Learned Factor Graphs for Inference From Stationary Time Sequences

Nir Shlezinger [ID], *Member, IEEE*, Nariman Farsad [ID], *Member, IEEE*, Yonina C. Eldar [ID], *Fellow, IEEE*, and Andrea J. Goldsmith, *Fellow, IEEE*

Abstract—The design of methods for inference from time sequences has traditionally relied on statistical models that describe the relation between a latent desired sequence and the observed one. A broad family of model-based algorithms have been derived to carry out inference at controllable complexity using recursive computations over the factor graph representing the underlying distribution. An alternative model-agnostic approach utilizes machine learning (ML) methods. Here we propose a framework that combines model-based algorithms and data-driven ML tools for stationary time sequences. In the proposed approach, neural networks are developed to separately learn specific components of a factor graph describing the distribution of the time sequence, rather than the complete inference task. By exploiting stationary properties of this distribution, the resulting approach can be applied to sequences of varying temporal duration. Learned factor graphs can be realized using compact neural networks that are trainable using small training sets, or alternatively, be used to improve upon existing deep inference systems. We present an inference algorithm based on learned stationary factor graphs, which learns to implement the sum-product scheme from labeled data, and can be applied to sequences of different lengths. Our experimental results demonstrate the ability of the proposed learned factor graphs to learn from small training sets to carry out accurate inference for sleep stage detection using the Sleep-EDF dataset, as well as for symbol detection in digital communications with unknown channels.

Index Terms—Factor graphs, deep learning, time sequences.

I. INTRODUCTION

A MULTITUDE of practical problems involve inference from time sequences. The need to accurately estimate a

hidden time series from a measured signal is frequently encountered in signal processing, communications, control, finance, and various other fields. Traditional algorithms, such as those based on the maximum a-posteriori probability (MAP) rule, are model-based, namely, they carry out inference based on complete knowledge of the underlying statistical model relating the desired time series and the observed one. The joint distribution of a large family of time sequences encountered in practice can be factorized, which facilitates inference at reduced complexity by representing their distribution as a Forney-style factor graph [1], [2], referred to henceforth as a *factor graph* for brevity. In factor graphs, variables are represented as edges connected to function nodes forming a graphical representation of joint distribution measures. When this graph is cycle-free, it can be used to evaluate marginal distributions in an efficient manner, i.e., with complexity that only grows linearly with the number of variables. As a result, many important model-based algorithms, such as the Viterbi algorithm [3], the sum-product (SP) method [4] also known as belief propagation (BP) [5], the BCJR detector [6], the Baum-Welch scheme for estimating the parameters of hidden Markov models (HMMs) [7], and the Kalman filter [8, Ch. 7], all process time sequences via recursive computations over a factor graph [9].

Often in practice, the underlying statistical model relating the observations and the desired time series is highly complex or poorly understood. In such cases, model-based algorithms, which are typically sensitive to inaccurate knowledge of the underlying statistics, cannot be reliably applied, and model-agnostic data-driven schemes are preferable. Consequently, recent years have witnessed extensive interest in the application of machine learning (ML), and particularly of deep neural networks (DNNs), for time sequence inference, with various architectures proposed to exploit the presence of temporal correlation [10]–[12]. However, training these deep architectures typically requires a massive amount of labeled data, which may not always be available. Furthermore, applying inference using highly parameterized DNNs may not be feasible on devices with limited hardware capabilities.

The individual challenges of model-based signal processing and model-agnostic ML has given rise to various hybrid systems combining ML and model-based algorithms [13]. Such hybrid model-based/data-driven attempt to benefit from the best of both worlds. These include the usage of DNNs to learn a possibly analytically intractable regularization in compressed sensing applications [14], [15], as well as the use of deep denoisers in

regularized optimization via plug-and-play networks [16], [17]. A systematic strategy to combine model-based algorithms and ML is based on deep unfolding or unrolling [18]–[21]. Deep unfolding sets the layers of a DNN in light of the iterations of some iterative optimization algorithm, while using the resulting unfolded network for the complete inference task. Deep unfolding typically requires knowledge of the underlying model, up to perhaps some unknown parameters, in order to unfold the optimization method. Unfolding commonly results in a highly parameterized DNN utilized for the complete inference task, whose architecture is inspired by a model-based algorithm.

In this work, we propose an alternative strategy that combines model-based signal processing algorithms based on factor graph computations with data-driven ML tools. Here, instead of using DNNs for inference, they are utilized to *learn only the function nodes of the factor graph*, which in turn is used for inference via conventional factor graph methods, such as the SP algorithm. This approach builds upon the fact that the statistical behavior of time sequences can often be approximated using stationary factorizable distributions, which allows incorporating domain knowledge in the structure of the graph while learning its nodes from data. This results in a hybrid model-based/data-driven inference scheme that only requires prior knowledge of the factorization of the underlying distribution. This is in contrast to deep unfolding where the distribution in parametric form is typically required. Moreover, the hybrid scheme can also incorporate additional domain knowledge in its learned nodes. As opposed to previous works that used highly-parameterized deep architectures to represent messages along a factor graph, and trained the overall system in an end-to-end manner [22]–[29], our strategy learns the function nodes separately from the task, exploiting stationarity by reusing a DNN for multiple function nodes. Consequently, this approach can use relatively compact networks that are trained with small training sets and employed on hardware-limited devices, as well as to improve upon existing DNN architectures by utilizing them for learning the factor graph instead of for inference. Furthermore, the same learned factor graph may be used for sequences of varying length, as well as combined with multiple inference algorithms.

In particular, we present a data-driven inference scheme based on learned factor graphs that learns to implement the SP method over factor graphs of stationary Markovian time sequences from labeled data. While the SP scheme requires accurate knowledge of the underlying statistical model, its data-driven implementation allows this algorithm to be utilized in scenarios involving time sequences with complex and possibly analytically intractable distributions. We detail how such hybrid model-based/data-driven inference is derived from the SP algorithm by learning the underlying factor graph. We also show that, by assuming stationarity, the complete factor graph can be learned using a single relatively compact neural network. We discuss how ViterbiNet, proposed in [30] for real-time data-driven symbol detection in digital communications, can be obtained as a special case of our framework, and in fact be implemented using the same learned factor graph as that used for SP-based inference. We then discuss how the ability to learn factor graphs of stationary distributions from small training sets may be exploited to facilitate adaptation to blockwise statistical variations when some future indication on the inference correctness is available, as in, e.g., coded communications setups.

We evaluate the usage of learned factor graphs for sleep pattern prediction as well as in a digital communications setup with unknown channel settings. For sleep pattern prediction, we use the Sleep-EDF dataset [31], and show how using a neural predictor as a learned node in the SP method improves the accuracy by 4% compared with using it for inference. We also demonstrate how this facilitates the usage of compact networks trainable with small data sets, enabling training using only part of the data of a single patient, while achieving accurate inference on its remaining data. For the communications setup, we demonstrate that inference over learned factor graphs is capable of approaching the performance of the MAP detector, which requires full knowledge of the underlying statistical model, while achieving improved robustness to model uncertainty compared with the conventional SP algorithm. Furthermore, we demonstrate that by utilizing compact networks that are trainable with small training sets, learned factor graphs can be tuned to accurately track temporal variations in the statistical model via online training.

The rest of this paper is organized as follows: In Section II, we detail the problem of inference over stationary time sequences, and briefly review model-based factor graph methods. Section III details the proposed framework for inference over learned factor graphs by deriving it from the SP algorithm applied to Markovian signals. Experimental results are presented in Section IV. Finally, Section V provides concluding remarks.

Throughout the paper, we use upper-case letters for random variables (RVs), e.g. X. Boldface lower-case letters denote vectors, e.g., $\boldsymbol{x}$ is a deterministic vector, and $\boldsymbol{X}$ is a random vector; the ith element of $\boldsymbol{x}$ is written as x_i. The probability measure of an RV X evaluated at x is denoted $P_X(x)$, while $\mathcal{N}(\cdot, \cdot)$ represents the Gaussian distribution. We use caligraphic letters for sets, e.g., $\mathcal{X}$, where $|\mathcal{X}|$ is the cardinality of a finite set $\mathcal{X}$, while $\mathbb{R}$ denotes the set of real numbers. Finally, for a sequence $\{x_i\}$ and integers $i_1 < i_2$ we use $\boldsymbol{x}_{i_1}^{i_2}$ to denote the stacking $[x_{i_1}, x_{i_1+1}, \ldots, x_{i_2}]^T$ while $\boldsymbol{x}^{i_2} \triangleq \boldsymbol{x}_1^{i_2}$.

II. SYSTEM MODEL

In this section, we present the system model for which we propose the concept of data-driven factor graphs in Section III. We begin by formulating the considered time series inference problem in Section II-A. We then discuss the model-based approach for the problem at hand in Section II-B, after which we briefly review conventional model-based factor graph methods in Section II-C.

A. Problem Formulation

We consider the problem of recovering a desired time series $\{S_i\}$ taking values in a set $\mathcal{S}$ from an observed sequence $\{Y_i\}$ taking values in a set $\mathcal{Y}$. The subscript i denotes the time index. The joint distribution of $\{S_i\}$ and $\{Y_i\}$ obeys an lth-order Markovian stationary model,

$$P_{Y_i, S_i | \{Y_j, S_j\}_{j<i}} \left(y_i, s_i | \{y_j, s_j\}_{j<i}\right)$$
$$= P_{Y_i | \boldsymbol{S}_{i-l}^i} \left(y_i | \boldsymbol{s}_{i-l}^i\right) P_{S_i | \boldsymbol{S}_{i-l}^{i-1}} \left(s_i | \boldsymbol{s}_{i-l}^{i-1}\right), \qquad (1)$$

for some integer $l \geq 1$, representing the memory of the sequences. Consequently, when the initial state $\boldsymbol{S}_{-l}^0$ is given, the joint distribution of $\boldsymbol{Y}^t$ and $\boldsymbol{S}^t$ satisfies

$$P_{\boldsymbol{Y}^t, \boldsymbol{S}^t}(\boldsymbol{y}, \boldsymbol{s}) = \prod_{i=1}^{t} P_{Y_i|\boldsymbol{S}_{i-l}^i}(y_i|\boldsymbol{s}_{i-l}^i) P_{S_i|\boldsymbol{S}_{i-l}^{i-1}}(s_i|\boldsymbol{s}_{i-l}^{i-1}), \quad (2)$$

for any fixed sequence length $t > 0$. The joint distribution in (2) is a special case of an lth order Markov model, where given $\boldsymbol{S}_{i-l}^i$, the observed sequence Y_i does not depend on the past observations $\boldsymbol{Y}_{i-l}^{i-1}$. In principle, the lth order Markov model in (2) can be expressed as an order-one Markov model by replacing S_i with the multivariate $\{\boldsymbol{S}_{i-l+1}^i\}_{i=1}^t$. However, since we focus on the recovery of the sample S_i from $\boldsymbol{Y}^t$ (rather than the stacking of l samples $\boldsymbol{S}_{i-l+1}^i$), we keep the above formulation without limiting our attention to the case where $l = 1$.

In general, the above statistical relationship can change over time. Here, we assume that the marginal distribution of the desired sequence $\{S_i\}$ remains static over time, i.e., $P_{S_i|\boldsymbol{S}_{i-l}^{i-1}}(\cdot)$ does not depend on i. We allow the conditional probability density function (PDF) in (2) to evolve over time in the following manners:

- *Stationary sequence* - the conditional PDF in (2) remains invariant over time, i.e., $P_{Y_i|S_i}(\cdot)$ does not depend on the time index i.
- *Blockwise stationary* - the conditional PDF $P_{Y_i|S_i}(\cdot)$ changes every t_b time instances.

The stationarity assumption implies that within a given block of t_b time instances, (2) represents the joint distribution as the product of the same function with different arguments.

The statistical relationship in (2) accurately represents a broad range of problems encountered in practice, including inference from bio-medical signals [32], [33] as well as symbol detection in digital communications [34, Ch. 3]. The common aspect of these problems is the presence of temporal correlation, implying that information regarding a state variable S_i is contained not only in its corresponding observation Y_i, but also in its preceding and subsequent measurements.

Our goal is to design a system that learns to reliably infer a block of desired variables $\boldsymbol{S}^t$ from its corresponding observations $\boldsymbol{Y}^t$ for arbitrary blocklength t. The system learns its inference mapping using a data set comprised of a sequence of n_t labeled samples denoted $\{s_k, y_k\}_{k=1}^{n_t}$, as well as prior knowledge that the joint distribution obeys the factorization in (2). Nonetheless, it is emphasized that distribution functions in (2), e.g., $P_{Y_i|\boldsymbol{S}_{i-l}^i}$, are unknown and may not be analytically tractable. The presence of data as well as partial domain knowledge motivates the design of hybrid model-based/data-driven schemes.

