Unsupervised Learned Kalman Filtering

Guy Revach, Nir Shlezinger, Timur Locher, Xiaoyong Ni, Ruud J. G. van Sloun, and Yonina C. Eldar

***Abstract*—In this paper we adapt KalmanNet, which is a recently proposed deep neural network (DNN)-aided system whose architecture follows the operation of the model-based Kalman filter (KF), to learn its mapping in an unsupervised manner, i.e., without requiring ground-truth states. The unsupervised adaptation is achieved by exploiting the hybrid model-based/data-driven architecture of KalmanNet, which internally predicts the next observation as the KF does. These internal features are then used to compute the loss rather than the state estimate at the output of the system. With the capability of unsupervised learning, one can use KalmanNet not only to track the hidden state, but also to adapt to variations in the state space (SS) model. We numerically demonstrate that when the noise statistics are unknown, unsupervised KalmanNet achieves a similar performance to KalmanNet with supervised learning. We also show that we can adapt a pre-trained KalmanNet to changing SS models without providing additional data thanks to the unsupervised capabilities.**

***Index Terms*—** Kalman filter, unsupervised learning.

I. Introduction

Real-time tracking of hidden state sequences from noisy observations plays a major role in many signal processing systems. Classic approaches are based on the Kalman filter (KF) [1] and its variants [2, Ch. 10]. These model-based (MB) techniques rely on accurate knowledge of an underlying statistical state space (SS) model capturing the system dynamics, which may not be available in some applications, and tend to notably degrade in the presence of model mismatch. To cope with missing model parameters, data is commonly used for parameter estimation, followed by plugging in the missing parameters into the MB KF and its variants [3], [4].

The unprecedented success of deep learning has spurred a multitude of deep neural networks (DNNs) based approaches for SS model related tasks, that are optimized in an end-to-end manner. This allows to achieve improved accuracy compared with MB algorithms when applied in complex, poorly understood, and partially known dynamics, by learning to carry out the task directly from data. Notable approaches include DNN feature extractors [5], variational inference techniques [6]–[11], and the usage of recurrent neural networks (RNNs) [12]–[15]. When the SS model is partially known, one can benefit from the available knowledge by using the hybrid MB/data-driven (DD) KalmanNet architecture proposed in [16] for the real-time filtering task, as a learned KF via MB deep learning [17], [18].

A key challenge in applying an end-to-end DNN-based filters, stems from their need to be trained using labeled data i.e., a large volume of pairs of noisy measurements and their corresponding ground-truth hidden state sequences from the underlying SS model. Obtaining such ground-truth sequences may be costly, particularly in setups where the underlying dynamics, i.e., the SS model, change over time. Previous works on DNN-based for SS model related tasks in the unsupervised setup, focused mostly on the imputation task of filling in missing observations [7]–[10]. This task notably differs from real-time state estimations, also known as filtering [2, Ch. 4].

In this work we extend KalmanNet [16], to learn its mapping in an unsupervised fashion by building upon its interpretable hybrid architecture, which learns to implement the KF while preserving its structure. Specifically, we define a loss measure that uses the noisy observations and their predictions taken from an internal feature of KalmanNet. We also propose a semi-supervised training method, which first trains KalmanNet offline, and then adapts in an unsupervised online manner to dynamics that differ from the offline trained model without providing ground-truth data. This mechanism results in KalmanNet tracking not only the latent state, but also changes in the underlying SS model. Our numerical evaluations demonstrate that the unsupervised KalmanNet that does not have access to the noise statistics, approaches the KF with full domain knowledge. Furthermore, its semi-supervised implementation allows to improve upon on its supervised counterpart due to the new ability to track variations in the SS model without requiring additional data.

The rest of this paper is organized as follows: Section II formulates the SS model and the problem. Section III presents unsupervised KalmanNet, which is evaluated in Section IV.

G. Revach. T. Locher, and X. Ni are with the Institute for Signal and Information Processing (ISI), D-ITET, ETH Zürich, Switzerland (e-mail: {grevach; tlocher}@ethz.ch, xiaoni@student.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@weizmann.ac.il). The authors thank Prof. Hans-Andrea Loeliger for his helpful comments and discussion.

II. System Model and Preliminaries

In this section we review the SS model and briefly recall the supervised KalmanNet. For simplicity, we focus on linear SS models, though the derivations can also be used for non-linear models in the same manner as the extended KF is applied [2, Ch. 10], as we demonstrate in Section IV.

