KALMANNET: DATA-DRIVEN KALMAN FILTERING

Guy Revach, Nir Shlezinger, Ruud J. G. van Sloun, and Yonina C. Eldar

ABSTRACT

The Kalman filter (KF) is a celebrated signal processing algorithm, implementing optimal state estimation of dynamical systems that are well represented by a linear Gaussian state-space model. The KF is model-based, and therefore relies on full and accurate knowledge of the underlying model. We present KalmanNet, a hybrid data-driven/model-based filter that does not require full knowledge of the underlying model parameters. KalmanNet is inspired by the classical KF flow and implemented by integrating a dedicated and compact neural network for the Kalman gain computation. We present an offline training method, and numerically illustrate that KalmanNet can achieve optimal performance without full knowledge of the model parameters. We demonstrate that when facing inaccurate parameters KalmanNet learns to achieve notably improved performance compared to KF.

Index Terms— Kalman filter, deep learning, model-based.

1. INTRODUCTION

State estimation of a dynamical system in discrete-time is one of the most fundamental problems in signal processing, as it arises in many real-world applications such as tracking and localization. In his pioneering work from 1960 [1], Rudolf E. Kalman introduced the Kalman filter (KF), which is a recursive and minimum mean-squared error (MMSE) optimal filtering algorithm for the linear Gaussian state-space (SS) model and is still one of the most widely used algorithms in signal processing. KF is a model-based (MB) algorithm; namely, its performance critically depends on accurate knowledge of the underlying SS model. In practice, the model parameters may be difficult to obtain accurately, and is often only a crude approximation of complex dynamics, degrading the performance of the algorithm. This gives rise to the need to enable the KF to operate reliably without accurate knowledge of the SS model, possibly using data to fill the gap.

This project has received funding from the European Union's Horizon 2020 research and innovation program under grant no. 646804-ERC-COG-BNYQ, and from the Israel Science Foundation under grant No. 0100101. G. Revach is with the Signal Processing Laboratory (ISI), Department of Information Technology and Electrical Engineering, ETH Zurich, Switzerland (e-mail: revach@isi.ee.ethz.ch). N. Shlezinger is with the School of ECE, Ben-Gurion University of the Negev, Beer Sheva, Israel (e-mail: nirshl@bgu.ac.il). R. J. G. van Sloun is with the EE Dpt., Eindhoven University of Technology, and with Phillips Research, Eindhoven, The Netherlands (e-mail: r.j.g.v.sloun@tue.nl). Y. C. Eldar is with the Faculty of Math and CS, Weizmann Institute of Science, Rehovot, Israel (e-mail: yonina.eldar@weizmann.ac.il).

Nowadays, massive quantities of data are available for many applications, and recent years have witnessed a dramatic growth in data-driven (DD) research and applications. The recent success of the state of the art deep neural networks (DNNs) are because they can be trained in an end-to-end and model-agnostic manner from a large quantity of data, thereby catching the subtleties of the true generative process [2]. Consequently, a properly trained DNN can replace the need to explicitly characterize the domain of interest, thus allowing optimization and search over complex domains [3]. The application of DNNs to SS models has been considered in several bodies of work. These include model-agnostic approaches, which utilized DNN architectures to carry out the complete estimation task [4–6], and combining DNNs with KF processing [7–9]. In particular, one work [7] used a recurrent neural network (RNN) to learn each of the SS parameters, and then applied the extended KF for state prediction in a pose estimation application. In [9], a graph neural network is applied in parallel to the MB KF to improve its accuracy as a form of neural augmentation. The work [8] proposed a variational auto-encoder system, referred to as deep KF, for counterfactual estimation; i.e., capturing how actions affect observations, which is a fundamentally different task from state estimation typically implemented by the KF. The aforementioned work all used highly parameterized DNNs that require large quantities of data to train, and involve a high computational complexity for training and inference. These requirements constitute major drawbacks when implementing state estimation applications such as tracking and localization on mobile devices; e.g., for use in drones and vehicular systems. Such systems are limited in their computational resources, and are likely to experience temporal variations in the statistical SS model, requiring frequent retraining to maintain accurate operation over long time periods. This motivates the derivation of a hybrid MB/DD implementation of the KF, capable of overcoming model uncertainty by relying on data in addition to domain knowledge while utilizing light DNNs.

