Deep Unfolding for Communications Systems: A Survey and Some New Directions

Conference Paper**Author(s):**

Balatsoukas-Stimming, Alexios; [Studer, Christoph](#)

Publication date:

2020-07

Permanent link:

<https://doi.org/10.3929/ethz-b-000460346>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

Originally published in:

<https://doi.org/10.1109/SiPS47522.2019.9020494>

Deep Unfolding for Communications Systems: A Survey and Some New Directions

Alexios Balatsoukas-Stimming^{1,2} and Christoph Studer³

¹*Telecommunications Circuits Laboratory, École polytechnique fédérale de Lausanne, Lausanne, Switzerland*

²*Electronic Systems Group, Eindhoven University of Technology, Eindhoven, The Netherlands (a.k.balatsoukas.stimming@tue.nl)*

³*School of Electrical and Computer Engineering, Cornell University, Ithaca, USA (studer@cornell.edu)*

Abstract—Deep unfolding is a method of growing popularity that fuses iterative optimization algorithms with tools from neural networks to efficiently solve a range of tasks in machine learning, signal and image processing, and communication systems. This survey summarizes the principle of deep unfolding and discusses its recent use for communication systems with focus on detection and precoding in multi-antenna (MIMO) wireless systems and belief propagation decoding of error-correcting codes. To showcase the efficacy and generality of deep unfolding, we describe a range of other tasks relevant to communication systems that can be solved using this emerging paradigm. We conclude the survey by outlining a list of open research problems and future research directions.

Index Terms—Machine learning, channel coding, massive MIMO, deep unfolding.

I. INTRODUCTION

A large number of signal processing tasks in communications systems, such as detection and decoding, can be formulated as optimization problems. These optimization problems are typically solved using numerical algorithms that iteratively refine the solution. Most practical communications applications require solving of these problems at high throughput and low latency, which implies that one can afford only a very small number of algorithm iterations (e.g., ten or fewer). In order to find accurate solutions with a small number of iterations, numerical solvers require careful parameter tuning (e.g., step-size selection). While the numerical optimization literature has focused extensively on analyzing convergence rates and stability given step-size conditions [1], only very little is known about optimal parameter tuning under stringent iteration constraints. In practice, the algorithm parameters are typically set using heuristics (e.g., tuned by hand using simulations) or pessimistic bounds (e.g., given by the Lipschitz constant). However, such conventional approaches are prone to result in suboptimal performance and may cause stability issues if the system conditions change (and the parameters would need to be adapted in real time).

A. Model-Driven Neural Networks via Deep Unfolding

In recent years, neural networks (NNs) have been proposed to replace a range of signal processing tasks in communica-

tions systems [2]–[6]. While the performance of such NN-assisted methods is promising in many applications, they suffer from the following drawbacks: (i) high computational complexity and memory requirements, and (ii) virtually no performance guarantees are available. As an alternative to such black-box methods, *model-driven* NNs are becoming increasingly popular in communications systems [7]. The idea of model-driven NNs is to fuse principled algorithms that have performance guarantees with tools from NNs, with the goal of combining the best of both worlds. *Deep unfolding* [8] is a powerful instance of such model-driven NNs and is also rapidly gaining popularity in the communications community.

In the words of the authors of [8], deep unfolding can be summarized as follows: “[...] *given a model-based approach that requires an iterative inference method, we unfold the iterations into a layer-wise structure analogous to a neural network.*” Put simply, deep unfolding takes an iterative algorithm with a fixed number of iterations T , unfolds its structure, and introduces a number of trainable parameters. These parameters are then learned using techniques from deep learning (with suitable loss functions, stochastic gradient descent, and back-propagation). The resulting unfolded algorithm with learned parameters can then be used to solve a range of tasks in communications systems. Deep unfolding has several practical advantages: (i) Existing performance guarantees for the original iterative algorithms may apply verbatim to learned unfolded networks and appropriate constraints can be imposed on the learned parameters. (ii) Most unfolded communications algorithms have a relatively small number of trainable parameters, which simplifies training. (iii) Unfolded algorithms are typically based on well-known methods for which efficient hardware implementations are readily available, which reduces design time. (iv) The resulting unfolded algorithms are often intuitive, interpretable, and have low complexity and memory requirements, which is in stark contrast to black-box NNs.

In this survey, we discuss several applications of deep unfolding to communications systems, with a particular focus on MIMO systems and belief-propagation-based decoding of error-correcting codes.¹ We also outline other applications and discuss a number of interesting open research directions.

The work of ABS was supported by the Swiss NSF project PZ00P2_179686. The work of CS was supported in part by Xilinx Inc. and the US NSF under grants ECCS-1408006, CCF-1535897, CCF-1652065, CNS-1717559, and ECCS-1824379. The authors would like to thank O. Castañeda and T. Goldstein for discussions on deep unfolding.

¹We note that our formulations of various algorithms may differ slightly from the notation used in the original papers for uniformity.

TABLE I
SUMMARY OF UNFOLDED LEARNED ALGORITHMS FOR MIMO DETECTION AND MIMO PRECODING.

