Learning to Detect

Neev Samuel, *Member, IEEE,* and Tzvi Diskin, *Member, IEEE* and Ami Wiesel, *Member, IEEE*

***Abstract*—In this paper we consider Multiple-Input-Multiple-Output (MIMO) detection using deep neural networks. We introduce two different deep architectures: a standard fully connected multi-layer network, and a Detection Network (DetNet) which is specifically designed for the task. The structure of DetNet is obtained by unfolding the iterations of a projected gradient descent algorithm into a network. We compare the accuracy and runtime complexity of the purposed approaches and achieve state-of-the-art performance while maintaining low computational requirements. Furthermore, we manage to train a single network to detect over an entire distribution of channels. Finally, we consider detection with soft outputs and show that the networks can easily be modified to produce soft decisions.**

***Index Terms*—MIMO Detection, Deep Learning, Neural Networks.**

I. Introduction

MULTIPLE input multiple output (MIMO) systems enable enhanced performance in communication systems, by using many dimensions that account for time and frequency resources, multiple users, multiple antennas and other resources. While improving performance, these systems present difficult computational challenges when it comes to detection since the detection problem is NP-Complete, and there is a growing need for sub-optimal solutions with polynomial complexity.

Recent advances in the field of machine learning, specifically the success of deep neural networks in solving many problems in almost any field of engineering, suggest that a data driven approach for detection using machine learning may present a computationally efficient way to achieve near optimal detection accuracy.

A. MIMO detection

MIMO detection is a classical problem in simple hypothesis testing [1]. The maximum likelihood (ML) detector involves an exhaustive search and is the optimal detector in the sense of minimum joint probability of error for detecting all the symbols simultaneously. Unfortunately, it has an exponential runtime complexity which makes it impractical in large real time systems.

In order and overcome the computational cost of the maximum likelihood decoder there is considerable interest in implementation of suboptimal detection algorithms which provide a better and more flexible accuracy vs complexity tradeoff. In the high accuracy regime, sphere decoding algorithms [2], [3], [4] were purposed, based on lattice search, and offering better computational complexity with a rather low accuracy performance degradation relatively to the full search. In the other regime, the most common suboptimal detectors are the linear receivers, i.e., the matched filter (MF), the decorrelator or zero forcing (ZF) detector and the minimum mean squared error (MMSE) detector. More advanced detectors are based on decision feedback equalization (DFE), approximate message passing (AMP) [5] and semidefinite relaxation (SDR) [6], [7]. Currently, both AMP and SDR provide near optimal accuracy under many practical scenarios. AMP is simple and cheap to implement in practice, but is an iterative method that may diverge in challenging settings. SDR is more robust and has polynomial complexity, but is limited in the settings it addresses and is much slower in practice.

B. Background on Machine Learning

Machine learning is the ability to solve statistical problems using examples of inputs and their desired outputs. Unlike classical hypothesis testing, it is typically used when the underlying distributions are unknown and are characterized via sample examples. It has a long history but was previously limited to simple and small problems. Fast forwarding to recent years, the field witnessed the deep revolution. The "deep" adjective is associated with the use of complicated and expressive classes of algorithms, also known as architectures. These are typically neural networks with many non-linear operations and layers. Deep architectures are more expressive than shallow ones and can theoretically solve much harder and larger problems [8], but were previously considered impossible to optimize. With the advances in big data, optimization algorithms and stronger computing resources, such networks are currently state of the art in different problems from speech processing [9], [10] and computer vision [11], [12] to online gaming [13]. Typical solutions involve dozens and even hundreds of layers which are slowly optimized off-line over clusters of computers, to provide accurate and cheap decision rules which can be applied in real-time. In particular, one promising approach to designing deep architectures is by unfolding an existing iterative algorithm [14]. Each iteration is considered a layer and the algorithm is called a network. The learning begins with the existing algorithm as an initial starting point and uses optimization methods to improve the algorithm. For example, this strategy has been shown successful in the context of sparse reconstruction [15], [16]. Leading algorithms as Iterative Shrinkage and Thresholding and a sparse version of AMP have both been improved by unfolding their iterations into a network and learning their optimal parameters.

Following this revolution, there is a growing body of works on deep learning methods for communication systems.

Manuscript received May, 2018;

N. Samuel, T. Diskin and A. Wiesel are with the School of Computer Science and Engineering, The Hebrew University of Jerusalem, Israel. E-mail: (neev.samuel@mail.huji.ac.il or see http://www.cs.huji.ac.il/∼amiw/).

This research was partly supported by the Heron Consortium and by ISF grant 1339/15.

Exciting contributions in the context of error correcting codes include [17]–[21]. In [22] a machine learning approach is considered in order to decode over molecular communication systems where chemical signals are used for transfer of information. In these systems an accurate model of the channel is impossible to find. This approach of decoding without CSI (channel state information) is further developed in [23]. Machine learning for channel estimation is considered in [24], [25]. End-to-end detection over continuous signals is addressed in [26]. And in [27] deep neural networks are used for the task of MIMO detection using an end-to-end approach where learning is deployed both in the transmitter in order to encode the transmitted signal and in the receiver where unsupervised deep learning is deployed using an autoencoder. Parts of our work on MIMO detection using deep learning have already appeared in [28], see also [29]. Similar ideas were discussed in [30] in the context of robust regression.

C. Main contributions

The main contribution of this paper is the introduction of two deep learning networks for MIMO detection. We show that, under a wide range of scenarios including different channels models and various digital constellations, our networks achieve near optimal detection performance with low computational complexity.

Another important result we show is their ability to easily provide soft outputs as required by modern communication systems. We show that for different constellations the soft output of our networks achieve accuracy comparable to that of the M-Best sphere decoder with low computational complexity.

In a more general learning perspective, an important contribution is DetNet's ability to perform on multiple models with a single training. Recently, there were works on learning to invert linear channels and reconstruct signals [15], [16], [31]. To the best of our knowledge, these were developed and trained to address a single fixed channel. In contrast, DetNet is designed for handling multiple channels simultaneously with a single training phase.

The paper is organized in the following order:

In section II we present the MIMO detection problem and how it is formulated as a learning problem including the use of one-hot representations. In section III we present two types of neural network based detectors, FullyCon and DetNet. In section IV we consider soft decisions. In section V we compare the accuracy and the runtime of the purposed learning based detectors against traditional detection methods both in the hard decision and the soft decision cases. Finally, section VI provides concluding remarks.