B. Model-Based Inference

When the joint distribution of $\boldsymbol{S}^t$ and $\boldsymbol{Y}^t$ is a-priori known and can be computed, the inference rule that minimizes the symbol error probability for each time instance is the MAP detector,

$$\hat{s}_i\left(\boldsymbol{y}^t\right) = \arg\max_{s \in \mathcal{S}} P_{S_i|\boldsymbol{Y}^t}(s|\boldsymbol{y}^t), \quad (3)$$

for each $i \in \{1, \ldots, t\} \triangleq \mathcal{T}$. This rule can be efficiently evaluated for finite-memory distributions using the SP algorithm [4]. An alternative common inference rule is the maximum likelihood sequence detector, given by

$$\hat{\boldsymbol{s}}^t\left(\boldsymbol{y}^t\right) \triangleq \arg\max_{\boldsymbol{s}^t \in \mathcal{S}^t} P_{\boldsymbol{Y}^t|\boldsymbol{S}^t}\left(\boldsymbol{y}^t|\boldsymbol{s}^t\right). \quad (4)$$

Unlike (3), the maximum likelihood sequence detector (4) has no optimality guarantee in general. However, similarly to (3), the inference rule in (4) is amenable to efficient iterative computation. In particular, for sequences obeying the structure (2), the detector (4) can be computed efficiently using the Viterbi algorithm [4], [35]. Viterbi detection allows real-time inference, i.e., it operates in a sequential manner and uses the partial vector $\boldsymbol{y}^{i+l-1}$, instead of the complete observations $\boldsymbol{y}^t$, when recovering S_i.

Both the SP method and the Viterbi scheme are model-based algorithms that employ recursive computations over the underlying factor graph encapsulating the joint distribution of $\boldsymbol{S}^t$ and $\boldsymbol{Y}^t$. Consequently, to design a system capable of learning from data how to carry out such inference, we first review factor graph methods in the following subsection.

C. Factor Graphs Methods

In the following we provide a brief introduction to factor graphs. We then review the SP algorithm for computing marginal distributions as a representative method for efficient inference over factor graphs [5].

A Forney-style factor graph, referred to henceforth as factor graph, is a graphical representation of the factorization of a function of several variables [9], commonly a joint distribution measure. Its main advantage over alternative graphical models of joint distributions, such as Bayesian (belief) networks [36] and junction graphs [37], stems from its suitability to hierarchical models and its resulting simple formulation of the SP message passing algorithm [2]. To present the concept of factor graphs and their usage, consider a $t \times 1$ random vector $\boldsymbol{X} \in \mathcal{X}^t$ where $\mathcal{X}$ is a finite set, i.e., the entries of $\boldsymbol{X}$, denoted $\{X_i\}$, are discrete RVs. The joint distribution of $\boldsymbol{X}$, $P_{\boldsymbol{X}}(\boldsymbol{x})$, is factorizable if it can be represented as the product of m functions $\{f_k(\cdot)\}_{k=1}^m$, i.e., there exist some partition variables $\{\mathcal{V}_k\}_{k=1}^m$, $\mathcal{V}_k \subset \{x_1, \ldots, x_t\}$, which are not subsets of one another, such that

$$P_{\boldsymbol{X}}(\boldsymbol{x}) = \prod_{k=1}^{m} f_k(\mathcal{V}_k). \quad (5)$$

In order to represent (5) as a factor graph, the functions $\{f_k(\cdot)\}_{k=1}^m$ should be set such that each variable x_i appears in no more than two partitions[1] $\{\mathcal{V}_k\}_{k=1}^m$. Subject to this assumption, the distribution $P_{\boldsymbol{X}}(\boldsymbol{x})$ can be described as a factor graph with m nodes, which are the functions $\{f_k(\cdot)\}_{k=1}^m$, while the variables $\{x_i\}_{i=1}^t$ represent edges or half-edges. In the sequel we focus on partitions in which the resulting graphical representation is cycle-free.

---

[1] A factorization in which a variable appears in more than two factors can always be modified to meet the above constraint by introducing additional variables and identity factors, see [2].

A major motivation for representing joint distributions via factor graphs is that they allow some statistical computations to be carried out with reduced complexity. One of the most common methods to exploit factorization via factor graphs for reduced complexity inference is the SP algorithm, that evaluates a marginal distribution from a factor graph representation of a joint probability measure [4]. To formulate the SP method, consider for simplicity a factorized distribution in which the ordering of the partitions $\{\mathcal{V}_i\}$ corresponds to the order of the variables $\{x_i\}$, e.g., $\mathcal{V}_1 = \{x_1, x_2\}$, $\mathcal{V}_2 = \{x_2, x_3, x_4\}$, $\mathcal{V}_3 = \{x_4, x_5\}$, etc. Furthermore, as noted above, assume that the factor graph does not contain cycles.[2] In this case, the marginal distribution of an RV X_i whose corresponding variable appears in the partitions $\mathcal{V}_j$ and $\mathcal{V}_{j+1}$ can be computed from the joint distribution $P_{\boldsymbol{X}}$ via

$$P_{X_i}(x_i) = \sum_{\{\boldsymbol{x}/x_i\}} P_{\boldsymbol{X}}(\boldsymbol{x})$$

$$= \underbrace{\left(\sum_{\{x_1,\dots,x_{i-1}\}} \prod_{k=1}^{j} f_k(\mathcal{V}_k) \right)}_{\triangleq \overrightarrow{\mu}_{X_i}(x_i)} \underbrace{\left(\sum_{\{x_{i+1},\dots,x_t\}} \prod_{k=j+1}^{m} f_k(\mathcal{V}_k) \right)}_{\triangleq \overleftarrow{\mu}_{X_i}(x_i)}.$$

$$(6)$$

The factorization of the joint distribution implies that the marginal distribution, whose computation typically requires summation over $|\mathcal{X}|^{t-1}$ variables, can now be evaluated as the product of two terms, $\overrightarrow{\mu}_{X_i}(x_i)$ and $\overleftarrow{\mu}_{X_i}(x_i)$. These terms may be viewed as messages propagating forward and backward along the factor graph, e.g., $\overrightarrow{\mu}_{X_i}(x_i)$ represents a forward message conveyed to edge x_i. In particular, these messages can be computed recursively. Writing $\mathcal{V}_j = \{x_{i-\tau}, \dots, x_i\}$ for some $\tau \geq 1$, the SP rule [2] implies that

$$\overrightarrow{\mu}_{X_i}(x_i) = \sum_{\{x_{i-\tau},\dots,x_{i-1}\}} f_j(\mathcal{V}_j) \sum_{\{x_1,\dots,x_{i-\tau-1}\}} \prod_{k=1}^{j-1} f_i(\mathcal{V}_i)$$

$$= \sum_{\{x_{i-\tau},\dots,x_{i-1}\}} f_j(\mathcal{V}_j) \overrightarrow{\mu}_{X_{i-\tau}}(x_{i-\tau}), \quad (7)$$

where the last equality follows from the fact that $x_{i-\tau}$ here is the variable with the largest index in $\mathcal{V}_{j-1}$. The computation of message terms in a recursive manner over a graphical model, as done in (7), is referred to as *message passing*. In particular, the method of computing marginals in (6) using message passing over factor graphs is referred to as the SP algorithm [4].

III. INFERENCE VIA LEARNED FACTOR GRAPHS

In this section we present our proposed system for ML-based inference applied to stationary time sequences. We begin by reviewing the application of the SP method to stationary signals in Section III-A. We next introduce the hybrid model-based/data-driven implementation of the SP algorithm in Section
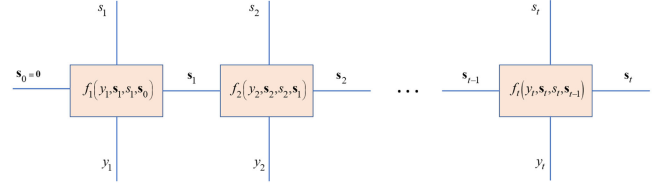
Fig. 1.   Factor graph of a Markovian time sequence.

III-B through the concept of learned factor graphs, which is followed by a discussion of this proposed method in Section III-C. Then, in Section III-D we show how the learned factor graph can be used to carry out Viterbi detection, detailing how the architecture proposed in [30] for receiver design in digital communications can be obtained as a special case of a learned factor graph. Finally, we present training of learned factor graphs in the presence of blockwise stationary distributions based on some future correctness indication in Section III-E.

A. SP Inference for Stationary Markovian Time Sequences

The SP algorithm computes the MAP rule in (3) for the signal model detailed in Section II by recursive message passing. To formulate this application of the SP scheme, define the vector variable $\boldsymbol{s}_i \triangleq \boldsymbol{s}_{i-l+1}^{i} \in \mathcal{S}^l$ (similarly, the random vector $\boldsymbol{S}_i \triangleq \boldsymbol{S}_{i-l+1}^{i} \in \mathcal{S}^l$). We can now represent the factorizable joint distribution $P_{\boldsymbol{Y}^t, \boldsymbol{S}^t}(\cdot)$ (2) as the factor graph illustrated in Fig. 1 (see [3, Fig. 15]), where the function nodes are

$$f_i(y_i, \boldsymbol{s}_i, \boldsymbol{s}_{i-1}) \triangleq P_{Y_i|\boldsymbol{S}_{i-l}^{i}}\left(y_i|\boldsymbol{s}_{i-l}^{i}\right) P_{S_i|\boldsymbol{S}_{i-l}^{i-1}}\left(s_i|\boldsymbol{s}_{i-l}^{i-1}\right).$$

Consequently, for the considered Markovian model, the partition sets $\{\mathcal{V}_k\}$ defined in Section II-C satisfy $\mathcal{V}_k = \{y_k. \boldsymbol{s}_k, \boldsymbol{s}_k, \boldsymbol{s}_{k-1}\}$. Due to the stationarity of the model, it holds that the mapping $f_i(\cdot)$ does not depend on the index i, and since s_i is an element of the vector $\boldsymbol{s}_i$, we can write the function nodes as

$$f_i(y_i, \boldsymbol{s}_i, \boldsymbol{s}_i, \boldsymbol{s}_{i-1}) = P_{Y_i|\boldsymbol{S}_i, \boldsymbol{S}_{i-1}}(y_i|\boldsymbol{s}_i, \boldsymbol{s}_{i-1})$$

$$\times P_{\boldsymbol{S}_i|\boldsymbol{S}_{i-1}}(\boldsymbol{s}_i|\boldsymbol{s}_{i-1})$$

$$\triangleq f(y_i, \boldsymbol{s}_i, \boldsymbol{s}_{i-1}). \quad (8)$$

In the special case where it holds that $P_{Y_i|\boldsymbol{S}_i, \boldsymbol{S}_{i-1}}(y_i|\boldsymbol{s}_i, \boldsymbol{s}_{i-1}) = P_{Y_i|\boldsymbol{S}_i,}(y_i|\boldsymbol{s}_i)$ in (8), the statistical model and the corresponding factor graph coincides with that of a hidden Markov model. Note that when $\boldsymbol{s}_i$ is a shifted version of $\boldsymbol{s}_{i-1}$, (8) coincides with $P_{Y_i|\boldsymbol{S}_{i-l}^{i}}(y_i|\boldsymbol{s}_{i-l}^{i}) P_{S_i|\boldsymbol{S}_{i-1}}(s_i|\boldsymbol{s}_{i-l}^{i-1})$, otherwise it equals zero. The fact that the distribution is stationary implies that the function node mapping $f(\cdot)$ is invariant to the time index i. Using its factor graph representation, one can compute the joint distribution of $\boldsymbol{S}^t$ and $\boldsymbol{Y}^t$ by recursive message passing along its factor graph. In particular,

$$P_{\boldsymbol{S}_k, \boldsymbol{S}_{k+1}, \boldsymbol{Y}^t}(s_k, s_{k+1}, \boldsymbol{y}^t) =$$

$$\overrightarrow{\mu}_{\boldsymbol{S}_k}(\boldsymbol{s}_k) f(y_{k+1}, \boldsymbol{s}_{k+1}, \boldsymbol{s}_k) \overleftarrow{\mu}_{\boldsymbol{S}_{k+1}}(\boldsymbol{s}_{k+1}), \quad (9)$$
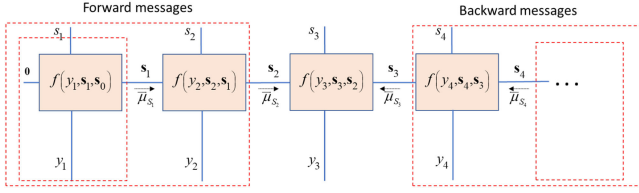
Fig. 2.   Message passing over the factor graph of a Markovian stationary time sequence.