A. Problem Formulation

We consider state estimation in discrete-time, linear, Gaussian SS models. Letting $\mathbf{x}_t$ denote the $m \times 1$ hidden state vector at time instance $t \in \mathbb{Z}$, which evolves in time via

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

Here, $\mathbf{F}$ is the $m \times m$ state evolution matrix, while $\mathbf{w}_t$ is additive white Gaussian noise (AWGN) with covariance $\mathbf{Q}$. The corresponding observation $\mathbf{y}_t$, is related to $\mathbf{x}_t$ via

$$\mathbf{y}_t = \mathbf{H} \cdot \mathbf{x}_t + \mathbf{v}_t, \qquad \mathbf{v}_t \sim \mathcal{N}(\mathbf{0}, \mathbf{R}), \quad \mathbf{y}_t \in \mathbb{R}^n, \tag{2}$$

where $\mathbf{H}$ is the $n \times m$ measurement matrix, and $\mathbf{v}_t$ is an AWGN with covariance $\mathbf{R}$. We focus on the filtering problem, where one needs to track the hidden state $\mathbf{x}_t$ from a known initial state $\mathbf{x}_0$. At each time instance t, the goal is to provide an instantaneous estimate $\hat{\mathbf{x}}_t$, based on the observations seen so far $\{\mathbf{y}_\tau\}_{\tau=1}^{t}$. We consider scenarios where one has partial domain knowledge, such that the statistics of the noises $\mathbf{w}_t$ and $\mathbf{v}_t$ are not known, while the matrices $\mathbf{F}$ and $\mathbf{H}$ are known. To fill the information gap we assume that we have access to an *unlabeled* data set containing a sequence of observations from which one has to learn to recover the hidden state.

B. Supervised KalmanNet

KalmanNet is a hybrid MB/DD implementation of the KF. The latter utilizes full knowledge of the SS model to estimate $\mathbf{x}_t$, based on the current observed $\mathbf{y}_t$ and the previous estimate $\hat{\mathbf{x}}_{t-1}$. This

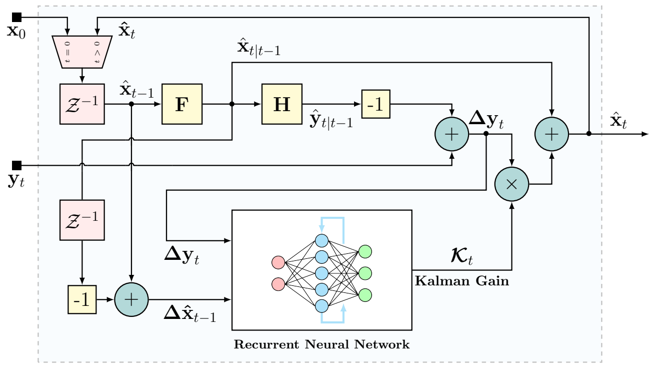

Fig. 1: KalmanNet block diagram.

is achieved by first predicting the next state and observation based solely on the previous estimate via

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{F} \cdot \hat{\mathbf{x}}_{t-1}, \tag{3a}$$

$$\hat{\mathbf{y}}_{t|t-1} = \mathbf{H} \cdot \hat{\mathbf{x}}_{t|t-1}, \tag{3b}$$

while computing the second-order moments of these estimates as $\mathbf{\Sigma}_{t|t-1} = \mathbf{F} \cdot \mathbf{\Sigma}_{t-1} \cdot \mathbf{F}^\top + \mathbf{Q}$, and $\mathbf{S}_{t|t-1} = \mathbf{H} \cdot \mathbf{\Sigma}_{t|t-1} \cdot \mathbf{H}^\top + \mathbf{R}$. Next, the KF computes the Kalman gain (KG) $\mathcal{K}_t$ as $\mathcal{K}_t = \mathbf{\Sigma}_{t|t-1} \cdot \mathbf{H}^\top \cdot \mathbf{S}^{-1}_{t|t-1}$, which is used to update the estimation covariance $\mathbf{\Sigma}_t = \mathbf{\Sigma}_{t|t-1} - \mathcal{K}_t \cdot \mathbf{S}_{t|t-1} \cdot \mathcal{K}_t^\top$, and provide the state estimate $\hat{\mathbf{x}}_t$ via

$$\hat{\mathbf{x}}_t = \hat{\mathbf{x}}_{t|t-1} + \mathcal{K}_t \cdot \Delta\mathbf{y}_t, \tag{4}$$

where $\Delta\mathbf{y}_t$ is the innovation process computed as

$$\Delta\mathbf{y}_t = \mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}. \tag{5}$$