D. Notation

In this paper, we define the normal distribution where μ is the mean and σ^2 is the variance as $\mathcal{N}\left(\mu, \sigma^2\right)$. The uniform distribution with the minimum value a and the maximum value b will be $\mathcal{U}\left(a, b\right)$. Boldface uppercase letters denote matrices. Boldface lowercase letters denote vectors. The superscript $(\cdot)^T$ denotes the transpose. The i'th element of the vector $\mathbf{x}$ will be denoted as $\mathbf{x}_i$. Unless stated otherwise, the term independent and identically distributed (i.i.d.) Gaussian matrix, refers to a matrix where each of its elements is i.i.d. sampled from the normal distribution $\mathcal{N}\left(0,1\right)$. The rectified linear unit defined as $\rho(x) = \max\{0, x\}$. When considering a complex matrix or vector the real and imaginary parts of it are defined as $\Re(\cdot)$ and $\Im(\cdot)$ respectively. An α-Toeplitz $\mathbf{M}$ matrix will be defined as a matrix such that $\mathbf{M}^T\mathbf{M}$ is a square matrix where the value of each element on the i'th diagonal is α^{i-1}.

II. Problem formulation

A. MIMO detection

We consider the standard linear MIMO model:

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

where $\bar{\mathbf{y}} \in \mathbb{C}^N$ is the received vector, $\bar{\mathbf{H}} \in \mathbb{C}^{N\times K}$ is the channel matrix, $\bar{\mathbf{x}} \in \bar{\mathbb{S}}^K$ is an unknown vector of independent and equal probability symbols from some finite constellation $\bar{\mathbb{S}}$ (e.g. PSK or QAM), $\bar{\mathbf{w}}$ is a noise vector of size N with independent, zero mean Gaussian variables of variance σ^2.

Our detectors do not assume knowledge of the noise variance σ^2. Hypothesis testing theory guarantees that it is unnecessary for optimal detection. Indeed, the ML rule does not depend on it. This is contrast to the MMSE and AMP decoders that exploit this parameter and are therefore less robust in cases where the noise variance is not known exactly.

B. Reparameterization

A main challenge in MIMO detection is the use of complex valued signals and various digital constellations $\bar{\mathbb{S}}$ which are less common in machine learning. In order to use standard tools and provide a unified framework, we re-parameterize the problem using real valued vectors and one-hot mappings as described below.

First, throughout this work, we avoid handling complex valued variables, and use the following convention:

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

where

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

where $\mathbf{y} \in \mathbb{R}^{2N}$ is the received vector, $\mathbf{H} \in \mathbb{R}^{2N\times 2K}$ is the channel matrix and $\mathbf{x} \in \mathbb{S}^{2K}$ where $\mathbb{S} = \Re\{\bar{\mathbb{S}}\}$ (which is also equal to $\Im\{\bar{\mathbb{S}}\}$ in the complex valued constellations we tested)

A second convention concerns the re-parameterization of the discrete constellations $\mathbb{S} = \{s_1, \cdots, s_{|\mathbb{S}|}\}$ using one-hot mapping. With each possible s_i we associate a unit vector $\mathbf{u}_i \in \mathbb{R}^{|\mathbb{S}|}$. For example, the 4 dimensional one-hot mapping of the real part of 16-QAM constellations is defined as

$$\begin{aligned}
s_1 = -3 \quad &\leftrightarrow \quad \mathbf{u}_1 = [1,0,0,0]\\
s_2 = -1 \quad &\leftrightarrow \quad \mathbf{u}_2 = [0,1,0,0]\\
s_3 = 1 \quad &\leftrightarrow \quad \mathbf{u}_3 = [0,0,1,0]\\
s_4 = 3 \quad &\leftrightarrow \quad \mathbf{u}_4 = [0,0,0,1]
\end{aligned} \tag{4}$$

We denote this mapping via the function $s = f_{oh}(\mathbf{u})$ so that $s_i = f_{oh}(\mathbf{u}_i)$ for $i = 1, \cdots, |\mathbb{S}|$. More generally, for approximate inputs which are not unit vectors, the function is defined as

$$x = f_{oh}(\mathbf{x}_{oh}) = \sum_{i=1}^{|\mathbb{S}|} s_i[\mathbf{x}_{oh}]_i \tag{5}$$

The description above holds for a scalar symbol. The MIMO model involves a vector of $2K$ symbols which is handled by stacking the one-hot mapping of each of its elements. Altogether, a vector $\mathbf{x}_{oh} \in \{0,1\}^{|\mathbb{S}|\cdot 2K}$is mapped to $\mathbf{x} \in \mathbb{S}^{2K}$.

C. Learning to detect

We end this section by formulating the MIMO detection problem as a machine learning task. The first step in machine learning is choosing a class of possible detectors, also known as an architecture. A network architecture is a function $\hat{\mathbf{x}}_{oh}(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta})$ parameterized by $\boldsymbol{\theta}$ that detects the unknown $\mathbf{x}_{oh}$ given $\mathbf{y}$ and $\mathbf{H}$. Learning is the problem of finding the $\boldsymbol{\theta}$ within some feasible set that will lead to strong detectors $\hat{\mathbf{x}}_{oh}(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta})$. For this purpose, we fix a loss function $l\left(\mathbf{x}_{oh}; \hat{\mathbf{x}}_{oh}\left(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta}\right)\right)$ that measures the distance between the true vectors and their estimates. Then, we find the network's parameter $\boldsymbol{\theta}$ by minimizing the loss function over the MIMO model distribution:

$$\min_{\boldsymbol{\theta}} \mathrm{E}\left\{l\left(\mathbf{x}_{oh}; \hat{\mathbf{x}}_{oh}(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta})\right)\right\}, \tag{6}$$

where the expectation is with respect to all the random variables in (2), i.e., $\mathbf{x}$, $\mathbf{w}$, and $\mathbf{H}$. Learning to detect is defined as finding the best parameters $\boldsymbol{\theta}$ of the networks' architecture that minimize the expected loss $l\left(\cdot;\cdot\right)$ over the distribution in (2).

We always assume perfect channel state information (CSI) which means that the channel $\mathbf{H}$ is exactly known during detection time. However, we differentiate between two possible cases:

- Fixed Channel (FC): In the FC scenario, $\mathbf{H}$ is deterministic and constant (or a realization of a degenerate distribution which only takes a single value). This means that during the training phase we know over which channel the detector will detect.
- Varying Channel (VC): In the VC scenario, we assume $\mathbf{H}$ random with a known continuous distribution. It is still completely known but changes in each realization, and a single detection algorithm must be designed for all its possible realizations. When detecting, the channel is randomly chosen, and the network must be able to generalize over the entire distribution of possible channels.

