Deep MIMO Detection

Neev Samuel, Tzvi Diskin and Ami Wiesel

School of Computer Science and Engineering
The Hebrew University of Jerusalem
The Edmond J. Safra Campus
9190416 Jerusalem, Israel

***Abstract*—In this paper, we consider the use of deep neural networks in the context of Multiple-Input-Multiple-Output (MIMO) detection. We give a brief introduction to deep learning and propose a modern neural network architecture suitable for this detection task. First, we consider the case in which the MIMO channel is constant, and we learn a detector for a specific system. Next, we consider the harder case in which the parameters are known yet changing and a single detector must be learned for all multiple varying channels. We demonstrate the performance of our deep MIMO detector using numerical simulations in comparison to competing methods including approximate message passing and semidefinite relaxation. The results show that deep networks can achieve state of the art accuracy with significantly lower complexity while providing robustness against ill conditioned channels and mis-specified noise variance.**

***Keywords*—MIMO Detection, Deep Learning, Neural Networks.**

I. Introduction

Multiple input multiple output (MIMO) systems arise in most modern communication channels. The dimensions can account for time and frequency resources, multiple users, multiple antennas and other resources. These promise substantial performance gains, but present a challenging detection problem in terms of computational complexity. In recent years, the world is witnessing a revolution in deep machine learning. In many fields of engineering, e.g., computer vision, it was shown that computers can be fed with sample pairs of inputs and desired outputs, and "learn" the functions which relates them. These rules can then be used to classify (detect) the unknown outputs of future inputs. The goal of this paper is to apply deep machine learning in the classical MIMO detection problem and understand its advantages and disadvantages.

A. Background on MIMO detection

The binary MIMO detection setting is a classical problem in simple hypothesis testing [1]. The maximum likelihood (ML) detector is the optimal detector in the sense of minimum joint probability of error for detecting all the symbols simultaneously. It can be implemented via efficient search algorithms, e.g., the sphere decoder [2]. The difficulty is that its worst case computational complexity is impractical for many applications. Consequently, several modified search algorithms have been purposed, offering improved complexity performance [3][4]. There has been much interest in implementing suboptimal detection algorithms. 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 problematic settings. SDR is more robust and has polynomial complexity, but is much slower in practice.

B. Background on Machine Learning

In the last decade, there is an explosion of machine learning success stories in all fields of engineering. Supervised classification is similar to statistical detection theory. Both observe noisy data and output a decision on the discrete unknown it originated from. Typically, the two fields differ in that detection theory is based on a prior probabilistic model of the environment, whereas learning is data driven and is based on examples. In the context of MIMO detection, a model is known and allows us to generate as many synthetic examples as needed. Therefore we adapt an alternative notion. We interpret "learning" as the idea of choosing a best decoder from a prescribed class of algorithms. Classical detection theory tries to choose the best estimate of the unknowns, whereas machine learning tries to choose the best algorithm to be applied. Indeed, the hypotheses in detection are the unknown symbols, whereas the hypotheses in learning are the detection rules [8]. Practically, this means that the computationally involved part of detection is applied every time we get a new observation. In learning, the expensive stage is learning the algorithm which is typically performed off line. Once the optimal rule algorithm is found, we can cheaply implement it in real time.

Machine learning has a long history but was previously limited to simple and small problems. Fast forwarding to the last 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 [9], 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 including speech processing and computer vision. In particular, one promising approach to designing deep architectures is by unfolding an existing iterative algorithm [10]. 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. 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 [11], [12].

In recent years, deep learning methods have been purposed for improving the performance of a decoder for linear codes in fixed channels[13]. And in [14] several applications of deep learning for communication applications have been considered, including decoding signals over fading channels, but the architecture purposed there does not seem to be scalable for higher dimension signals.

C. Main contributions

The main contribution of this paper is the introduction of DetNET, a deep learning network for MIMO detection. DetNet is derived by unfolding a projected gradient descent method. Simulations show that it achieves near optimal detection performance while being a fast algorithm that can be implemented in real-time. Its accuracy is similar to SDR with running time that is more than 30 times faster. Compared to AMP, another detector with optimality guarantees, DetNet is more robust. It shows promising performance in handling ill conditioned channels, and does not require knowledge of the noise variance.