Fig. 3.   SP inference over a learned stationary factor graph.

where for $i = 1, \ldots, k$, the forward messages satisfy

$$\overrightarrow{\mu}_{\boldsymbol{S}_i}(\boldsymbol{s}_i) = \sum_{\boldsymbol{s}_{i-1}} f(y_i, \boldsymbol{s}_i, \boldsymbol{s}_{i-1}) \overrightarrow{\mu}_{\boldsymbol{S}_{i-1}}(\boldsymbol{s}_{i-1}). \qquad (10)$$

Similarly, for $i = k+1, \ldots, t-1$, the backward messages are

$$\overleftarrow{\mu}_{\boldsymbol{S}_i}(\boldsymbol{s}_i) = \sum_{\boldsymbol{s}_{i+1}} f(y_{i+1}, \boldsymbol{s}_{i+1}, \boldsymbol{s}_i) \overleftarrow{\mu}_{\boldsymbol{S}_{i+1}}(\boldsymbol{s}_{i+1}). \qquad (11)$$

This message passing is illustrated in Fig. 2.

The ability to compute the joint distribution in (9) via message passing leads to computation of the MAP detector in (3) with complexity that grows linearly with t; without this message passing the computation grows exponentially with the block size. This reduction in complexity is achieved by noting that the MAP estimate satisfies

$$\hat{s}_i\left(\boldsymbol{y}^t\right) = \arg\max_{s_i \in \mathcal{S}} \sum_{\boldsymbol{s}_{i-1} \in \mathcal{S}^l} \overrightarrow{\mu}_{\boldsymbol{S}_{i-1}}(\boldsymbol{s}_{i-1}) f(y_i, [s_{i-l+1}, \ldots, s_i], \boldsymbol{s}_{i-1})$$

$$\times \overleftarrow{\mu}_{\boldsymbol{S}_i}([s_{i-l+1}, \ldots, s_i]), \qquad (12)$$

for each $i \in \mathcal{T}$, where the summands can be computed recursively. When t is large, the messages may tend to zero. Hence, the messages are commonly scaled [2], e.g., $\overleftarrow{\mu}_{\boldsymbol{S}_i}(\boldsymbol{s})$ is replaced with $\gamma_i \overleftarrow{\mu}_{\boldsymbol{S}_i}(\boldsymbol{s})$ for some scale factor that does not depend on $\boldsymbol{s}$, and thus does not affect the MAP rule. This instantiation of the SP algorithm is summarized in Algorithm 1 below.

Algorithm 1: The SP Algorithm for Stationary Markovian Sequences.

Init: Fix an initial forward message $\overrightarrow{\mu}_{\boldsymbol{S}_0}(\boldsymbol{s}) = 1$ and a
     final backward message $\overleftarrow{\mu}_{\boldsymbol{S}_t}(\boldsymbol{s}) \equiv 1$.
1   **for** $i = t-1, t-2, \ldots, 1$ **do**
2 | For each $\boldsymbol{s} \in \mathcal{S}^l$, compute backward message
       | $\overleftarrow{\mu}_{\boldsymbol{S}_i}(\boldsymbol{s})$ via (11) ;   // backward messages
3   **end**
4   **for** $i = 1, 2, \ldots, t$ **do**
5 | For each $\boldsymbol{s} \in \mathcal{S}^l$, compute forward message
       | $\overrightarrow{\mu}_{\boldsymbol{S}_i}(\boldsymbol{s})$ via (10) ;   // forward messages
6   **end**
Output: $\hat{\boldsymbol{s}}^t = [\hat{s}_1, \ldots, \hat{s}_t]^T$, each obtained using (12).

B. Learned Factor Graphs

Here, we propose a hybrid model-based/data-driven implementation of the SP scheme in Algorithm 1, which learns to implement M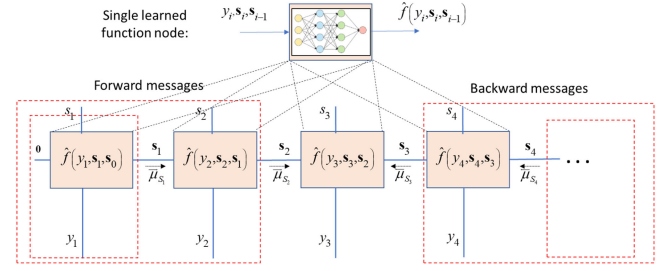AP detection of stationary Markovian time sequences from labeled data. Our framework builds upon the fact that in order to implement Algorithm 1, one must be able to specify the factor graph representing the underlying distribution. In particular, the stationarity assumption implies that the complete factor graph is encapsulated in the single function $f(\cdot)$ (8) *regardless of the block size t.* The Markovian nature of the signals implies that the structure of the graph is known to be of the form detailed in the previous subsection and illustrated in Fig. 2, regardless of the actual values of its function nodes. Building upon this insight, we utilize DNNs to learn the mapping carried out at the function node separately from the inference task. By doing so, one can train a system to learn an underlying factor graph, which can then be utilized for inference using conventional factor graph methods, such as the SP algorithm. The resulting learned stationary factor graph is then used to recover $\{S_i\}$ by message passing, as illustrated in Fig. 3.

In order to learn a stationary factor graph from samples, one must only learn its function node, which boils down to learning $P_{Y_i|\boldsymbol{S}_{i-l}^i}(\cdot)$ and $P_{S_i|\boldsymbol{S}_{i-l}^{i-1}}(\cdot)$ by (8). Specifically, for stationary sequences, only a single function node must be learned, as the mapping $f(\cdot)$ does not depend on the time index i. This implies that one can utilize a single learned mapping, denoted $\hat{f}(\cdot)$, to carry out SP-based inference over an arbitrary blocklength t, as illustrated in Fig. 3. When $\{S_i\}$ take values in a finite set, i.e., $\mathcal{S}$ is finite, the transition probability $P_{S_i|\boldsymbol{S}_{i-l}^{i-1}}(\cdot)$ can be learned via a histogram, as we do in our numerical study. For learning the distribution $P_{Y_i|\boldsymbol{S}_{i-l}^i}(\cdot)$ we consider two architectures, based on classification and density estimation networks, respectively.

1) Function Nodes as Classification Networks: Since y_i is given and may take continuous values while the desired variables take discrete values, a natural approach to evaluate $P_{Y_i|\boldsymbol{S}_{i-l}^i}(y_i|\boldsymbol{s})$ for each $\boldsymbol{s} \in \mathcal{S}^{l+1}$ is to estimate $P_{\boldsymbol{S}_{i-l}^i|Y_i}(\boldsymbol{s}|y_i)$, from which $P_{Y_i|\boldsymbol{S}_{i-l}^i}(\cdot)$ is obtained using Bayes rule as

$$P_{Y_i|\boldsymbol{S}_{i-l}^i}(y_i|\boldsymbol{s}) = P_{\boldsymbol{S}_{i-l}^i|Y_i}(\boldsymbol{s}|y_i) P_{Y_i}(y_i) \left(P_{\boldsymbol{S}_{i-l}^i}(\boldsymbol{s})\right)^{-1}. \quad (13)$$

A parametric estimate of $P_{\boldsymbol{S}_{i-l}^i|Y_i}(\boldsymbol{s}|y_i)$, denoted $\hat{P}_{\boldsymbol{\theta}}(\boldsymbol{s}|y_i)$, is obtained for each $\boldsymbol{s} \in \mathcal{S}^{l+1}$ by training classification networks with softmax output layers to minimize the cross entropy loss.

In general, the marginal PDF of Y_i can be estimated from the training data using mixture density estimation via, e.g., expectation maximization [40, Ch. 2], or any other finite mixture model fitting method. However, while the joint distribution in (2) depends on the marginal PDF of Y_i, the MAP rule is invariant
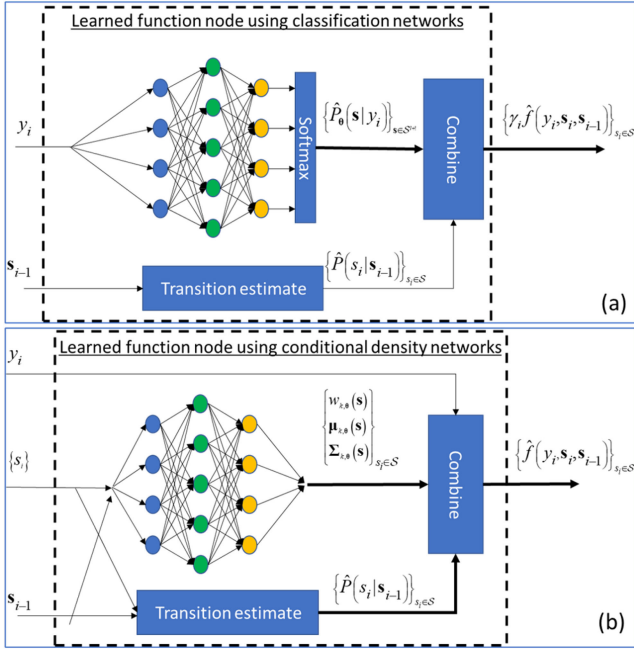
Fig. 4. Learned function node architectures for evaluating $\hat{f}(\cdot)$ based on (a) classification DNNs. (b) Conditional density estimation networks.

of it. This follows since $P_{Y_i}(y_i)$ does not depend on the variable s, and thus (12) can be written as

$$
\hat{s}_i\left(\boldsymbol{y}^t\right) = \arg\max_{s_i \in \mathcal{S}} \sum_{\boldsymbol{s}_{i-1} \in \mathcal{S}^l} \overrightarrow{\mu}_{\boldsymbol{S}_{i-1}}(\boldsymbol{s}_{i-1}) \frac{f(y_i, [s_{i-l+1}, \dots, s_i], \boldsymbol{s}_{i-1})}{P_{Y_i}(y_i)}
$$

$$
\times \overleftarrow{\mu}_{\boldsymbol{S}_i}([s_{i-l+1}, \dots, s_i]). \quad (14)
$$

Consequently, one can use a surrogate factor graph in which the function nodes are computed as $\frac{f(y_i, s_i], s_{i-1})}{P_{Y_i}(y_i)}$ instead of using (8) without altering the predictions of the SP algorithm. The surrogate function nodes, which differ from (8) yet yield the same inference rule, are equivalently computed by setting $P_{Y_i}(y_i) \equiv 1$ in (13). The resulting structure in which the parametric estimates are combined into a learned function node $\hat{f}(\cdot)$, scaled by some constant $\gamma_i = \frac{1}{P_{Y_i}(y_i)}$, is illustrated in the upper part of Fig. 4.

2) Function Nodes as Conditional Density Networks: An additional strategy is to directly estimate the conditional $P_{Y_i|\boldsymbol{S}_{i-l}^i}(y_i|s)$ from data. This can be achieved using conditional density estimation networks [40], [41] that are specifically designed to learn such PDFs. Alternatively, normalizing flow networks [42] can be used; these architectures which are typically used in the context of generative models, are capable of explicitly learning complex densities [43]. For example, mixture density networks [40] model the conditional PDF $P_{Y_i|\boldsymbol{S}_{i-l}^i}(y_i|s)$ as a mixture of K Gaussians, and train a DNN to learn a parametric estimate of its mixing parameters, mean values, and covariances, denoted $w_{k,\boldsymbol{\theta}}(s)$, $\boldsymbol{\mu}_{k,\boldsymbol{\theta}}(s)$ and $\boldsymbol{\Sigma}_{k,\boldsymbol{\theta}}(s)$, respectively, by maximizing the likelihood of

$$
\hat{P}_{\boldsymbol{\theta}}(y_i|s) = \sum_{k=1}^{K} w_{k,\boldsymbol{\theta}}(s) \mathcal{N}\left(y_i|\boldsymbol{\mu}_{k,\boldsymbol{\theta}}(s), \boldsymbol{\Sigma}_{k,\boldsymbol{\theta}}(s)\right), \quad (15)
$$

as illustrated in the lower part of Fig. 4.