Reference	[9]	[10]	[11]	[12]	[13] and [14]	[15]
Task	Detection	Detection	Detection	Detection	Detection	Precoding
Algorithm	PGD	PGD	PGD	OAMP	PGD	PGD
Parameters	$T(12M \times 2K + 2K + 6M)$	$T(12M \times 2K + 2K + 4M)$	$T(8M \times 2K + 2K + 4M)$	$2T$	$2T+1$	$2T$

II. DEEP UNFOLDING FOR MIMO SYSTEMS

We now describe applications of deep unfolding to signal processing tasks in multiple-input multiple-output (MIMO) wireless systems. More specifically, we discuss recent results on MIMO detection [9]–[14], [16], [17] and MIMO precoding [15], which are also summarized in Table I.

A. MIMO Data Detection

The common baseband input-output relation of data transmission over a frequency-flat MIMO channel is as follows:

$$\tilde{\mathbf{y}} = \tilde{\mathbf{H}}\tilde{\mathbf{x}} + \tilde{\mathbf{n}}. \quad (1)$$

Here, $\tilde{\mathbf{y}} \in \mathbb{C}^N$ is the complex-valued receive vector, $\tilde{\mathbf{H}} \in \mathbb{C}^{N \times M}$ is the $N \times M$ complex-valued MIMO channel matrix, $\tilde{\mathbf{x}} \in \mathcal{L}^M$ is the vector of transmit symbols, where $\mathcal{L}$ is the transmit constellation set, and $\tilde{\mathbf{n}} \in \mathbb{C}$ is a complex Gaussian noise vector distributed according to $\mathcal{CN}(0, \sigma^2 \mathbf{I})$.

The goal of MIMO data detection is to compute an estimate $\hat{\mathbf{x}}$ of the transmitted data vector $\tilde{\mathbf{x}}$, given the receive vector $\tilde{\mathbf{y}}$, the channel matrix $\tilde{\mathbf{H}}$, and knowledge of the statistics of $\tilde{\mathbf{n}}$. Maximum likelihood (ML) data detection for this model amounts to solving the following optimization problem:

$$\hat{\mathbf{x}} = \arg \min_{\tilde{\mathbf{x}} \in \mathcal{L}^M} \|\tilde{\mathbf{y}} - \tilde{\mathbf{H}}\tilde{\mathbf{x}}\|_2^2. \quad (2)$$

Since the ML data detection problem is NP-hard, computationally efficient approximate methods are used in practice. To use NN-based methods for detection, one often operates on real-valued data using the real-valued decomposition of (1):

$$\mathbf{y} = \mathbf{H}\mathbf{x} + \mathbf{n}, \quad (3)$$

where the quantities $\mathbf{y} \in \mathbb{R}^{2N}$, $\mathbf{H} \in \mathbb{R}^{2N \times 2M}$, $\mathbf{x} \in \mathbb{R}^{2M}$, and $\mathbf{n} \in \mathbb{R}^{2N}$, are defined as follows:

$$\mathbf{y}^T = [\Re\{\tilde{\mathbf{y}}\}^T \quad \Im\{\tilde{\mathbf{y}}\}^T], \quad \mathbf{H} = \begin{bmatrix} \Re\{\tilde{\mathbf{H}}\} & -\Im\{\tilde{\mathbf{H}}\} \\ \Im\{\tilde{\mathbf{H}}\} & \Re\{\tilde{\mathbf{H}}\} \end{bmatrix}, \quad (4)$$

$$\mathbf{x}^T = [\Re\{\tilde{\mathbf{x}}\}^T \quad \Im\{\tilde{\mathbf{x}}\}^T], \quad \mathbf{n}^T = [\Re\{\tilde{\mathbf{n}}\}^T \quad \Im\{\tilde{\mathbf{n}}\}^T]. \quad (5)$$

The NN-based MIMO data detection methods in [9], [10] are obtained by unfolding the iterations of projected gradient descent data detection algorithms (e.g., the method in [18]) and enriching these algorithms with additional dimensions and trainable parameters. Specifically, in [9] the following updates are used for $t = 0, \dots, T-1$:

$$\mathbf{z}_t = \text{ReLU} \left(\mathbf{W}_{1t} \begin{bmatrix} \mathbf{H}^T \mathbf{y} \\ \hat{\mathbf{x}}_t \\ \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_t \\ \mathbf{v}_t \end{bmatrix} + \mathbf{b}_{1t} \right), \quad (6)$$

$$\hat{\mathbf{x}}_{t+1} = \psi_{\mathbf{k}_t}(\mathbf{W}_{2t} \mathbf{z}_t + \mathbf{b}_{2t}), \quad (7)$$

$$\hat{\mathbf{v}}_{t+1} = \mathbf{W}_{3t} \mathbf{z}_t + \mathbf{b}_{3t}, \quad (8)$$

where $\text{ReLU}(x) = \max\{x, 0\}$, $\hat{\mathbf{x}}_0 = \mathbf{0}$, $\psi_{\mathbf{k}_t}(\cdot)$ is a soft sign operator parameterized by the trainable parameter vector $\mathbf{k}_t$, and $\mathbf{z}_t$ is of dimension $2K > 2M$. The quantity $\mathbf{H}^T \mathbf{y}$ is included in the input vector of (6) because it is a sufficient detection statistic, while $\mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_t$ is an estimate of the (noiseless) sufficient detection statistic at iteration t [9]. The set of trainable parameters is $\{\mathbf{W}_{1t}, \mathbf{W}_{2t}, \mathbf{W}_{3t}, \mathbf{b}_{1t}, \mathbf{b}_{2t}, \mathbf{b}_{3t}, \mathbf{k}_t : t = 0, \dots, T-1\}$; these parameters can be learned by specifying a suitable loss function and using tools from deep neural networks (such as stochastic gradient descent and back-propagation). Finally, hard decisions are obtained by computing $\hat{\mathbf{x}} = \text{sign}(\hat{\mathbf{x}}_T)$.