Another important contribution, in the general context of deep learning, is DetNet's ability to perform on multiple models with a single training. Recently, there were many works on learning to invert linear channels and reconstruct signals [11], [12], [15]. To the best of our knowledge, all of 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.

D. Notation

In this paper, we shall 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 $\left(\cdot\right)^T$ 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, will refer 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 $\rho(x)=\max\{0,x\}$ will be denoted as ρ.

II. Learning to detect

In this section, we formulate the MIMO detection problem in a machine learning framework. We consider the standard linear MIMO model:

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

where $\mathbf{y}\in\mathbb{R}^N$ is the received vector, $\mathbf{H}\in\mathbb{R}^{N\times K}$ is the channel matrix, $\mathbf{x}\in\{\pm1\}^K$ is an unknown vector of independent and equal probability binary symbols, $\mathbf{w}\in\mathbb{R}^N$ is a noise vector with independent, zero mean Gaussian variables of variance σ^2. We do not assume knowledge of the variance as hypothesis testing theory guarantees that this is unnecessary for optimal detection. Indeed, the optimal ML rule does not require knowledge of σ^2. This is contrast to the MMSE and AMP decoders that exploit this parameter and are therefore less robust.

We assume perfect channel state information (CSI) and that the channel $\mathbf{H}$ is exactly known. 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).
- Varying Channel (VC): In the VC scenario, we assume $\mathbf{H}$ random with a known distribution.

Our goal is to detect $\mathbf{x}$, using an algorithm that receives $\mathbf{y}$ and $\mathbf{H}$ as inputs and estimates $\hat{\mathbf{x}}$.

The first step is choosing and fixing a detection architecture. An architecture is a function $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{H},\mathbf{y})$ that detects the unknown symbols given $\mathbf{y}$ and $\mathbf{H}$. The architecture is parametrized by $\boldsymbol{\theta}$. Learning is the problem of finding the $\boldsymbol{\theta}$ within the feasible set that will lead to strong detectors $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{H},\mathbf{y})$. By choosing different functions and parameter sets, we characterize competing types of detectors which tradeoff accuracy with complexity.

To find the best detector, we fix a loss function $l\left(\mathbf{x};\hat{\mathbf{x}}_{\boldsymbol{\theta}}\left(\mathbf{H},\mathbf{y}\right)\right)$ that measures the distance between the true symbols and their estimates. Then, we find $\boldsymbol{\theta}$ by minimizing the loss function we chose over the MIMO model distribution:

$$\min_{\boldsymbol{\theta}}\mathrm{E}\left\{l\left(\mathbf{x};\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{H},\mathbf{y})\right)\right\}, \tag{2}$$

where the expectation is with respect to all the random variables in (1), i.e., $\mathbf{x}$, $\mathbf{w}$, and $\mathbf{H}$. Learning to detect is defined as finding the best set of parameters $\boldsymbol{\theta}$ of the architecture $\hat{\mathbf{x}}_{\boldsymbol{\theta}}\left(\mathbf{y},\mathbf{H}\right)$ that minimize the expected loss $l\left(\cdot;\cdot\right)$ over the distribution in (1).

The next examples illustrate how the choice of architecture $\hat{\mathbf{x}}_{\boldsymbol{\theta}}\left(\mathbf{y},\mathbf{H}\right)$ leads to different detectors that tradeoff accuracy for complexity.

Example 1: The goal in detection is to decrease the probability of error. Therefore, the best loss function in this problem

$$l\left(\mathbf{x};\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{H},\mathbf{y})\right)=\begin{cases}1 & \mathbf{x}\neq\hat{\mathbf{x}}_{\boldsymbol{\theta}}\left(\mathbf{y},\mathbf{H}\right)\\ 0 & \text{else.}\end{cases} \tag{3}$$

By choosing an unrealistically flexible architecture with unbounded parameterization and no restrictions such that

$$\left\{\hat{\mathbf{x}}_{\boldsymbol{\theta}}\left(\mathbf{y},\mathbf{H}\right)\ :\ \boldsymbol{\theta}\right\}=\left\{\begin{array}{c}\text{all the functions}\\ \mathbb{R}^N\times\mathbb{R}^{N\times K}\mapsto\{\pm1\}^K\end{array}\right\}. \tag{4}$$

Then, the solution to (2) is the ML decoder:

$$\hat{\mathbf{x}}_{\boldsymbol{\theta}}\left(\mathbf{y},\mathbf{H}\right)=\arg\min_{\mathbf{x}\in\{\pm1\}^K}\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2. \tag{5}$$

This rule is optimal in terms of accuracy but requires a computationally intensive search of $O\left(2^K\right)$. Obviously, this example is theoretical since the architecture of all possible functions cannot be parametrized and (2) cannot be optimized.