KalmanNet learns to implement the KF from labeled data in partially known SS models. This is achieved by noting that the available knowledge allows to compute the predictions in (3), while the missing domain knowledge is needed to compute the KG. Thus, KalmanNet augments the flow of the KF with an RNN, which estimates the KG and implicitly tracks the second-order moments computed by the KF, while the state estimate is obtained via (4) (see [16] for a detailed description). The KalmanNet architecture, depicted in Fig. 1, is trained to estimate the state in a supervised manner based on labeled data. The data set comprises N pairs of hidden state trajectories and its corresponding observations of the form $\mathcal{D} = \{(\mathbf{Y}_i, \mathbf{X}_i)\}_{i=1}^N$, where

$$\mathbf{Y}_i = \left[\mathbf{y}_1^{(i)}, \ldots, \mathbf{y}_{T_i}^{(i)}\right], \quad \mathbf{X}_i = \left[\mathbf{x}_0^{(i)}, \mathbf{x}_1^{(i)}, \ldots, \mathbf{x}_{T_i}^{(i)}\right], \tag{6}$$

and T_i is the length of the ith training trajectory. The training procedure aims at minimizing the regularized ℓ_2 loss. Letting $\mathbf{\Theta}$ be the trainable parameters of the RNN and $\hat{\mathbf{x}}_t^{(i)}(\mathbf{\Theta})$ be the output of KalmanNet with parameters $\mathbf{\Theta}$ at time t applied to $\mathbf{Y}_i$, the loss is computed for the ith trajectory as

$$l_i(\mathbf{\Theta}) = \frac{1}{T_i}\sum_{t=1}^{T_i}\left\|\hat{\mathbf{x}}_t^{(i)}(\mathbf{\Theta}) - \mathbf{x}_t^{(i)}\right\|^2 + \gamma \cdot \|\mathbf{\Theta}\|^2, \tag{7}$$

where $\gamma > 0$ is a regularization coefficient. The loss measure (7) is used to optimize $\mathbf{\Theta}$ via stochastic gradient descent (SGD) optimization combined with the backpropagation through time (BPTT) algorithm [19].

III. Unsupervised KalmanNet

A. Unsupervised Training Algorithm

KalmanNet, described in Subsection II-B, as well as other previously proposed DNN-based state estimators such as [14], are designed to estimate $\mathbf{x}_t$, and thus are trained so that the output approaches the ground-truth hidden state sequence. KalmanNet admits an interpretable architecture owing to its hybrid MB/DD design, which preserves the flow of the KF. We exploit this fact to propose a training algorithm for KalmanNet that does not rely on ground-truth labels.

Unsupervised loss: KalmanNet uses its state estimates to predict the next observation via (3b) as an internal feature. While the accuracy of this prediction, e.g., the squared magnitude of the innovation process (5), depends on the accuracy in estimating $\mathbf{x}_t$ and the observation noise, (5) can be computed based solely on the observed sequence. Consequently, one can adapt KalmanNet in an unsupervised manner by training it to minimize $\|\Delta\mathbf{y}_t\|^2$. This quantity can be used to compute the gradient with respect to the parameters of the RNN, which outputs the learned KG, by the derivative chain rule. Indeed,

$$\begin{aligned}\frac{\partial\|\Delta\mathbf{y}_t\|^2}{\partial\mathcal{K}_{t-1}} &\overset{(a)}{=} \frac{\partial}{\partial\mathcal{K}_{t-1}}\left\|\mathbf{H}\cdot\mathbf{F}\cdot\mathcal{K}_{t-1}\cdot\Delta\mathbf{y}_{t-1} - \Delta\mathbf{y}_t^-\right\|^2 \\ &= 2\cdot\mathbf{H}^\top\cdot\mathbf{F}^\top\cdot\left(\mathcal{K}_{t-1}\cdot\Delta\mathbf{y}_{t-1} - \Delta\mathbf{y}_t^-\right)\cdot\Delta\mathbf{y}_{t-1}^\top,\end{aligned} \tag{8}$$