The work of [10] mimics a projected gradient algorithm more closely, thus simplifying the algorithm iteration (6)–(8):

$$\mathbf{z}_t = \text{ReLU} \left(\mathbf{W}_{1t} \begin{bmatrix} \hat{\mathbf{x}}_t - \delta_{1t} \mathbf{H}^T \mathbf{y} - \delta_{2t} \mathbf{H}^T \mathbf{H} \hat{\mathbf{x}}_t \\ \hat{\mathbf{v}}_t \end{bmatrix} + \mathbf{b}_{1t} \right), \quad (9)$$

$$\hat{\mathbf{x}}_{t+1} = \sigma(\mathbf{W}_{2t} \mathbf{z}_t + \mathbf{b}_{2t}), \quad (10)$$

$$\hat{\mathbf{v}}_{t+1} = \mathbf{W}_{3t} \mathbf{z}_t + \mathbf{b}_{3t}, \quad (11)$$

where $\sigma(\cdot)$ is a logistic sigmoid, and the trainable parameters $\{\mathbf{W}_{1t}, \mathbf{W}_{2t}, \mathbf{W}_{3t}, \mathbf{b}_{1t}, \mathbf{b}_{2t}, \mathbf{b}_{3t}, \delta_{1t}, \delta_{2t} : t = 0, \dots, T-1\}$ are reduced as the $\mathbf{W}_{1t}$ matrices are of lower dimension. To aid parameter learning, the cost function in [9], [10] is based on the outputs of all layers and a *residual feature* where the output of each layer is a weighted average with the output of the previous layer is also added. Details on how to obtain soft outputs and to extend the method to high-order constellations are provided in [10]. The unfolded data detectors are compared with a wide range of existing detectors in the literature as well as with a standard NN-based detector over different channel models and is shown to provide competitive performance.

To enable support for higher-order constellations, the work of [11] proposes to use a sum of shifted sigmoid functions:

$$\psi(x) = \sum_{i=1}^L \sigma(x - \tau_i) + A. \quad (12)$$

Here, the shifts τ_i are pre-defined based on the constellation set $\mathcal{L}$ and A is a fixed offset. Moreover, two distinct NN-based detectors are trained with different initialization strategies and the best output is kept. It is also argued that only ML-detectable training samples should be used for training. Simulation results show that close-to-ML performance is achieved for a constellation with $L = 5$ levels over a fixed (during training and data detection) MIMO channel with $N = M = 8$.

The work of [12] unfolds the iterations of an orthogonal approximate message passing (OAMP) detector [18], where only

Fig. 1. Computation graph structure of the unfolded MU-MIMO 1-bit precoding algorithm used in [15], where learnable parameters are highlighted in red. The other MIMO-related work discussed in this section [9]–[14] build upon very similar unfolded structures.

trainable scalars $\{\gamma_t, \theta_t : t = 0, \dots, T-1\}$ are introduced. The following updates are used for $t = 0, \dots, T-1$:

$$\mathbf{r}_t = \hat{\mathbf{x}}_t + \gamma_t \mathbf{W}_t (\mathbf{y} - \mathbf{H} \hat{\mathbf{x}}_t), \quad (13)$$

$$\hat{\mathbf{x}}_{t+1} = \mathbb{E}[\mathbf{x} | \mathbf{r}_t, \tau_t], \quad (14)$$

$$v_t^2 = \frac{\|\mathbf{y} - \mathbf{H} \hat{\mathbf{x}}_t\|_2^2 - M \sigma^2}{\text{tr}(\mathbf{H}^T \mathbf{H})}, \quad (15)$$

$$\tau_t^2 = \frac{1}{2N} \text{tr}(\mathbf{C}_t \mathbf{C}_t^T) v_t^2 + \frac{\theta_t^2 \sigma^2}{4N} \text{tr}(\mathbf{W}_t \mathbf{W}_t^T), \quad (16)$$

where $\mathbf{W}_t$ is a function of the channel matrix $\mathbf{H}$, v_t^2 , and σ^2 , and $\mathbf{C}_t$ is a function of $\mathbf{H}$ and $\mathbf{W}_t$, as defined in [12]. Simulation results for Rayleigh and correlated channels demonstrate that data-driven tuning of γ_t and θ_t can lead to significant performance improvements compared to standard OAMP.

The algorithms in [13], [14] target detection for massive overloaded MIMO channels, i.e., channels where $N \gg M$. The proposed data detection algorithm is based on projected gradient descent for a total of T iterations and with the introduced trainable scalars $\{\alpha, \gamma_t, \theta_t : t = 0, \dots, T-1\}$:

$$\mathbf{r}_t = \hat{\mathbf{x}}_t + \gamma_t \mathbf{W} (\mathbf{y} - \mathbf{H} \hat{\mathbf{x}}_t), \quad (17)$$

$$\hat{\mathbf{x}}_{t+1} = \tanh(\mathbf{r}_t / |\theta_t|), \quad (18)$$

where $\hat{\mathbf{x}}_0 = \mathbf{0}$ and $\mathbf{W} = \mathbf{H}^T (\mathbf{H} \mathbf{H}^T + \alpha \mathbf{I})^{-1}$. The authors use incremental training to avoid vanishing gradient problems. Simulation results show that the trained projected gradient detector provides similar performance to other detectors but at a significantly lower complexity. Moreover, the trained projected gradient detector is also shown to perform well in traditional, i.e., non-overloaded, MIMO systems.