In this work we propose KalmanNet, a hybrid DD and MB recursive filter inspired by KF flow that can track the state of a dynamical system. By using labeled data to train our filter offline, we circumvent the need for full and accurate knowledge of the underlying model. Our derivation of KalmanNet builds upon the success of our previous work on hybrid architectures [10–13], which integrated DNNs into symbol detection algorithms. Here, we identify that the Kalman gain (KG) is the critical component in the KF flow that depends on the missing model knowledge. In KalmanNet, this component is

replaced by a compact neural network (NN) of limited complexity that requires relatively small data sets for training. We formulate a training scheme to minimize the mean-squared error (MSE) loss. Our numerical study demonstrates that KalmanNet achieves the MMSE when facing missing model parameters, and outperforms the KF in the presence of inaccurate SS parameters.

The rest of this paper is organized as follows: Section 2 presents the system model and recalls the basics of Kalman filtering. Section 3 details the proposed KalmanNet. Numerical examples are presented in Section 4.

2. SYSTEM MODEL

2.1. Problem Formulation

We consider a linear Gaussian and partially observable continuous SS model for a time-invariant and autonomous dynamical system in a discrete time [14, Ch. 7]:

$$\mathbf{x}_t = \mathbf{F}\cdot\mathbf{x}_{t-1} + \mathbf{e}_t, \quad \mathbf{e}_t \sim \mathcal{N}(\mathbf{0},\mathbf{Q}), \quad \mathbf{x}_t \in \mathbb{R}^m, \tag{1a}$$

$$\mathbf{y}_t = \mathbf{H}\cdot\mathbf{x}_t + \mathbf{v}_t, \quad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0},\mathbf{R}), \quad \mathbf{y}_t \in \mathbb{R}^n. \tag{1b}$$

Here, $\mathbf{x}_t$ is the latent state vector of the system in time $t \in \mathbb{Z}$, which is evolved by a state evolution matrix $\mathbf{F}$, and by additive Gaussian process noise $\mathbf{e}_t$ without any external control. $\mathbf{y}_t$ is a state observation vector in time t that is generated from the latent state by a linear transformation $\mathbf{H}$ and by additive Gaussian observation noise $\mathbf{v}_t$. We consider a state estimation problem, where in each t we are given access to a new noisy observation $\mathbf{y}_t$. For given initial conditions $\mathbf{x}_0$, our goal is to track and estimate the current latent state $\mathbf{x}_t$ in an online manner, given the observations available so far; $\mathbf{y}_1, \dots \mathbf{y}_t$. We assume that the transformations $\mathbf{F}$ and $\mathbf{H}$ are known (though we also consider the case in which this knowledge is inaccurate). However, as opposed to the KF, the noise statistics $\mathbf{Q}$ and $\mathbf{R}$ are not known. Instead, we have access to a labeled data set comprised of a sequence of observations and their corresponding states. Since our architecture is inspired by the structure of the classical KF, we review it in the following.

2.2. Kalman Filter

KF is a online recursive linear MMSE (LMMSE) filter that is also MMSE optimal for the SS model (1). In every time step t, KF estimates $\mathbf{x}_t$ using only the new observation $\mathbf{y}_t$ and the previous estimate $\hat{\mathbf{x}}_{t-1}$, with constant computational complexity. KF can be described by a two-step procedure: *prediction* and *update*. In the *prediction* step, the *a priori* first and second order moments are obtained by computing:

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F}\cdot\hat{\mathbf{x}}_{t-1}, \qquad \boldsymbol{\Sigma}_{t|t-1} = \mathbf{F}\cdot\boldsymbol{\Sigma}_{t-1}\cdot\mathbf{F}^\top + \mathbf{Q},$$

$$\hat{\mathbf{y}}_{t|t-1} = \mathbf{H}\cdot\hat{\mathbf{x}}_{t|t-1}, \quad \mathbf{S}_{t|t-1} = \mathbf{H}\cdot\boldsymbol{\Sigma}_{t|t-1}\cdot\mathbf{H}^\top + \mathbf{R}.$$

In the *update* step, the *a posteriori* moments are computed given the new observed $\mathbf{y}_t$, via

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t-1} + \boldsymbol{\mathcal{K}}_t\cdot\Delta\mathbf{y}_t, \tag{3a}$$

$$\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_{t|t-1} - \boldsymbol{\mathcal{K}}_t\cdot\mathbf{S}_{t|t-1}\cdot\boldsymbol{\mathcal{K}}_t^\top. \tag{3b}$$