Both of the above architectures can be used for learning the conditional distribution $P_{Y_i|\boldsymbol{S}_{i-l}^i}(\cdot)$ utilized by the learned stationary factor graph. To provide guidelines for choosing between these architectures, we note that when $\mathcal{Y}$ is high-dimensional, directly learning the conditional density is difficult and likely to be inaccurate. In such cases, the classification-based architecture, which avoids the need to explicitly learn the density by accounting for the invariance of Algorithm 1 to message scaling, may be preferable. When the state cardinality $|\mathcal{S}|^{l+1}$ is large, conditional density networks are expected to be more reliable. However, the SP algorithm, which computes the messages for each possible state, becomes computationally infeasible when $|\mathcal{S}|^{l+1}$ grows, making the application of the SP algorithm over learned factor graphs non-suitable for such setups. Consequently, in our numerical study we use the classification network architecture for learning the function nodes.

C. Using Learned Factor Graphs for Inference

The proposed approach of inference over learned factor graphs has several key advantages: First, as learning a single function node is expected to be a simpler task compared with learning the overall inference method for recovering $\boldsymbol{S}^t$ from $\boldsymbol{Y}^t$, this approach uses relatively compact DNNs, which can be learned from a relatively small set of labeled data. Furthermore, the learned function node describes the factor graph for different values of t, implying that the same architecture can be used for inference from sequences with different lengths. When the learned function node is an accurate estimate of the true one, message passing over it effectively implements the MAP detection rule (3), and thus approaches the minimal probability of error for each time instance.

Learned factor graphs rely on prior knowledge of the graph structure, which directly follows from the Markovian and stationarity assumptions. As such, it incorporates this limited level of domain knowledge in the structure of the factor graph, but does not impose any assumptions on the function nodes. These properties allow the representation of complex and possibly analytically intractable joint distributions as learned factor graphs, as long as they obey the Markovian stationary structure. The mapping of the function nodes is learned in a model-invariant manner from labeled data comprised of realizations of the observed sequence along with the corresponding realizations of the hidden state. In practice, the latter can be obtained from measurements or based on human annotations, as in the numerical study detailed in Section IV-A. If additional domain knowledge is present, it can be incorporated via imposing some parametric model on the function nodes $\{\hat{f}(\cdot)\}$, which follows from our understanding of the behavior of the setup at hand. Moreover, partial domain knowledge can be exploited to facilitate unsupervised training of the learned modules, thus relieving the dependence on known realizations of the state sequence, as recently proposed for DNN-aided tracking based in [44]. The operation of the proposed learned SP inference is invariant of whether its DNN-based function nodes were trained in a supervised or in an unsupervised manner. Nonetheless, we leave the study of unsupervised training of the learned function nodes for future investigation.

The proposed strategy significantly simplifies inference for scenarios represented by cycle-free factor graphs, such as the considered Markovian setup, compared with previously proposed DNNs whose structure imitates the message passing operation trained end-to-end, such as factor graph neural networks [26], [29]. In particular, this approach enables the usage of compact DNNs which can be trained using small data sets and achieve improved performance over factor graph neural networks for cycle-free factor graphs, as numerically demonstrated in Section IV. Furthermore, the learned factor graph can be applied to stationary time sequences of different lengths, without having to change its architecture and train anew, as well as be utilized with different message passing mechanisms, as we show in Section III-D. For blockwise stationary sequences, one only needs to learn a different function node for each block, as discussed in Section III-E. Factor graph neural networks are expected to be advantageous for scenarios characterized by loopy factor graphs, i.e., setups not obeying the system model detailed in Section II-A, where SP inference does not coincide with the MAP rule, and thus training end-to-end with sufficient data can lead to an improved inference rule.

The proposed approach of inference over learned factor graphs bears some similarity to previous works on the optimization of information bounds for communication channels with memory [45], [46]. In particular, the channel model considered in [45] specializes to the stationary Markovian setup of (2) under the additional constraint that the observations take values in a finite set. The main similarity to our work follows from the optimization over an auxiliary observations model, which [45] uses for formulating tight bounds on the information rates. Here, we use DNNs to capture the subtleties of this model from data such that the data-driven model can be integrated into factor graph based inference. Despite the similarity in the model and the approach, our work is somewhat different from these prior works on information rate bounds in the considered task as well as the usage of deep learning tools combined with principled model-based algorithms.

The proposed approach for using learned factor graphs trains the function nodes separately from the inference algorithm which utilizes the factor graph. Consequently, the factor graph learned can be processed using various message passing algorithms other than the SP scheme, e.g., the max-product method [2] and the Viterbi algorithm [3]. In particular, in the following subsection we show that the function nodes of learned factor graphs assuming equiprobable $\{S_i\}$ produce the same learned quantities as that used by ViterbiNet, proposed in [30] for symbol detection in finite-memory communication channels. Furthermore, the rationale of inference over learned factor graphs can be combined with alternative factor graph structures and message passing mechanisms, such as those used for joint channel estimation and iterative detection in [47]–[50]. Finally, we expect the same design to be applicable when training in an end-to-end manner, i.e., by backpropating through the message passing algorithm, as was done in [51] for image segmentation problems. We leave these extensions for future research.

D. Application as the Viterbi Algorithm

SP inference over learned factor graphs detailed in Section III-B is based on the ability to learn a parametric estimate of the function node mapping for factor graphs of stationary time sequences. However, once the factor graph is learned from data, it can also be used by factor graph inference algorithms other than the SP method. One such alternative designed for Markovian observations is the Viterbi scheme [3], originally proposed for decoding convolutional channel codes in digital communications. While the Viterbi algorithm can be applied with both forward and backward recursions over the underlying factor graph, we focus here on its implementation with a single forward recursive computation, following the description of the Viterbi scheme in [52, Ch. 3.4] and [34, Ch. 8.3].

1) The Viterbi Algorithm: The Viterbi detector aims at recovering the maximum likelihood sequence estimator in (4) via

$$\hat{s}^t\left(y^t\right) = \operatorname*{arg\,min}_{s^t \in \mathcal{S}^t} -\log P_{\boldsymbol{Y}^t|\boldsymbol{S}^t}\left(\boldsymbol{y}^t|\boldsymbol{s}^t\right)$$

$$= \operatorname*{arg\,min}_{s^t \in \mathcal{S}^t} \sum_{i=1}^{t} -\log P_{Y_i|\boldsymbol{S}_i}\left(y_i|\boldsymbol{s}_i\right), \qquad (16)$$

where in (16) we use the abbreviated term $\boldsymbol{S}_i$ defined in Section III-A, i.e., $\boldsymbol{S}_i = \boldsymbol{S}_{i-l+1}^{i}$. The optimization problem (16) can be solved recursively using dynamic programming, by iteratively updating a *path cost* $c_i(\boldsymbol{s})$ for each state $\boldsymbol{s} \in \mathcal{S}^l$. The resulting scheme, known as the Viterbi algorithm, is given below as Algorithm 2.

Algorithm 2: The Viterbi Algorithm [4].

Init: Fix an initial path $\boldsymbol{p}_0\left(\boldsymbol{s}\right) = \varnothing$ and path cost $c_0(\boldsymbol{s}) = 0$, for each $\boldsymbol{s} \in \mathcal{S}^l$.

1 **for** $i = 1, 2, \ldots, t$ **do**

2   For each $\boldsymbol{s} \in \mathcal{S}^l$, compute previous state with shortest path, denoted $\boldsymbol{u_s}$, via

$$\boldsymbol{u_s} = \operatorname*{arg\,min}_{\boldsymbol{u} \in \mathcal{S}^l: P_{\boldsymbol{S}_i|\boldsymbol{S}_{i-1}}(\boldsymbol{s}|\boldsymbol{u})>0} \left(c_{i-1}(\boldsymbol{u}) - \log P_{Y_i|\boldsymbol{S}_i}(y_i|\boldsymbol{s})\right).$$
$$(17)$$

3   Update cost and path via

$$c_i\left(\boldsymbol{s}\right) = c_{i-1}\left(\boldsymbol{u_s}\right) - \log P_{Y_i|\boldsymbol{S}_i}\left(y_i|\boldsymbol{s}\right), \quad (18)$$

    and $\boldsymbol{p}_i\left(\boldsymbol{s}\right) = \left[\boldsymbol{p}_{i-1}\left(\boldsymbol{u_s}\right), \boldsymbol{u_s}\right]$;

4 **end**

Output: $\hat{\boldsymbol{s}}^t = \boldsymbol{p}_t\left(\boldsymbol{s}^*\right)$ where $\boldsymbol{s}^* = \arg\min_{\boldsymbol{s}} c_t(\boldsymbol{s})$.

Algorithm 2 outputs its estimate of the complete unknown sequence, i.e., $\hat{\boldsymbol{s}}^t$, at time instance t. However, its output can also be approximated in real-time, since all paths at time $i + l$, $\{\boldsymbol{p}_{i+l}(\boldsymbol{s})\}_{\boldsymbol{s} \in \mathcal{S}^l}$ typically include the same states corresponding to time instances not larger then i [35], [53]. Consequently, the output of Algorithm 2 can be approached by estimating $\hat{\boldsymbol{s}}_i$ when processing y_{i+l}, and thus produce its estimates with a constant delay of l time instances. It is emphasized though that while this implementation typically comes at a negligible performance loss compared to Algorithm 2, it is no longer guaranteed to recover the maximum likelihood sequence estimator.

2) Viterbi Detection Over Learned Factor Graphs: To see that the learned factor graph detailed in Section III-B can also be applied for Viterbi detection, we focus on the case where the conditional distribution of $\boldsymbol{S}_i$ given $\boldsymbol{S}_{i-1}$ is uniform, as is commonly the case in digital communications systems for which Algorithm 2 was originally derived.

In such cases the shortest path (17) can be written as

$$
\boldsymbol{u_s} = \underset{\boldsymbol{u} \in \mathcal{S}^l}{\arg\min} \left(c_{i-1}\left(\boldsymbol{u}\right) - \log P_{Y_i|\boldsymbol{S}_i}\left(y_i|\boldsymbol{s}\right) - \log P_{\boldsymbol{S}_i|\boldsymbol{S}_{i-1}}\left(\boldsymbol{s}|\boldsymbol{u}\right) \right)
$$

$$
= \underset{\boldsymbol{u} \in \mathcal{S}^l}{\arg\min} \left(c_{i-1}\left(\boldsymbol{u}\right) - \log f_i\left(y_i, \boldsymbol{s}, \boldsymbol{u}\right) \right), \tag{19}
$$

where $f_i(\cdot)$ is defined in (8).

Similarly, the cost update (18) can be replaced with

$$
c_i\left(\boldsymbol{s}\right) = c_{i-1}\left(\boldsymbol{u_s}\right) - \log f_i\left(y_i, \boldsymbol{s}, \boldsymbol{u_s}\right), \tag{20}
$$

without affecting the resulting inference rule. Furthermore, scaling the function nodes $f_i(\cdot)$ by some γ_i that is independent of the state does not affect the Viterbi algorithm due to the $\arg\min$ statements in Algorithm 2. This implies that the same learned factor graph proposed in Section III-B for SP inference, which trains parametric estimates of $f_i(\cdot)$, can be utilized to also carry out Viterbi detection in a data-driven manner. In fact, carrying out the Viterbi algorithm (rather than the SP method) over the learned factor graphs coincides with ViterbiNet, proposed in [30] for symbol detection in finite-memory communications. The application of the model-based SP algorithm in such scenarios, i.e., symbol detection in finite-memory channels, specializes to the BCJR algorithm [6]. Thus SP inference over learned factor graphs of the joint input-output distribution of finite memory communications implements BCJR detection from data.

3) Discussion: Learned factor graphs can thus be applied, once trained, to carry out multiple inference algorithms, including the SP scheme (as proposed in Section III-B) as well as the Viterbi algorithm (via ViterbiNet). Furthermore, while Algorithm 2 considers the combination of learned factor graphs with Viterbi detection to produce hard decisions, where the output is the vector $\hat{\boldsymbol{s}}^t \in \mathcal{S}^t$, similar computations can be used to output soft decisions by utilizing the soft-output Viterbi algorithm over the learned factor factor graph. Once the factor graph encapsulating the underlying distribution is learned, one can decide which inference algorithm to apply to a learned factor graph. The preference of one method over the other is invariant of the learned factor graph, and follows from the differences between model-based message passing schemes, e.g., the differences between the SP algorithm and the Viterbi algorithm. The main advantages of Algorithm 2 over Algorithm 1, and thus of using a learned factor graph as part of ViterbiNet over SP inference, are its reduced complexity and the fact that it can be approached using a real-time operation. In particular, while the complexity of both algorithms grows linearly with the block size t, the Viterbi scheme as detailed in Algorithm 2 computes only a forward recursion and can thus provide its estimations in real time within a given delay from each incoming observation, while the SP scheme implements both forward and backward recursions, and can thus infer only once the complete block is observed. One can also implement the SP method using only

the forward messages, and thus share the real-time operation and reduced complexity of Algorithm 2, at the cost of reduced accuracy and deviation from the MAP rule.