Very recently, the papers [16] and [17] used deep unfolding for MIMO data detection based on conjugate gradients and projected gradient descent, respectively; both methods achieve near-ML performance at low complexity.

B. Multi-User (MU) MIMO Precoding

MU-MIMO precoding consists of multiplying the transmit symbol vector $\hat{\mathbf{x}}$ with a precoding matrix $\mathbf{P}$ so that a suitably defined performance metric (e.g., the SNR at the receiver) is maximized. The complex-valued system model in (1) is

$$\tilde{\mathbf{y}} = \tilde{\mathbf{H}} \mathbf{P} \tilde{\mathbf{x}} + \tilde{\mathbf{n}} = \tilde{\mathbf{H}} \tilde{\mathbf{v}} + \tilde{\mathbf{n}}, \quad (19)$$

where $\tilde{\mathbf{v}} = \mathbf{P} \tilde{\mathbf{x}}$ and the corresponding real-valued model of (3) using $\mathbf{y}$, $\mathbf{H}$, $\mathbf{v}$, and $\mathbf{n}$ can be derived accordingly.

The results in [15] describe a projected gradient descent algorithm for precoding in massive MU-MIMO systems with 1-bit quantization at the transmitter. In this scenario, each element of the vector $\mathbf{v}$ is constrained to the binary set $\{-v, +v\}$, where $v^2 = \frac{P}{2M}$ is selected to satisfy a transmit

power constraint P . The following updates with trainable scalars $\{\tau_t, \rho_t : t = 0, \dots, T-1\}$ are used for a total of T iterations:

$$\mathbf{z}_{t+1} = \mathbf{v}_{t+1} - \tau_t \mathbf{A}^H \mathbf{A} \mathbf{v}_t, \quad (20)$$

$$\mathbf{v}_{t+1} = \text{prox}_g(\mathbf{z}_{t+1}; \rho_t, \xi). \quad (21)$$

where $\text{prox}_g(\mathbf{z}; \rho_t, \xi) = \text{clip}(\rho \Re\{\mathbf{z}\}, \xi) + j \text{clip}(\rho \Im\{\mathbf{z}\}, \xi)$ is the *proximal operator*, $\mathbf{A} = (\mathbf{I} - \mathbf{x} \mathbf{x}^T / \|\mathbf{x}\|_2^2) \mathbf{H}$, and $\mathbf{v}_T$ is quantized to $\{-v, +v\}$. Simulation results for a range of channel models show that learning suitable parameters τ_t, ρ_t allows one to decrease the number of iterations T by a factor of two for the same error-rate performance. The computational graph corresponding to the unfolded version of (20) and (21) along with the final quantization $Q(\cdot)$ and transmission over $\mathbf{H}$ is shown in Fig. 1.

III. DEEP UNFOLDING FOR BELIEF-PROPAGATION-BASED CHANNEL DECODING

Belief propagation is an iterative message-passing algorithm that is commonly used to decode error-correcting codes. The message-passing strategy is typically described by a bipartite Tanner graph that represents the parity-check matrix of the code. These Tanner graphs consist of two types of nodes, namely *variable nodes* and *check nodes*. Each variable node is associated with a codeword bit and each check node is associated with a parity-check equation. Let $\mathcal{V}$ denote the set of variable nodes, $\mathcal{C}$ denote the set of check nodes, and $\mathcal{N}(x)$ denote the set of (one-hop) neighbors of a node x . Then, for each $v \in \mathcal{V}$, $c \in \mathcal{N}(v)$, the variable-to-check messages $m_t^{v \rightarrow c}$ at iteration $t \in \{1, \dots, T\}$ are:

$$m_t^{v \rightarrow c} = l^v + \sum_{c' \in \mathcal{N}(v) \setminus c} m_{t-1}^{c' \rightarrow v}, \quad (22)$$

where l^v denotes the channel log-likelihood ratio (LLR) for variable node v and $m_0^{c \rightarrow v} = 0$ by convention. Moreover, for each $c \in \mathcal{C}$, $v \in \mathcal{N}(c)$, the check-to-variable messages $m_t^{c \rightarrow v}$ at iteration t are given by

$$m_t^{c \rightarrow v} = 2 \tanh^{-1} \left(\prod_{v' \in \mathcal{N}(c) \setminus v} \tanh \left(\frac{m_{t-1}^{v' \rightarrow c}}{2} \right) \right). \quad (23)$$

For each $v \in \mathcal{V}$, the bit-decision metric is calculated as

$$m_t^v = l^v + \sum_{c \in \mathcal{N}(v)} m_{t-1}^{c \rightarrow v}, \quad (24)$$

and final bit-decisions are generated as follows:

$$\hat{u}_t^v = \frac{1}{2} (1 - \text{sign}(m_t^v)). \quad (25)$$

In [19], the variable-to-check BP equation in (22) is modified by adding trainable weights w_t^v and $w_t^{c'}$, which yields:

$$m_t^{v \rightarrow c} = w_t^v l^v + \sum_{c' \in \mathcal{N}(v) \setminus c} w_{t-1}^{c'} m_{t-1}^{c' \rightarrow v}. \quad (26)$$

Moreover, bit-decisions are generated using the following soft (i.e., differentiable) version of (25):

$$\hat{u}_t^v = \sigma(m_t^v). \quad (27)$$

The authors use a binary cross-entropy loss function that uses the $\hat{u}_t^v$ values from all iterations t in order to aid learning and avoid vanishing gradient problems. The unfolded BP decoder is trained using synthetically-generated training data for a range of different signal-to-noise-ratio (SNR) values. Simulation results for a variety of BCH codes show that the unfolded BP decoder with learned weights significantly outperforms traditional BP decoders.