where $\Delta\mathbf{y}_t^- \triangleq \mathbf{y}_t - \mathbf{H}\cdot\mathbf{F}\cdot\hat{\mathbf{x}}_{t-1|t-2}$. In (8), (a) holds since

$$\begin{aligned}\hat{\mathbf{y}}_{t|t-1} &= \mathbf{H}\cdot\mathbf{F}\cdot\hat{\mathbf{x}}_{t-1|t-1} \\ &= \mathbf{H}\cdot\mathbf{F}\cdot\left(\hat{\mathbf{x}}_{t-1|t-2} + \mathcal{K}_{t-1}\cdot\Delta\mathbf{y}_{t-1}\right).\end{aligned} \tag{9}$$

The gradient in (8) indicates that the ℓ_2 norm of the innovation process can be used to learn the computation of the KG, which involves the trainable parameters of KalmanNet $\mathbf{\Theta}$. Similarly to (7), the resulting loss for the ith trajectory is

$$\tilde{l}_i(\mathbf{\Theta}) = \frac{1}{T_i}\sum_{t=1}^{T_i}\left\|\hat{\mathbf{y}}_{t|t-1}^{(i)}(\mathbf{\Theta}) - \mathbf{y}_t^{(i)}\right\|^2 + \gamma\cdot\|\mathbf{\Theta}\|^2. \tag{10}$$

Unsupervised KalmanNet is thus trained using solely observed trajectories based on the loss measure (10) using SGD variants combined with BPTT for gradient computation.

Offline versus online training: The ability to train KalmanNet without providing ground-truth state sequences gives rise to two possible training approaches: a purely unsupervised offline training scheme, and an online semi-supervised strategy. The offline approach follows conventional unsupervised learning using unlabeled data of the form $\tilde{\mathcal{D}} = \{(\mathbf{Y}_i)\}_{i=1}^N$. This data set is used to optimize $\mathbf{\Theta}$ via mini-batch SGD-based optimization, where for every batch indexed by k, we choose $M < N$ trajectories indexed by $i_1^k, \ldots, i_M^k$, computing the mini-batch mean-squared error (MSE) loss as

$$\mathcal{L}_k(\mathbf{\Theta}) = \frac{1}{M}\sum_{j=1}^{M}\tilde{l}_{i_j^k}(\mathbf{\Theta}). \tag{11}$$

Online training builds upon the ability to learn without labels to adapt a pre-trained KalmanNet to dynamics that differ from those used during training. Pretraining can be done using labeled data obtained by mathematical modelling and/or past measurements without altering the architecture of KalmanNet. Then, the deployed model is further adapted in an unsupervised manner using observations acquired during operation to form training trajectories from realizations of the data. Such a training procedure provides KalmanNet with the ability to be adaptive to changes in the distribution of the data. Specifically, once every $\tilde{T}$ time steps, we compute the loss (10) over the last $\tilde{T}$ observations online and optimize the RNN parameters $\mathbf{\Theta}$ accordingly.

B. Discussion

The ability to train KalmanNet in an unsupervised manner without relying on ground-truth sequences, follows directly from the hybrid MB/DD architecture of KalmanNet, where one can identify the observations innovation process as an internal feature and use it to compute the loss. The training procedure does not affect the

$1/\mathbf{r}^2$ [dB]	KF MSE [dB]	KalmanNet MSE [dB]
0	-2.31	-2.27
3	-5.31	-5.26
10	-12.3	-12.3
20	-22.3	-22.3
30	-32.3	-32.3

TABLE I: Generalizing for long trajectories, 2×2, $\nu = 0\,$[dB].

KalmanNet architecture, and one can use the same supervised model designed in [16]. Since the KalmanNet methodology can be extended to carry out offline smoothing in a learned fashion [20], we expect the proposed unsupervised learning approach to be useful for tasks involving state-space inference other than the real-time filtering problem considered here, though we leave this study for future investigation. The proposed online semi-supervised technique allows to adapt a pre-trained KalmanNet state estimator after deployment, coping with setups in which the original training is based on data that does not fully capture the true underlying SS model. This gain, numerically demonstrated in Section IV, shows the potential of MB deep learning in enabling application-oriented, efficient training algorithms. In the current work, we focus on partially-known SS models where $\mathbf{F}$ and $\mathbf{H}$ are available from, e.g., a physical model. While supervised KalmanNet was shown in [16] to operate reliably when using inaccurate approximations of $\mathbf{F}$ and $\mathbf{H}$, we leave such a study in unsupervised setups for future work.