Example 2: On the other extreme, consider the architecture of fixed linear detectors:

$$\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{y},\mathbf{H}) = \mathbf{A}\mathbf{y}, \tag{6}$$

where the parameter $\boldsymbol{\theta}$ is a single fixed matrix to be optimized within $\mathbb{R}^{K\times N}$. In the FC model, choosing $\|\mathbf{x}-\hat{\mathbf{x}}(\mathbf{y},\mathbf{H})\|^2$ as the loss function and assuming $\sigma^2 \to 0$, the optimal decoder is the well known decorrelator:

$$\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{y},\mathbf{H}) = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{y}. \tag{7}$$

The resulting detector involves a simple matrix multiplication that requires $O(NK)$ operations, but is not very accurate. On the other hand, if we consider the more challenging VC model, then the optimal linear transformation is simply $\mathbf{H}=\mathbf{0}$. A single linear decoder cannot decode arbitrary channels simultaneously, and the decoder is completely useless.

These two examples emphasize how fixing an architecture and a loss function determines what will be the optimal detector for the MIMO detection problem. The more expressive we choose $\hat{\mathbf{x}}$ to be, the more accurate the final detector can be, on the expense of the computational complexity.

We close this section with a technical note on the numerical implementation of the optimization in (2). In practice, it is intractable to optimize over an abstract class of functions as in (4). Numerical minimization is typically performed with respect to a finite parameter set as in (6). Thus, our deep architectures are based on multiple layers with multivariate linear operations and element-wise non-linear operators. These allow rich decoding functions while resorting to a finite and tractable parameterization. In addition, analytic computation of the expectation in the objective is usually impossible. Instead, we approximate it using an empirical mean of samples drawn from a data set of examples (thus the 'learning' notion). In our case, the data set is composed of synthetically generated samples satisfying (1). Both these technicalities, were considered unthinkable just a decade ago, but are now standard procedures in the deep learning community. Easy to use, open source tools, make it possible to create deep architectures and optimize them in a straight forward manner. Specifically, in this work, all the experiments were implemented on the TensorFlow framework [16].

III. Deep MIMO detector

In this section, we propose a deep detector with an architecture which is specifically designed for MIMO detection that will be named from now on 'DetNet' (Detection Network). First, we note that an efficient detector should not work with $\mathbf{y}$ directly, but use the compressed sufficient statistic:

$$\mathbf{H}^T\mathbf{y} = \mathbf{H}^T\mathbf{H}\mathbf{x} + \mathbf{H}^T\mathbf{w}. \tag{8}$$

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 ML optimization in (5). Such an algorithm would lead to iterations of the form

$$\begin{aligned}\hat{\mathbf{x}}_{k+1} &= \Pi\left[\hat{\mathbf{x}}_k - \delta_k \left.\frac{\partial\|\mathbf{y}-\mathbf{H}\mathbf{x}\|^2}{\partial\mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}_k}\right] \\ &= \Pi\left[\hat{\mathbf{x}}_k - \delta_k\mathbf{H}^T\mathbf{y} + \delta_k\mathbf{H}^T\mathbf{H}\mathbf{x}_k\right],\end{aligned} \tag{9}$$

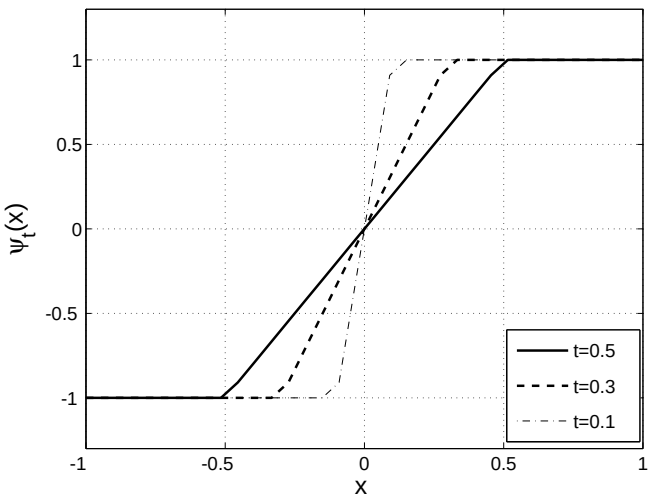

Fig. 1. A graph illustrating the linear soft sign function $\psi_t(\mathbf{x})$ for different values of the parameter t.