Altogether, our goal is to detect $\mathbf{x}$, using a neural network that receives $\mathbf{y}$ and $\mathbf{H}$ as inputs and provides an estimate $\hat{\mathbf{x}}$. In the next section, we will introduce two competing architectures that tradeoff accuracy and complexity.

III. Deep MIMO detectors

A. FullyCon

The fully connected multi-layer network is a well known architecture which is considered to be the basic deep neural network architecture, and from now on will be named simply as 'FullyCon'. It is composed of L layers, where the output of each layer is the input of the next layer. Each layer can be described by the following equations:

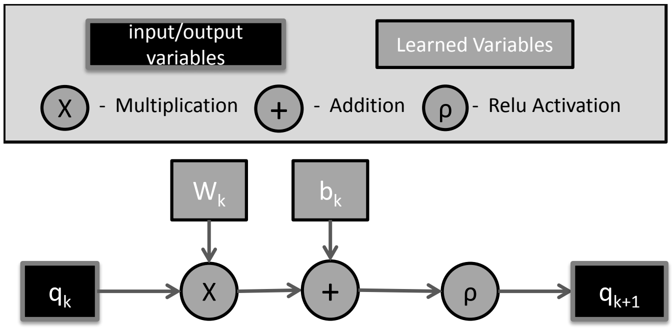

Fig. 1. A flowchart representing a single layer of the fully connected network.

$$\begin{aligned} \mathbf{q}_1 &= \mathbf{y} \\ \mathbf{q}_{k+1} &= \rho\left(\mathbf{W}_k\mathbf{q}_k + \mathbf{b}_k\right) \\ \hat{\mathbf{x}}_{oh} &= \mathbf{W}_L\mathbf{q}_L + \mathbf{b}_L \\ \hat{\mathbf{x}} &= \mathbf{f}_{oh}(\hat{\mathbf{x}}_{oh}) \end{aligned} \tag{7}$$

An illustration of a single layer of FullyCon can be seen in Fig 1. The parameters of the network that are optimized during the learning phase are:

$$\boldsymbol{\theta} = \left\{\mathbf{W}_k, \mathbf{b}_k\right\}_{k=1}^{L}. \tag{8}$$

The loss function used is a simple l_2 distance between the estimated signal and the true signal:

$$l\left(\mathbf{x}_{oh}; \hat{\mathbf{x}}_{oh}\left(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta}\right)\right) = \left\|\mathbf{x}_{oh} - \hat{\mathbf{x}}_{oh}\right\|^2 \tag{9}$$

FullyCon is simple and general purpose. It has a relatively small number of parameters to optimize. It only uses the input $\mathbf{y}$, and does not exploit the channel $\mathbf{H}$ within (7). The dependence on the channel is indirect via the expectation in (6) which depends on $\mathbf{H}$ and leads to parameters that depend on its moments. The result is a simple and straight forward structure which is ideal for detection over the FC model. As will be detailed in the simulations section, it manages to achieve almost optimal accuracy with low complexity. On the other hand, our experiences with FullyCon for the VC model led to disappointing results. It was not expressive enough to capture the dependencies of changing channels. We also tried to add the channel matrix $\mathbf{H}$ as an input, and this attempt failed too. In the next subsection, we propose a more expressive architecture specifically designed for addressing this challenge.

B. DetNet

In this section we present an architecture designed specifically for MIMO detection that will be named from now on 'DetNet' (abbreviation of 'detection network'). The derivation begins by noting that an efficient MIMO detector should not work with $\mathbf{y}$ directly, but use the compressed sufficient statistic:

$$\mathbf{H}^T\mathbf{y} = \mathbf{H}^T\mathbf{H}\mathbf{x} + \mathbf{H}^T\mathbf{w}. \tag{10}$$

This hints that two main ingredients in the architecture should be $\mathbf{H}^T\mathbf{y}$ and $\mathbf{H}^T\mathbf{H}\mathbf{x}$. Second, our construction is based on mimicking a projected gradient descent like solution for the maximum likelihood optimization. Such an algorithm would lead to iterations of the form

$$\begin{aligned} \hat{\mathbf{x}}_{k+1} &= \Pi\left[\hat{\mathbf{x}}_k - \delta_k \left.\frac{\partial\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2}{\partial\mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k}\right] \\ &= \Pi\left[\hat{\mathbf{x}}_k - \delta_k\mathbf{H}^T\mathbf{y} + \delta_k\mathbf{H}^T\mathbf{H}\mathbf{x}_k\right], \end{aligned} \tag{11}$$

where $\hat{\mathbf{x}}_k$ is the estimate in the k'th iteration, $\Pi[\cdot]$ is a nonlinear projection operator, and δ_k is a step size. Intuitively, each iteration is a linear combination of the $\mathbf{x}_k$, $\mathbf{H}^T\mathbf{y}$, and $\mathbf{H}^T\mathbf{H}\mathbf{x}_k$ followed by a non-linear projection. We enrich these iterations by lifting the input to a higher dimension in each iteration and applying standard non-linearities which are common in deep neural networks. In order to further improve the performance we treat the gradient step sizes δ_K at each step as a learned parameter and optimize them during the training phase. This yields the following architecture:

$$\begin{aligned} \mathbf{q}_k &= \hat{\mathbf{x}}_{k-1} - \delta_{1k}\mathbf{H}^T\mathbf{y} + \delta_{2k}\mathbf{H}^T\mathbf{H}\mathbf{x}_{k-1} \\ \mathbf{z}_k &= \rho\left(\mathbf{W}_{1k}\left[\begin{array}{c}\mathbf{q}_k \\ \mathbf{v}_{k-1}\end{array}\right] + \mathbf{b}_{1k}\right) \\ \hat{\mathbf{x}}_{oh,k} &= \mathbf{W}_{2k}\mathbf{z}_k + \mathbf{b}_{2k} \\ \hat{\mathbf{x}}_k &= \mathbf{f}_{oh}(\hat{\mathbf{x}}_{oh,k}) \\ \hat{\mathbf{v}}_k &= \mathbf{W}_{3k}\mathbf{z}_k + \mathbf{b}_{3k} \\ \hat{\mathbf{x}}_0 &= \mathbf{0} \\ \hat{\mathbf{v}}_0 &= \mathbf{0}, \end{aligned} \tag{12}$$

with the trainable parameters

$$\boldsymbol{\theta} = \{\mathbf{W}_{1k}, \mathbf{b}_{1k}, \mathbf{W}_{2k}, \mathbf{b}_{2k}, \mathbf{W}_{3k}, \mathbf{b}_{1k}, \delta_{1k}, \delta_{2k}\}_{k=1}^{L}. \tag{13}$$

To enjoy the lifting and non-linearities, the parameters $\mathbf{W}_{1k}$ are defined as tall and skinny matrices. The final estimate is defined as $\hat{\mathbf{x}}_L$. For convenience, the structure of each DetNet layer is illustrated in Fig. 2.