The main advantage of Algorithm 1 over Algorithm 2, i.e., of using the learned factor graphs for SP inference with forward and backward recursions rather than as part of the ViterbiNet system, stems from the fact that it implements the MAP rule (3), which minimizes the symbol error probability. The maximum likelihood sequence detector (4) computed by the Viterbi algorithm, requires the states to be equiprobable in order to be able to approach the performance of the MAP rule. In digital communications, where the Viterbi algorithm originates, the states correspond to transmitted symbols, which are commonly equiprobable, and thus the Viterbi detector is far more popular and widely used compared with the SP scheme. However, in many other problems involving inference from time sequences, such as sleep pattern detection considered in Section IV-A, the states do not obey a uniform distribution, making the combination of learned factor graphs with SP inference the more attractive and natural data-driven method for such tasks.

E. Application for Blockwise-Stationary Statistical Variations

In the previous subsections we discussed how one can learn to infer from time sequences by using DNNs for estimating the function nodes in factor graphs with known structures, rather than using these networks for the complete end-to-end inference task. As a result, inference over learned factor graphs can be carried out using relatively compact networks which are trainable with a small number of training samples, as we also numerically demonstrate in our experimental study detailed in Section IV. This property of learned factor graphs facilitates their operation in the presence of blockwise stationary distributions.

As detailed in Section II-A, under a blockwise stationary distribution, the conditional PDF $P_{Y_i|\boldsymbol{S}_i}(\cdot)$, which is the mapping produced by a neural network in learned factor graphs, changes every t_b time instances. Such scenarios correspond to, e.g., communication over block-fading channels. In general, one can tackle these time variations by joint learning [54], [55], i.e., training the function nodes using labeled data corresponding to a broad set of expected distributions. However, this approach requires a large data set, and the resulting mapping may be inaccurate for each of the observed $P_{Y_i|\boldsymbol{S}_i}(\cdot)$. Alternatively, one can train in advance a different network for each statistical block via ensemble models [56]. Such a strategy is feasible when one has prior access to training data corresponding to each statistical relationship as well as knowledge regarding the order in which these statistical models are observed during inference. An additional approach involves the usage of meta-learning tools as in [57], [58] for tuning the architecture and training hyperparameters of the data-driven function node architecture such that it can be rapidly re-trained to accurately represent a given family of expected distributions.

Finally, when some future indication on the inference correctness is available, the ability to train data-driven function nodes individually using a small number of labeled samples can be exploited to track temporal variations in the underlying

statistics, without requiring prior knowledge of these variations. This is achieved by using the future indication to re-train the network in a self-supervised manner. For example, in a digital communications setup, the desired $\boldsymbol{S}^t$ represents the transmitted symbols, while the observed $\boldsymbol{Y}^t$ is the output of the channel used for recovering these symbols. Such communications are typically protected using error correction coding, implying that even when some of the symbols are inaccurately estimated, the conveyed message is still recoverable, as long as the number of errors does not exceed the code distance [34, Ch. 8]. In such cases, the recovered message can be re-encoded, generating the postulated transmitted symbols, which in turn can be used along with corresponding observations for re-training the learned parameters, as proposed in [30], [59], [60]. When this network can be effectively adapted using a small training set of the order of a communication codeword, the resulting data-driven digital communication receiver is capable of tracking the variations of the underlying statistical model, thereby avoiding errors caused by these variations. We numerically demonstrate the gains of this approach for adapting learned factor graphs to track block-wise variations in the context of digital communications in our experimental study described in Section IV-B.

IV. EXPERIMENTAL RESULTS

We next numerically evaluate inference over learned factor graphs in an experimental study. First, we consider the problem of sleep pattern detection, using the PhysioNet Sleep-EDF Expanded database [31]. Then, we focus on symbol detection in communication over finite-memory channels, using simulated data of common channel models. We conclude the section with an evaluation of the computational complexity associated with inference via learned factor graphs. Throughout this section, we use the classification network architecture (upper part of Fig. 4), trained with the Adam optimizer [61] to minimize the cross-entropy loss for learning the function nodes.[3] The state transition probability is estimated from the training data provided in each experiment via a histogram, i.e., for n_t labeled training samples $\{s_k, y_k\}_{k=1}^{n_t}$, it is computed as

$$\hat{P}(s_i|\boldsymbol{s}_{i-1}) = \frac{\sum_{k=l+1}^{n_t} \mathbb{1}\left([\boldsymbol{s}_{i-1}^T, s_i] = [s_{k-l}, \ldots, s_k]\right)}{\sum_{k=l+1}^{n_t} \mathbb{1}\left(\boldsymbol{s}_{i-1}^T = [s_{k-l}, \ldots, s_{k-1}]\right)}, \quad (21)$$

where $\mathbb{1}(\cdot)$ is the indicator function.

We utilize relatively compact networks which require only several minutes to train on a standard CPU. The specific architectures used for each setup are detailed in the description of the corresponding scenarios below.

A. Sleep Pattern Detection

Here, we consider the problem of sleep pattern detection from EEG signals. We use the PhysioNet Sleep-EDF Expanded database [31], which consists of 197 whole-night PolySomno-Graphic sleep recordings, containing EEG, EOG, chin EMG,
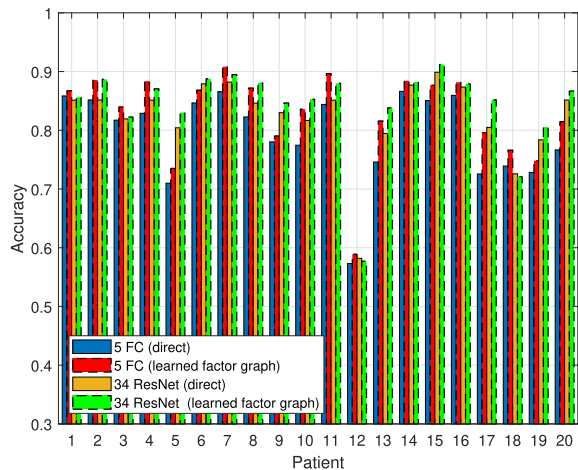

Fig. 5. The 5 FC network used for learning the function nodes.

and event markers. Similar to many prior works in this area [62]–[64], we use 20 patients from one of the two studies in this dataset that investigates the age effect in healthy subjects, known as the Sleep Cassette (SC) dataset. We focus on using a single EEG channel recording (the EEG channel Fpz-Cz) to classify five stages of sleep: awake (AWA), REM, and non-REM sleep stages (N1-N3), i.e., $|\mathcal{S}| = 5$. In particular, every 30 seconds of recording (i.e., 3000 EEG samples at 100 Hz), which is called an epoch, is labeled by human experts. For some of our experiments, we apply the feature extraction method proposed in [33] to extract 150 features in each epoch, used as the observation Y_i.
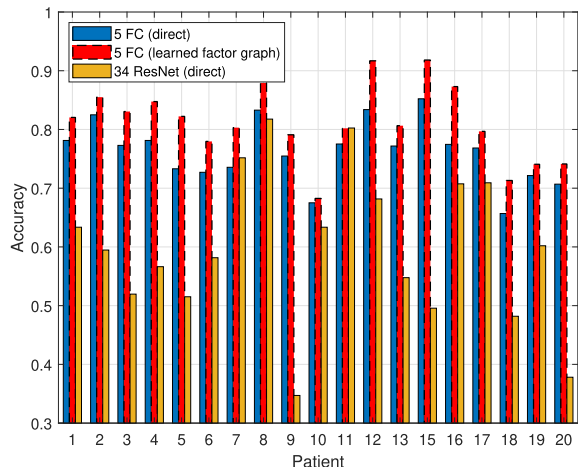
The task here is to identify the sleep states from the observed EEG signal. The common strategy is to train highly-parameterized DNNs to predict S_i from Y_i, based on convolutional layers [62], [64] or bidirectional recurrent neural networks (RNNs) [63]. Our goal here is to show that by using the compact networks associated with learned factor graphs, one can achieve comparable performance to previously proposed deep detectors. These compact networks can be trained using smaller data sets and are simpler to implement compared with previously proposed DNNs. We also show that the learned factor graph framework can be used to improve upon existing architectures, using their predictors for learning the function nodes instead of for directly recovering the desired $\{S_i\}$.

We learn the factor graph assuming that the sleep states follow a first-order Markov chain model, i.e., $l = 1$, and the joint distribution of states and measurements is stationary. In particular, we utilize a five layer fully-connected (FC) network (referred to as 5 FC) consisting of: 150×1200 FC layer with sigmoid activation; 1200×600, 600×300, and 300×150 FC layers with ReLU activations; and 150×5 FC layer with softmax output layer. An illustration of this network is depicted in Fig. 5. Here we use the 150 features extracted from EEG signals at each epoch as the network input. We also use a 34 layer residual network (referred to as 34 ResNet) proposed in [64], where the network input is the raw EEG signals during each epoch. For both of these networks, we compare the accuracy of the trained network directly applied for sleep pattern detection with that achieved when used as learned function nodes for SP inference.

First, we apply leave-one-out (LOO) cross-validation, where the data from one patient is left out for evaluation, while the data from the 19 other patients is used for training the models. The average accuracy of the algorithms for each patient is shown in Fig. 6(a), while the confusion matrices achieved using learned factor graphs when using the 5 FC network and with 34 ResNet are shown in Fig. 7(a)–(b), respectively. Here, the 5 FC network is trained with an initial learning rate of 0.001 over 50 epochs with a mini-batch size of 60 samples. Directly applying 5 FC

---

[3] The source code used in this section is available online at https://github.com/nirshlezinger1/LearnedFactorGraphs

(a) LOO cross validation accuracy.

(b) Train over 1000 samples accuracy.

Fig. 6.   Sleep pattern detection numerical results.

TABLE I
LOO OVERALL ACCURACY

Method	Accuracy
CNN-BLSTM [63]	82.0%
BLSTM-SVM [64]	82.5%
5 FC (direct)	79.2%
5 FC (learned factor graph)	82.8%
34 ResNet (direct)	82.3%
34 ResNet (learned factor graph)	**84.2%**

achieves an average accuracy of 79.2%, while using it to learn the factor graph in SP inference achieves an improvement of 3.6%, resulting in an accuracy of 82.8%. Similarly, the 34 ResNet achieves an accuracy of 82.3% while applying it to learn the factor graph achieves an average improvement of about 2%, resulting in an accuracy of 84.2%. As summarized by Table I, when using either the 5 FC network or the 34 ResNet classifier to learn the factor graph, our algorithm outperforms the state-of-the-art deep learning algorithms applied to this dataset [62], [63].

We also observe in Fig. 6(a) that for some patients, such as patient 12, the samples appear to obey a considerably different

statistical model from that of the remaining patients. This degrades the classification accuracy of the compact network, which in turn leads to an inaccurate estimate of the function nodes, resulting in accuracy below 75%. This motivates us to evaluate learned factor graphs when trained and tested using samples from the same patient, exploiting the compact networks of then learned factor graphs, which facilitates training from small data sets. We thus compute the accuracy when, for each patient, the networks are trained using its first 1000 samples and tested using the remaining samples, except for patient 14 for which less than 1000 samples are available.

The average accuracy for each patient is depicted in Fig. 6(b), and the confusion matrix achieved by learned factor graphs using the 5 FC network for learning the function nodes is shown in Fig. 7(c). Here, directly applying the 5 FC network as a classifier achieves an accuracy of 76%, as the DNN is trained using only 1000 samples, using it to form a learned factor graph improves the accuracy to 81%. 34 ResNet achieves only 60% accuracy due to its inability to properly train its highly parameterized network using small datasets. Building upon the ability of the relatively compact network to adapt with few samples, learned factor graphs achieve improved accuracy when applied to patients whose measurements obey a unique statistical model. For example, the SP method over the learned factor graph achieves an accuracy of 92% when applied to patient 12, improving by over 8% compared with using the network as predictor, and by 25%, compared with its performance when trained over the remaining 19 patients in Fig. 6(a). These results demonstrate the potential of combining neural networks for learning the function nodes rather than to carry out the complete classification tasks, as well as the advantages of this approach in allowing the usage of compact networks, which can be trained with small training sets, for accurate inference from time sequences with non-synthetic data.