The methods in [20], [21] improve upon [19] by using a recurrent neural network (RNN) structure so that the weights w_t^v and $w_t^{c'}$ do not change over the iterations. Moreover, the authors use a technique called *relaxation* where consecutive messages are combined using learned weights, which improves the convergence behavior of the BP decoder. Moreover, [21], [22] simplify [19] by using normalized min-sum (MS) decoding for the check nodes with a learned parameter w :

$$m_t^{c \rightarrow v} = w \times \prod_{v' \in \mathcal{N}(c) \setminus v} \text{sign}(m_t^{v \rightarrow c'}) \times \min_{v' \in \mathcal{N}(c) \setminus v} |m_t^{v \rightarrow c'}|. \quad (28)$$

The method in [23] uses an unfolded normalized MS algorithm for the decoding of polar codes. The main difference with [21], [22], apart from the slightly different message scheduling required to decode polar codes, is that the normalization parameter w is allowed to differ for every message and for every iteration. Simulation results for polar codes of various block-lengths and rate $R = 1/2$ show that the unfolded MS decoder with per-message learned normalization parameters outperforms the standard normalized MS decoder by approximately 0.5 dB. The authors also provide a high-level discussion of hardware implementation considerations.

The authors of [24] propose a hybrid BP-NN decoder for polar codes, where a fraction of the messages is calculated using standard BP message-passing rules, while the remaining messages are calculated using trained NNs. This approach enables the scaling of NN-assisted decoders to large block-lengths, while simulation results show very competitive performance with respect to conventional decoders for polar codes.

The method in [25] unfolds the MS algorithm to decode polar codes. The authors first use a method to convert the message-passing graph of polar codes into a conventional sparse Tanner graph so that the standard BP message-passing rules of (22) and (23) can be used verbatim, thus avoiding the different message schedule used in [23], [24]. Moreover, a single weight w' is used for all variable-to-check messages at all iterations so that (26) is simplified to:

$$m_t^{v \rightarrow c} = l^v + \sum_{c' \in \mathcal{N}(v) \setminus c} w' m_{t-1}^{c' \rightarrow v}. \quad (29)$$

The non-normalized MS update rule is used for the check-to-variable messages, i.e., (28) with $w = 1$. Simulation results show that the use of a single weight w' has a negligible effect on the error rate of the decoder, while significantly reducing the complexity of both learning and decoding.

The papers [26], [27] propose to unfold the normalized-offset MS algorithm to decode LDPC codes and polar codes, respectively. The minimum-finding part in (28) is replaced by:

$$\alpha_t^{v \rightarrow c} \cdot \min_{v' \in \mathcal{N}(c) \setminus v} \max(|m_t^{v \rightarrow c'}| - \beta_t^{v \rightarrow c}, 0), \quad (30)$$

where $\alpha_t^{v \rightarrow c}$ and $\beta_t^{v \rightarrow c}$ are per-message and per-iteration trainable parameters. Simulation results show that these additional parameters can improve the performance of unfolded MS decoding with respect to standard MS and BP decoding as well as previous works on unfolded MS decoding.

In [28], a joint CRC-polar MS decoding algorithm is proposed, which exploits the concatenated factor graph of a polar code and a CRC. Similarly to previously described works, trainable weights are assigned to the edges of the unfolded factor graph. Moreover, a multi-loss cost function is used to improve the training process. In general, multi-loss functions are a sum of multiple loss functions from different parts of the NN to be trained. In this case, the multi-loss function takes the outputs of both the MS part and the CRC part of the factor graph into account. Simulation results show improved performance with respect to [21], [23].

The method in [29] uses an unfolded structure that resembles that of [21], with the main difference that a single weight is used for all messages and all iterations. The authors also argue that the binary cross-entropy function that is commonly minimized to train unfolded decoders does not necessarily minimize the bit error rate (BER). Instead, they propose a new cost function that is based on the so-called *soft bit error* concept. For a single bit, if the actual bit-value is $a \in \{0, 1\}$ and the estimated (soft) bit-value at the output of the unfolded decoder is $b \in [0, 1]$, the soft bit error $L_{\text{sbe}}(a, b)$ is given by:

$$L_{\text{sbe}}(a, b) = (1 - b)^a b^{1-a}, \quad (31)$$

whereas the standard binary cross-entropy $L_{\text{bce}}(a, b)$ would be:

$$L_{\text{bce}}(a, b) = -\log(b^a (1 - b)^{1-a}). \quad (32)$$

Instead of training for a single SNR or a set of SNR points, the authors of [29] use an auxiliary NN that learns parameter values given the SNR as an input.