The proposed online semi-supervised technique allows one to adapt a pre-trained KalmanNet state estimator after deployment, coping with setups in which the original training is based on data that does not fully capture the true underlying SS model. This gain, numerically demonstrated in Section IV, bears some similarity to online training mechanisms proposed for hybrid MB/DD communication receivers in [21]–[23]. Despite the similarity, the proposed technique, obtained from the interpretable operation of KalmanNet, is fundamentally different from that proposed in [21]–[23], where structures in communication data were exploited to generate confident labels from decisions. Nonetheless, both the current work and [21]–[23] demonstrate the potential of MB deep learning in enabling application-oriented, efficient training algorithms.

IV. Numerical Evaluations

In this section we numerically[1] evaluate unsupervised KalmanNet on a linear SS model and on the non-linear Lorenz attractor model, and compare it to the KF and extended KF.

In the linear setup, $\mathbf{F}$ and $\mathbf{H}$ take a canonical form, and $\mathbf{Q}$ and $\mathbf{R}$ are the diagonal matrices $\mathbf{q}^2\cdot\mathbf{I}_m$ and $\mathbf{r}^2\cdot\mathbf{I}_n$, respectively, while defining

$$\nu \triangleq \frac{\mathbf{q}^2}{\mathbf{r}^2}.$$

In Fig. 2 we compare the performance of unsupervised KalmanNet to the MB KF, which achieves the minimum mean-squared error (MMSE) here, for 2×2 and 5×5 SS models and trajectory length $T = 80$. We observe in Fig. 2 that the *offline* trained unsupervised KalmanNet learns to achieve the MMSE lower bound.

Next, the previously trained model is evaluated on a longer trajectory length of $T = 10{,}000$. The results reported in Table I show that KalmanNet does not overfit to the trajectory length and that the unsupervised training of KalmanNet is not tailored to the trajectories presented during training, tuning the filter with dependency only on the SS model.

[1]The source code used in our numerical study along with the complete set of hyper-parameters used in each numerical evaluation can be found online at https://github.com/KalmanNet/Unsupervised_EUSIPCO_22.

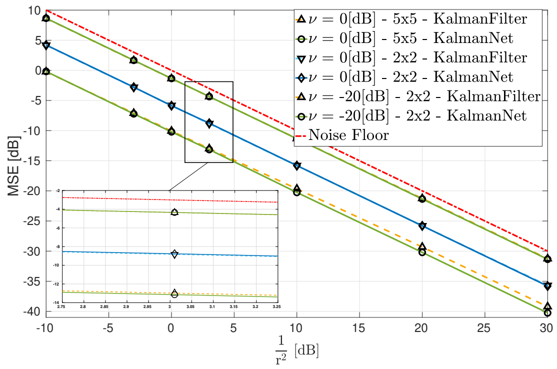

Fig. 2: MSE vs $\frac{1}{\mathbf{r}^2}$ for a 2×2 and a 5×5 linear systems, $T = 80$. An MSE offset is added to prevent overlapping.

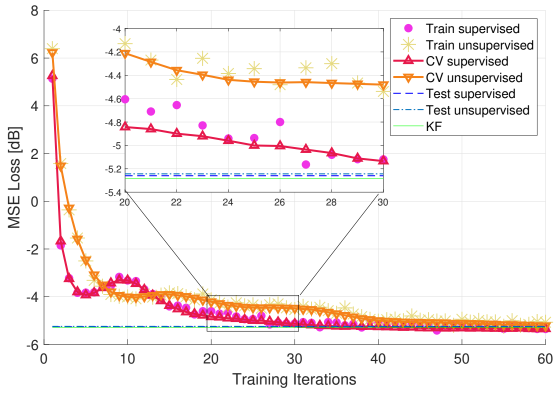

Fig. 3: MSE versus epoch of supervised and unsupervised KalmanNet for a 2×2 linear SS model, $\nu = 0\,$[dB].

The results so far indicate that for the considered SS model, unsupervised training does not degrade the performance of KalmanNet observed for supervised training in [16]. To understand the benefits of supervision, we depict in Fig. 3 the MSE convergence of unsupervised KalmanNet compared with its supervised counterpart. We observe in Fig. 3 that the lack of labeled data in unsupervised KalmanNet and the fact that it is not explicitly encouraged to minimize the state estimation MSE, results in slower convergence to the MMSE compared to supervised KalmanNet.