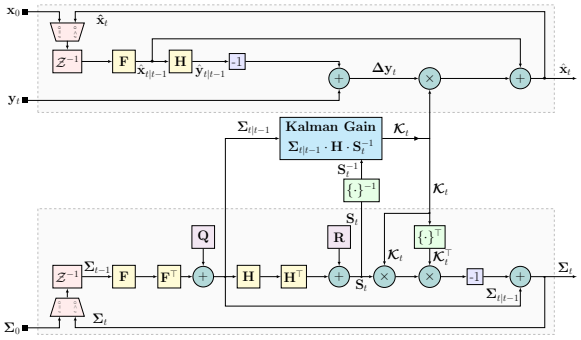

Fig. 1. Kalman filter block diagram. The upper and lower parts correspond to the first and second order moments, respectively. $\{\cdot\}^\top$ denotes transpose and $\mathcal{Z}^{-1}$ is the unit delay.

Here, $\Delta\mathbf{y}_t = \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}$, and $\boldsymbol{\mathcal{K}}_t$ is the KG

$$\boldsymbol{\mathcal{K}}_t = \boldsymbol{\Sigma}_{t|t-1}\cdot\mathbf{H}^\top\cdot\mathbf{S}_{t|t-1}^{-1}. \tag{4}$$

The KF computation is depicted in Fig. 1. The fact that the knowledge of the noise statistics is accounted only for the KG computation (the lower part of Fig. 1) indicates that to reduce the dependence of the KF on this knowledge, one must incorporate DD methods into that computation. This motivates our derivation of KalmanNet in the following.

3. KALMANNET

We present KalmanNet, a hybrid DD/MB system that learns to implement Kalman filtering from partial domain knowledge and from data. We exploit the structure of the SS model in (1) to design a recursive filter operating in the same manner as the KF. It uses the previous estimate $\hat{\mathbf{x}}_{t-1}$ as a sufficient statistic with respect to the previous observations, along with the current observation $\mathbf{y}_t$, to recover $\mathbf{x}_t$ in an online fashion.

We assume that the transformation matrices $\mathbf{F}$ and $\mathbf{H}$ are known (maybe inaccurately), but the noise covariance matrices $\mathbf{Q}$ and $\mathbf{R}$ are not known at all. Instead we have access to labeled training data that were generated from the true SS model. Our design process starts by identifying the specific computations that are based on unavailable knowledge. It can be seen from Fig. 1 and from (4) that these missing statistical moments are used only for computing the KG[1]. Therefore, we design KalmanNet to learn the KG, rather than the complete state estimation task. To design a NN to learn how to compute the KG as part of an overall Kalman filtering flow, one must answer three key questions: (i) from what input signals will the network learn the KG? (ii) what should be the architecture of such a network? and (iii) how will this NN be trained from data? In KF, KG in (4) does not depend on the current observation $\mathbf{y}_t$, and is not a function of the previous estimate $\hat{\mathbf{x}}_{t-1}$. Nonetheless, as it is dictated by the statistical relationship of the SS model (1), which is encapsulated in these quantities, we provide these signals as input to the NN

[1] KG is a vector if $\mathbf{y}_t$ is scalar, and a matrix otherwise.

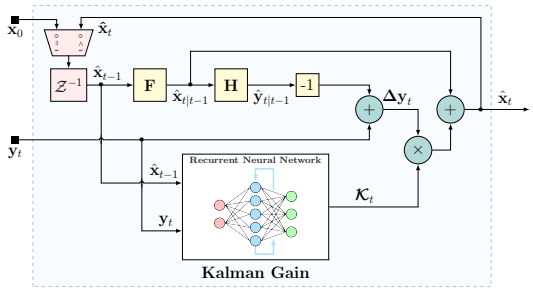

Fig. 2. KalmanNet block diagram.

that computes the KG. The resulting structure that combines the learned KG in the overall KF flow is called KalmanNet, and is illustrated in Fig. 2, where the computation of $\boldsymbol{\mathcal{K}}_t$ in Fig. 1 is replaced with a dedicated NN. In particular, KalmanNet uses the previous posterior estimate $\hat{\mathbf{x}}_{t-1}$ to compute the next prior estimate $\hat{\mathbf{x}}_{t|t-1}$ by a forward pass as in the MB KF. Similarly, it predicts the next observation $\hat{\mathbf{y}}_t$, and computes the innovation value $\Delta\mathbf{y}_t$. State estimation is then carried out via (3), where the KG $\boldsymbol{\mathcal{K}}_t$ is estimated using the NN.