Training deep networks is a difficult task due to vanishing gradients, saturation of the activation functions, sensitivity to initialization and more [32]. To address these challenges and following the notion of auxiliary classifiers feature in GoogLeNet [12], we adopted a loss function that takes into account the outputs of all of the layers:

$$l\left(\mathbf{x}_{oh}; \hat{\mathbf{x}}_{oh}\left(\mathbf{H}, \mathbf{y}; \boldsymbol{\theta}\right)\right) = \sum_{l=1}^{\mathbf{L}} \log(l)\|\mathbf{x}_{oh} - \hat{\mathbf{x}}_{oh,l}\|^2. \tag{14}$$

In our final implementation, in order to further enhance the performance of DetNet, we added a residual feature from ResNet [11] where the output of each layer is a weighted average with the output of the previous layer.

IV. Soft decision output

In this section, we consider a more general setting in which the MIMO detector needs to provide soft outputs. High end communication systems typically resort to iterative decoding where the MIMO detector and the error correcting decoder iteratively exchange information on the unknowns until convergence. For this purpose, the MIMO detector must replace its hard estimates with soft posterior distributions $\text{Prob}(x_j = s_i|\mathbf{y})$ for each unknown $j = 1, \cdots, 2K$ and each possible symbol $i = 1, \cdots, |\mathbb{S}|$. More precisely, it also needs to allow additional soft inputs but we leave this for future work.

Computation of the posteriors is straight forward based on Bayes law, but its complexity is exponential in the size of the signal and constellation. Similarly to the maximum likelihood algorithm in the hard decision case, this computation yields optimal accuracy yet is intractable. Thus, the goal in this section is to design networks that output approximate the posteriors. On first glance, this seems difficult to learn as we have no training set of posteriors and cannot define a loss function. Remarkably, this is not a problem and the probabilities of arbitrary constellations can be easily recovered using the standard l_2 loss function with respect to the one-hot representation x_{oh}. Indeed, consider a scalar x and a single $s \in \mathbb{S}$ associated with its one-hot bit x_{oh} then it is well known that

$$\begin{aligned} \arg\min_{\hat{x}_{oh}} E[||x_{oh} - \hat{x}_{oh}||^2|\mathbf{y}] &= E[x_{oh}|\mathbf{y}] \\ &= \underset{\mathrm{s}\in\mathbb{S}}{\mathrm{Prob}}(\mathrm{x_{oh,s}} = 1|\mathbf{y}) \\ &= \underset{\mathrm{s}\in\mathbb{S}}{\mathrm{Prob}}(\mathrm{x} = \mathrm{s}|\mathbf{y}) \end{aligned} \tag{15}$$

Thus, assuming that our network is sufficiently expressive and globally optimized, the one-hot output $\mathbf{x}\hat{_{oh}}$ will provide the exact posterior probabilities.

V. Numerical Results

In this section, we provide numerical results on the accuracy and complexity of the proposed networks in comparison to competing methods.

In the FC case, the results are over the 0.55-Toeplitz channel.

In the VC case and when testing the soft output performance, the results presented are over random channels, where each element is sampled i.i.d. from the normal distribution $\mathcal{N}(0, 1)$.

A. Implementation details

We train both networks using a variant of the stochastic gradient descent method [33], [34] for optimizing deep networks, named Adam Optimizer [35]. All networks were implemented using the Python based TensorFlow library [36].

To give a rough idea of the computation needed during the learning phase, optimizing the detectors in our numerical results in both architectures took around 3 days on a standard Intel i7-6700 processor. Each sample was independently generated from (2) according to the statistics of $\mathbf{x}$, $\mathbf{H}$ (either in the

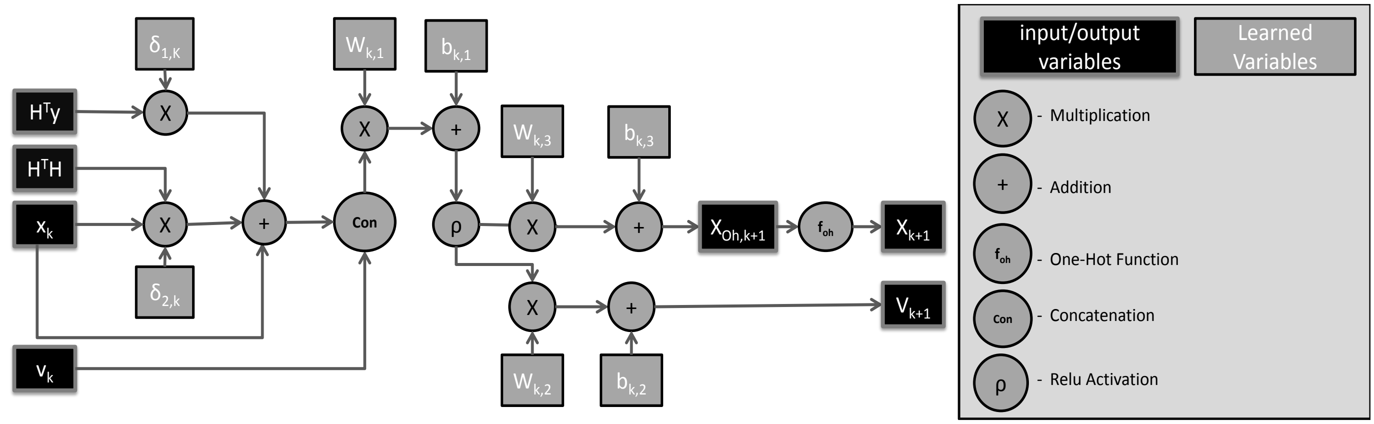

Fig. 2. A flowchart representing a single layer of DetNet. The network is composed out of L layers as such where each layers' output is the ext layers' input

FC or VC model) and $\mathbf{w}$. During training, the noise variance was randomly generated so that the SNR will be uniformly distributed on $\mathcal{U}\left(\mathrm{SNR}_{\min}, \mathrm{SNR}_{\max}\right)$.