Next, we train unsupervised KalmanNet for the non-linear SS model of the chaotic Lorenz attractor [24]. Here, the noiseless state-evolution of the continuous-time process $\mathbf{x}_\tau$ with $\tau \in \mathbb{R}^+$ is given by

$$\frac{\partial}{\partial\tau}\mathbf{x}_\tau = \mathbf{A}(\mathbf{x}_\tau)\cdot\mathbf{x}_\tau,\ \ \mathbf{A}(\mathbf{x}_\tau) = \begin{pmatrix} -10 & 10 & 0 \\ 28 & -1 & -x_{1,\tau} \\ 0 & x_{1,\tau} & -\frac{8}{3} \end{pmatrix}. \tag{12}$$

To get a discrete-time state-evolution model, we repeat the steps used in [25]. First, we sample the noiseless process with sampling interval $\Delta\tau$ and assume that $\mathbf{A}(\mathbf{x}_\tau)$ can be kept constant in a small neighborhood of $\mathbf{x}_\tau$; i.e., $\mathbf{A}(\mathbf{x}_\tau) \approx \mathbf{A}(\mathbf{x}_{\tau+\Delta\tau})$. Then, the continuous-time solution of the differential system (12), which is valid in the neighborhood of $\mathbf{x}_\tau$ for a short time interval $\Delta\tau$, is

$$\mathbf{x}_{\tau+\Delta\tau} = \exp\left(\mathbf{A}(\mathbf{x}_\tau)\cdot\Delta\tau\right)\cdot\mathbf{x}_\tau. \tag{13}$$

Finally, we take the Taylor series expansion of (13) and a *finite* series

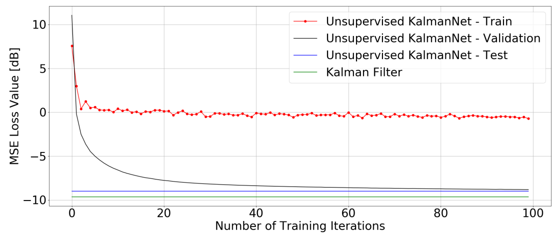

Fig. 4: Learning curve of unsupervised KalmanNet, Lorenz attractor. $\mathbf{r}^2 = 0\,[\text{dB}]$, $\mathbf{q}^2 = 0\,[\text{dB}]$, $T = 100$.

approximation (with J coefficients), which results in

$$\mathbf{F}(\mathbf{x}_\tau) \triangleq \exp(\mathbf{A}(\mathbf{x}_\tau) \cdot \Delta\tau) \approx \mathbf{I} + \sum_{j=1}^{J} \frac{(\mathbf{A}(\mathbf{x}_\tau) \cdot \Delta\tau)^j}{j!}. \tag{14}$$

The resulting discrete-time evolution process is given by

$$\mathbf{x}_{t+1} = \mathbf{f}(\mathbf{x}_t) = \mathbf{F}(\mathbf{x}_t) \cdot \mathbf{x}_t. \tag{15}$$

The discrete-time state-evolution model in (15), with additional process noise, is used for generating the simulated Lorenz attractor data, using $J = 5$ Taylor order and $\Delta\tau = 0.02$ sampling interval.

In Fig. 4 we can see that we were able to train KalmanNet for this challenging setup. Although the training test is bounded by the observation noise $\mathbf{r}^2 = 0\,[\text{dB}]$, the MSE achieved by unsupervised KalmanNet is within a minor gap of $0.5\,[\text{dB}]$ from the MSE achieved by the extended KF which has full knowledge of the SS model. Furthermore, the DNN-aided KalmanNet is observed to require 0.15 seconds to infer for each trajectory, which is quicker compared to the extended KF, that involves matrix inversions and requires 0.2 seconds per trajectory. This indicates that KalmanNet may be preferable even when one can estimate the noise statistics.

Finally, we evaluate the online training mechanism when the testing distribution differs from the SS model from which the training data is generated. We again consider a linear 2×2 SS model where the true (testing) observation distribution is generated with $\mathbf{r}^2 = 25\,[\text{dB}]$, while the model is pre-trained on data from an SS model with $\mathbf{r}^2 = 10\,[\text{dB}]$. For online adaptation, we train every incoming $\tilde{T} = 10$ samples. In Fig. 5 we can see that KalmanNet smoothly adapts to the test distribution while training on the observed trajectory over multiple time steps. This shows that the proposed training algorithm enables KalmanNet to track variations in the SS model.