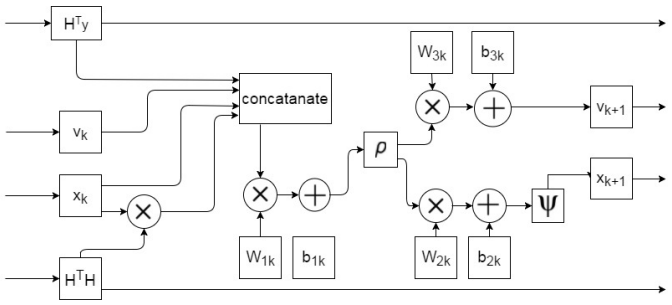

Fig. 2. A flowchart representing a single layer of DetNet.

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 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 and applying standard non-linearities which are common in deep neural networks. This yields the following architecture:

$$\begin{aligned}\mathbf{z}_k &= \rho\left(\mathbf{W}_{1k}\begin{bmatrix}\mathbf{H}^T\mathbf{y}\\ \hat{\mathbf{x}}_k\\ \mathbf{H}^T\mathbf{H}\hat{\mathbf{x}}_k\\ \mathbf{v}_k\end{bmatrix} + \mathbf{b}_{1k}\right)\\ \hat{\mathbf{x}}_{k+1} &= \psi_{t_k}\left(\mathbf{W}_{2k}\mathbf{z}_k + \mathbf{b}_{2k}\right)\\ \hat{\mathbf{v}}_{k+1} &= \mathbf{W}_{3k}\mathbf{z}_k + \mathbf{b}_{3k}\\ \hat{\mathbf{x}}_1 &= \mathbf{0},\end{aligned} \tag{10}$$

where $k=1,\cdots,L$ and $\psi_t(\cdot)$ is a piecewise linear soft sign operator defined as:

$$\psi_t(x) = -1 + \frac{\rho(x+t)}{|t|} - \frac{\rho(x-t)}{|t|}. \tag{11}$$

The operator is plotted in Fig. 1, and the structure of each DetNet layer is illustrated in Fig. 2. The final estimate is defined as $\hat{\mathbf{x}}_{\boldsymbol{\theta}}(\mathbf{y},\mathbf{H}) = sign(\hat{\mathbf{x}}_L)$.

The parameters of DetNet that are optimized during the

learning phase are:

$$\boldsymbol{\theta} = \{\mathbf{W}_{1k}, \mathbf{b}_{1k}, \mathbf{W}_{2k}, \mathbf{b}_{2k}, \mathbf{W}_{3k}, \mathbf{b}_{1k}, \mathbf{t}_k\}_{k=1}^{L}. \tag{12}$$

Training deep networks is a difficult task due to vanishing gradients, saturation of the activation functions, sensitivity to initializations and more [17]. To address these challenges, we adopted a loss function that takes into account the outputs of all of the layers. Moreover, since the errors depend on the channel's realization, we decided to normalize the errors with those of the decorrelator. Together, this led to the following loss function:

$$l\left(\mathbf{x}; \hat{\mathbf{x}}_{\boldsymbol{\theta}}\left(\mathbf{H}, \mathbf{y}\right)\right) = \sum_{k=1}^{\mathbf{L}} \log(k) \frac{\|\mathbf{x} - \hat{\mathbf{x}}_k\|^2}{\|\mathbf{x} - \tilde{\mathbf{x}}\|^2}, \tag{13}$$

where:

$$\tilde{\mathbf{x}} = \left(\mathbf{H}^T\mathbf{H}\right)^{-1}\mathbf{H}^T\mathbf{y}. \tag{14}$$

is the standard decorrelator decoder.

In our final implementation, in order to further enhance the performance of DetNet, we added a residual feature from ResNet [18] where the output of each layer is a weighted average with the output of the previous layer. Note also that our loss function is motivated by the auxiliary classifiers feature in GoogLeNet [19].