B. *Competing algorithms*

When presenting our network performance we shall use the following naming conventions:

FullyCon: The basic fully-connected deep architecture.

DetNet: The DetNet deep architecture.

In the hard decision scenarios, we tested our deep networks against the following detection algorithms:

ZF: This is the classical decorrelator, also known as least squares or zero forcing (ZF) detector [1].

AMP: Approximate message passing algorithm from [5].

SDR: A decoder based on semidefinite relaxation implemented using an efficient interior point solver [6], [7]. For the 8-PSK constellation we implemented the SDR variation suggested in [37].

SD: An implementation of the sphere decoding algorithm as presented in [38].

In the soft output case, we tested our networks against the M-Best sphere decoding algorithm as presented in [3] (originally named K-Best, but changed here to avoid confusion with K the transmitted signal size):

M-Best SD M=5: The M-Best sphere decoding algorithm, where the number of candidates we keep is 5.

M-Best SD M=7: Same as M-Best SD M=5 with 7 candidates.

C. *Accuracy results*

1) Fixed Channel (FC): In the case of the FC scenario, where we know during the learning phase over what realization of the channel we need to detect, the performance of both our network was comparable to most of the competitors except SD. Both DetNet and FullyCon managed to achieve accuracy results comparable to SDR and AMP. This result emphasizes the notion that when learning to detect over simple scenarios as FC, a simple network is expressive enough. And since a simple network is easier to optimize and has lower complexity, it is preferable. Fig. 3 we present the accuracy rates over a range of SNR values in the FC model. This is a rather difficult setting and algorithms such as AMP did not succeed to converge.

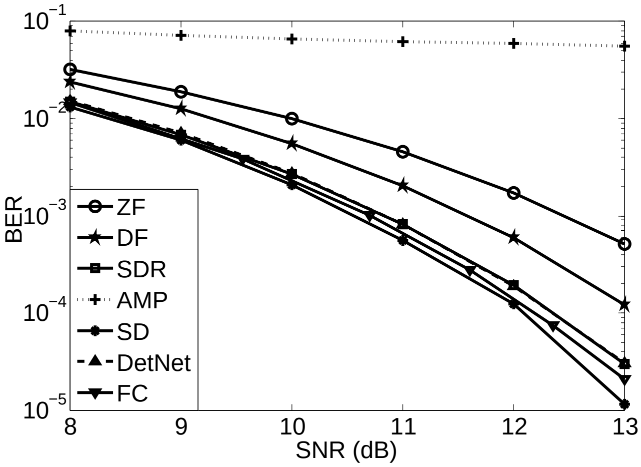

Fig. 3. Comparison of the detection algorithms BER performance in the fixed channel channel case over a BPSK modulated signal.

2) Varying channel: In the VC case, the accuracy results of FullyCon were poor and the network did not manage to learn how to detect properly. DetNet managed to achieve accuracy rates comparable to those of SDR and AMP, and almost comparable to those of SD, while being computationally cheaper (see next section regarding computational resources). In Fig. 4 we compare the accuracy results over a 30 × 60 real valued channel with BPSK signals and in Fig. 5 we compare the accuracy of a 20 × 30 complex channel with QPSK symbols. In both cases DetNet achieves accuracy rates comparable to SDR and AMP and near SD, and accuracy much better than ZF and DF. Results over larger constellations are presented in Fig. 6 and 7 where we compare the accuracy rates over complex channels of size 15×25 for the 16-QAM and 8-PSK constellations respectively.We can see that in those larger constellations DetNet performs better then AMP and SDR. For both constellations we can observe that DetNet reaches accuracy levels topped only by SD.

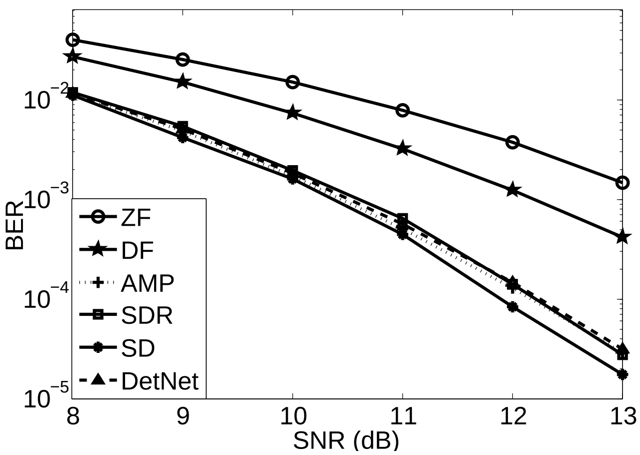

Fig. 4. Comparison of the detection algorithms BER performance in the varying channel case over a BPSK modulated signal. All algorithms were tested channels of size 30x60.

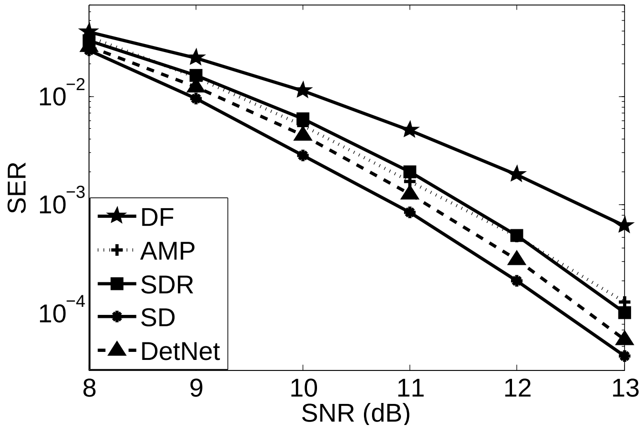

Fig. 5. Comparison of the detection algorithms BER performance in the varying channel case over a QPSK modulated signal. All algorithms were tested on channels of size 20x30.