V. Conclusions

In this work we proposed an unsupervised training scheme that enables KalmanNet to learn its mapping without requiring ground-truth sequences. The training scheme exploits the interpretable nature of KalmanNet to formulate an unsupervised loss based on an internal feature that predicts the next observation. Our numerical evaluations demonstrate that the proposed unsupervised training allows KalmanNet to approach the MMSE, without access to the noise statistics.

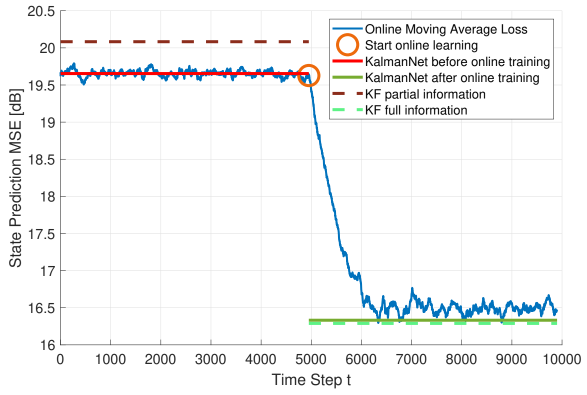

Fig. 5: A pre-trained KalmanNet trained in an online unsupervised manner. 2×2 linear SS model, $\mathbf{q}^2 = 10\,[\text{dB}]$

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] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*. Oxford University Press, 2012.
[3] P. Abbeel, A. Coates, M. Montemerlo, A. Y. Ng, and S. Thrun, "Discriminative training of Kalman filters." in *Robotics: Science and Systems*, vol. 2, 2005, p. 1.
[4] L. Xu and R. Niu, "EKFNet: Learning system noise statistics from measurement data," in *Proc. IEEE ICASSP*, 2021, pp. 4560–4564.
[5] L. Zhou, Z. Luo, T. Shen, J. Zhang, M. Zhen, Y. Yao, T. Fang, and L. Quan, "KFNet: Learning temporal camera relocalization using Kalman filtering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2020, pp. 4919–4928.
[6] R. G. Krishnan, U. Shalit, and D. Sontag, "Deep Kalman filters," *preprint arXiv:1511.05121*, 2015.
[7] 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.
[8] M. Fraccaro, S. D. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," in *Advances in Neural Information Processing Systems*, 2017.
[9] C. Naesseth, S. Linderman, R. Ranganath, and D. Blei, "Variational sequential Monte Carlo," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2018, pp. 968–977.
[10] E. Archer, I. M. Park, L. Buesing, J. Cunningham, and L. Paninski, "Black box variational inference for state space models," *arXiv preprint arXiv:1511.07367*, 2015.
[11] R. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.
[12] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop KF: Learning discriminative deterministic state estimators," in *Advances in Neural Information Processing Systems*, 2016, pp. 4376–4384.
[13] 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.
[14] 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.
[15] P. Becker, H. Pandya, G. Gebhardt, C. Zhao, C. J. Taylor, and G. Neumann, "Recurrent Kalman networks: Factorized inference in high-dimensional deep feature spaces," in *International Conference on Machine Learning*. PMLR, 2019, pp. 544–552.
[16] 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," *IEEE Trans. Signal Process.*, vol. 70, pp. 1532–1547, 2022.
[17] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," *arXiv preprint arXiv:2012.08405*, 2020.
[18] N. Shlezinger, Y. C. Eldar, and S. P. Boyd, "Model-based deep learning: On the intersection of deep learning and optimization," *arXiv preprint arXiv:2205.02640*, 2022.
[19] P. J. Werbos, "Backpropagation through time: What it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
[20] X. Ni, G. Revach, N. Shlezinger, R. J. van Sloun, and Y. C. Eldar, "RTSNET: Deep learning aided Kalman smoothing," in *Proc. IEEE ICASSP*, 2022.
[21] 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.

[22] 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, 2021.

[23] 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.*, 2020.

[24] W. Gilpin, "Chaos as an interpretable benchmark for forecasting and data-driven modelling," *arXiv preprint arXiv:2110.05266*, 2021.

[25] 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.