Next, we consider the architecture of the NN that learns the KG. Computing KG in KF involves tracking the second-order statistical moment $\boldsymbol{\Sigma}_t$, which is used merely for producing the KG. This indicates that the learned computation of the KG should involve some internal memory element; e.g., gated recurrent units (GRUs) [15], to be able to carry out the KG computation of KF in a DD fashion. The resulting NN architecture, illustrated in Fig. 3, is composed of three main layers. The input layer, a fully connected layer, takes as input both $\hat{\mathbf{x}}_{t-1}$ and $\mathbf{y}_t$, and maps it into higher dimensions, where the resulting dimensions are linearly proportional to the original input dimensions. The middle layer is implemented as a GRU, providing the necessary capacity for implicit learning of the unknown second-order statistical moments enabling the explicit learning of the KG. The output of a GRU layer is its hidden state, denoted $\mathbf{h}_t$. The dimensions of this layer are linearly proportional to those of the covariance matrices. The output layer is implemented as a fully connected layer, and its role is to restructure the output to the dimensions of the KG.

KalmanNet is trained offline using labeled data. The data set is composed of N length T trajectories, denoted $\{\mathbf{Y}_i, \mathbf{X}_i\}_1^N$, which start from $\mathbf{x}_0 = \mathbf{0}$. Here,

$$\mathbf{Y}_i = \left[\mathbf{y}_1^{(i)}, \ldots, \mathbf{y}_T^{(i)}\right] \qquad \mathbf{X}_i = \left[\mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_T^{(i)}\right]$$

are sequences of observation and true hidden state vectors, respectively, from trajectory i. We unroll the recurrence to T time steps and get a sequence-to-sequence supervised training scheme. For each trajectory i we use the empirical MSE loss measure

$$\ell_i\left(\boldsymbol{\Theta}\right) = \frac{1}{T}\sum_{t=1}^{T}\left\|\Psi_{\boldsymbol{\Theta}}\left(\hat{\mathbf{x}}_{t-1}^{(i)}, \mathbf{y}_t^{(i)}\right) - \mathbf{x}_t^{(i)}\right\|^2,$$

where, $\Psi_{\boldsymbol{\Theta}}\left(\cdot\right)$ is the output of KalmanNet and $\boldsymbol{\Theta}$ is its trainable parameters. To optimize $\boldsymbol{\Theta}$, we use a variant of mini-batch stochastic gradient descent in which for every batch indexed by k, we randomly choose $M < N$ trajectories indexed $i_1^k, \ldots, i_M^k$ from the training set, computing the batch loss as

$$\mathcal{L}_k\left(\boldsymbol{\Theta}\right) = \frac{1}{M}\sum_{j=1}^{M}\ell_{i_j^k}\left(\boldsymbol{\Theta}\right).$$

In Section 4 we numerically demonstrate that this training scheme allows KalmanNet to implement the KF with arbitrary initial conditions $\mathbf{x}_0$ and arbitrary sequence lengths T.

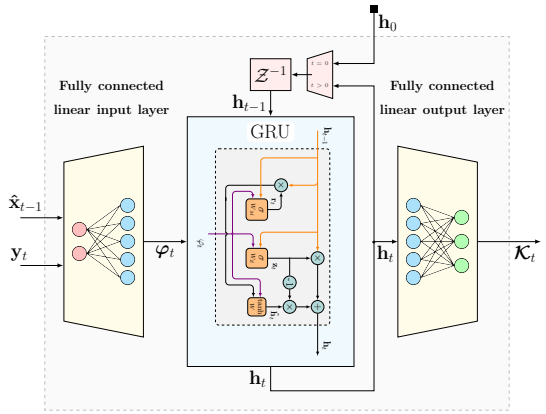

Fig. 3. Kalman gain neural network block diagram.

Discussion. We design KalmanNet as a hybrid DD/MB architecture that combines deep learning with the classic KF. Identifying the specific noise-model-dependent computations of the KF and replacing them with a dedicated NN enables simpler training compared to learning an end-to-end state estimation task. This facilitates the use of relatively compact NN, with dimensions linearly proportional to the SS model, which can be trained with a small data set, as observed in Section 4. The computation complexity for each time step t is also linear w.r.t NN dimensions and does not involve matrix inversion. This implies that KalmanNet is a good candidate to apply for high dimensional SS model and on computationally limited devices.