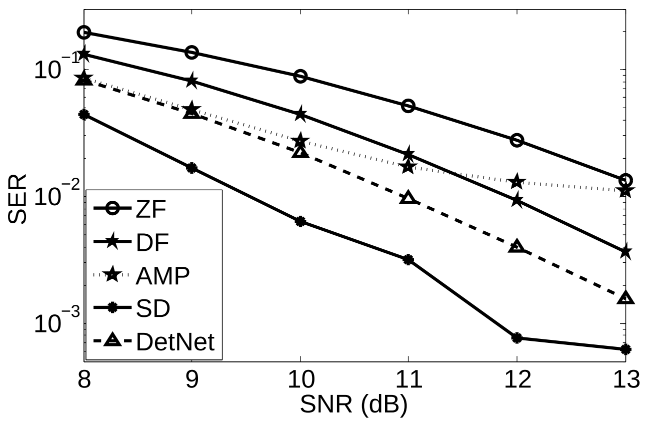

Fig. 6. Comparison of the detection algorithms SER performance in the varying channel case over a 16-QAM modulated signal. All algorithms were tested on channels of size 15X25.

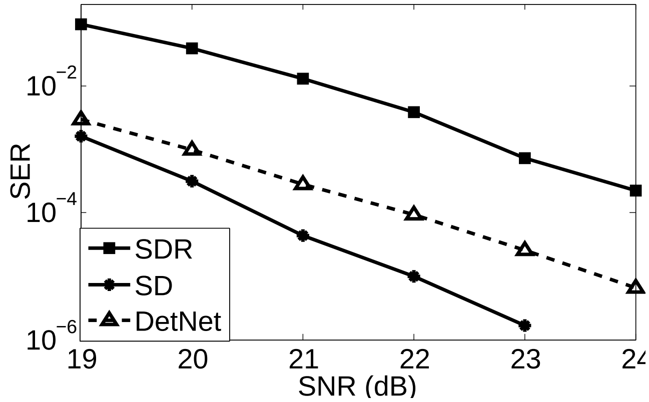

Fig. 7. Comparison of the detection algorithms SER performance in the varying channel case over a 8-PSK modulated signal. All algorithms were tested on channels of size 15X25.

3) Soft Outputs: We also experimented with soft decoding. Implementing a full iterative decoding scheme is outside the scope of this paper, and we only provide initial results on the accuracy of our posterior estimates. For this purpose, we examined smaller models where the exact posteriors can be computed exactly and measured their statistical distance to our estimates.

We shall define the following statistical distance function:

Given two probability distributions P and Q over the symbol set $\mathbb{S}$ (that is, the probability of each symbol to be the true symbol), the distance $\delta(P,Q)$ shall be:

$$\delta(P,Q) = \sum_{s \in \mathbb{S}} |P(s) - Q(s)| \qquad (16)$$

As reference, we compare our results to the M-Best detectors [3]. In Fig. 8 we present accuracy in the case of a BPSK signal over a 10x20 real channel. In this setting we reach accuracy levels better than those achieved by the M-Best algorithm. As seen in Fig. 8 adding additional layers improves the accuracy of the soft output. In Fig. 9 we present the results over a 4x8 complex channel with 16-QAM constellation. We can see the performance of DetNet is comparable to the M-Best Sphere decoding algorithm. For completeness, in Fig. 10 we added the 8-PSK constellation soft output where DetNet is comparable to the M-Best algorithms only in the high SNR region.

D. Computational Resources

1) FullyCon and DetNet run time: In order and estimate the computational complexity of the different detectors we compared their run time. Comparing complexity is non-trivial due to many complicating factors as implementation details and platforms. To ensure fairness, all the algorithms were tested on the same machine via python 2.7 environment using the Numpy package. The networks were converted from TensorFlow objects to Numpy objects. We note that the run-time of SD depends on the SNR, and we therefore report a range of times.

An important factor when considering the run time of the neural networks is the effect the batch size. Unlike classical detectors as SDR and SD, neural networks can detect over entire batches of data which speeds up the detection process. This is true also for the AMP algorithm, where computation can be made on an entire batch of signals at once. However, the

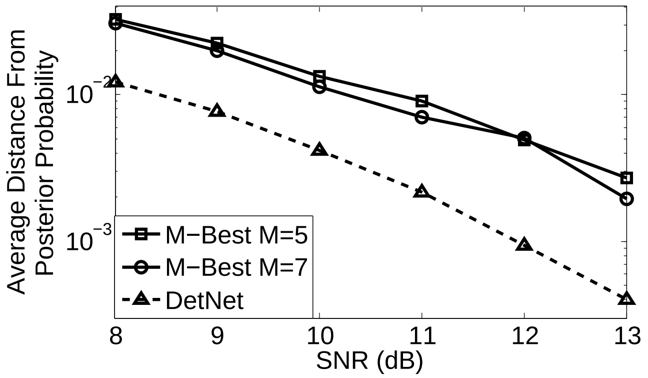

Fig. 8. Comparison of the accuracy of the soft output relative to the posterior probability in the case of a BPSK signal over a 10×20 real valued channel. We present the results for 2 types of DetNet, one with 30 layers and the second one with 50 layers

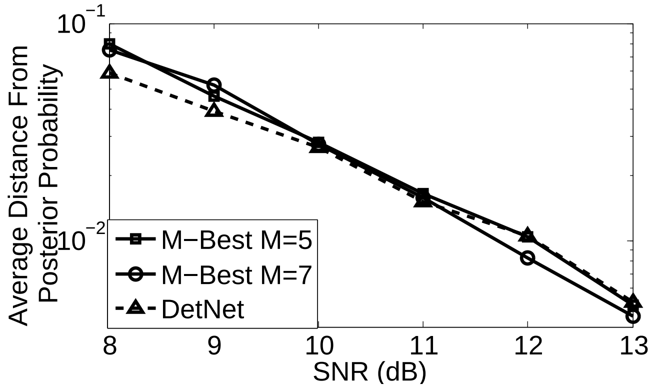

Fig. 9. Comparison of the accuracy of the soft output relative to the posterior probability for a 16-QAM signal over an 4×8 complex valued channel.

improvement introduced by using batches is highly dependent on the platform used (CPU/GPU/FPGA etc). Therefore, for completeness, we present the run time for several batch sizes including batch size equal to one.

In table I the run times are presented for hard decision detection in a FC case. We can see that FullyCon is faster than all other detection algorithms, even without using batches. DetNet is slightly faster than traditional detection algorithms without using batches, yet when using batches, the run time improves significantly compared to other detection methods.