B. Symbol Detection in Digital Communications

Next, we apply learned factor graphs for detection in a finite-memory communication setup. We use simulated data based on common channel models, which allows us to compare the performance of the SP method over learned factor graphs with that of the model-based SP algorithm, as well as to that of existing deep detectors. Here, a transmitter sends a sequence of symbols $S_i \in \mathcal{S}$, $i \in \mathcal{T}$, and a receiver uses the channel output $Y_i \in \mathbb{R}$ to recover the symbols. Each Y_i is affected only by the last l transmitted symbols, where l is the memory length.

We consider two channels with memory length $l = 4$: A Gaussian channel and a Poisson channel. Let $\boldsymbol{h}(\gamma) \in \mathbb{R}^l$ be a vector whose entries obey an exponentially decaying profile $h_\tau(\gamma) \triangleq e^{-\gamma(\tau-1)}$ for $\gamma > 0$ and $\tau \in \{1, \ldots, l\}$. For the Gaussian channel, the symbols take values in the set $\mathcal{S} = \{-1, 1\}$, representing a binary phase shift keying constellation, and the channel output is generated via

$$Y_i | \boldsymbol{S}^t \sim \mathcal{N}\left(\sqrt{\rho} \cdot \sum_{\tau=1}^{l} h_\tau(\gamma) S_{i-\tau+1}, 1\right), \quad (22)$$

where $\rho > 0$ represents the signal-to-noise ratio (SNR). For the Poisson channel, the channel input represents on-off keying, i.e.,

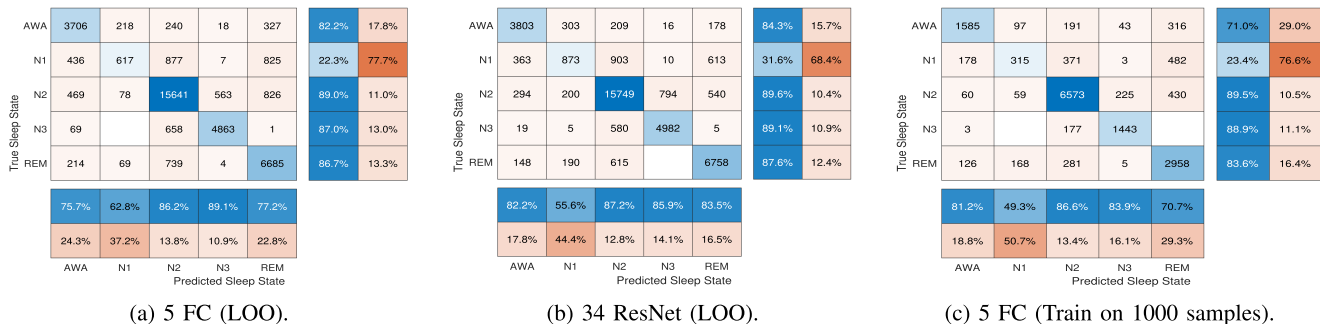(a) 5 FC (LOO). (b) 34 ResNet (LOO). (c) 5 FC (Train on 1000 samples).

Fig. 7. Sleep pattern detection confusion matrices of SP inference over learned factor graphs.

Fig. 8. The 3 FC network used for learning the function nodes.

$\mathcal{S} = \{0, 1\}$, and the channel output Y_i obeys

$$Y_i | \boldsymbol{S}^t \sim \mathcal{P}\left(\sqrt{\rho} \cdot \sum_{\tau=1}^{l} h_\tau(\gamma) S_{i-\tau+1} + 1\right), \qquad (23)$$

where $\mathcal{P}(\cdot)$ is the Poisson distribution. Namely, for each realization $\boldsymbol{S}^t = \boldsymbol{s}^t$, the channel output for every time instance is generated independently, where the output at corresponding to the ith time instance is generated from a Poisson distribution with parameter which equals $\sqrt{\rho} \cdot \sum_{\tau=1}^{l} h_\tau(\gamma) s_{i-\tau+1} + 1$ [65].

We implement the classification network with three FC layers: 1×100, 100×50, and 50×16 layers, using sigmoid and ReLU activation functions, respectively. The network is trained using 5000 samples, which is the order of a typical preamble sequence in wireless networks [66], with learning rate 0.01. The training is carried out over 100 epochs with mini-batch size of 27. Using these training samples to compute the transition probability via (21) yields an estimate with normalized mean-squared error of merely $1.01 \cdot 10^{-4}$ with respect to the true transition probability which equals $\frac{1}{|\mathcal{S}|} = \frac{1}{2}$ here. The DNN architecture is depicted in Fig. 8.

For each channel, we compute the symbol error rate (SER) achieved using learned factor graphs for different values of the SNR ρ, and the DNN-aided function node is trained anew for each value of ρ. For every SNR, the SER values are averaged over 20 different channel vectors $\boldsymbol{h}(\gamma)$, obtained by letting γ vary in the range $[0.1, 2]$. For comparison, we evaluate the SER of the model-based SP algorithm, as well as that of the data-driven sliding bidirectional RNN (SBRNN) deep detector proposed in [65] and the factor graph neural network of [26], for which we use the same architecture as that utilized in the numerical study in [26, Sec. 4.1]. We consider two cases: The first is *perfect channel state information (CSI)*, in which the SP method knows the exact $\boldsymbol{h}(\gamma)$, while the data-driven systems are trained using data consisting of samples generated with the same $\boldsymbol{h}(\gamma)$ used for the test data. The second setup considered,

referred to as *CSI uncertainty*, considers the scenario in which the SP algorithm is implemented using an estimate of $\boldsymbol{h}(\gamma)$ corrupted by Gaussian noise with variance which equals 10% and 8% of the magnitude of the channel tap for the Gaussian and Poisson channels, respectively. Using the considered setup for CSI uncertainty represents the operation of the model-based SP algorithm in scenarios where the underlying model is not accurately known, in a manner that is invariant of how this uncertainty is obtained. For the DNN-aided systems, the training data is generated with the noisy $\boldsymbol{h}(\gamma)$, allowing us to study resiliency to inaccurate training. In all cases, the information symbols are uniformly randomized in an i.i.d. fashion from $\mathcal{S}$, and the test samples are generated from their corresponding channel with the true vector $\boldsymbol{h}(\gamma)$.

The SER values, averaged over 50000 Monte Carlo simulations, are depicted in Fig. 9(a)–(b) for the Gaussian and Poisson channels, respectively. We observe in Fig. 9(a)–(b) that the SER achieved using learned factor graphs approaches that of the SP algorithm from which it originates, while the latter requires accurate prior knowledge of the underlying distribution. In the presence of CSI uncertainty, carrying out SP inference over a learned factor graph significantly outperforms applying it over the inaccurate model-based factor graphs. When the function nodes are trained with a variety of different channels, learned factor graphs achieve relatively good SER when inferring under each of the channels for which it is trained, while the performance of the conventional SP method is significantly degraded due to imperfect CSI. We also observe that the SBRNN receiver, shown in [65] to approach the performance of the MAP rule when sufficient training is provided, is outperformed by SP inference over learned factor graphs here due to the small training set. A similar observation is noted for the factor graph neural network of [26], whose SER performance is within a notable gap from by the proposed approach of learned factor graphs for the considered scenario. These results demonstrate the ability of learned factor graphs to enable accurate implementation of the SP method while requiring small training sets and improving robustness to uncertainty.

Next, we consider blockwise stationary channels, showing how the approach discussed in Section III-E can exploit coded communications for channel tracking. Here, transmission consists of multiple codewords of length $t_b = 2040$, each representing 1784 bits encoded using a Reed-Solomon (RS) [255, 223] channel code, and protected with a cyclic redundancy check (CRC) for error detection. We simulate the Gaussian and Poisson channels in (22)–(23), respectively. To
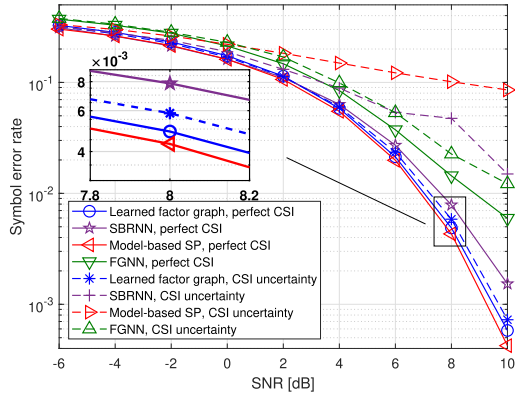
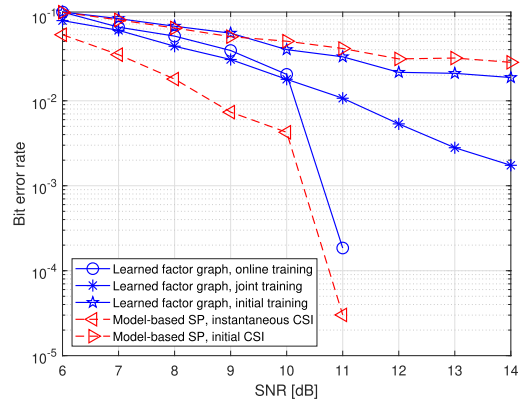(a) SER performance of different detectors in the Gaussian channel.
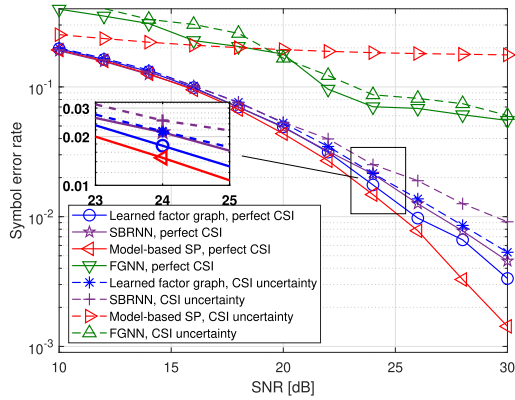
(b) SER performance of different detectors in the Poisson channel.

Fig. 9. Symbol detection accuracy of the SP algorithm over learned factor graphs compared the data-driven SBRNN detector [65] and the purely model-based SP method.
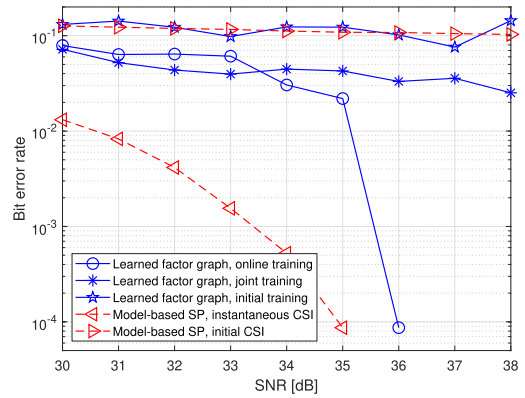

(a) BER performance of different detectors in the Gaussian channel.

(b) BER performance of different detectors in the Poisson channel.

Fig. 10. BER for blockwise stationary channels.

simulate block-wise temporal variations, we let the entries of $\boldsymbol{h}(\gamma)$ vary between codewords. As in [30], for the jth codeword, we use $h_\tau(\gamma) \triangleq e^{-\gamma(\tau-1)} \cdot (0.8 + 0.2\cos(\frac{2\pi \cdot j}{p_\tau}))$ for each $\tau \in \{1, \ldots, l\}$, with a fixed exponential decay parameter $\gamma = 0.2$, and with $\boldsymbol{p} = [51, 39, 33, 21]^T$, representing block-wise periodic variations in the channel coefficients.

Before the first block is transmitted, a factor graph representing a stationary distribution is learned using 5000 training samples taken using the initial channel coefficients. In order to track channel variations via online training, we use successful decoding to re-train the function nodes in a decision-directed manner [30], [59]. In particular, each recovered block $\hat{\boldsymbol{S}}^{t_b}$ is decoded to its corresponding 1784 bits using an RS decoder, and validated using a CRC check. If the CRC check passes, the bits are re-encoded using an RS encoder into a postulated symbol block $\tilde{\boldsymbol{S}}^{t_b}$, which is used along with its corresponding observed $\boldsymbol{Y}^{t_b}$ to retrain the learned function node using 50 epochs with an initial learning rate of 0.002.

In addition to evaluating learned factor graphs with online training, we also compute the coded bit error rate (BER) when the function node is trained only once using the 5000 training samples representing the initial channel, referred to as *initial training*, as well as when trained once using 5000 training samples corresponding to the channels observed at blocks $j \in 3 \cdot \{1, \ldots, 10\}$, referred to as *joint training*. The coded BER

of SP inference over learned factor graphs is compared with that of the SP detector with full instantaneous CSI as well as to that with knowledge of only the initial channel conditions. The coded BER results, averaged over 200 consecutive blocks, are depicted in Fig. 10(a)–(b) for the Gaussian and Poisson channels, respectively.

