmit or receive data from multiple users thanks to the MIMO processing. When the transmitters have knowledge about the channel state information, they can preprocess signals such that the transmitted signals do not interfere with each other at the receiver. However, requiring the channel information at the transmitters can be a big burden when there are many antennas. Thus, in this open loop MIMO system, the transmitter sends multiple data symbols without using the channel information, which creates co-channel interference at the receiver. In this case, all signal processing burdens are placed on the receivers.

In order to achieve good performance, the receivers have to detect multiple symbols jointly. We consider single-user MIMO systems, in which there are one transmitter with multiple antennas and one receiver with multiple antennas, and uplink MIMO systems, where one receiver with multiple antennas simultaneously supports multiple single-antenna transmitters. However, the optimal joint MIMO detection problem is a non-deterministic polynomial-time hard (NP-hard) [2] and non-convex problem. Thus, there have been a lot of algorithms that tried to solve this problem with low computational complexity [3]–[6]. Near-optimal algorithms such as the sphere decoder have extremely high complexity, while low complexity algorithms such as linear detectors do not perform well. Between these two extreme cases, there are many kinds of algorithms with reasonable computational complexity. For the history of the open-loop MIMO detection algorithms, the readers are referred to [3].

Recently, signal processing algorithms such as the sparse signal recovery and the MIMO detection are designed with data-driven learning approaches based on deep neural networks. Thanks to the universal approximation theorem, neural networks have capability to approximate any continuous functions if they are well trained with enough data [7], [8]. For example, sparse signal recovery problems have been solved based on the system model-based iterative algorithms such as orthogonal matching pursuit (OMP) [9] and iterative shrinkage thresholding algorithm (ISTA) [10]. The data-driven signal recovery algorithms based on deep neural networks [11]–[13] outperform the model-based algorithms in terms of signal recovery error and convergence speed. The neural networks are more advantageous due to faster convergence to good approximate solutions [14]. In addition, they are well suited for parallel processing, enabling fast batch processing. Thus, we can run the neural networks in batches using parallel processors such as Graphics Processing Units (GPUs) to speed up execution very quickly.

As one of learnable MIMO detection networks, the detection network (DetNet) was proposed [15], [16]. DetNet was designed based on the proximal gradient descent method with additional learnable parameters. Its error performance is close to the sphere decoder in the case of BPSK with a large number of receive antennas, but is far from the optimal performance in the case of higher modulations with not so many receive antennas.

In this article, we present a new MIMO detection algorithm using an inexact alternating direction method of multiplier (ADMM) algorithm [17], [18]. It has some parameters that significantly affect the error performance. We optimize them using a data set of the received vectors and the ground truth transmitted signals. Our main contributions are summarized as follows.

Manuscript received October 11, 2019; revised March 8, 2020 and July 21, 2020; accepted September 17, 2020. Date of publication October 1, 2020; date of current version January 8, 2021. This work was supported in part by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (No. 2019R1A2C1005512) and in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) funded by the Korea Government (MSIT) (Artificial Intelligence Convergence Research Center, Inha University) under Grant 2020-0-01389. The associate editor coordinating the review of this article and approving it for publication was A. El Gamal. *(Corresponding author: Daeyoung Park.)*

The authors are with the Department of Information and Communication Engineering, Inha University, Incheon 22212, South Korea (e-mail: dpark@inha.ac.kr).

Color versions of one or more of the figures in this article are available online at https://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TWC.2020.3026471

1) We present a new MIMO detection algorithm with learnable parameters based on ADMM. Since an algorithm directly derived from ADMM is as hard as the original algorithm, we propose an inexact ADMM update based on the majorization-minimization method, which greatly reduces complexity while achieving near-optimal performance. For convex compressive sensing problems, the inexact linearlized ADMM algorithms were also used [19], [20], but they do not provide how to demodulate higher order modulation symbols. Thus, we introduce a projection operator using hyperbolic tangent for the higher order demodulation operator. In [21], MIMO detection problems were transformed into convex compressive sensing problems and the ADMM algorithms were applied to the transformed problems. Their performance is quite good but is not close to the performance of the optimal detector. For low latency communications, proximal Jacobian ADMM algorithms were applied to the MIMO detection problems [22], which speedups the detection at the cost of degraded performance. In our proposed algorithm, we introduce the relaxation parameter and the momentum acceleration parameter to the inexact ADMM in order to accelerate the algorithm convergence. Instead of performing exhaustive numerical simulations to search for the optimal parameters, we learn the parameters from a training data set of the received vectors and the ground truth transmitted signals.
2) The proposed ADMM network has many advantages. First, we can train the proposed ADMM Net much faster than DetNet with less amount of training data. Second, thanks to the parallel processing of GPUs, when its batch size is 10, it is about 50 times faster than the sphere decoder in terms of the detection time and even faster than the simple zero-forcing detector without batch processing. Third, it brings significant performance improvement over DetNet in terms of symbol errors.

This article is organized as follows: Section II briefly describes the MIMO system model. We present learnable detection networks in Section III and propose a new MIMO detection scheme based on ADMM in Section IV. Section V makes a comparison with DetNet in terms of network structure and the number of learnable parameters. We provide numerical results in Section VI and conclude this article in Section VII.

II. System Model

We consider the MIMO system model

$$\bar{\mathbf{y}} = \bar{\mathbf{H}}\bar{\mathbf{x}} + \bar{\mathbf{w}}, \tag{1}$$

where $\bar{\mathbf{H}} \in \mathbb{C}^{\bar{N}\times\bar{M}}$ is a channel matrix with $\bar{N} \geq \bar{M}$, $\bar{\mathbf{x}} \in \bar{\mathcal{S}}^{\bar{M}}$ is a transmitted signal from finite square signal constellation $\bar{\mathcal{S}}$ such as 4QAM, 16QAM, and 64QAM, and $\bar{\mathbf{w}} \sim \mathcal{CN}(0, \sigma^2\mathbf{I})$ is an additive white Gaussian noise vector. We consider open-loop MIMO systems where the transmitters do not know the channel information. We assume $\bar{\mathbf{H}}$ is a full rank matrix and each transmit antenna conveys independent signals. The receiver jointly detects the $\bar{M}$ symbols. The MIMO system models in (1) include the two cases:

- Single user MIMO (SU-MIMO): One transmitter with $\bar{M}$ antennas and one receiver with $\bar{N}$ antennas
- Uplink MIMO: $\bar{M}$ transmitters each with single antenna and one receiver with $\bar{N}$ antennas

The MIMO detection problem is to recover $\bar{\mathbf{x}}$ from $\bar{\mathbf{y}}$ and $\bar{\mathbf{H}}$, which was intensively reviewed in [3].

The receiver may rearrange (1) to obtain the equivalent real channel model

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

with

$$\mathbf{y} = \begin{bmatrix} \Re(\bar{\mathbf{y}}) \\ \Im(\bar{\mathbf{y}}) \end{bmatrix}, \quad \mathbf{x} = \begin{bmatrix} \Re(\bar{\mathbf{x}}) \\ \Im(\bar{\mathbf{x}}) \end{bmatrix}, \quad \mathbf{w} = \begin{bmatrix} \Re(\bar{\mathbf{w}}) \\ \Im(\bar{\mathbf{w}}) \end{bmatrix}$$
$$\mathbf{H} = \begin{bmatrix} \Re(\bar{\mathbf{H}}) & -\Im(\bar{\mathbf{H}}) \\ \Im(\bar{\mathbf{H}}) & \Re(\bar{\mathbf{H}}) \end{bmatrix}, \tag{3}$$

where $\mathbf{y} \in \mathbb{R}^N$, $\mathbf{x} \in \mathcal{S}^M$, $\mathbf{w} \in \mathbb{R}^N$, $\mathbf{H} \in \mathbb{R}^{N\times M}$ with $N = 2\bar{N}$ and $M = 2\bar{M}$ are the received signal, the transmitted signal, the noise, and the channel matrix, respectively. Here, $\mathcal{S} = \Re(\bar{\mathcal{S}})$ represents the real part of complex constellations. For example, the real parts of 4QAM and 16QAM corresponds to the BPSK and 4PAM, respectively. Thus, we focus on the real MIMO channel model (2) in the rest of the paper.

The channel matrix $\mathbf{H}$ can be modelled in two ways: The fixed channel (FC) system is the system in which the channel matrix is constant all the time, and the varying channel (VC) system is the system in which the channel changes over time, but we assume the receiver has full knowledge of $\mathbf{H}$ with the aid of a channel estimator. The MIMO detection problem is to recover the transmitted signal $\mathbf{x}$ from $\mathbf{y}$ for given $\mathbf{H}$

$$\begin{aligned} &\text{minimize } \|\mathbf{y} - \mathbf{H}\mathbf{x}\|^2 \\ &\text{subject to } \mathbf{x} \in \mathcal{S}^M. \end{aligned} \tag{4}$$

III. Learnable Detection Network

Any detection algorithms can be expressed as an explicit function of channel $\mathbf{H}$ and received signal $\mathbf{y}$

$$\hat{\mathbf{x}} = g(\mathbf{H}, \mathbf{y}). \tag{5}$$

For example, zero-forcing (ZF) detector is $\mathbf{H}^\dagger\mathbf{y}$ followed by an element-wise symbol demodulation, where $\mathbf{H}^\dagger$ is the pseudo-inverse. The maximum likelihood (ML) detector is optimal, but its huge computational complexity is a big burden in a massive MIMO detection problem. Instead of fixing g, we may use a collection of functions that are parameterized by parameter Ω

$$\hat{\mathbf{x}} = g(\mathbf{H}, \mathbf{y}; \Omega). \tag{6}$$

We suppose g is differentiable with respect to Ω, which enables to optimize g using the gradient descent algorithms. We adjust parameter Ω by using the stochastic gradient method to minimize some loss function $l(\hat{\mathbf{x}}, \mathbf{x}^*)$ such as $\|\hat{\mathbf{x}} - \mathbf{x}^*\|_2^2$, where $\mathbf{x}^*$ is the ground truth signal. In many cases, we may update the current estimate of $\hat{\mathbf{x}}$ using a previous estimate. Let $\mathbf{x}_k$ denote the k-th iterate estimation of $\mathbf{x}$. Then, we can

express the $k+1$-th iteration as the function of the k-th iteration

$$\begin{aligned}\mathbf{x}_{k+1} &= g_k(\mathbf{H}, \mathbf{y}, \mathbf{x}_k; \Omega_k), \quad k = 0, 1, \cdots, L-1 \\ \hat{\mathbf{x}} &= \mathbf{x}_L \end{aligned} \tag{7}$$

with parameter Ω_k, where L is the total number of iterations or layers. Universal approximation theorem says any continuous function on a compact domain can be well approximated within any desired accuracy by neural networks [7], [8]. Thanks to this theorem, we can limit g_k to a function expressed as a neural network layer mapping. For neural networks, g_k is a parameterized mapping from the k-th layer to the $k+1$-th layer.