Finally, the work of [30] proposes the idea of unfolding in order to learn finite-alphabet (FA) decoding of LDPC codes. In FA decoding, messages are quantized using a very small number of quantization bits and it is thus crucial that the quantization thresholds and levels are designed very carefully. The authors show that by unfolding and learning FA decoders, gains of up to 0.25 dB can be achieved for a (1296, 972) QC-LDPC code when using 3 quantization bits.

IV. DEEP UNFOLDING FOR OTHER COMMUNICATIONS APPLICATIONS

There exist a plethora of other communications applications in which the idea of unfolding has been used—we now briefly summarize some of these applications. For channel decoding that is not based on BP decoding, references [31]–[33] study unfolding of Turbo decoding, whereas [34] discusses successive cancellation decoding of polar codes. The work in [35] proposes to replace the channel-dependent parts of the Viterbi

detection algorithm by a DNN. NNs have also been used extensively for non-linear signal processing tasks. In this case, unfolding does not refer to the iterations of some algorithm, but rather to the non-linear equations themselves (e.g., parallel Hammerstein model [36], [37] or Schrödinger wave equation). This approach has been recently applied to optical communications (e.g., [38], [39]) and to full-duplex communications (e.g., [40]). Finally, unfolding has been extensively applied to the iterative shrinkage-thresholding algorithm (ISTA) to solve sparse linear inverse problems [41]–[46], which is a general tool that finds use in communications systems (e.g., for sparse channel estimation).

V. FUTURE RESEARCH DIRECTIONS

Even though the idea of deep unfolding is relatively novel and many open research questions remain, it has already found wide applicability in communication systems and is likely to transform a range of other signal processing tasks. For example, proximal algorithms [47], [48] solve a wide range of optimization problems in communication systems and they are generally well-suited for unfolding. We conclude this brief survey by outlining some interesting future research directions.

A. Unfolded Structures with Acceleration Methods

Several techniques that have been used in the optimization literature to accelerate the convergence of optimization algorithms can be incorporated into unfolded architectures with trainable parameters. Some examples of these techniques include preconditioning, momentum and Onsager terms, restart, and adaptive step-size rules. Such methods are particularly interesting for severely iteration-constrained applications where obtaining the fastest possible convergence is of the utmost importance. It may also be beneficial, from both a complexity and performance perspective, to derive and optimize unfolded structures that directly operate on complex-valued signals.

B. Loss Functions

Novel application-tailored cost functions can improve the convergence of the training process. This can not only lead to better results for the same computational effort, but it may also enable real-time and online training of unfolded structures. Some examples of customized loss functions already exist (e.g., [29] uses a soft bit error function, [49] proposes a syndrome-based cost function, and [50] uses a cost function that is tailored to the quantum error-correction scenario), but they are mostly limited to channel decoding. All works on MIMO detection/precoding that we have described [9]–[15] use the standard mean-squared error (MSE) cost function. The MSE cost function has the advantage that closed-form solutions or very accurate iterative approximations can be derived in many cases. However, the MSE is not necessarily a good proxy for the error rate performance of the system. For example, in a multi-user MIMO setting, the system error rate will most likely be dominated by the user with the largest MSE. When optimizing an unfolded algorithm, even when the algorithm itself has been derived assuming an MSE cost function, it is typically easy to learn the optimal set

of parameters for a different cost function. In the multi-user MIMO example mentioned previously, this could be the maximum MSE over all users, or even the maximum MSE over all users and all channel realizations in the training dataset.

C. Training

Even though in many applications training can be carried out offline, it is still a task that requires considerable effort and thus deserves attention. In applications where it is sufficient to unroll a small number of iterations, training is mostly straightforward. However, when more iterations are considered, numerous problems arise. A common problem is that of vanishing gradients, where it becomes increasingly difficult to find suitable parameters in early iterations. This problem can be addressed by using multi-loss functions (like most of the works presented in this survey), incremental training [13], or by simply using a set of known good initial values to minimize training [15]. Another solution would be to perform windowed training, where unfolded iterations are trained only over a moving window of fixed size. This approach can also significantly reduce the memory required for training, which may become a limiting factor when considering algorithms with high-dimensional inputs (e.g., in massive MIMO) and a large number of iterations, since all intermediate output values of each mini-batch need to be stored for back-propagation. Online training methods that adapt to, e.g., changing channel or SNR conditions, is another important problem. Finally, it is often unclear what the best dataset for training is. We note that some preliminary works already focused specifically on this direction [51], [52], but more research is required.

D. Hardware Implementation

Unfolded learned algorithms are particularly attractive from a hardware implementation perspective, as they strongly resemble known algorithms for which efficient hardware architectures already exist. However, the hardware implementation complexity aspect is typically not considered in the literature, with some notable exceptions being [23], where high-level hardware considerations for unfolded MS decoding are discussed, and [53], where FPGA and ASIC implementation results of the method in [40] are presented. As such, it remains largely unclear how the additional trainable parameters required by unfolded algorithms affect the hardware implementation complexity and the achieved throughput. Moreover, efficient hardware implementations of the training step are necessary for situations that require online learning.

REFERENCES

- [1] D. P. Bertsekas, *Convex Optimization Algorithms*. Athena Scientific, Belmont, Massachusetts, 2015.
- [2] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Transactions on Cognitive Communications and Networking*, vol. 3, no. 4, pp. 563–575, Dec. 2017.
- [3] 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, Nov. 2017.
- [4] Q. Mao, F. Hu, and Q. Hao, "Deep learning for intelligent wireless networks: A comprehensive survey," *IEEE Communications Surveys Tutorials*, vol. 20, no. 4, pp. 2595–2621, Fourth Quarter 2018.