KalmanNet is derived for state estimation in linear SS models when noise statistics are not specified explicitly. The classical strategy to implement the KF in such setups is to use the data to estimate $\mathbf{Q}$ and $\mathbf{R}$, and utilize these estimates to compute the KG. As covariance estimation can be a challenging task when dealing with high-dimensional signals, KalmanNet bypasses this need by directly learning the KG while utilizing deep learning. KalmanNet is more robust to model mismatch than the classical KF, as demonstrated in Section 4. In particular, KF is sensitive to inaccuracies in $\mathbf{F}$ and $\mathbf{H}$, while KalmanNet can overcome such uncertainty by learning an alternative KG that yields accurate estimation.

Our design of KalmanNet, which is based on a linear time-invariant SS model with known $\mathbf{F}$ and $\mathbf{H}$ gives rise to many interesting future extensions. An obvious extension to KalmanNet is to operate with knowing $\mathbf{F}$ and $\mathbf{H}$. Also, as

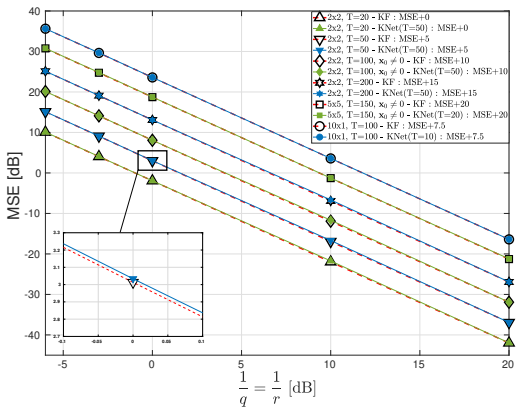

Fig. 4. MSE vs inverse noise variance with accurate SS knowledge. We add an MSE offset to avoid overlapping.

the SS model is linear, we do not fully exploit the model-agnostic nature of NNs and their ability to learn accurately in complex environments [3]. An additional extension is thus state estimation for non-linear models. While various extensions of KF have been proposed [14, Ch. 7], they do not share the MMSE optimality of the classical KF. We also expect KalmanNet to be applicable in non-Gaussian noise models, e.g., for measurements with outliers, adapting KG to the measurements. Finally, one can extend KalmanNet to time varying models by incorporating an online learning scheme to adapt to changes in the stochastic processes.

4. NUMERICAL EVALUATIONS

We present numerical evaluations of the proposed KalmanNet and compare its performance to KF [16]. We show that when using the true $\mathbf{F}$ and $\mathbf{H}$, KalmanNet is empirically optimal, without explicit access to the noise statics $\mathbf{Q}$ and $\mathbf{R}$. We then show that KalmanNet is more robust to inaccurate model knowledge compared to the KF with the same inaccuracies.

In the following, the training and test data are generated from the same SS model with dimensions $m \times n$, where m and n are the size of $\mathbf{x}$ and $\mathbf{y}$ respectively. KalmanNet are trained using the ADAM optimizer [17]. We set $\mathbf{F}$ and $\mathbf{H}$ to take the controllable canonical and inverse canonical forms, respectively, and use diagonal covariance matrices $\mathbf{Q} = q^2 \cdot \mathbf{I}$ and $\mathbf{R} = r^2 \cdot \mathbf{I}$ for the process noise and the observation noise, respectively.

In the simulations presented in Fig. 4 the design parameters that were plugged in to KF and KalmanNet are equal to these of the data model. To show that KalmanNet is invariant to the training set trajectory length, we tested KalmanNet on trajectories with different lengths. The training trajectories were generated from initial conditions $\mathbf{x}_0 = \mathbf{0}$, and some of the testing sets with $\mathbf{x}_0 \neq \mathbf{0}$. Fig. 4 demonstrates that KalmanNet achieves the optimal MSE lower bound; i.e., its MSE coincides with the MSE of KF w.r.t the gain of the observation noise r. We report our findings for the case where q is equal to r, but we observed similar results when $q \neq r$.

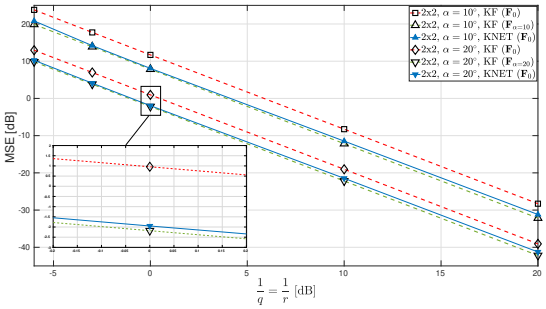

Fig. 5. MSE vs inverse noise variance with inaccurate SS knowledge. We add an MSE offset to avoid overlapping.