A most simple straight-forward approach to adopt neural networks in algorithm design is to use fully connected multi-layer networks (FullyCon). The non-linear activation function $\phi(\mathbf{x})$ such as rectified linear unit (ReLU) maps the weighted sum of the previous layer hidden variables to the next layer hidden variables [15], [16]

$$\begin{aligned}\mathbf{q}_0 &= \mathbf{y} \\ \mathbf{q}_{k+1} &= \phi(\mathbf{W}_k \mathbf{q}_k + \mathbf{b}_k), \quad k = 0, 1, \cdots, L-1 \\ \mathbf{x}_{oh} &= \mathbf{W}_L \mathbf{q}_L + \mathbf{b}_L \\ \hat{\mathbf{x}} &= \mathbf{f}_{oh}(\mathbf{x}_{oh}), \end{aligned} \tag{8}$$

where $\mathbf{f}_{oh}(\mathbf{x})$ denotes the one-hot mapping from binary vectors to signal constellations [16]. Due to this one-hot mapping, as the number of transmit antenna increases, there is the problem of curse of dimensionality. In the next section, the proposed algorithm works only on the symbol level not on the bit level to avoid this problem. In FullyCon, the trainable parameters are $\Omega = \{\mathbf{W}_k, \mathbf{b}_k\}_{k=0}^{L-1}$. Since FullyCon relies only on $\mathbf{y}$ and does not take $\mathbf{H}$ as an input, it does not work well in varying channels. Even an attempt to employ $\mathbf{H}$ as input for fully connected networks also failed [15], [16]. Thus, it is necessary to design better neural networks that can be applied to general varying channels.

For better performance we may consider recurrent neural networks (RNN) that introduces hidden state $\mathbf{s}_k$ that summarizes all necessary information about the past states of network

$$\begin{aligned}[\mathbf{x}_{k+1}, \mathbf{s}_{k+1}] &= g_k(\mathbf{H}, \mathbf{y}, \mathbf{x}_k, \mathbf{s}_k; \Omega_k), \quad k = 0, 1, \cdots, L-1 \\ \hat{\mathbf{x}} &= \mathbf{x}_L. \end{aligned} \tag{9}$$

We update Ω_k by using the stochastic gradient method to minimize some loss function.

We can design g_k based on the conventional optimization algorithms instead of simply applying fully connected layers. The derived iterative algorithms can play role of the skeleton of trainable neural networks. For this purpose, we consider a regularized minimization problem

$$\text{minimize } f(\mathbf{x}) + f_r(\mathbf{x}) \tag{10}$$

where $f(\mathbf{x})$ is a smooth convex loss function such as l_2-norm and $f_r(\mathbf{x})$ is a non-smooth (possibly non-convex) regularizer. We derive layer-wise structure from the forward step, i.e., the one-step gradient descent of smooth function $f(\mathbf{x})$. We may augment some adjustable parameters to this layer-wise structure, which makes the overall network a differentiable function with those parameters that can be learned by the gradient method. Then, we introduce non-linearity based on the backward step, i.e., the proximal operator of $f_r(\mathbf{x})$. The non-linear functions can also have parameters that can be learned by the gradient descent. Each iteration in the proximal gradient descent method corresponds to a hidden layer in the unfolded neural networks. We can train the network to find optimal parameters using training data. This kind of methods to learn algorithms have been applied to sparse signal recovery problems [11]–[13].

More specifically, we now derive a proximal gradient descent. We may use a surrogate function $\hat{f}(\mathbf{x}; \mathbf{x}_k)$ to upper-bound $f(\mathbf{x})$

$$\begin{aligned} f(\mathbf{x}) &\le \hat{f}(\mathbf{x}; \mathbf{x}_k) \\ &\equiv f(\mathbf{x}_k) + \nabla f(\mathbf{x}_k)^T (\mathbf{x} - \mathbf{x}_k) + \frac{1}{2\delta_k} \|\mathbf{x} - \mathbf{x}_k\|_2^2, \end{aligned} \tag{11}$$

where $\mathbf{x}_k$ is the estimate in the k-th iteration and δ_k is small enough to satisfy the inequality.[1] Then, the majorization-minimization method provides a series of optimization problems

$$\begin{aligned}\mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}} \hat{f}(\mathbf{x}; \mathbf{x}_k) + f_r(\mathbf{x}) \\ &= \arg\min_{\mathbf{x}} f_r(\mathbf{x}) + \frac{1}{2\delta_k} \|\mathbf{x} - (\mathbf{x}_k - \delta_k \nabla f(\mathbf{x}_k))\|_2^2 \\ &= \text{Prox}_{\delta_k f_r(\mathbf{x})}(\mathbf{x}_k - \delta_k \nabla f(\mathbf{x}_k)), \end{aligned} \tag{12}$$

where $\text{Prox}_{\delta f_r(\mathbf{w})}(\mathbf{x})$ is called the proximal operator

$$\text{Prox}_{\delta f_r(\mathbf{w})}(\mathbf{x}) = \arg\min_{\mathbf{w}} f_r(\mathbf{w}) + \frac{1}{2\delta} \|\mathbf{w} - \mathbf{x}\|_2^2. \tag{13}$$

The iterative method to update $\mathbf{x}$ in (12) is called the proximal gradient descent method, or the forward-backward splitting method [18]. It refines $\mathbf{x}_{k+1}$ by using the gradient descent update of $\mathbf{x}_k$ followed by the proximal operator. It converges to the global optimal solution when the original objective function is convex. Proximal operators are generalized projection operators. Some functions have closed-form expressions for their proximal operators. Interested readers are referred to [18].

MIMO detection algorithms can also be designed using the proximal gradient descent methods [15], [16]. The critical problem of FullyCon detector is that it only relies on the $\mathbf{y}$. For optimal detection of transmitted signals, sufficient statistics are $\mathbf{H}^T\mathbf{y} = \mathbf{H}^T\mathbf{H}\mathbf{x} + \mathbf{H}^T\mathbf{w}$. In other words, we do not lose any optimality if we use $\mathbf{H}^T\mathbf{y}$ instead of $\mathbf{y}$ in the detector. Thus, the main ingredients in the trainable networks are $\mathbf{H}^T\mathbf{y}$ and $\mathbf{H}^T\mathbf{H}\mathbf{x}$. From (12), the proximal gradient descent method for the MIMO detectors takes the expression

$$\begin{aligned}\mathbf{x}_{k+1} &= \Pi_{\mathcal{S}^M} \left(\mathbf{x}_k - \delta_k \nabla_{\mathbf{x}} \frac{1}{2} \|\mathbf{y} - \mathbf{H}\mathbf{x}\|_2^2 \Big|_{\mathbf{x}=\mathbf{x}_k} \right) \\ &= \Pi_{\mathcal{S}^M}(\mathbf{x}_k - \delta_k \mathbf{H}^T\mathbf{y} + \delta_k \mathbf{H}^T\mathbf{H}\mathbf{x}_k). \end{aligned} \tag{14}$$

$\Pi_{\mathcal{S}^M}(\mathbf{y}) = \arg\min_{\mathbf{x} \in \mathcal{S}^M} \|\mathbf{x} - \mathbf{y}\|_2^2$ denotes the element-wise demodulator for given signal constellations, which plays the role of a proximal operator.

[1] If f is an L_f-smooth function, i.e., $\|\nabla f(\mathbf{x}) - \nabla f(\mathbf{y})\| \le L_f \|\mathbf{x} - \mathbf{y}\|$, then we require $0 < \delta_k \le 1/L_f$ to satisfy (11).

One of neural networks for MIMO detection based on the proximal gradient descent method is DetNet (Detection Network) [16]

$$\begin{aligned}\mathbf{q}_{k+1} &= \mathbf{x}_k - \delta_{1,k}\mathbf{H}^T\mathbf{y} + \delta_{2,k}\mathbf{H}^T\mathbf{H}\mathbf{x}_k \\ \mathbf{z}_{k+1} &= \phi\left(\mathbf{W}_{1,k}\begin{bmatrix}\mathbf{q}_{k+1}\\ \mathbf{v}_k\end{bmatrix} + \mathbf{b}_{1,k}\right) \\ \mathbf{x}_{oh,k+1} &= \mathbf{W}_{2,k}\mathbf{z}_{k+1} + \mathbf{b}_{2,k} \\ \mathbf{x}_{k+1} &= \mathbf{f}_{oh}(\mathbf{x}_{oh,k+1}) \\ \mathbf{v}_{k+1} &= \mathbf{W}_{3,k}\mathbf{z}_{k+1} + \mathbf{b}_{3,k}. \end{aligned} \tag{15}$$

There are two important steps borrowed from the proximal gradient descent method: First, $\mathbf{q}_k$ is obtained by the one-step gradient descent. Second, $\phi(\mathbf{x})$ is the element-wise non-linear operator such as ReLU and sigmoid to mimic the proximal operator. However, this network also has still the one-hot mapping which causes the curse of dimensionality problem when the number of transmit antenna is large. The trainable parameters are $\Omega = \{\mathbf{W}_{1,k}, \mathbf{b}_{1,k}, \mathbf{W}_{2,k}, \mathbf{b}_{2,k}, \mathbf{W}_{3,k}, \mathbf{b}_{3,k}, \delta_{1,k}, \delta_{2,k}\}_{k=0}^{L-1}$.

DetNet performs better than FullyCon in varying channels due to the following reasons: First, thanks to $\mathbf{H}^T\mathbf{y}$ and $\mathbf{H}^T\mathbf{H}\mathbf{x}_{k-1}$ in calculating the gradient descent $\mathbf{q}_k$, the designed network performs well in general varying channels, because the trainable parameters are independent of channel realizations $\mathbf{H}$. However, this is not the case for FullyCon. Second, DetNet introduces a mysterious lifting variable $\mathbf{v}_k$ to transfer the previous layer information to the next layer, which reminds us of a state in the recurrent neural network, i.e., $\mathbf{s}_k = \mathbf{v}_k$ in (9). It will be very demanding to investigate what kind of information $\mathbf{v}_k$ conveys to the next layer. The drawback of DetNet is that its performance is not good enough for higher order constellations and its training time is very long. In the next section, we derive a new detection network based on ADMM.

IV. Inexact ADMM-Based Detection Network

In the last section, we discussed the design of layer-wise structures from iterative algorithms with adjustable parameters. As one example, DetNet was designed based on a proximal gradient descent method applied to a regularized minimization problem (10). This kind of regularized minimization problems can also be solved using the alternating direction method of multipliers (ADMM) [17]. For example, ADMM is much faster than the proximal gradient descent method in the case of sparse signal recovery problems that are also formulated in the form of (10) [17]. Motivated by this, we apply ADMM to the MIMO detection problem.