- [5] D. Gunduz, P. de Kerret, N. D. Sidiropoulos, D. Gesbert, C. Murthy, and M. van der Schaar, "Machine learning in the air," Apr. 2019. [Online]. Available: <https://arxiv.org/abs/1904.12385>
- [6] Z. Qin, H. Ye, G. Y. Li, and B.-H. F. Juang, "Deep learning in physical layer communications," Feb. 2019. [Online]. Available: <https://arxiv.org/abs/1807.11713>
- [7] H. He, S. Jin, C.-K. Wen, F. Gao, G. Y. Li, and Z. Xu, "Model-driven deep learning for physical layer communications," Feb. 2019. [Online]. Available: <https://arxiv.org/abs/1809.06059>
- [8] J. R. Hershey, J. Le Roux, and F. Weninger, "Deep unfolding: Model-based inspiration of novel deep architectures," Nov. 2014. [Online]. Available: <https://arxiv.org/abs/1409.2574>
- [9] N. Samuel, T. Diskin, and A. Wiesel, "Deep MIMO detection," in *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Jul. 2017, pp. 1–5.
- [10] N. Samuel, T. Diskin, and A. Wiesel, "Learning to detect," *IEEE Transactions on Signal Processing*, vol. 67, no. 10, pp. 2554–2564, May 2019.
- [11] V. Corlay, J. J. Boutros, P. Ciblat, and L. Brunel, "Multilevel MIMO detection with deep learning," in *Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018, pp. 1805–1809.
- [12] H. He, C.-K. Wen, S. Jin, and G. Y. Li, "A model-driven deep learning network for MIMO detection," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Nov. 2018, pp. 584–588.
- [13] S. Takabe, M. Imanishi, T. Wadayama, and K. Hayashi, "Trainable projected gradient detector for massive overloaded MIMO channels: Data-driven tuning approach," Dec. 2018. [Online]. Available: <https://arxiv.org/abs/1812.10044>
- [14] S. Takabe, M. Imanishi, T. Wadayama, and K. Hayashi, "Deep learning-aided projected gradient detector for massive overloaded MIMO channels," Dec. 2018. [Online]. Available: <https://arxiv.org/abs/1806.10827>
- [15] A. Balatsoukas-Stimming, O. Castañeda, S. Jacobsson, G. Durisi, and C. Studer, "Neural-network optimized 1-bit precoding for massive MU-MIMO," Mar. 2019. [Online]. Available: <https://arxiv.org/abs/1903.03718>
- [16] Y. Wei, M.-M. Zhao, M. Hong, M. Jian Zhao, and M. Lei, "Learned conjugate gradient descent network for massive MIMO detection," Jun. 2019. [Online]. Available: <https://arxiv.org/abs/1906.03814>
- [17] M. Khani, M. Alizadeh, J. Hoydis, and P. Fleming, "Adaptive neural signal detection for massive MIMO," Jun. 2019. [Online]. Available: <https://arxiv.org/abs/1906.04610>
- [18] C. Jeon, A. Maleki, and C. Studer, "On the performance of mismatched data detection in large MIMO systems," in *IEEE International Symposium on Information Theory (ISIT)*, July 2016, pp. 180–184.
- [19] E. Nachmani, Y. Be'ery, and D. Burshtein, "Learning to decode linear codes using deep learning," in *Annual Allerton Conference*, Sep. 2016, pp. 341–346.
- [20] E. Nachmani, E. Marciano, D. Burshtein, and Y. Be'ery, "RNN decoding of linear block codes," Feb. 2017. [Online]. Available: <https://arxiv.org/abs/1702.07560>
- [21] 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*, vol. 12, no. 1, pp. 119–131, Feb. 2018.
- [22] L. Lugosch and W. J. Gross, "Neural offset min-sum decoding," in *IEEE International Symposium on Information Theory (ISIT)*, Jun. 2017, pp. 1361–1365.
- [23] W. Xu, Z. Wu, Y. Ueng, X. You, and C. Zhang, "Improved polar decoder based on deep learning," in *IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct. 2017, pp. 1–6.
- [24] S. Cammerer, T. Gruber, J. Hoydis, and S. ten Brink, "Scaling deep learning-based decoding of polar codes via partitioning," in *IEEE Global Communications Conference (GLOBECOM)*, Dec. 2017, pp. 1–6.
- [25] W. Xu, X. You, C. Zhang, and Y. Be'ery, "Polar decoding on sparse graphs with deep learning," in *Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018, pp. 599–603.
- [26] X. Wu, M. Jiang, and C. Zhao, "Decoding optimization for 5G LDPC codes by machine learning," *IEEE Access*, vol. 6, pp. 50 179–50 186, Sep. 2018.
- [27] B. Dai, R. Liu, and Z. Yan, "New min-sum decoders based on deep learning for polar codes," in *IEEE International Workshop on Signal Processing Systems (SiPS)*, Oct. 2018, pp. 252–257.
- [28] N. Doan, S. A. Hashemi, E. N. Mambou, T. Tonnellier, and W. J. Gross, "Neural belief propagation decoding of CRC-polar concatenated codes," Oct. 2018. [Online]. Available: <https://arxiv.org/abs/1811.00124>
- [29] M. Lian, F. Carpi, C. Häger, and H. D. Pfister, "Learned belief-propagation decoding with simple scaling and SNR adaptation," Jan. 2019. [Online]. Available: <https://arxiv.org/abs/1901.08621>