Observing Fig. 10(a)–(b), we note that for both channels, as the SNR increases, learned factor graphs with online training approaches the performance of the model-based SP detector with instantaneous CSI, which implements MAP detection here. The latter require accurate knowledge of the complete input-output statistical relationship for each block. For low SNRs, the performance of SP inference over the online-trained factor graph is only slightly improved compared with training only using the initial channel. This can be explained by noting that for high SNR values, the number of symbol detection errors does not grow above the code distance as the channel changes between block, and thus the proposed online training scheme is capable of generating reliable labels. The online-trained factor graphs can accurately track the channel, allowing inference with optimal-approaching performance. However, for low SNR values, the RS decoder frequently fails to correctly decode the bits, and the online training methods does not frequently update its neural network, thus achieving only a minor improvement over using only the initial training data set. The ability to track

channel variations in a decision-directed manner can be potentially improved by utilizing syndrome decoders, as proposed in [59]. We leave this study for future work.

We also observe in Fig. 10(a)–(b) that the (offline) joint training approach allows inference over learned factor graphs to achieve improved BER performance compared with using only initial training. This follows since the resulting decoder is capable of operating in a broader range of different channel conditions. Still, joint training is notably outperformed by the SP detector with instantaneous CSI, whose BER performance is approached only when using online training at high SNRs.

The results reported in Fig. 10(a)–(b) indicate that in order to reliably cope with non-stationary conditions, learned factor graphs designed for stationary setups should be combined with additional mechanisms for tracking the statistical variations, such as the online training mechanism discussed in Section III-E. Nonetheless, one can also design learned factor graphs to cope with non-stationarity by learning different function nodes for different time instances, at the cost of requiring larger datasets for training and possible limitations on the duration of the time sequences. We leave this extension of learned factor graphs for future study.

C. Inference Computational Complexity

We conclude this section by comparing the computational complexity of inference over learned factor graphs with that of end-to-end deep learning models. Specifically, we focus on the symbol detection setup detailed in Section IV-B and consider the SBRNN algorithm as the end-to-end deep learning model, as this architecture is shown to achieve the best performance among the end-to-end deep learning models evaluated in Fig. 9(a)–(b).

Recall that t is the length of the sequence, $\mathcal{S}$ is the symbol set, and l is the memory length of the channel. Let N_f be the computational complexity of the learned factor node in the factor graph, i.e., the number of operations required to map a single observation y_i and state vector $\boldsymbol{s}_{i-1}$ into an estimate of the function node $\{\hat{f}(y_i, \boldsymbol{s}_i, \boldsymbol{s}_{i-1})\}_{s_i \in \mathcal{S}}$. For instance, using the classification DNN architecture illustrated in Fig. 4(a), this operation involves passing the value y_i through a fully-connected DNN, which using the DNN in Fig. 8 is comprised of approximately $6 \cdot 10^3$ multiplications, and computing $|\mathcal{S}|$ transition estimates from $\boldsymbol{s}_i$ using a look-up table. Once the messages are computed for each state and for each time instance i, i.e., $|\mathcal{S}|^l \cdot t$ times, inference is carried out by computing the forward and backward messages via (10)–(11), resulting in an overall computational burden of the order of $\mathcal{O}(N_f \cdot |\mathcal{S}|^l \cdot t)$. For comparison, letting N_r be the computational complexity of the RNN block corresponding to one time-step in the SBRNN architecture, the computational complexity of the SBRNN is on the order of $\mathcal{O}(N_r \cdot l \cdot t)$. Generally, we expect $N_r \gg N_f$; For example, in the scenario reported in Section IV-B computing the function nodes is done using $N_f \approx 6 \cdot 10^3$ multiplications, while the SBRNN architecture requires $N_r \approx 1.3 \cdot 10^5$ multiplications per time instance. Therefore, for the symbol detection scenario, SP over learned factor graphs does not only train with less data compared with the SBRNN system, but also infers at reduced computational complexity.

When the state cardinality $|\mathcal{S}|^{l+1}$ is large, SP inference, and thus also its application over learned factor graphs, may be computationally prohibitive. In these regimes, one can still utilize learned factor graphs for inference at controllable computational burden by utilizing alternative inference methods based on, e.g., state reduction [67]. This indicates that it is possible to have learned factor graphs that are efficient both in terms of sample efficiency during training, and in terms of computational complexity during inference. Nonetheless, we leave the detailed study of the combination of learned factor graphs with such reduced complexity inference methods for future work.

V. Conclusion

In this work we proposed a framework for inference from stationary time sequences via learned factor graphs, combining the model-based SP algorithm with data-driven ML tools. By exploiting domain knowledge of a stationary and Markovian characteristics, encountered in many applications in signal processing and communications, the factor graph encapsulating the underlying distribution can be learned separately from the overall inference task. This results in a hybrid model-based/data-driven system based on compact neural networks which can be trained with relatively small training sets. The resulting algorithm carries out inference over the learned factor graph in a manner which is not restricted to a specific number of input samples. The integration of DNNs for learning the function nodes combined with domain knowledge which determines the structure of the graph results in a system which learns from data to carry out MAP-approaching detection in complex setups. Furthermore, a learned factor graph can be used with different message passing based inference algorithms, other than the SP method. Our numerical evaluations demonstrate the ability of learned factor graphs to facilitate accurate inference and improve upon existing classifiers for sleep pattern detection and symbol recovery. We also show that its ability to train with small training sets enables personalized learning as well as tracking of blockwise temporal variations in the statistical model, which are both extremely challenging to carry out using conventional highly-parameterized DNNs.

References

[1] G. D. Forney, "Codes on graphs: Normal realizations," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 520–548, Jun. 2001.

[2] H.-A. Loeliger, "An introduction to factor graphs," *IEEE Signal Process. Mag.*, vol. 21, no. 1, pp. 28–41, Jan. 2004.

[3] A. Viterbi, "Error bounds for convolutional codes and an asymptotically optimum decoding algorithm," *IEEE Trans. Inf. Theory*, vol. IT-13, no. 2, pp. 260–269, Apr. 1967.

[4] F. R. Kschischang, B. J. Frey, and H.-A. Loeliger, "Factor graphs and the sum-product algorithm," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 498–519, Feb. 2001.

[5] J. Pearl, "Fusion, propagation, and structuring in belief networks," *Artif. Intell.*, vol. 29, no. 3, pp. 241–288, 1986.

[6] L. Bahl, J. Cocke, F. Jelinek, and J. Raviv, "Optimal decoding of linear codes for minimizing symbol error rate," *IEEE Trans. Inf. Theory*, vol. IT-20, no. 2, pp. 284–287, Mar. 1974.

[7] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *Ann. Math. Statist.*, vol. 41, no. 1, pp. 164–171, 1970.

[8] S. S. Haykin, *Adaptive Filter Theory*. Noida, India: Pearson Education India, 2005.

[9] H.-A. Loeliger, J. Dauwels, J. Hu, S. Korl, L. Ping, and F. R. Kschischang, "The factor graph approach to model-based signal processing," *Proc. IEEE*, vol. 95, no. 6, pp. 1295–1322, Jun. 2007.

[10] J. C. B. Gamboa, "Deep learning for time-series analysis," 2017, *arXiv:1701.01887*.

[11] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: A review," *Data Mining Knowl. Discov.*, vol. 33, no. 4, pp. 917–963, 2019.

[12] A. Vaswani *et al.*, "Attention is all you need," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 5998–6008.

[13] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," 2020, *arXiv:2012.08405*.

[14] A. Bora, A. Jalal, E. Price, and A. G. Dimakis, "Compressed sensing using generative models," in *Proc. 34th Int. Conf. Mach. Learn.*, 2017, pp. 537–546.

[15] D. Van Veen, A. Jalal, M. Soltanolkotabi, E. Price, S. Vishwanath, and A. G. Dimakis, "Compressed sensing with deep image prior and learned regularization," 2018, *arXiv:1806.06438*.

[16] S. V. Venkatakrishnan, C. A. Bouman, and B. Wohlberg, "Plug-and-play priors for model based reconstruction," in *Proc. IEEE Global SIP*, 2013, pp. 945–948.

[17] R. Ahmad *et al.*, "Plug-and-play methods for magnetic resonance imaging: Using denoisers for image recovery," *IEEE Signal Process. Mag.*, vol. 37, no. 1, pp. 105–116, Jan. 2020.

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

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

[20] V. Monga, Y. Li, and Y. C. Eldar, "Algorithm unrolling: Interpretable, efficient deep learning for signal and image processing," *IEEE Signal Process. Mag.*, vol. 38, no. 2, pp. 18–44, Mar. 2021.

[21] A. Balatsoukas-Stimming and C. Studer, "Deep unfolding for communications systems: A survey and some new directions," in *Proc. IEEE Int. Workshop on Signal Process. Syst. (SiPS)*, 2019, pp. 266–271.

[22] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun, "Spectral networks and locally connected networks on graphs," in *Proc. 2nd Int. Conf. Learn. Representations*, 2014.

[23] S. Zheng *et al.*, "Conditional random fields as recurrent neural networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2015, pp. 1529–1537.

[24] G. Lin, C. Shen, I. Reid, and A. van den Hengel, "Deeply learning the messages in message passing inference," in *Proc. Adv. Neural Inf. Process. Syst.*, 2015, pp. 361–369.

[25] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, "Neural message passing for quantum chemistry," in *Proc. Int. Conf. Mach. Learn.*, 2017, pp. 1263–1272.

[26] Z. Zhang, F. Wu, and W. S. Lee, "Factor graph neural network," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 8577–8587.

[27] K. Yoon *et al.*, "Inference in probabilistic graphical models by graph neural networks," in *Proc. IEEE 53rd Asilomar Conf. Signals, Syst. Comput.*, 2019, pp. 868–875.

[28] V. G. Satorras, Z. Akata, and M. Welling, "Combining generative and discriminative models for hybrid inference," in *Proc. Adv. Neural Inf. Process. Syst.*, 2019, pp. 13802–13812.

[29] V. G. Satorras and M. Welling, "Neural enhanced belief propagation on factor graphs," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2021, pp. 685–693.

[30] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "ViterbiNet: A deep learning based Viterbi algorithm for symbol detection," *IEEE Trans. Wireless Commun.*, vol. 19, no. 5, pp. 3319–3331, May 2020.

[31] B. Kemp, "The sleep-EDF database online," 2013. [Online]. Available: http://www.physionet.org/physiobank/database/sleep-edf

[32] R. Vullings, B. De Vries, and J. W. Bergmans, "An adaptive Kalman filter for ECG signal enhancement," *IEEE Trans. Biomed. Eng.*, vol. 58, no. 4, pp. 1094–1103, Apr. 2010.

[33] D. Jiang, Y.-N. Lu, M. Yu, and W. Yuanyuan, "Robust sleep stage classification with single-channel EEG signals using multimodal decomposition and HMM-based refinement," *Expert Syst. Appl.*, vol. 121, pp. 188–203, 2019.

[34] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[35] G. D. Forney, "The Viterbi algorithm," *Proc. IEEE*, vol. 61, no. 3, pp. 268–278, Mar. 1973.

[36] J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. New York, NY, USA: Elsevier, 2014.

[37] S. M. Aji and R. J. McEliece, "The generalized distributive law," *IEEE Trans. Inf. Theory*, vol. 46, no. 2, pp. 325–343, Mar. 2000.

[38] Y. Weiss and W. T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs," *IEEE Trans. Inf. Theory*, vol. 47, no. 2, pp. 736–744, Feb. 2001.

[39] G. McLachlan and D. Peel, *Finite Mixture Models*. Hoboken, NJ, USA: Wiley, 2004.

[40] C. M. Bishop, "Mixture density networks," 1994. [Online]. Available: http://publications.aston.ac.uk/id/eprint/373/

[41] J. Rothfuss, F. Ferreira, S. Walther, and M. Ulrich, "Conditional density estimation with neural networks: Best practices and benchmarks," 2019, *arXiv:1903.00954*.

[42] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *Proc. Int. Conf. Mach. Learn.*, 2015, pp. 1530–1538.

[43] I. Kobyzev, S. Prince, and M. Brubaker, "Normalizing flows: An introduction and review of current methods," *IEEE Trans. Pattern Anal. Mach. Intell.*, 2020.