A. ADMM-Based Detection

We formulate the MIMO detection problem with $\lambda > 0$

$$\begin{aligned}\min_{\mathbf{x}\in\mathbb{R}^M, \mathbf{z}\in\mathbb{R}^N} \quad & I_{\mathcal{S}^M}(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{z}\|_2^2 \\ \text{s.t. } & \mathbf{z} = \mathbf{H}\mathbf{x} - \mathbf{y}, \end{aligned} \tag{16}$$

where $I_{\mathcal{S}^M}(\mathbf{x})$ denotes an indicator function for signal constellation, where it takes 0 if $\mathbf{x} \in \mathcal{S}^M$ and ∞ otherwise. The augmented scaled Lagrangian is given by

$$\begin{aligned}L(\mathbf{x},\mathbf{z},\mathbf{u}) = I_{\mathcal{S}^M}(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{z}\|_2^2 + \rho\mathbf{u}^T(\mathbf{H}\mathbf{x}-\mathbf{y}-\mathbf{z}) \\ + \frac{\rho}{2}\|\mathbf{H}\mathbf{x}-\mathbf{y}-\mathbf{z}\|_2^2, \end{aligned} \tag{17}$$

where $\rho \geq 0$. We need to minimize $L(\mathbf{x},\mathbf{z},\mathbf{u})$ with respect to the primal variables $(\mathbf{x},\mathbf{z})$ and to maximize $L(\mathbf{x},\mathbf{z},\mathbf{u})$ with respect to the dual variable $\mathbf{u}$ by using the dual ascent method. Instead of the joint minimization, we alternatively minimize $L(\mathbf{x},\mathbf{z},\mathbf{u})$ over $\mathbf{x}$ and $\mathbf{z}$ in a Gauss-Seidel fashion to derive the ADMM [17]

$$\begin{aligned}\mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}\in\mathbb{R}^M} I_{\mathcal{S}^M}(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{H}\mathbf{x}-\mathbf{y}-\mathbf{z}_k+\mathbf{u}_k\|_2^2 \\ \mathbf{z}_{k+1} &= \arg\min_{\mathbf{z}\in\mathbb{R}^N} \frac{1}{2\lambda}\|\mathbf{z}\|_2^2 + \frac{\rho}{2}\|\mathbf{H}\mathbf{x}_{k+1}-\mathbf{y}-\mathbf{z}+\mathbf{u}_k\|_2^2 \\ &= \frac{\lambda\rho}{1+\lambda\rho}(\mathbf{H}\mathbf{x}_{k+1}-\mathbf{y}+\mathbf{u}_k) \\ \mathbf{u}_{k+1} &= \mathbf{u}_k + \mathbf{H}\mathbf{x}_{k+1} - \mathbf{y} - \mathbf{z}_{k+1}. \end{aligned} \tag{18}$$

We can speedup the convergence of the optimization algorithm by introducing some momentum terms [23]. From (18), we derive the (exact) relaxed and accelerated ADMM (R-A-ADMM) [24]

$$\begin{aligned}\mathbf{x}_{k+1} &= \arg\min_{\mathbf{x}\in\mathbb{R}^M} I_{\mathcal{S}^M}(\mathbf{x}) + \frac{\rho}{2}\|\mathbf{H}\mathbf{x}-\mathbf{y}-\hat{\mathbf{z}}_k+\hat{\mathbf{u}}_k\|_2^2 \\ \mathbf{z}_{k+1} &= \arg\min_{\mathbf{z}\in\mathbb{R}^N} \frac{1}{2\lambda}\|\mathbf{z}\|_2^2 \\ &\quad + \frac{\rho}{2}\|\alpha(\mathbf{H}\mathbf{x}_{k+1}-\mathbf{y}) + (1-\alpha)\hat{\mathbf{z}}_k - \mathbf{z} + \hat{\mathbf{u}}_k\|_2^2 \\ &= \frac{\lambda\rho}{1+\lambda\rho}(\alpha(\mathbf{H}\mathbf{x}_{k+1}-\mathbf{y}) + (1-\alpha)\hat{\mathbf{z}}_k + \hat{\mathbf{u}}_k) \\ \mathbf{u}_{k+1} &= \hat{\mathbf{u}}_k + \alpha(\mathbf{H}\mathbf{x}_{k+1}-\mathbf{y}) + (1-\alpha)\hat{\mathbf{z}}_k - \mathbf{z}_{k+1} \\ \hat{\mathbf{z}}_{k+1} &= \mathbf{z}_{k+1} + \gamma_k(\mathbf{z}_{k+1}-\mathbf{z}_k) \\ \hat{\mathbf{u}}_{k+1} &= \mathbf{u}_{k+1} + \gamma_k(\mathbf{u}_{k+1}-\mathbf{u}_k), \end{aligned} \tag{19}$$

where $\gamma_k = k/(k+\mu)$ and $\alpha \in (0,2)$. Relaxation parameter α speedups the convergence and the case of $\alpha = 1$ and $\gamma_k = 0$ corresponds to the conventional ADMM given in (18). In addition, γ_k introduces the momentum terms [23] and the case of $\alpha = 1$ and $\gamma_k = k/(k+5)$ reduces to the accelerated ADMM (A-ADMM) [25]. Thus, the algorithm in (19) has a very general ADMM form with the relaxation parameter α and the momentum acceleration parameter γ_k. Moreover, in the case of strongly convex objective functions, R-A-ADMM attains the linear convergence rate, which implies the algorithm converges very fast [24].

However, R-A-ADMM in (19) has prohibitive computational complexity. Eq. (16) can be expressed as $\min_{\mathbf{x}} I_{\mathcal{S}^M}(\mathbf{x}) + \frac{1}{2\lambda}\|\mathbf{H}\mathbf{x}-\mathbf{y}\|_2^2$, which has the nearly identical form with the x-update in (19). Therefore, the x-update in Eq. (19) is as hard as Eq. (16). Since the x-update in (19) itself is hard to perform, it is a nonsense to use (19) to solve the MIMO detection problem formulated in (16) due to the huge computational complexity.

To reduce computational complexity, we have to design an inexact low-complexity solver that replaces the x-update. We may approximately solve the x-update by introducing

a proximal term as we did in deriving Eq. (12), which is inspired by [19] in the case of compressive sensing problems. We define $h(\mathbf{x}) = \frac{1}{2}\|\mathbf{H}\mathbf{x} - \mathbf{y} - \hat{\mathbf{z}}_k + \hat{\mathbf{u}}_k\|_2^2$. $h(\mathbf{x})$ is a smooth convex function that takes the expression

$$h(\mathbf{x}) \le h(\mathbf{x}_k) + \nabla h(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) + \frac{1}{2\delta_k}\|\mathbf{x} - \mathbf{x}_k\|_2^2 \quad (20)$$

for $\delta_k \le 1/L_f$, where L_f is the largest eigenvalue of $\mathbf{H}^T\mathbf{H}$. By applying the majorization-minimization method, the approximate x-update can take the expression

$$\begin{aligned}
&\mathbf{x}_{k+1} \\
&= \arg\min_{\mathbf{x}\in\mathbb{R}^M} I_{\mathcal{S}^M}(\mathbf{x}) + \rho\nabla h(\mathbf{x}_k)^T(\mathbf{x} - \mathbf{x}_k) + \frac{\rho}{2\delta_k}\|\mathbf{x} - \mathbf{x}_k\|_2^2 \\
&= \arg\min_{\mathbf{x}\in\mathbb{R}^M} I_{\mathcal{S}^M}(\mathbf{x}) + \frac{\rho}{2\delta_k}\|\mathbf{x} - \mathbf{x}_k + \delta_k\nabla h(\mathbf{x}_k)\|_2^2 \\
&= \Pi_{\mathcal{S}^M}(\mathbf{x}_k - \delta_k\nabla h(\mathbf{x}_k)) \\
&= \Pi_{\mathcal{S}^M}(\mathbf{x}_k - \delta_k\mathbf{H}^T(\mathbf{H}\mathbf{x}_k - \mathbf{y} - \hat{\mathbf{z}}_k + \hat{\mathbf{u}}_k)), \quad (21)
\end{aligned}$$

where the third equality holds due to the definition of the projection operator $\Pi_{\mathcal{S}^M}(\mathbf{y}) = \arg\min_{\mathbf{x}\in\mathcal{S}^M}\|\mathbf{x} - \mathbf{y}\|_2^2 = \arg\min_{\mathbf{x}\in\mathbb{R}^M} I_{\mathcal{S}^M}(\mathbf{x}) + c\|\mathbf{x} - \mathbf{y}\|_2^2$ for any $c > 0$. Note that it reminds us of a gradient descent followed by a proximal operator, i.e., the proximal gradient descent method given in (14). The key difference from (14) is that we correct the gradient $\mathbf{H}^T(\mathbf{H}\mathbf{x}_k - \mathbf{y})$ in the gradient descent by adding $\mathbf{H}^T(-\hat{\mathbf{z}}_k + \hat{\mathbf{u}}_k)$ to the gradient. On the other hand, DetNet does not use the proximal gradient descent method directly, but updates the gradient descent $\mathbf{q}_k$ using the multiplication of $\mathbf{W}_{1,k}$ with $\mathbf{q}_{k+1}$ and state variable $\mathbf{v}_k$ in (15). This kind of multiplication makes the training hard due to the vanishing gradient problem [26]. Thus, we expect our additive gradient correction is better than the multiplicative gradient correction in DetNet.

B. Demodulation Operator

To train neural networks, all operations should be differentiable for the gradient back-propagation. Thus, we need to design a differentiable projection operator $\Pi(\mathbf{x})$. For BPSK, we may use the element-wise sign function for $\Pi(\mathbf{x})$. However, it is not appropriate for learnable networks because it is not differentiable. In addition, it is not straightforward to demodulate higher order constellations with a simple differentiable function. In [27], the sigmoid function is used for multi-level demodulation. Similarly, we propose to use the following function

$$\begin{aligned}
\eta_{\mathcal{S}}(x;\theta) &= \sum_{i=1}^{|\mathcal{S}|-1} \tanh(\theta(x - \tau_i)) \\
\tau_i &= \frac{1}{2}(c_i + c_{i+1}), \quad i = 1, \cdots, |\mathcal{S}| - 1, \quad (22)
\end{aligned}$$

where $c_i < c_{i+1}$ for $c_i, c_{i+1} \in \mathcal{S}$. For BPSK case, we have

$$\eta_{\mathcal{S}}(x;\theta) = \tanh(\theta x), \quad (23)$$

which is close to $\mathrm{sign}(\mathbf{x})$ for large θ. In the case of 4-PAM, we have $\mathcal{S} = \{-3, -1, 1, 3\}$ and

$$\eta_{\mathcal{S}}(x;\theta) = \tanh(\theta(x+2)) + \tanh(\theta x) + \tanh(\theta(x-2)). \quad (24)$$

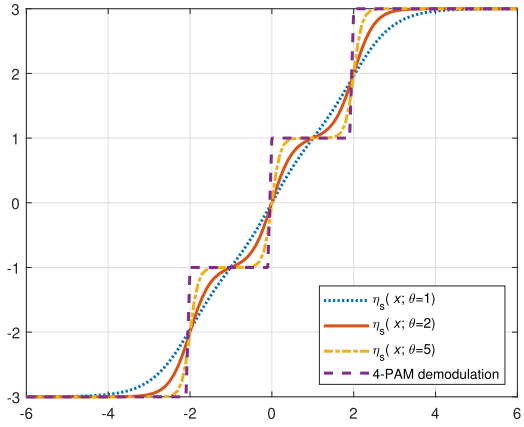


Fig. 1. Approximated demodulation for 4-PAM.