Channel size	Batch size	FullyCon	DetNet	SDR	AMP	SD
Top055 30x60	1	0.0004	0.0045	0.009	0.005	0.001 -0.01
Top055 30x60	10	6.6E-05	0.0007	0.009	0.001	0.001 -0.01
Top055 30x60	100	2.4E-05	1.6E-04	0.009	0.0003	0.001 -0.01
Top055 30x60	1000	1.6E-05	1.1E-04	0.009	0.0003	0.001 -0.01

TABLE I
FIXED CHANNEL RUNTIME COMPARISON

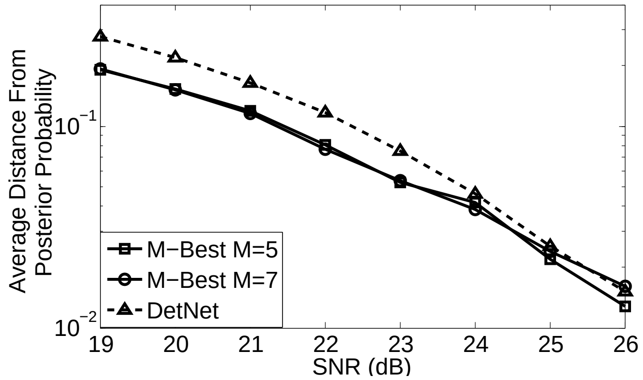

Fig. 10. Comparison of the accuracy of the soft output relative to the posterior probability for a 8-PSK signal over an 4×8 complex valued channel.

In table II we present the results for the VC setting. In the BPSK case the relative time difference between the different detection algorithms is similar to the FC case, with the exception of SD being relatively slower. In larger constellations (8-PSK/16-QAM) DetNet's relative advantage when comparing against AMP/SDR is smaller than in the BPSK case (and in the 16-QAM constellation AMP was slightly faster without using batches). The reason is that these accurate detection with these constellations requires larger networks. On the other hand, the relative performance vs SD improved.

Constellation channel size	Batch size	DetNet	SDR	AMP	SD
BPSK 30X60	1	0.0066	0.024	0.0093	0.008 -0.1
BPSK 30X60	10	0.0011	0.024	0.0016	0.008 -0.1
BPSK 30X60	100	0.0005	0.024	0.00086	0.008 -0.1
16-QAM 15X25	1	0.006	-	0.01	0.01 -0.4
16-QAM 15X25	10	0.0014	-	0.002	0.01 -0.4
16-QAM 15X25	100	0.0003	-	0.001	0.01 -0.4
8-PSK 15X25	1	0.019	0.021	-	0.004 -0.06
8-PSK 15X25	10	0.0029	0.021	-	0.004 -0.06
8-PSK 15X25	100	0.0005	0.021	-	0.004 -0.06

TABLE II
RUN TIME COMPARISON IN VC. DETNET IS COMPARED WITH THE SDR,AMP AND SPHERE DECODING ALGORITHMS

In table III we compare the run time of the detection algorithms in the soft-output case.As we can see, in the BPSK case without using batches the performance of DetNet is comparable to the performance of the M-Best sphere decoders, and using batches improves the performance significantly. In the 16-QAM/8-PSK cases DetNet is slightly faster than the M-Best decoders even without using batches.

2) Accuracy-Complexity Trade-Off: An interesting feature of DetNet is that the complexity-accuracy trade-off can be decided during run-time. Each of the network's layers outputs an estimated signal, and our loss optimizes all of them. We usually use the output of the last layer as the result since it is the most accurate, but it is possible to take the estimated

Constellation channel size	Batch size	DetNet	M-Best (M=5)	M-Best (M=7)
BPSK 10X20	1	0.0075	0.006	0.008
BPSK 10X20	10	0.00092	0.006	0.008
BPSK 10X20	100	0.00029	0.006	0.008
16-QAM 4X8	1	0.006	0.008	0.01
16-QAM 4X8	10	0.0008	0.008	0.01
16-QAM 4X8	100	0.0001	0.008	0.01
8-PSK 4X8	1	0.02	0.05	0.07
8-PSK 4X8	10	0.003	0.05	0.07
8-PSK 4X8	100	0.0012	0.05	0.07

TABLE III

RUN TIME COMPARISON OF SOFT OUTPUT IN VC. THE DETNET IS COMPARED WITH THE M-BEST SPHERE DECODING ALGORITHM

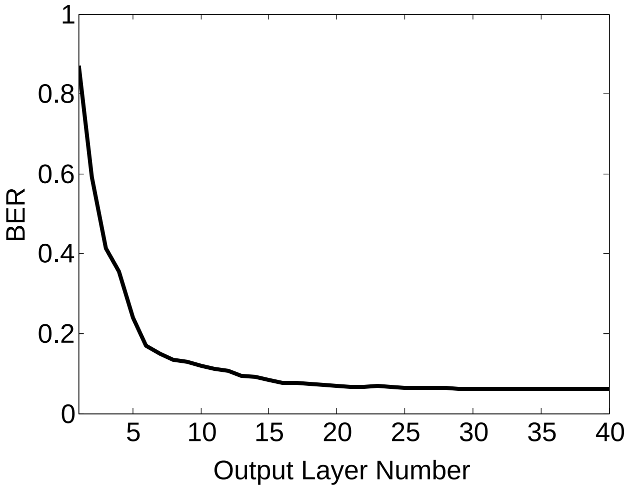

Fig. 11. Comparison of the average BER as a function of the layer chosen to be the output layer.

output $\mathbf{x}_i$ of previous layers to allow faster detection. In Fig. 11 we present the accuracy as a function of the number of layers.

VI. CONCLUSION

In this paper we investigated into the ability of deep neural networks to serve as MIMO detectors. We introduced two deep learning architectures that provide promising accuracy with low and flexible computational complexity. We demonstrated their application to various digital constellations, and their ability to provide accurate soft posterior outputs. An important feature of one of our network is its ability to detect over multiple channel realizations with a single training.

Using neural networks as a general scheme in MIMO detection still a long way to go and there are many open questions. These include their hardware complexity, robustness, and integration into full communication systems. Nonetheless, we believe this approach is promising and has the potential to impact future communication systems. Neural networks can be trained on realistic channel models and tune their performance for specific environments. Their architectures and batch operation are more natural to hardware implementation than algorithms as SDR and SD. Finally, their multi-layer structure allows a flexible accuracy vs complexity nature as required by many modern applications.

ACKNOWLEDGMENTS

We would like to thank Shai Shalev-Shwartz for many discussions throughout this research. In addition, we thank Amir Globerson and Yoav Wald for their ideas and help with the soft output networks.

REFERENCES

[1] S. Verdu, *Multiuser detection*. Cambridge university press, 1998.