[44] G. Revach, N. Shlezinger, X. Ni, A. L. Escoriza, R. J. van Sloun, and Y. C. Eldar, "KalmanNet: Neural network aided Kalman filtering for partially known dynamics," 2021, *arXiv:2107.10043*.

[45] P. Sadeghi, P. O. Vontobel, and R. Shams, "Optimization of information rate upper and lower bounds for channels with memory," *IEEE Trans. Inf. Theory*, vol. 55, no. 2, pp. 663–688, Feb. 2009.

[46] F. Rusek and D. Fertonani, "Bounds on the information rate of intersymbol interference channels based on mismatched receivers," *IEEE Trans. Inf. Theory*, vol. 58, no. 3, pp. 1470–1482, Mar. 2012.

[47] C. Novak, G. Matz, and F. Hlawatsch, "IDMA for the multiuser MIMO-OFDM uplink: A factor graph framework for joint data detection and channel estimation," *IEEE Trans. Signal Process.*, vol. 61, no. 16, pp. 4051–4066, Aug. 2013.

[48] E. Riegler, G. E. Kirkelund, C. N. Manchon, M. A. Badiu, and B. H. Fleury, "Merging belief propagation and the mean field approximation: A free energy approach," *IEEE Trans. Inf. Theory*, vol. 59, no. 1, pp. 588–602, Jan. 2013.

[49] T. L. Hansen, P. B. Jørgensen, M. Badiu, and B. H. Fleury, "An iterative receiver for OFDM with sparsity-based parametric channel estimation," *IEEE Trans. Signal Process.*, vol. 66, no. 20, pp. 5454–5469, Oct. 2018.

[50] G. E. Kirkelund, C. N. Manchon, L. P. B. Christensen, E. Riegler, and B. H. Fleury, "Variational message-passing for joint channel estimation and decoding in MIMO-OFDM," in *Proc. IEEE Glob. Telecommun. Conf.*, 2010, pp. 1–6.

[51] P. Knobelreiter, C. Sormann, A. Shekhovtsov, F. Fraundorfer, and T. Pock, "Belief propagation reloaded: Learning BP-layers for labeling problems," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 7900–7909.

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

[53] H.-L. Lou, "Implementing the Viterbi algorithm," *IEEE Signal Process. Mag.*, vol. 12, no. 5, pp. 42–52, Sep. 1995.

[54] T. O'Shea and J. Hoydis, "An introduction to deep learning for the physical layer," *IEEE Trans. Cogn. Commun. Netw.*, vol. 3, no. 4, pp. 563–575, Dec. 2017.

[55] J. Xia, D. Deng, and D. Fan, "A note on implementation methodologies of deep learning-based signal detection for conventional MIMO transmitters," *IEEE Trans. Broadcast.*, vol. 66, no. 3, pp. 744–745, Sep. 2020.

[56] T. Raviv, N. Raviv, and Y. Be'ery, "Data-driven ensembles for deep and hard-decision hybrid decoding," in *Proc. IEEE Int. Symp. Inf. Theory*, 2020, pp. 321–326.

[57] S. Park, H. Jang, O. Simeone, and J. Kang, "Learning to demodulate from few pilots via offline and online meta-learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 226–239, 2021.

[58] T. Raviv, S. Park, N. Shlezinger, O. Simeone, Y. C. Eldar, and J. Kang, "Meta-ViterbiNet: Online meta-learned Viterbi equalization for non-stationary channels," 2021, *arXiv:2103.13483*.

[59] C.-F. Teng and Y.-L. Chen, "Syndrome enabled unsupervised learning for neural network based polar decoder and jointly optimized blind equalizer," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 10, no. 2, pp. 177–188, Jun. 2020.

[60] N. Shlezinger, R. Fu, and Y. C. Eldar, "DeepSIC: Deep soft interference cancellation for multiuser MIMO detection," *IEEE Trans. Wireless Commun.*, vol. 20, no. 2, pp. 1349–1362, Feb. 2021.

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

[62] A. Supratak, H. Dong, C. Wu, and Y. Guo, "DeepSleepNet: A model for automatic sleep stage scoring based on raw single-channel EEG," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 25, no. 11, pp. 1998–2008, Nov. 2017.

[63] H. Phan, F. Andreotti, N. Cooray, O. Y. Chén, and M. De Vos, "Automatic sleep stage classification using single-channel EEG: Learning sequential features with attention-based recurrent neural networks," in, *Proc. IEEE Eng. Med. Biol. Soc.*, 2018, pp. 1452–1455.

[64] A. I. Humayun, A. S. Sushmit, T. Hasan, and M. I. H. Bhuiyan, "End-to-end sleep staging with raw single channel EEG using deep residual convnets," in *Proc. IEEE Biomed. Health Infor.*, 2019, pp. 1–5.

[65] N. Farsad and A. Goldsmith, "Neural network detection of data sequences in communication systems," *IEEE Trans. Signal Process.*, vol. 66, no. 21, pp. 5663–5678, Nov. 2018.

[66] E. Dahlman, S. Parkvall, J. Skold, and P. Beming, *3G Evolution: HSPA and LTE for Mobile Broadband*. San Francisco, CA, USA: Academic, 2010.

[67] X. Lingyun and D. Limin, "Efficient Viterbi beam search algorithm using dynamic pruning," in *Proc. Int. Conf. Signal Process.*, 2004, pp. 699–702.

Nir Shlezinger (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees in electrical and computer engineering from Ben-Gurion University, Beersheba, Israel, in 2011, 2013, and 2017, respectively. He is currently an Assistant Professor with the School of Electrical and Computer Engineering, Ben-Gurion University. From 2017 to 2019, he was a Postdoctoral Researcher with the Technion, Haifa, Israel, and from 2019 to 2020, he was a Postdoctoral Researcher with the Weizmann Institute of Science, Rehovot, Israel, where he was awarded the FGS Prize for outstanding research achievements. His research interests include communications, information theory, signal processing, and machine learning.

Nariman Farsad (Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees all from York University, in Toronto, Canada. He is currently an Assistant Professor with Ryerson University, Toronto, ON, Canada. He was a Senior Machine Learning and a Algorithm Researcher with Apple, Inc., and from 2018 to 2020, he was a Visiting Research Fellow with Stanford University, Stanford, CA, USA. From 2015 to 2018, he was a Postdoctoral Research Fellow with Stanford University, where he was the recipient of the Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellowship. His research interests include improving the performance of communication systems through machine learning and deep learning and on emerging communication technologies such as molecular communications. He was a Member of the Technical Program Committees of ICC2015, ICC2018, BICT2015, GLOBCOM2015, GLOBCOM2016, and GLOBECOM2017. He was the recipient of the Natural Sciences and Engineering Research Council of Canada Postdoctoral Fellowship, the Second Prize from the 2014 IEEE ComSoc Student Competition, the Best Demo Award at INFOCOM2015, and was recognized as a finalist for the 2014 Bell Labs Prize. He is an Area Associate Editor of the IEEE JOURNAL OF SELECTED AREAS OF COMMUNICATIONS-Special Issue on Emerging Technologies in Communications and a Technical Reviewer for a number of journals, including the IEEE TRANSACTIONS ON SIGNAL PROCESSING and the IEEE TRANSACTIONS ON COMMUNICATIONS.

Yonina C. Eldar (Fellow, IEEE) received the B.Sc. degree in physics in 1995, and the B.Sc. degree in electrical engineering in 1996, from Tel-Aviv University, Tel-Aviv, Israel, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2002.

She is currently a Professor with the Department of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel. She was previously a Professor with the Department of Electrical Engineering, Technion, Haifa, Israel, where she held the Edwards Chair in Engineering. She is also a Visiting Professor with MIT, a Visiting Scientist with the Broad Institute, Cambridge, MA, USA, and an Adjunct Professor with Duke University, Durham, NC, USA, and was a Visiting Professor with Stanford, Stanford, CA, USA. She is author of the book *Sampling Theory: Beyond Bandlimited Systems* and co-author of five other books published by Cambridge University Press. Her research interests include the broad areas of statistical signal processing, sampling theory and compressed sensing, learning and optimization methods, and their applications to biology, medical imaging and optics.

She was the recipient of many awards for excellence in research and teaching, including the IEEE Signal Processing Society Technical Achievement Award (2013), the IEEE/AESS Fred Nathanson Memorial Radar Award (2014), and the IEEE Kiyo Tomiyasu Award (2016), the Michael Bruno Memorial Award from the Rothschild Foundation, the Weizmann Prize for Exact Sciences, the Wolf Foundation Krill Prize for Excellence in Scientific Research, the Henry Taub Prize for Excellence in Research (twice), the Hershel Rich Innovation Award (three times), the Award for Women with Distinguished Contributions, the Andre and Bella Meyer Lectureship, the Career Development Chair at the Technion, the Muriel & David Jacknow Award for Excellence in Teaching, and the Technion's Award for Excellence in Teaching (two times). She was the recipient of the several best paper awards and best demo awards together with her research students and colleagues including the SIAM outstanding Paper Prize, the UFFC Outstanding Paper Award, the Signal Processing Society Best Paper Award and the IET Circuits, Devices and Systems Premium Award, and was selected as one of the 50 most influential women in Israel and in Asia, and is a highly cited Researcher.

She was a Member of the Young Israel Academy of Science and Humanities and the Israel Committee for Higher Education. She is the Editor in Chief of *Foundations and Trends in Signal Processing*, a Member of the IEEE Sensor Array and Multichannel Technical Committee and serves on several other IEEE committees. In the past, she was a Signal Processing Society Distinguished Lecturer, Member of the IEEE Signal Processing Theory and Methods and Bio Imaging Signal Processing technical committees, and was an Associate Editor for the IEEE TRANSACTIONS ON SIGNAL PROCESSING, the *EURASIP Journal of Signal Processing*, the *SIAM Journal on Matrix Analysis and Applications*, and the *SIAM Journal on Imaging Sciences*. She was the Co-Chair and Technical Co-Chair of several international conferences and workshops. She was a Horev Fellow of the Leaders in Science and Technology program at the Technion and an Alon Fellow. She is a Member of the Israel Academy of Sciences and Humanities (elected 2017), and a EURASIP Fellow.

Andrea J. Goldsmith (Fellow, IEEE) received the B.S., M.S., and Ph.D. degrees in electrical engineering from the University of California, Berkeley, CA, USA. She is currently the Dean of engineering and applied science and the Arthur LeGrand Doty Professor of electrical and computer engineering with Princeton University, Princeton, NJ, USA. She was previously the Stephen Harris Professor of engineering and Professor of electrical engineering with Stanford University, Stanford, CA, USA, where she is also a Harris Professor Emerita. She is author of the book *Wireless Communications,* and coauthor of the books *MIMO Wireless Communications*, *Principles of Cognitive Radio*, and *Machine Learning and Wireless Communications*, all published by Cambridge University Press, and also an Inventor on 29 patents. Her research interests include information theory, communication theory, and signal processing, and their application to wireless communications, interconnected systems, and biomedical devices. She founded and was the Chief Technical Officer of Plume WiFi (formerly Accelera, Inc.) and of Quantenna (QTNA), Inc, and serves on the Board of Directors for Intel (INTC), Medtronic (MDT), Crown Castle Inc (CCI), and the Marconi Society. She also serves on the Presidential Council of Advisors on Science and Technology (PCAST). She was the recipient of several awards for her work, including the Marconi Prize, the IEEE Sumner Technical Field Award, the ACM Athena Lecturer Award, the ComSoc Armstrong Technical Achievement Award, the Kirchmayer Graduate Teaching Award, the WICE Mentoring Award, and the Silicon Valley/San Jose Business Journal's Women of Influence Award. She is a Member of the National Academy of Engineering and the American Academy of Arts and Sciences. Dr. Goldsmith is currently the founding Chair of the IEEE Board of Directors Committee on Diversity, Inclusion, and Ethics and was the founding Chair of the IEEE TAB Committee on Diversity and Inclusion. She was a President of the IEEE Information Theory Society in 2009, as founding Chair of its Student Committee, and as founding Editor-in-Chief of the IEEE JOURNAL ON SELECTED AREAS OF INFORMATION THEORY. She has also served on the Board of Governors for both the IEEE Information Theory and Communications Societies. At Stanford, she was the Chair of Stanford's Faculty Senate and for multiple terms as a Senator, and on its Academic Council Advisory Board, Budget Group, Committee on Research, Planning and Policy Board, Commissions on Graduate and on Undergraduate Education, Faculty Women's Forum Steering Committee, and Task Force on Women and Leadership.