For vector input, we also simply represent $\eta_{\mathcal{S}}(\mathbf{x};\theta)$ for computing (22) in an element-wise fashion. Fig. 1 depicts the differentiable approximation $\eta_{\mathcal{S}}(x;\theta)$ of the 4-PAM demodulation with various θ. We observe that $\eta_{\mathcal{S}}(x;\theta)$ with large θ is close to the 4-PAM demodulation.

The proposed demodulation using the hyperbolic tangent can be used to produce soft decision output. Since the demodulation operation performs in an element-wise fashion, we consider a single-antenna BPSK-modulated system with $\mathcal{S} = \{-1, 1\}$ for a simple analysis. Suppose $s \in \mathcal{S}$ is an equiprobable symbol and $x = s + n$ is the symbol corrupted by Gaussian noise $n \sim \mathcal{N}(0, \sigma^2)$. Then, we have $p(x|s) = \frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{(x-s)^2}{2\sigma^2}}$ and the posteriori probability is given by

$$\begin{aligned}
p(s=+1|x) &= \frac{p(x|s=+1)p(s=+1)}{p(x|s=+1)p(s=+1) + p(x|s=-1)p(s=-1)} \\
&= \frac{1}{1 + \frac{p(x|s=-1)}{p(x|s=+1)}} = \frac{1}{1 + e^{-2x/\sigma^2}} \\
&= \frac{1}{2}(1 + \tanh(x/\sigma^2)) \\
p(s=-1|x) &= \frac{1}{2}(1 - \tanh(x/\sigma^2)). \quad (25)
\end{aligned}$$

Therefore, if the noise is Gaussian distributed with variance σ^2 and we set $\theta = 1/\sigma^2$, then $\tanh(\theta x)$ can be used to extract the soft information. This theoretically verifies our choice of $\tanh(\theta x)$ for the demodulation operator. It is for further study for the extension to the higher-order modulation.

C. Learnable ADMM Network for MIMO Detection

We simplify the proposed ADMM algorithm by introducing some auxiliary variables; the residual error $\mathbf{e}_{k+1}$, the accumulated error $\mathbf{v}_{k+1}$, and the accelerated accumulated error $\hat{\mathbf{v}}_{k+1}$

$$\begin{aligned}
\mathbf{e}_{k+1} &= \mathbf{H}\mathbf{x}_{k+1} - \mathbf{y} \\
\mathbf{v}_{k+1} &= \alpha\mathbf{e}_{k+1} + (1 - \alpha)\hat{\mathbf{z}}_k + \hat{\mathbf{u}}_k \\
\hat{\mathbf{v}}_{k+1} &= \mathbf{v}_{k+1} + \gamma_k(\mathbf{v}_{k+1} - \mathbf{v}_k). \quad (26)
\end{aligned}$$

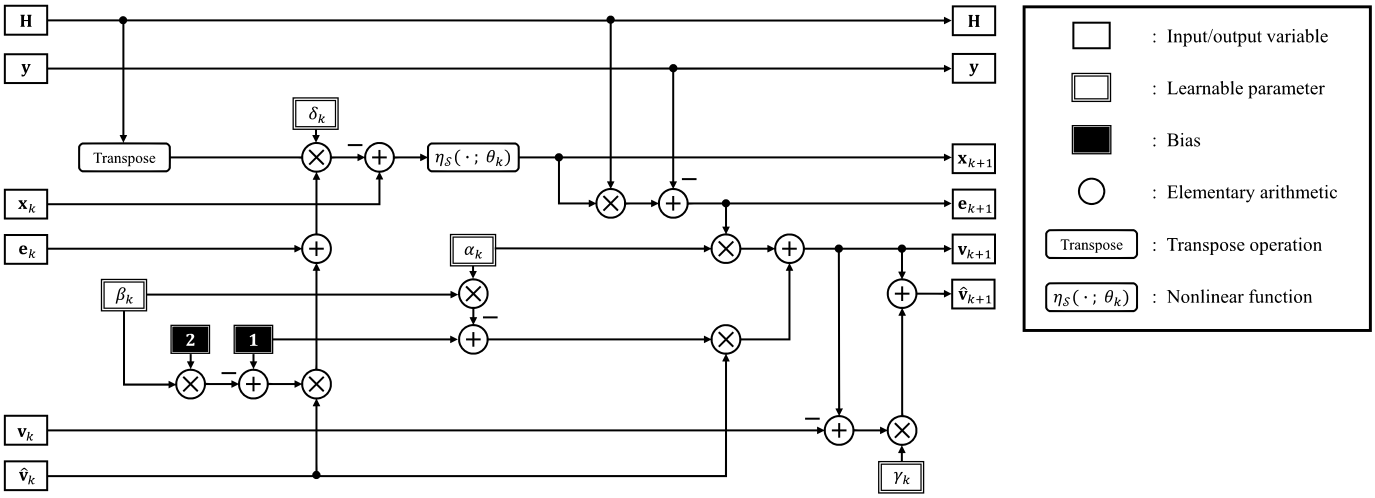


Fig. 2. Block diagram of ADMM Net.

Using these variables, we have

$$\begin{aligned}
&\mathbf{z}_{k+1} = \beta\mathbf{v}_{k+1} \quad \hat{\mathbf{z}}_{k+1} = \beta\hat{\mathbf{v}}_{k+1}\\
&\mathbf{u}_{k+1} = (1-\beta)\mathbf{v}_{k+1} \quad \hat{\mathbf{u}}_{k+1} = (1-\beta)\hat{\mathbf{v}}_{k+1}\\
&\mathbf{v}_{k+1} = \alpha\mathbf{e}_{k+1} + (1-\alpha\beta)\hat{\mathbf{v}}_k,
\end{aligned} \tag{27}$$

where $\beta = \lambda\rho/(1+\lambda\rho)$. Note that $\mathbf{z}_k$ represents the k-th estimate of the additive noise at the receiver and $\mathbf{u}_k$ represents the dual variable in the k-th iteration associated with the constraint in (16). In our problem, $\mathbf{z}_k$ and $\mathbf{u}_k$ are just a scalar multiple of the same variable $\mathbf{v}_k$. In summary, the proposed MIMO detection based on inexact ADMM is

$$\begin{aligned}
\mathbf{x}_{k+1} &= \eta_{\mathcal{S}}(\mathbf{x}_k - \delta_k\mathbf{H}^T(\mathbf{e}_k + (1-2\beta)\hat{\mathbf{v}}_k);\theta)\\
\mathbf{e}_{k+1} &= \mathbf{H}\mathbf{x}_{k+1} - \mathbf{y}\\
\mathbf{v}_{k+1} &= \alpha\mathbf{e}_{k+1} + (1-\alpha\beta)\hat{\mathbf{v}}_k\\
\hat{\mathbf{v}}_{k+1} &= \mathbf{v}_{k+1} + \gamma_k(\mathbf{v}_{k+1} - \mathbf{v}_k)
\end{aligned} \tag{28}$$

for $k = 0, 1, \ldots, L-1$.

Due to the non-convexity of $I_{\mathcal{S}^M}(\mathbf{x})$ in (16), the derived algorithm is not guaranteed to converge to the optimal solution, which is similar to DetNet. Since its performance is sensitive to the parameters $(\delta_k, \alpha, \lambda, \rho, \mu)$, we have to find the optimal ones by performing intensive numerical simulations with many different possible values.

Instead of manually searching for the optimal parameters, we can compute them with the aid of learnable neural networks. Thus, the derived iterative algorithm in (28) can play the role of a skeleton of a learnable neural network. The proposed learnable inexact ADMM network is given in Algorithm 1, where $\Omega = \{\delta_k, \alpha_k, \beta_k, \gamma_k, \theta_k\}_{k=0}^{L-1}$ are learnable parameters. Fig. 2 depicts the block diagram of the proposed inexact ADMM based detection network (ADMM Net). In general, layer-wise distinct parameters (i.e., the untied case) provide more degrees of freedom than layer-wise identical parameters (i.e., the tied case). Thus, in addition to δ_k and γ_k in (28), we allow layer-wise distinct parameters for $(\alpha_k, \beta_k, \theta_k)$, which brings forth better performance due to more expressiveness of the neural network.

Algorithm 1 ADMM Network for MIMO Detection

Input: $\mathbf{y}, \mathbf{H}$
Output: $\mathbf{x}_L$
Initialize $\mathbf{x}_0 := \mathbf{H}^\dagger\mathbf{y}$, $\mathbf{e}_0 := 0$, $\mathbf{v}_0 := \hat{\mathbf{v}}_0 := 0$,
for $k = 0 : L-1$
  $\mathbf{x}_{k+1} := \eta_{\mathcal{S}}(\mathbf{x}_k - \delta_k\mathbf{H}^T(\mathbf{e}_k + (1-2\beta_k)\hat{\mathbf{v}}_k);\theta_k)$
  $\mathbf{e}_{k+1} := \mathbf{H}\mathbf{x}_{k+1} - \mathbf{y}$
  $\mathbf{v}_{k+1} := \alpha_k\mathbf{e}_{k+1} + (1-\alpha_k\beta_k)\hat{\mathbf{v}}_k$
  $\hat{\mathbf{v}}_{k+1} := \mathbf{v}_{k+1} + \gamma_k(\mathbf{v}_{k+1} - \mathbf{v}_k)$
end

In the view of RNN in (9), ADMM-based detection network has a state variable $\mathbf{s}_k = \hat{\mathbf{v}}_k$ that transfers the summary of all necessary information about past states from k-th layer to the $k+1$-th layer. More specifically, $\mathbf{v}_k$ is a scalar multiple of the k-th estimate of the additive noise $\mathbf{z}_k$. The $k+1$-th layer uses its momentum $\gamma_k(\mathbf{v}_{k+1} - \mathbf{v}_k)$ to obtain the accelerated versions $\hat{\mathbf{v}}_{k+1}$.