We train the network using a variant of the stochastic gradient descent method [20],[21] for optimizing deep networks, named Adam Optimizer [22]. We used batch training with 5000 random data samples at each iteration, and trained the network for 50000 iterations. To give a rough idea of the complexity, learning the detectors in our numerical results took 2 days on a standard Intel i7-6700 processor. Each sample was independently generated from (1) according to the statistics of $\mathbf{x}$, $\mathbf{H}$ (either in the FC or VC model) and $\mathbf{w}$. With respect to the noise, its variance is unknown and therefore this too was randomly generated so that the SNR will be uniformly distributed on $\mathcal{U}\left(\mathrm{SNR}_{\min}, \mathrm{SNR}_{\max}\right)$. This approach allows us to train the network to detect over a wide range of SNR values.

IV. Numerical results

In this section, we demonstrate the advantages of our proposed detector using computer simulations.

All the experiments address a MIMO channel with an input of size $K = 30$ and output of size $N = 60$. It is well known that performance is highly dependent on the type of MIMO channel. Therefore, we tried two scenarios:

FC: In this model, we chose to test the algorithms on a deterministic and constant ill-conditioned matrix which is known to be challenging for detection [23]. The matrix was generated such that $\mathbf{H}^T\mathbf{H}$ would have a Toeplitz structure with $\left[\mathbf{H}^T\mathbf{H}\right]_{i,j} = 0.55^{|i-j|}$. We shall denote this matrix as the 0.55-Toeplitz matrix. This defines the singular values and right singular vectors of $\mathbf{H}$. Its left singular vectors were randomly generated uniformly in the space of orthogonal matrices, and then fixed throughout the simulations.

VC: In this model, the matrices $\mathbf{H}$ were randomly generated with i.i.d. $\mathcal{N}\left(0, 1\right)$ elements. Each example was independently generated within the same experiment.

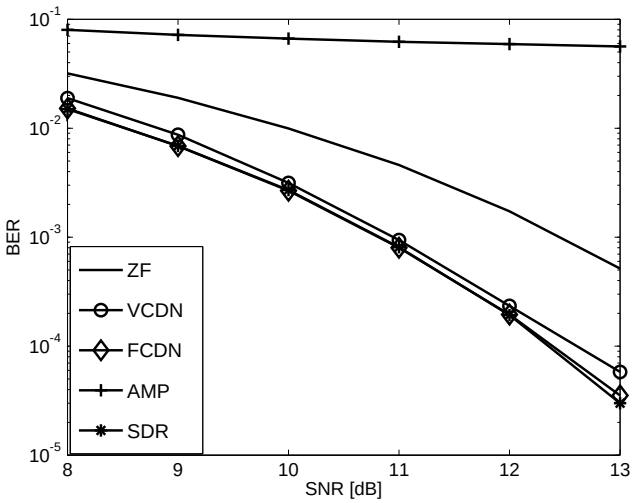

Fig. 3. Comparison of BER performance in the fixed channel case between the detection algorithms. all algorithms were tested on the 0.55-Toeplitz channel.

We have tested the performance of the following detection algorithms:

FCDN: DetNet algorithm described in (10) with $3K$ layers, $\mathbf{z}_k$ of size $8K$, and $\mathbf{v}_k$ of size $2K$. FCDN was trained using the FC model described above, and is specifically designed to handle a specific ill conditioned channel matrix.

VCDN: Same architecture as the FCDN but the training is on the VC model and is supposed to cope with arbitrary channel matrices.

ShVCDN : Same as the VCDN algorithm, but with a shallow network architecture using only K layers.

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

AMP: Approximate message passing algorithm from [5]. The algorithm was adjusted to the real-valued case and was implemented with $3K$ iterations.

AMP2: Same as the AMP algorithm but with a mis-specified SNR. The SNR in dB has an additional $\mathcal{N}(0, 2)$ bias.

SDR: A decoder based on semidefinite relaxation implemented using a specifically tailored and efficient interior point solver [6], [7].

In our first experiment, we focused on the FC model in which the channel is known and fixed, yet challenging due to its condition number. Figure 3 shows the results of all the algorithms in this setting. FCDN manages to reach the accuracy rates of the computationally expensive SDR algorithm which in our simulations took 30 times longer to detect. AMP does not manage to detect with reasonable accuracy in this challenging channel. It is interesting to notice that VCDN, which was not designed for this challenging channel, also manages to achieve good accuracy. This result indicates that VCDN generalizes itself during the training phase to detect over arbitrary random channels.