[2] E. Agrell, T. Eriksson, A. Vardy, and K. Zeger, “Closest point search in lattices,” *IEEE transactions on information theory*, vol. 48, no. 8, pp. 2201–2214, 2002.

[3] Z. Guo and P. Nilsson, “Algorithm and implementation of the k-best sphere decoding for mimo detection,” *IEEE Journal on selected areas in communications*, vol. 24, no. 3, pp. 491–503, 2006.

[4] S. Suh and J. R. Barry, “Reduced-complexity MIMO detection via a slicing breadth-first tree search,” *IEEE Transactions on Wireless Communications*, vol. 16, no. 3, pp. 1782–1790, 2017.

[5] C. Jeon, R. Ghods, A. Maleki, and C. Studer, “Optimality of large MIMO detection via approximate message passing,” in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2015, pp. 1227–1231.

[6] Z. Q. Luo, W. K. Ma, A. M. So, Y. Ye, and S. Zhang, “Semidefinite relaxation of quadratic optimization problems,” *IEEE Signal Processing Magazine*, vol. 27, no. 3, pp. 20–34, 2010.

[7] J. Jald’en and B. Ottersten, “The diversity order of the semidefinite relaxation detector,” *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1406–1422, 2008.

[8] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[9] G. Hinton, L. Deng, D. Yu, G. Dahl, A. Mohamed, N. Jaitly, A. Senior, V. Vanhoucke, P. Nguyen, T. Sainath *et al.*, “Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups,” *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 82–97, 2012.

[10] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *International Conference on Machine Learning*, 2014, pp. 1764–1772.

[11] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 770–778.

[12] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, “Going deeper with convolutions,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1–9.

[13] D. Silver, A. Huang, C. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, “Mastering the game of Go with deep neural networks and tree search,” *nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[14] J. R. Hershey, J. L. Roux, and F. Weninger, “Deep unfolding: Model-based inspiration of novel deep architectures,” *arXiv preprint arXiv:1409.2574*, 2014.

[15] K. Gregor and Y. LeCun, “Learning fast approximations of sparse coding,” in *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, 2010, pp. 399–406.

[16] M. Borgerding and P. Schniter, “Onsager-corrected deep learning for sparse linear inverse problems,” in *Signal and Information Processing (GlobalSIP), 2016 IEEE Global Conference on*. IEEE, 2016, pp. 227–231.

[17] E. Nachmani, Y. Be’ery, and D. Burshtein, “Learning to decode linear codes using deep learning,” in *Communication, Control, and Computing (Allerton), 2016 54th Annual Allerton Conference on*. IEEE, 2016, pp. 341–346.

[18] E. Nachmani, E. Marciano, D. Burshtein, and Y. Be’ery, “RNN decoding of linear block codes,” *arXiv preprint arXiv:1702.07560*, 2017.

[19] E. Nachmani, E. Marciano, L. Lugosch, W. J. Gross, D. Burshtein, and Y. Be’ery, “Deep learning methods for improved decoding of linear codes,” *IEEE Journal of Selected Topics in Signal Processing*, 2018.

[20] T. J. O’Shea and J. Hoydis, “An introduction to machine learning communications systems,” *arXiv preprint arXiv:1702.00832*, 2017.

[21] T. Gruber, S. Cammerer, J. Hoydis, and S. ten Brink, “On deep learning-based channel decoding,” in *Information Sciences and Systems (CISS), 2017 51st Annual Conference on*. IEEE, 2017, pp. 1–6.

[22] N. Farsad and A. Goldsmith, “Detection algorithms for communication systems using deep learning,” *arXiv preprint arXiv:1705.08044*, 2017.

[23] ——, “Neural network detection of data sequences in communication systems,” *arXiv preprint arXiv:1802.02046*, 2018.

[24] H. Ye, G. Li, and B. Juang, “Power of deep learning for channel estimation and signal detection in OFDM systems,” *IEEE Wireless Communications Letters*, 2017.

[25] T. O’Shea, K. Karra, and T. Clancy, “Learning approximate neural estimators for wireless channel state information,” *arXiv preprint arXiv:1707.06260*, 2017.

[26] S. Dörner, S. Cammerer, J. Hoydis, and S. ten Brink, “Deep learning based communication over the air,” *IEEE Journal of Selected Topics in Signal Processing*, vol. 12, no. 1, pp. 132–143, 2018.

[27] T. O’Shea, T. Erpek, and T. Clancy, “Deep learning based MIMO communications,” *arXiv preprint arXiv:1707.07980*, 2017.

[28] N. Samuel, T. Diskin, and A. Wiesel, “Deep MIMO detection,” *arXiv preprint arXiv:1706.01151*, 2017.

[29] T. Wang, C. Wen, H. Wang, F. Gao, T. Jiang, and S. Jin, “Deep learning for wireless physical layer: Opportunities and challenges,” *China Communications*, vol. 14, no. 11, pp. 92–111, 2017.

[30] T. Diskin, G. Draskovic, F. Pascal, and A. Wiesel, “Deep robust regression,” in *Computational Advances in Multi-Sensor Adaptive Processing (CAMSAP), 2017 IEEE 7th International Workshop on*. IEEE, 2017, pp. 1–5.

[31] A. Mousavi and R. G. Baraniuk, “Learning to invert: Signal recovery via deep convolutional networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2017 IEEE International Conference on*. IEEE, 2017, pp. 2272–2276.

[32] X. Glorot and Y. Bengio, “Understanding the difficulty of training deep feedforward neural networks.” in *Aistats*, vol. 9, 2010, pp. 249–256.

[33] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning representations by back-propagating errors,” *Cognitive modeling*, vol. 5, no. 3, p. 1, 1988.

[34] L. Bottou, “Large-scale machine learning with stochastic gradient descent,” in *Proceedings of COMPSTAT’2010*. Springer, 2010, pp. 177–186.

[35] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[36] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin *et al.*, “Tensorflow: Large-scale machine learning on heterogeneous distributed systems,” *arXiv preprint arXiv:1603.04467*, 2016.

[37] W. Ma, P. Ching, and Z. Ding, “Semidefinite relaxation based multiuser detection for m-ary PSK multiuser systems,” *IEEE Transactions on Signal Processing*, vol. 52, no. 10, pp. 2862–2872, 2004.

[38] A. Ghasemmehdi and E. Agrell, “Faster recursions in sphere decoding,” *IEEE Transactions on Information Theory*, vol. 57, no. 6, pp. 3530–3536, 2011.