For training, we consider the loss function is given by

$$l = \sum_{k=1}^{L} \|\eta_S(\mathbf{x}_k;\bar{\theta}) - \mathbf{x}^*\|_2^2, \tag{29}$$

where $\bar{\theta}$ is a large fixed constant (e.g., 100), which is not learned from data. Since the overall networks are differentiable with respect to Ω, the layer-wise structure can be used to back-propagate the gradients of the loss function.[2] Thanks to the auto-differentiation of machine-learning frameworks such as TensorFlow and PyTorch, we can learn Ω from data using the gradient descent method.

D. Learnable Gated ADMM Network for MIMO Detection

We introduce additional gating operations for performance improvement. In the numerical results, the proposed ADMM

[2]If we choose the sign function for demodulation operation, the gradient back-propagation is not possible due to the non-differentiability of the sign function. Thus, the hyperbolic tangent for the projection operator is a good choice for learnable networks.

Algorithm 2 Gated ADMM Network for MIMO Detection

1: Input: $\mathbf{y}, \mathbf{H}$
2: Output: $\mathbf{x}_L$
3: Initialize $\mathbf{x}_0 := \mathbf{H}^\dagger \mathbf{y}$, $\mathbf{e}_0 := 0$, $\mathbf{v}_0 := \hat{\mathbf{v}}_0 := 0$
4: for $k = 0 : L-1$
5: $\quad \mathbf{x}_{k+1} := \eta_{\mathcal{S}}(\mathbf{x}_k - \delta_k \mathbf{H}^T(\mathbf{e}_k + (1-2\beta_k)\hat{\mathbf{v}}_k); \theta_k)$
6: $\quad \mathbf{e}_{k+1} := \mathbf{H}\mathbf{x}_{k+1} - \mathbf{y}$
7: $\quad \alpha_{k+1} = 2\sigma\left(\mathbf{w}_{1,k}^T \begin{bmatrix} \mathbf{x}_{k+1} \\ \hat{\mathbf{v}}_k \end{bmatrix} + b_{1,k}\right)$
8: $\quad \gamma_{k+1} = \sigma\left(\mathbf{w}_{2,k}^T \begin{bmatrix} \mathbf{x}_{k+1} \\ \hat{\mathbf{v}}_k \end{bmatrix} + b_{2,k}\right)$
9: $\quad \mathbf{v}_{k+1} := \alpha_{k+1}\mathbf{e}_{k+1} + (1-\alpha_{k+1}\beta_k)\hat{\mathbf{v}}_k$
10: $\quad \hat{\mathbf{v}}_{k+1} := \mathbf{v}_{k+1} + \gamma_{k+1}(\mathbf{v}_{k+1} - \mathbf{v}_k)$
11: end

Net in Algorithm 1 performs well in various modulations such as BPSK, 4QAM, and 16QAM. However, there is still room for improvement in the case of higher order modulations. Inspired by the gating operations in Long Short-Term Memory (LSTM) [28], we also introduce the gating operations to the ADMM Net. In Algorithm 1, α_k and γ_k, which control the effect of the previous layers' information on the current layer, are learned from the training data but fixed during the test time. Thus, we expect better performance by dynamically adapting α_k and γ_k based on the demodulation output $\mathbf{x}_k$ and the hidden state $\hat{\mathbf{v}}_k$

$$\alpha_{k+1} = 2\sigma\left(\mathbf{w}_{1,k}^T \begin{bmatrix} \mathbf{x}_{k+1} \\ \hat{\mathbf{v}}_k \end{bmatrix} + b_{1,k}\right)$$
$$\gamma_{k+1} = \sigma\left(\mathbf{w}_{2,k}^T \begin{bmatrix} \mathbf{x}_{k+1} \\ \hat{\mathbf{v}}_k \end{bmatrix} + b_{2,k}\right), \tag{30}$$

where $\sigma(x) = 1/(1+e^{-x})$ is the sigmoid function. Algorithm 2 describes the proposed gated ADMM network for MIMO detection (Gated ADMM Net). The set of learnable parameters of Gated ADMM Net is $\Omega = \{\mathbf{w}_{1,k}, b_{1,k}, \mathbf{w}_{2,k}, b_{2,k}, \delta_k, \beta_k, \theta_k\}_{k=0}^{L-1}$.

V. Comparison of Proposed ADMM Based Detection Network and DetNet

In this section, we compare DetNet and the proposed ADMM Net in terms of the network structures and the number of learnable parameters.

A. Structure

Both of DetNet and ADMM Net are designed based on gradient descent methods. DetNet is based on the proximal gradient descent method that consists of the one-step gradient descent followed by the proximal operator. In addition, it introduces internal hidden state $\mathbf{s}_k = \mathbf{v}_k$ for a lossy summary of past history of input sequences up to k. More specifically, the hidden state $\mathbf{v}_{k+1}$ is updated using the one-step gradient descent $\mathbf{q}_{k+1}$ and the previous hidden state $\mathbf{v}_k$

$$\mathbf{v}_{k+1} = \mathbf{W}_{3,k}\phi\left(\mathbf{W}_{1,k}\begin{bmatrix} \mathbf{q}_{k+1} \\ \mathbf{v}_k \end{bmatrix} + \mathbf{b}_{1,k}\right) + \mathbf{b}_{3,k}. \tag{31}$$

The challenge in this RNN model is that back-propagated gradients tend to either vanish or explode depending on the eigenvalues of $\mathbf{W}_{1,k}$ and $\mathbf{W}_{3,k}$ [26]. A popular solution is LSTM that is designed to capture long-term dependencies in sequence data [28]. LSTM cells have self-loops where gradients can easily flow for long duration. The self-loops are controlled by dynamically activating forget gates, which require additional learnable parameters. Since DetNet has already a large number of learnable parameters, LSTM variants of DetNet may have a big training burden.

Instead of the proximal gradient descent, we used ADMM for the MIMO detection problem in this article. In some applications such as LASSO problems, ADMM is much faster than the proximal gradient descent method [18]. In Algorithm 1, the x-update in Line 5 looks similar to the proximal gradient descent. In the v-update in Line 7, $\mathbf{v}_{k+1}$ is updated by a weighted sum of the residual $\mathbf{e}_{k+1}$ and the previous $\hat{\mathbf{v}}_k$ that is an accelerated version of $\mathbf{v}_k$, where α_k controls their priority. Thus, it seems that α_k plays the role of a gate that controls the amount of information flowing through layers in the LSTM cells. We introduce sigmoid layers that control α_k and γ_k dynamically with other variables $\mathbf{x}_k$ and $\hat{\mathbf{v}}_k$ in Gated ADMM Net, which improves the error performance at the cost of a lot of additional learnable parameters.

The MIMO detection requires the demodulation operation over the given constellation. The hard demodulation such as the sign function in BPSK demodulation is not differentiable, which prevents the network from back-propagating gradients. DetNet alleviates this problem by the weight matrix multiplication followed by the one-hot mapping, which increases the number of learnable parameters significantly especially in the case of higher order modulations. In ADMM Net, we propose differentiable $\eta_{\mathcal{S}}(\mathbf{x}; \theta_k)$ for the soft demodulation. It has two advantages: First, its complexity is not high for higher order modulations. Second, it only has a scalar learnable parameter θ_k that is easy to learn from data samples.

There are similar properties for DetNet and ADMM Net. Neural networks can have problems such as vanishing gradients and activation function saturation. As a remedy, both networks use the loss functions that take into account of all layer outputs similarly to the GoogLeNet auxiliary classifier [29].

B. Number of Learnable Parameters

We consider the untied parameters, i.e., the L-layer networks has layer-dependent parameters. This implies the total learnable parameters is proportional to L. We count the number of learnable parameters of DetNet and ADMM Net.

The learnable parameters of DetNet is $\Omega = \{\mathbf{W}_{1,k}, \mathbf{b}_{1,k}, \mathbf{W}_{2,k}, \mathbf{b}_{2,k}, \mathbf{W}_{3,k}, \mathbf{b}_{3,k}, \delta_{1,k}, \delta_{2,k}\}_{k=0}^{L-1}$. Let D and F denote the length of $\mathbf{z}_k$ and the length of $\mathbf{v}_k$. From (15), we have $\mathbf{W}_{1,k} \in \mathbb{R}^{D\times(M+F)}$, $\mathbf{W}_{2,k} \in \mathbb{R}^{|\bar{\mathcal{S}}|\bar{M}\times D}$, $\mathbf{W}_{3,k} \in \mathbb{R}^{F\times D}$, $\mathbf{b}_{1,k} \in \mathbb{R}^D$, $\mathbf{b}_{2,k} \in \mathbb{R}^{|\bar{\mathcal{S}}|\bar{M}}$, $\mathbf{b}_{3,k} \in \mathbb{R}^F$, $\delta_{1,k} \in \mathbb{R}$, and $\delta_{2,k} \in \mathbb{R}$. Therefore, the total number of learnable parameters in DetNet is

$$\left((M + |\bar{\mathcal{S}}|\bar{M} + 2F + 1)D + |\bar{\mathcal{S}}|\bar{M} + F + 2\right) L. \tag{32}$$

For the Python code uploaded in GitHub [16], the DetNet with BPSK modulation has $|\bar{\mathcal{S}}|\bar{M} = 2M$, $D = 4M$ and $F = 2M$.

In this case, the total number of learnable parameters is

$$\left(28M^2 + 8M + 2\right) L. \tag{33}$$

However, the proposed ADMM Net has only $5L$ learnable parameters, because it has scalar learnable parameters $\Omega = \{\delta_k, \alpha_k, \beta_k, \gamma_k, \theta_k\}_{k=0}^{L-1}$, which is independent of the number of transmit antennas, M. Since DetNet introduces too many learning parameters in order to replace the projection operator and improve performance, it takes very long time to learn the optimal parameters with a large amounts of data. On the other hand, ADMM Net has small number of learning parameters that are much easier to learn from small amount of data.

For performance improvement, we proposed Gated ADMM Net. From Algorithm 2, Gated ADMM Net has parameters $\mathbf{w}_{1,k} \in \mathbb{R}^{M+N}$, $\mathbf{w}_{2,k} \in \mathbb{R}^{M+N}$, $b_{1,k} \in \mathbb{R}$, $b_{2,k} \in \mathbb{R}$, $\delta_k \in \mathbb{R}$, $\beta_k \in \mathbb{R}$ and $\theta_k \in \mathbb{R}$. Therefore, the total number of learnable parameters is $(2M + 2N + 5)L$, which is strictly less than DetNet.