In our second experiment which results are presented in figure 4, we examine the performance in the VC model. SDR and AMP are theoretically known to be optimal in this setting, and VCDN manages to provide similar accuracy. Compared

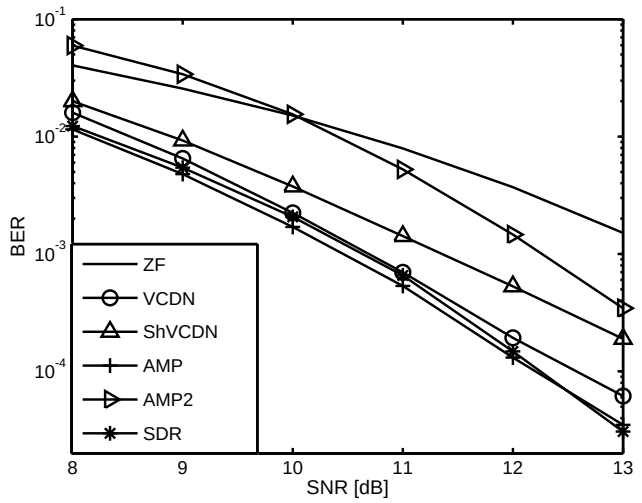

Fig. 4. Comparison of the detection algorithms BER performance in the varying channel case . All algorithms were tested on random i.i.d. Gaussian channels.

to SDR, VCDN runs 30 times faster. Compared to AMP in a scenario where the SNR values are not given accurately, we can notice a negative effect on the accuracy of the AMP, compared to VCDN that does not require any knowledge regarding the SNR.

Another important feature of DetNet is the ability to trade-off complexity and accuracy by adding or removing additional layers. In figure 4 we test the ShVCDN algorithm that is a shallow version on VCDN with only K layers , which is much faster, but less accurate. Since every layer in DetNet outputs a predicted signal $\hat{\mathbf{x}}_k$, we can decide in real-time what layer will be the final output layer, and trade-off complexity for accuracy in real-time, without any further training.

V. Conclusion

In this paper we have presented deep neural networks as a general framework for MIMO detection. We have tested the performance in the fixed channel scenario over challenging channels, and in the more complicated VC scenario. The DetNet architecture we have suggested has proven to be computationally inexpensive and has near-optimal accuracy without any knowledge regarding the SNR level. The ability of DetNet to optimize over an entire distribution of channels, rather than a single or even a large-finite set of channels, makes it robust and enables implementation in systems where the channel is not fixed. Simulations show that DetNet succeeds to generalize and detect accurately over channels with different characteristics than those of the channels used in the training phase of DetNet. For more details, see [24], where further information is presented.

Acknowledgments

We would like to thank Prof. Shai Shalev-Shwartz for his help during the research and his insights. This research was partly supported by the Heron Consortium and by ISF grant 1339/15.

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] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, New York, NY, USA, 2014.

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

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

[11] 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.

[12] M. Borgerding and P. Schniter, "Onsager-corrected deep learning for sparse linear inverse problems," *arXiv preprint arXiv:1607.05966*, 2016.

[13] E. Nachmani, Y. Beery, and D. Burshtein, "Learning to decode linear codes using deep learning," *arXiv preprint arXiv:1607.04793*, 2016.

[14] T. J. O'Shea and j. Hoydis, "An introduction to machine learning communications systems," *arXiv preprint arXiv:1702.00832*, 2017.

[15] A. Mousavi and R. G. Baraniuk, "Learning to invert: Signal recovery via deep convolutional networks," *arXiv preprint arXiv:1701.03891*, 2017.

[16] 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.

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

[18] 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.

[19] 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.

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

[21] L. Bottou, "Large-scale machine learning with stochastic gradient descent," in *Proceedings of COMPSTAT'2010*, pp. 177–186. Springer, 2010.

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

[23] S. Rangan, P. Schniter, and A. Fletcher, "On the convergence of approximate message passing with arbitrary matrices," in *IEEE International Symposium on Information Theory (ISIT)*. IEEE, 2014, pp. 236–240.

[24] N. Samuel and A. Wiesel, "Learning to detect," *In preparation for a journal*.