- [30] B. Vasić, X. Xiao, and S. Lin, "Learning to decode LDPC codes with finite-alphabet message passing," in *Information Theory and Applications Workshop (ITA)*, Feb. 2018, pp. 1–9.
- [31] H. Kim, Y. Jiang, R. Rana, S. Kannan, S. Oh, and P. Viswanath, "Communication algorithms via deep learning," May 2018. [Online]. Available: <https://arxiv.org/abs/1805.09317>
- [32] Y. Jiang, H. Kim, H. Asnani, S. Kannan, S. Oh, and P. Viswanath, "DeepTurbo: Deep turbo decoder," Apr. 2019. [Online]. Available: <https://arxiv.org/abs/1903.02295>
- [33] Y. He, J. Zhang, C.-K. Wen, and S. Jin, "TurboNet: A model-driven DNN decoder based on max-log-MAP algorithm for turbo code," May 2019. [Online]. Available: <https://arxiv.org/abs/1905.10502>
- [34] N. Doan, S. Ali Hashemi, and W. J. Gross, "Neural successive cancellation decoding of polar codes," in *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Jun. 2018, pp. 1–5.
- [35] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: A deep learning based Viterbi algorithm for symbol detection," May 2019. [Online]. Available: <https://arxiv.org/abs/1905.10750>
- [36] A. Janczak, "Neural network approach for identification of Hammerstein systems," *International Journal of Control*, vol. 76, no. 17, pp. 1749–1766, 2003.
- [37] H. Yu, J. Peng, and Y. Tang, "Identification of nonlinear dynamic systems using Hammerstein-type neural network," *Mathematical Problems in Engineering*, Oct. 2014.
- [38] C. Häger and H. D. Pfister, "Nonlinear interference mitigation via deep neural networks," in *Optical Fiber Communications Conference and Exposition (OFC)*, Mar. 2018, pp. 1–3.
- [39] C. Häger and H. D. Pfister, "Deep learning of the nonlinear Schrödinger equation in fiber-optic communications," in *IEEE International Symposium on Information Theory (ISIT)*, Jun. 2018, pp. 1590–1594.
- [40] A. Balatsoukas-Stimming, "Non-linear digital self-interference cancellation for in-band full-duplex radios using neural networks," in *IEEE International Workshop on Signal Processing Advances in Wireless Communications (SPAWC)*, Jun. 2018, pp. 1–5.
- [41] K. Gregor and Y. LeCun, "Learning fast approximations of sparse coding," in *International Conference on International Conference on Machine Learning*, 2010, pp. 399–406.
- [42] M. Borgerding and P. Schniter, "Onsager-corrected deep learning for sparse linear inverse problems," in *IEEE Global Conference on Signal and Information Processing (GlobalSIP)*, Dec. 2016, pp. 227–231.
- [43] U. S. Kamilov and H. Mansour, "Learning optimal nonlinearities for iterative thresholding algorithms," *IEEE Signal Processing Letters*, vol. 23, no. 5, pp. 747–751, May 2016.
- [44] M. Borgerding, P. Schniter, and S. Rangan, "AMP-inspired deep networks for sparse linear inverse problems," *IEEE Transactions on Signal Processing*, vol. 65, no. 16, pp. 4293–4308, Aug. 2017.
- [45] D. Ito, S. Takabe, and T. Wadayama, "Trainable ISTA for sparse signal recovery," *IEEE Transactions on Signal Processing*, vol. 67, no. 12, pp. 3113–3125, Jun. 2019.
- [46] S. Takabe and T. Wadayama, "Complex field-trainable ISTA for linear and nonlinear inverse problems," Apr. 2019. [Online]. Available: <https://arxiv.org/abs/1904.07409>
- [47] N. Parikh and S. Boyd, *Proximal Algorithms*. Now Foundations and Trends, 2014.
- [48] T. Goldstein, C. Studer, and R. Baraniuk, "A field guide to forward-backward splitting with a FASTA implementation," *Technical Report*, Nov. 2014. [Online]. Available: <https://arxiv.org/abs/1411.3406>
- [49] L. Lugosch and W. J. Gross, "Learning from the syndrome," in *Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018, pp. 594–598.
- [50] Y.-H. Liu and D. Poulin, "Neural belief-propagation decoders for quantum error-correcting codes," *Physical Review Letters*, vol. 122, p. 200501, May 2019.
- [51] M. Benammar and P. Piantanida, "Optimal training channel statistics for neural-based decoders," in *Asilomar Conference on Signals, Systems, and Computers*, Oct. 2018, pp. 2157–2161.
- [52] I. Be'ery, N. Raviv, T. Raviv, and Y. Be'ery, "Active deep decoding of linear codes," Jun. 2019. [Online]. Available: <https://arxiv.org/abs/1906.02778>
- [53] Y. Kurzo, A. Burg, and A. Balatsoukas-Stimming, "Design and implementation of a neural network aided self-interference cancellation scheme for full-duplex radios," in *Asilomar Conference on Signals, Systems, and Computers*, Nov. 2018.