C. Complexity

We compare the complexity in terms of multiplications. In the test phase, the matrix multiplication operations are dominant factors for computational complexity. Each layer of DetNet has $M^2 + 2M + D(M + F) + |\bar{\mathcal{S}}|\bar{M}D + FD$ multiplications. On the other hand, each layer of ADMM Net has $M^2 + MN + 5M$ multiplication. Therefore, both algorithms have a complexity of $\mathcal{O}(M^2)$. More specifically, we consider the case of $N = 2M$. For BPSK, DetNet with $|\bar{\mathcal{S}}|\bar{M} = 2M$, $D = 4M$ and $F = 2M$ has $29M^2 + 2M$ multiplications per layer. On the other hand, ADMM Net has $3M^2 + 5M$ multiplications per layer. For example of $(M, N) = (30, 60)$, DetNet has 26160 multiplications per layer, while ADMM Net has 2850 multiplications per layer. For higher modulation order, the difference is more significant. For 16QAM, DetNet with $|\bar{\mathcal{S}}|\bar{M} = 4M$, $D = 8M$ and $F = 4M$ has $105M^2 + 2M$ multiplications per layer, but ADMM Net still has $3M^2 + 5M$. For example of $(M, N) = (30, 60)$, DetNet and ADMM Net have 94560 and 2850 multiplications per layer.

For complexity comparison of the training phase, we can compute the number of multiplications in the gradient back-propagations. In this case, the matrix multiplication operations are also dominant in the computational complexity. Both of DetNet and ADMM Net have a complexity of $\mathcal{O}(M^2)$.

VI. Numerical Results

In this section, we compare our proposed ADMM Network with the conventional detection algorithms such as sphere decoding (SD), and zero-forcing (ZF), 2-Lasso-ADMM [21], PJADMM [22], and DetNet [16] in terms of symbol error rate and simulation times. In the simulation, the number of transmit antennas is less than the number of receive antennas, and BPSK, 4QAM, and 16QAM modulations are used for digital modulation. Each entry of the MIMO channel matrices is drawn from an i.i.d. complex Gaussian distribution, each with zero mean and unit variance. The MIMO system with M transmit and N receive antennas is denoted by $M \times N$ MIMO channels. We consider the varying channel model in which the channel is random and is known at the receiver. The loss function is differentiable with learnable parameters. Thanks to the auto-differentiation of deep learning framework such as TensorFlow [30], we can back-propagate the gradient of the loss function with respect to the learnable parameters. Then, we update them in the direction of their negative gradients using Adam optimizer [31].

In the training phase, we trained DetNet and ADMM Net using randomly generated data samples. The total number of layers, or iterations, in DetNet and ADMM Net is fixed to 30 layers. For initial parameter values, we set $\delta_k = 0.001$, $\alpha_k = 1.5$, $\beta_k = 0.1$, $\gamma_k = k/(k+5)$, $\theta_k = 1.5$ as suggested by the R-A-ADMM [24]. We used 2×10^7 training data samples to train ADMM Network, which takes 2 hours for training. One data sample indicates a triplet of $(\mathbf{H}, \mathbf{y}, \mathbf{x})$ for training. The training data samples are generated so that the SNR is uniformly distributed in [8 dB, 13 dB]. For DetNet, we used the code uploaded to the GitHub[3] [16] considering two training cases: We trained DetNet using 2×10^7 training data samples, which takes similar learning time to our proposed ADMM Network. Since its performance is not good, we also trained DetNet using 6×10^8 training data samples, which requires more than two days for training. Except Fig. 3, which includes both data sample cases, all performances of DetNet were obtained with 6×10^8 training data samples.

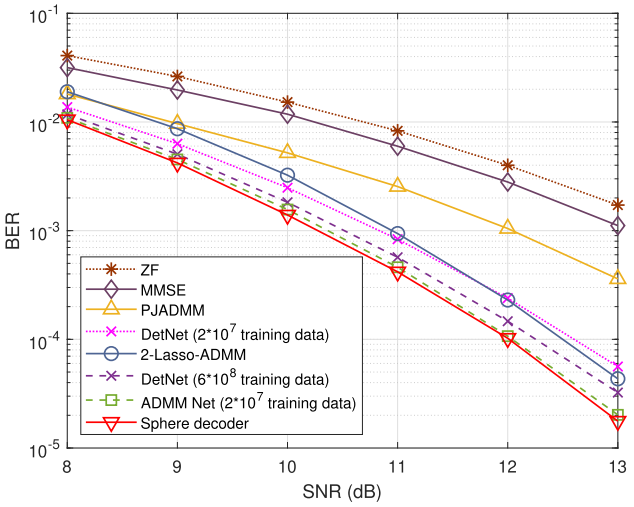
We evaluated the detection algorithms using 10^6 random channel realizations per SNR that are independent of training channels. In the test phase, the neural networks are not trained any more with the test data samples.

A. Symbol Error Rate Performance

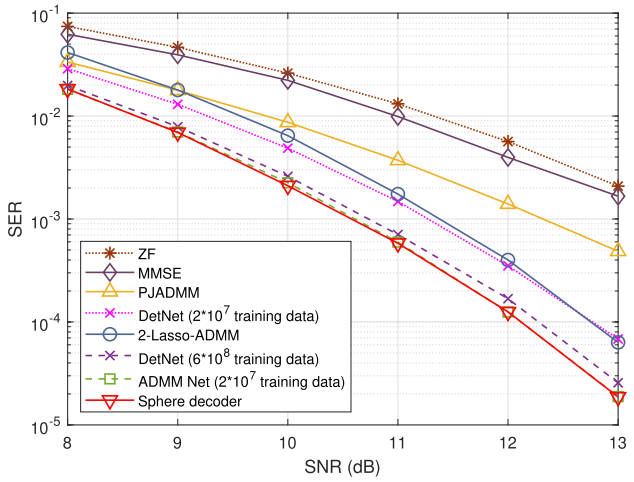
Figs. 3 (a)–(c) depict the symbol error rate (SER) performance with respect to SNR when there are twice as many receive antennas as transmit antennas. Figs. 4 (a)–(c) depict the SER performance when there are 5 or 10 more receive antennas than transmit antennas. Thus, Figs. 3 (a)–(c) have more favorable channels than Figs. 4 (a)–(c). We compare SER performance of each algorithm as follows:

1) **Sphere decoder**: Sphere decoder achieves near maximal likelihood (ML), or sometimes exact ML performance, while its worst-case complexity is exponential [3].
2) **Zero forcing**: Zero forcing detector has a low complexity but has the worst performance in all scenarios.
3) **PJADMM** : The performance of PJADMM [22] is far from the sphere decoding performance.
4) **DetNet**: In BPSK and 4QAM, Figs. 3 (a), (b) and Fig. 4 (a) show DetNet with 6×10^8 training data samples works well with less than 0.7 dB from the performance of the sphere decoder and is worse than the ADMM Net. In the higher order modulation schemes, Fig. 3 (c) and Figs. 4 (b) and (c) show DetNet has a performance degradation and the gap with the sphere decoder increases to up to 2 dB as SNR increases. In summary,

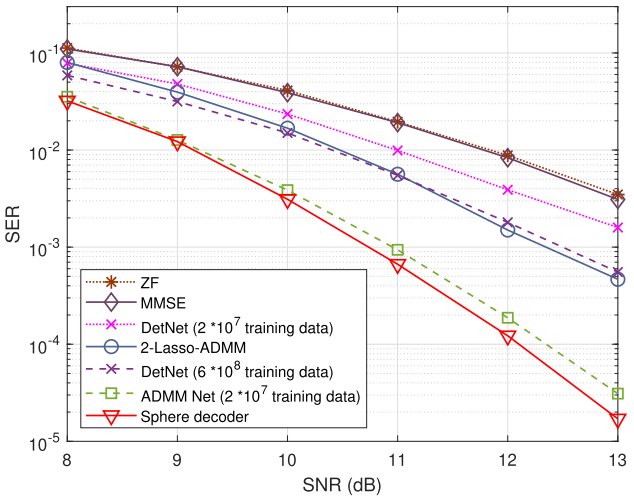
[3]https://github.com/neevsamuel/



(a)

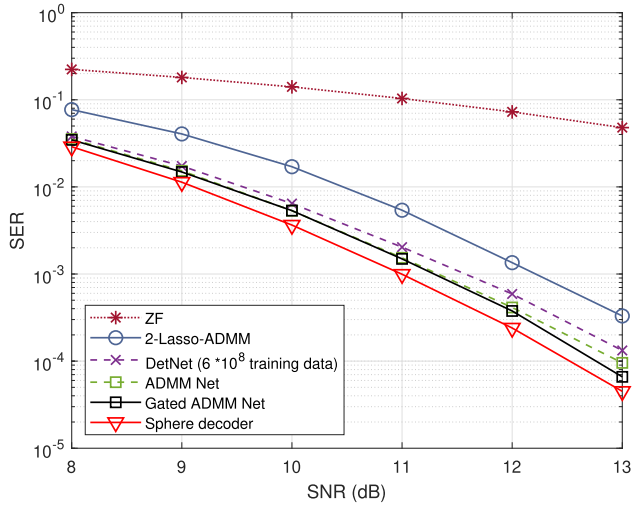


(b)

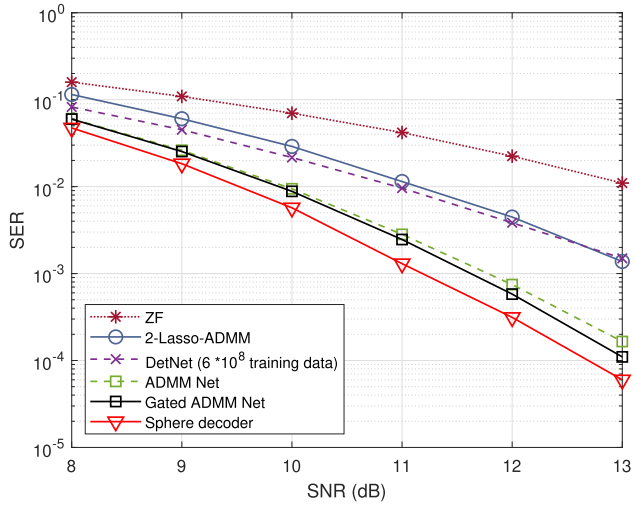


(c)

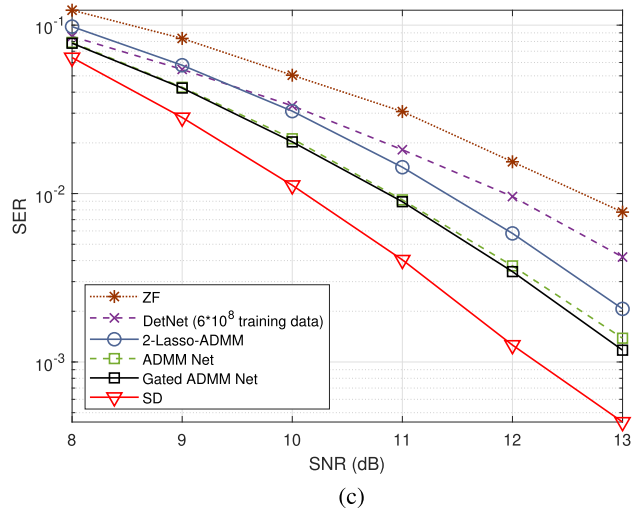
Fig. 3. Comparison of error rates: (a) BPSK in real 30×60 MIMO channels, (b) 4QAM in complex 30×60 MIMO channels, (c) 16QAM in complex 30×60 MIMO channels.