The results demonstrate that KalmanNet is not tailored to the actual trajectory presented during training (i.e., $\mathbf{x}_0$ and T); it learns how to do Kalman filtering in the same manner as KF, where its dependency is only in the SS model.

The simulations presented in Fig. 5 demonstrate that for 2×2 SS model KalmanNet outperforms KF when both face inaccurate model knowledge. We plugged in $\mathbf{F}_0$ as a design parameter for both filters, but evaluate with testing data that were generated from SS model based on rotated matrix $\mathbf{F}_{\alpha^\circ}$:

$$\mathbf{F}_{\alpha^\circ} = \mathbf{A}\left(\alpha^\circ\right) \cdot \mathbf{F}_0, \qquad \mathbf{A}\left(\alpha^\circ\right) = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}.$$

KalmanNet achieves a 3 [dB] gain over the KF. Furthermore, despite the fact that KalmanNet implements the KF with inaccurate $\mathbf{F}$, it learns from its data to apply an alternative KG, resulting in an overall MSE within a minor gap from the MMSE; i.e., from the KF with the true $\mathbf{F}_{\alpha^\circ}$ plugged in. Such scenarios represent a setup in which the analytical approximation of the SS model used as a plugged-in parameter in the KF flow differs from the true model. The results demonstrate that KalmanNet is more robust to model uncertainty than KF by extending its flow to operate in a hybrid DD/MB fashion.

For our first experiment KalmanNet was trained till convergence over merely 50 training mini-batches of 100 trajectories, i.e., using $N = 5,000$ i.i.d. trajectories from the same SS model as the test data. For our second experiment we used 300 training mini-batches. Since KalmanNet is invariant to the trajectory length in the training set we can train it using short trajectories, saving data and training time.

5. CONCLUSIONS

We proposed KalmanNet, which is a DD implementation of the KF. KalmanNet is designed by integrating a dedicated NN for the purpose of learning to compute the KG into the KF flow. The resulting hybrid MB/DD implements the KF without prior knowledge of the statistics of the noise signals in the SS model. Our numerical evaluations demonstrate that KalmanNet achieves the MMSE when properly trained, and that it is notably more robust to inaccurate knowledge of the SS parameters compared to the MB KF.

6. REFERENCES

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[3] Y. Bengio, "Learning deep architectures for AI," *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009.

[4] S. S. Rangapuram, M. W. Seeger, J. Gasthaus, L. Stella, Y. Wang, and T. Januschowski, "Deep state space models for time series forecasting," in *Advances in Neural Information Processing Systems (NIPS)*, 2018, pp. 7785–7794.

[5] X. Zheng, M. Zaheer, A. Ahmed, Y. Wang, E. P. Xing, and A. J. Smola, "State space LSTM models with particle MCMC inference," *preprint arXiv:1711.11179*, 2017.

[6] M. Karl, M. Soelch, J. Bayer, and P. Van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," *preprint arXiv:1605.06432*, 2016.

[7] H. Coskun, F. Achilles, R. DiPietro, N. Navab, and F. Tombari, "Long short-term memory Kalman filters: Recurrent neural estimators for pose regularization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 5524–5532.

[8] R. G. Krishnan, U. Shalit, and D. Sontag, "Deep Kalman filters," *preprint arXiv:1511.05121*, 2015.

[9] V. G. Satorras, Z. Akata, and M. Welling, "Combining generative and discriminative models for hybrid inference," in *Advances in Neural Information Processing Systems*, 2019, pp. 13 802–13 812.

[10] 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, 2020.

[11] N. Shlezinger, R. Fu, and Y. C. Eldar, "DeepSIC: Deep soft interference cancellation for multiuser MIMO detection," *IEEE Trans. Wireless Commun.*, early access, 2020.

[12] N. Shlezinger, N. Farsad, Y. C. Eldar, and A. J. Goldsmith, "Data-driven factor graphs for deep symbol detection," *preprint arXiv:2002.00758*, 2020.

[13] N. Farsad, N. Shlezinger, A. J. Goldsmith, and Y. C. Eldar, "Data-driven symbol detection via model-based machine learning," *preprint arXiv:2002.07806*, 2020.

[14] S. S. Haykin, *Adaptive Filter Theory*. Pearson Education India, 2005.

[15] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *preprint arXiv:1412.3555*, 2014.

[16] *KalmanNet source code*, 2020, http://people.ee.ethz.ch/~grevach/kalmannet.html, version 1.0.

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