(a)



(b)



(c)

Fig. 4. Comparison of error rates: (a) 4QAM in complex 20×25 MIMO channels, (b) 16QAM in complex 15×25 MIMO channels, (c) 64QAM in complex 5×10 MIMO channels.

DetNet performs well in lower order modulations but not in higher order modulations. It requires a large number of training data samples to achieve good performance.

5) **2-Lasso-ADMM**: In Figs. 3 (a)–(b), 2-Lasso-ADMM [22] has similar performance with DetNet that is trained with 2×10^7 training data samples. However, it is

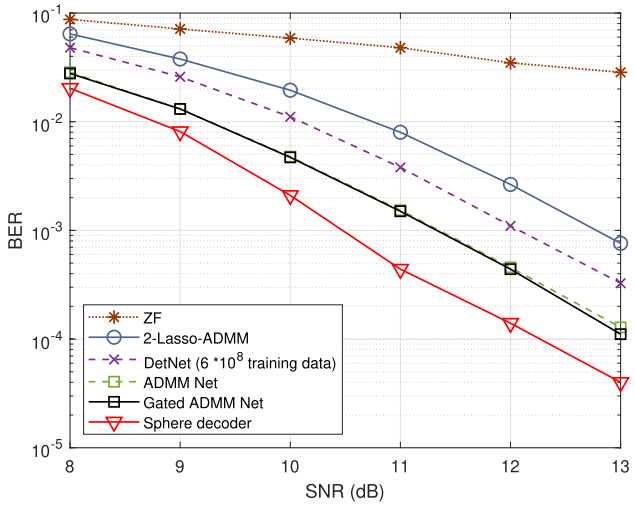


Fig. 5. Comparison of error rates for BPSK in real 50×50 MIMO channels.

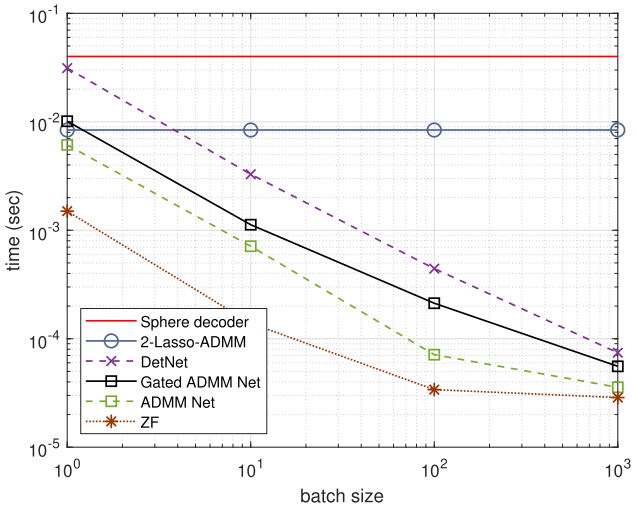


Fig. 6. Comparison of average running times (30×60 16QAM, SNR 13 dB, 30 layers).

outperformed by DetNet trained with 6×10^8 training data samples. In higher order modulation, 2-LASSO-ADMM is better than DetNet trained with 6×10^8 training data samples in Fig. 4 (c).

6) **ADMM Net**: ADMM Net works well with less than 0.05 dB from the performance of the sphere decoder in Figs. 3 (a)–(b) and 0.5 dB in Fig. 3 (c) in the case that there are much more receive antennas than transmit antennas. In particular, we can see that ADMM Net achieves nearly the same performance with the sphere decoder in Fig. 3 (b). When there are 5 or 10 more receive antennas than transmit antennas, ADMM Net has 0.2–1.2 dB gap with the sphere decoder in Figs. 4 (a)–(c). In the case of $M = N = 50$ in Fig. 5, there is 0.7 dB gap with the sphere decoder. Therefore, the deep neural networks based on ADMM can bring forth significant performance gain over DetNet but there is still room for improvement.
7) **Gated ADMM Net**: We applied Gated ADMM Net in case of $M \leq N \leq M + 10$. Gated ADMM Net slightly outperforms ADMM Net in Fig. 4 (a)–(c) and Fig. 5.
8) **Summary**: DetNet only works well with lower order modulation. 2-Lasso-ADMM is quite good for higher order modulations, but its performance is not close to the sphere decoder. In all cases, both of ADMM Net and Gated ADMM Net outperform DetNet and 2-Lasso-ADMM. In addition, ADMM Net achieves near optimal performance in Figs. 3 (a)–(b).

B. Average Running Time

To compare the computational complexity, we measure the average running time for each algorithm. The average running time is defined as the running time divided by the number of data samples in the test stage. For a fair comparison, we used the same PC with Intel Xeon E5 CPU and NVIDIA GTX Titan Xp GPU to measure the running time. Thanks to the parallel processing of GPU, neural networks run faster with a large batch size. In addition, GPU also accelerates the ZF detector, because ZF detector can be considered as an 1-layer neural network consisting of matrix multiplication followed by non-linear demodulation without learnable parameters. However, it is hard to perform the sphere decoding in the batch processing due to its complex structure. In addition, the running time of sphere decoder depends on the operating SNR because it considers more candidates at lower SNR. For fairness, all experiments were conducted at the same SNR 13 dB.

Fig. 6 shows the average running time according to the batch size with 16QAM in complex 30×60 MIMO channels at 13dB. We fix the number of layers is 30. The average running time of ZF, DetNet, ADMM Net, and Gated ADMM Net, which can be batch processed, is inversely proportional to the the batch size. When the batch size is B, we take B pairs of $(\mathbf{y}, \mathbf{H})$ to perform the MIMO detection for recovery of $\mathbf{x}$. Thus, as long as the GPU can handle B pairs of $(\mathbf{y}, \mathbf{H})$ simultaneously, the total running time is irrelevant to B. Therefore, the average running time is inversely proportional to the batch size B if the GPU can handle B pairs. If we take a 10 times bigger batch size, we get 10 times less average running time. There is a lower limit of about 3×10^{-5} seconds in the average running time possibly due to the data transfer overhead between CPU and GPU and the number of GPU cores. This lower limit may be dependent on the experimental environments. ADMM Net has a shorter average running time than DetNet and is 50 times faster than sphere decoder when the size of the batch is 10 or larger. In addition, ADMM Net with the batch size of 10 or larger is even faster than ZF without batch processing.

So far we compared the average running time in the test phase. We may compare the overall simulation times including the training and the test phases. In Fig. 3 (c), ADMM Net with $B = 10^3$ required 2 hours for the training phase and 180 seconds for the test phase. However, it took more than 30 hours for the sphere decoder to perform the detection.

C. SER Improvement Over Layers

Fig. 7 depicts SER versus the number of layers of ADMM Net. We observe that ADMM Net converges in less than

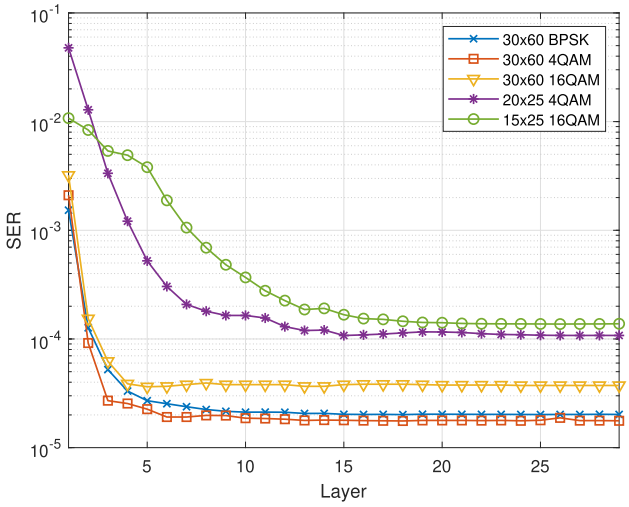


Fig. 7. SER versus layers of ADMM Net (SNR 13dB).

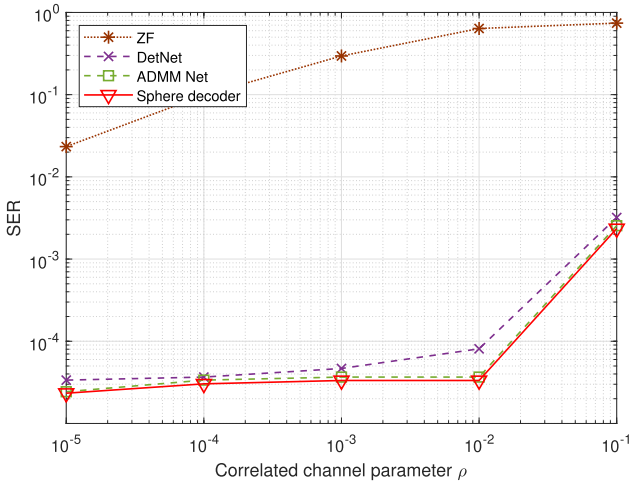


Fig. 8. Effect of channel correlations on error rates (4QAM in complex 30×60 MIMO correlated channels).

20 layers for all modulations and antenna configurations. When the number of receive antennas are large, ADMM Net converges in less than 5 layers. There is a trade-off between the running time and the error performance. As the neural networks are deeper, we expect lower error probability. Therefore, we can adjust the number of the network layers to achieve the target error performance.

D. Effect of Channel Correlation

In the previous subsections, each entry of the channel matrix $\bar{\mathbf{H}}$ is independent Gaussian distributed. For correlated channel models, the channel matrix $\bar{\mathbf{H}}$ takes the expression

$$\bar{\mathbf{H}} = \mathbf{R}_{RX}^{1/2} \bar{\mathbf{H}}_w \mathbf{R}_{TX}^{1/2}, \tag{34}$$

where each entry of $\bar{\mathbf{H}}_w$ is drawn from an i.i.d. complex Gaussian distribution with zero mean and unit variance, and $\mathbf{R}_{RX}$ and $\mathbf{R}_{TX}$ are the correlation matrices at the receiver and the transmitter, respectively. To investigate the effect of amount of correlation, $\mathbf{R}_{RX}$ is expressed in terms of a single parameter ρ using a Toeplitz correlation structure [32]

$$\mathbf{R}_{RX} = \begin{bmatrix} 1 & \rho & \rho^4 & \cdots & \rho^{(\bar{N}-1)^2} \\ \rho & 1 & \rho & \ddots & \rho^{(\bar{N}-2)^2} \\ \rho^4 & \rho & 1 & \ddots & \rho^{(\bar{N}-3)^2} \\ \vdots & \ddots & \ddots & \ddots & \vdots \\ \rho^{(\bar{N}-1)^2} & \cdots & & \cdots & 1 \end{bmatrix} \in \mathbb{R}^{\bar{N} \times \bar{N}} \tag{35}$$

and $\mathbf{R}_{TX}$ is also similarly defined. For simplicity, we assume both of $\mathbf{R}_{RX}$ and $\mathbf{R}_{TX}$ are characterized by the same correlation parameter ρ. $\rho = 0$ corresponds the case that each entry of $\bar{\mathbf{H}}$ is i.i.d. Gaussian distributed. On the other hand, $\rho = 1$ indicates the full correlations among antennas. Fig. 8 depicts the error performance with correlated channels with correlation parameter ρ. As ρ increases, all detector performances deteriorate. ADMM Net performs as well as the sphere decoder and better than DetNet.

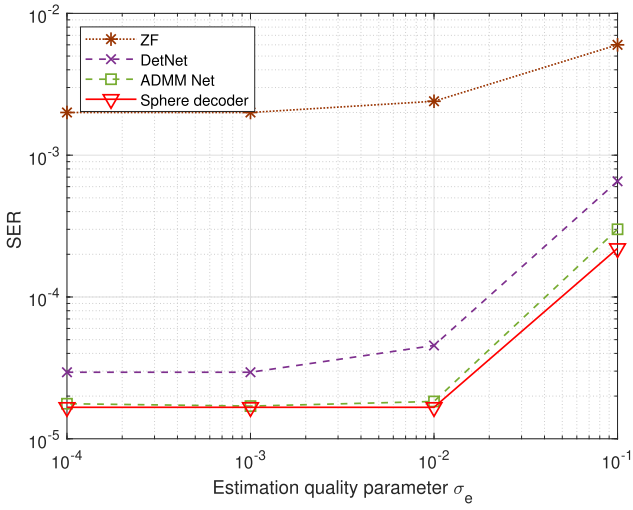


Fig. 9. Effect of channel estimation errors on error rates (4QAM in complex 30×60 MIMO).

E. Effect of Channel Estimation Errors

So far, we assume the channel estimation is perfect so that the receiver knows the exact channel matrix $\bar{\mathbf{H}} \in \mathbb{C}^{\bar{N} \times \bar{M}}$. In this subsection, we investigate the effect of imperfect channel estimation on the error performance. We consider a simple channel estimation error model. The received signal is given by $\bar{\mathbf{y}} = \bar{\mathbf{H}}\bar{\mathbf{x}} + \bar{\mathbf{w}}$ and the receiver only has a noisy channel estimate

$$\bar{\mathbf{H}}_\sigma = \sqrt{1 - \sigma_e^2}\bar{\mathbf{H}} + \sigma_e \bar{\mathbf{E}}, \tag{36}$$

where each entry of $\bar{\mathbf{E}}$ is i.i.d. complex Gaussian distributed with zero mean and unit variance. σ_e is the channel estimation quality parameter: $\sigma_e = 0$ corresponds to a perfect channel estimation, while $\sigma_e = 1$ corresponds to the case that the estimated channel is uncorrelated with the true channel. The MIMO detection problem is to recover $\bar{\mathbf{x}}$ from $\bar{\mathbf{y}}$ for given $\bar{\mathbf{H}}_\sigma$. The receiver rearranges complex vectors to real vectors as did in (3) and then applies various MIMO detection algorithms. Fig. 9 depicts the effect of channel estimation errors on error rates. As σ_e increases, all the detector performances

deteriorate. Since the performance gap between ADMM Net and the sphere decoder is very small for wide ranges of σ_e, ADMM Net is robust to channel estimation errors.

VII. Conclusion

We proposed a new learnable MIMO detection network based on inexact ADMM that achieves near-optimal error performance with reasonable computational complexity. The proposed ADMM Net borrowed a parameterized mapping structure from relaxed accelerated ADMM and its trainable layers consist of linear processing for gradient descents and non-linear hyperbolic tangent operations for demodulation.

ADMM Net has the following advantages: First, it achieves near optimal performance with a low computational complexity. It works well with the higher order modulation such as 16QAM. Second, ADMM Net is implemented with a fewer learning parameters. It can learn parameters with much less data samples than one of the latest method such as DetNet. Third, it can run in batch, processing large amounts of data at once, which reduces the average running time. Therefore, ADMM Net is well suited as a feasible massive MIMO detection algorithm with a low complexity.

Inspired by LSTM, we proposed Gated ADMM Net to control the effect of the previous layers' information on the current layer. Gating operation improves SER performance by actively controlling amount of information flows. It is for further study to design a more intelligent low-complexity algorithm for the massive MIMO detection.

References

[1] D. Tse and P. Viswanath, *Fundamentals of Wireless Communication*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[2] S. Verdú, "Computational complexity of optimum multiuser detection," *Algorithmica*, vol. 4, nos. 1–4, pp. 303–312, Jun. 1989.

[3] 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, Sep. 2015.

[4] A. Chockalingam and B. S. Rajan, *Large MIMO Systems*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

[5] A. Elghariani and M. Zoltowski, "Low complexity detection algorithms in large-scale MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 15, no. 3, pp. 1689–1702, Mar. 2016.

[6] R. Hayakawa and K. Hayashi, "Convex optimization-based signal detection for massive overloaded MIMO systems," *IEEE Trans. Wireless Commun.*, vol. 16, no. 11, pp. 7080–7091, Nov. 2017.

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

[8] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Netw.*, vol. 2, no. 5, pp. 359–366, Jan. 1989.

[9] J. A. Tropp and A. C. Gilbert, "Signal recovery from random measurements via orthogonal matching pursuit," *IEEE Trans. Inf. Theory*, vol. 53, no. 12, pp. 4655–4666, Dec. 2007.

[10] T. Blumensath and M. E. Davies, "Iterative thresholding for sparse approximations," *J. Fourier Anal. Appl.*, vol. 14, nos. 5–6, pp. 629–654, Dec. 2008.

[11] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *Proc. 27th Int. Conf. Int. Conf. Mach. Learn.*, 2010, pp. 399–406.

[12] Z. Wang, Q. Ling, and T. S. Huang, "Learning deep ℓ_0 encoders," in *Proc. AAAI Conf. Artif. Intell.*, 2016, pp. 2194–2200.

[13] M. Borgerding, P. Schniter, and S. Rangan, "AMP-inspired deep networks for sparse linear inverse problems," *IEEE Trans. Signal Process.*, vol. 65, no. 16, pp. 4293–4308, Aug. 2017.

[14] D. Wipf, "Replacing/enhancing iterative algorithms with deep neural networks," in *Proc. IEEE ICASSP Tutorial*, Apr. 2018.

[15] N. Samuel, T. Diskin, and A. Wiesel, "Deep MIMO detection," in *Proc. IEEE 18th Int. Workshop Signal Process. Adv. Wireless Commun. (SPAWC)*, Jul. 2017, pp. 690–694.

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

[17] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein, "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, Jan. 2011.

[18] N. Parikh and S. Boyd, "Proximal algorithms," *Found. Trends Optim.*, vol. 1, no. 3, pp. 123–231, 2014.

[19] J. Yang and Y. Zhang, "Alternating direction algorithms for ℓ_1-problems in compressive sensing," *SIAM J. Sci. Comput.*, vol. 33, no. 1, pp. 250–278, 2011.

[20] X. Xie, J. Wu, G. Liu, Z. Zhong, and Z. Lin, "Differentiable linearized ADMM," in *Proc. ICML*, 2019, pp. 6902–6911.

[21] A. Elgabli, A. Elghariani, A. O. Al-Abbasi, and M. Bell, "Two-stage LASSO ADMM signal detection algorithm for large scale MIMO," in *Proc. 51st Asilomar Conf. Signals, Syst., Comput.*, Oct. 2017, pp. 1660–1664.

[22] A. Elgabli, A. Elghariani, V. Aggarwal, M. Bennis, and M. R. Bell, "A proximal jacobian ADMM approach for fast massive MIMO signal detection in low-latency communications," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2019.

[23] Y. Nesterov, *Introductory Lectures Convex Optimization: A Basic Course*. Boston, MA, USA: Kluwer, 2004.

[24] G. Franca, D. P. Robinson, and R. Vidal, "A dynamical systems perspective on nonsmooth constrained optimization," 2018, *arXiv:1808.04048v3*. [Online]. Available: https://dev.arxiv.org/abs/1808.04048v3

[25] T. Goldstein, B. O'Donoghue, S. Setzer, and R. Baraniuk, "Fast alternating direction optimization methods," *SIAM J. Imag. Sci.*, vol. 7, no. 3, pp. 1588–1623, 2014.

[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016.

[27] V. Corlay, J. J. Boutros, P. Ciblat, and L. Brunel, "Multilevel MIMO detection with deep learning," in *Proc. 52nd Asilomar Conf. Signals, Syst., Comput.*, Oct. 2018, pp. 1805–1809.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

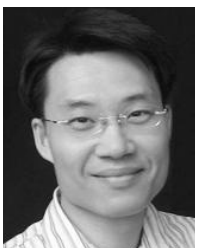
[29] J. C. Szegedy *et al.*, "Going deeper with convolutions," in *Proc. IEEE CVPR*, Jun. 2015, pp. 1–9.

[30] M. Abadi *et al.*, "Tensorflow: A system for large-scale machine learning," in *Proc. OSDI*, 2016, pp. 265–283.

[31] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. ICLR*, 2015.

[32] A. van Zelst and J. S. Hammerschmidt, "A single coefficient spatial correlation model for multiple-input multiple-output (MIMO) radio channels," in *Proc. URSI 27th Gen. Assem.*, 2002, pp. 1–4.

Minsik Kim received the B.E. and M.E. degrees in information and communication from Inha University, Incheon, South Korea, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree in electrical and computer engineering. His research interests include wireless communications, convex optimization, and machine learning.



Daeyoung Park (Member, IEEE) received the B.S. and M.E. degrees in electrical engineering and the Ph.D. degree in electrical engineering and computer science from Seoul National University, Seoul, South Korea, in 1998, 2000, and 2004, respectively. He was with Samsung Electronics as a Senior Engineer from 2004 to 2007. He has held visiting positions at the University of Southern California and the University of California, San Diego. Since 2008, he has been with Inha University, Incheon, South Korea, where he is currently a Professor. He is a coauthor of *Wireless Communications Resource Management* (John Wiley & Sons, 2009). His research interests include sparsity-aware signal processing, multiple antenna communication systems, optimization, and machine learning.