# Bayesian KalmanNet: Quantifying Uncertainty in Deep Learning Augmented Kalman Filter

Yehonatan Dahan , *Student Member, IEEE*, Guy Revach , *Senior Member, IEEE*, Jindrich Dunik , *Senior Member, IEEE*, and Nir Shlezinger , *Senior Member, IEEE*

*Abstract*—Recent years have witnessed a growing interest in tracking algorithms that augment Kalman filters (KFs) with deep neural networks (DNNs). By transforming KFs into trainable deep learning models, one can learn from data to reliably track a latent state in complex and partially known dynamics. However, unlike classic KFs, conventional DNN-based systems do not naturally provide an uncertainty measure, such as error covariance, alongside their estimates, which is crucial in various applications that rely on KF-type tracking. This work bridges this gap by studying error covariance extraction in DNN-aided KFs. We begin by characterizing how uncertainty can be extracted from existing DNN-aided algorithms and distinguishing between approaches by their ability to associate internal features with meaningful KF quantities, such as the Kalman gain and prior covariance. We then identify that uncertainty extraction from existing architectures necessitates additional domain knowledge not required for state estimation. Based on this insight, we propose *Bayesian KalmanNet*, a novel DNN-aided KF that integrates Bayesian deep learning techniques with the recently proposed KalmanNet and transforms the KF into a *stochastic* machine learning architecture. This architecture employs sampling techniques to predict error covariance reliably without requiring additional domain knowledge, while retaining KalmanNet's ability to accurately track in partially known dynamics. Our numerical study demonstrates that Bayesian KalmanNet provides accurate and reliable tracking in various scenarios representing partially known dynamic systems.

*Index Terms*—Kalman filter, Bayesian deep learning, uncertainty estimation.

Yehonatan Dahan and Nir Shlezinger are with the School of ECE, Ben-Gurion University of the Negev, Be'er Sheva 8410501, Israel (e-mail: yehonatd@post.bgu.ac.il; nirshl@bgu.ac.il).

Guy Revach is with the Institute for Signal and Information Processing, D-ITET, ETH Zürich, 8092 Zurich, Switzerland (e-mail: grevach@ethz.ch).

Jindrich Dunik is with the Faculty of Applied Science, University of West Bohemia in Pilsen, 301 00 Pelsen, Czech Republic (e-mail: dunikj@kky.zcu.cz).

## I. INTRODUCTION

TRACKING the latent state of a dynamic system is one of the most common tasks in signal processing [2]. Often, one is required not only to track the state, but also to provide a reliable assessment of the uncertainty in its estimate. For instance, navigation systems must not only provide the instantaneous position but also characterize its accuracy [3]. Moreover, uncertainty quantification in state estimation provides a measure of confidence in the predictions and enables informed, risk-aware decision-making, which is particularly critical in safety-sensitive domains. Uncertainty quantification is inherently provided by classic algorithms, such as the Kalman filter (KF) and the extended KF (EKF), that rely on the ability to faithfully describe the dynamics as a closed-form state space (SS) model. Such model-based algorithms track not only the state, but also compute the error covariance [4].

The emergence of deep learning gave rise to a growing interest in using deep neural networks (DNNs) to carry out tasks involving time sequences [5]. Various DNN architectures can learn from data to track dynamic systems without relying on full knowledge of an underlying SS model. These include generic recurrent neural networks (RNNs) [6, Ch. 10] and attention mechanisms [7], as well as RNN-type architectures inspired by SS models [8] and KF processing [9]. Such DNNs are trained end-to-end without using any statistical modeling.

When one has partial knowledge of the underlying dynamics, it can be leveraged by hybrid model-based/data-driven designs [10], [11], [12]. Existing approaches to exploit partial domain knowledge can be roughly divided following the machine learning paradigms of *generative* and *discriminative* learning [13]. Generative approaches are SS *model-oriented*, using DNNs to learn the missing aspects of the underlying SS model, i.e., for system identification, while adopting parameterizations varying from physics-based SS models [14], [15] to fully DNN-based ones [16], [17], [18]. The learned models are then used by classical algorithms, such as the EKF, thus typically preserving the limitations of these methods to, e.g., Gaussian temporally independent noises [19].

The discriminative strategy is *task-oriented*, learning to directly output the state estimate, following the conventional usage of DNNs for classification and regression [6]. Existing designs vary in the knowledge they require on the underlying model. For settings with relatively high level of domain knowledge, e.g., when one can approximately characterize

the dynamics as a linear Gaussian SS model, DNNs can enhance classic KFs by providing correction terms [20] or pre-processing [21] trained alongside the tracking algorithm. In scenarios with only partially known SS models, DNN augmented KFs were proposed, converting the KF into a machine learning algorithm [22], [23], [24], [25], [26], [27], [28], [29]. Doing so bypasses the need to estimate the missing aspects of the SS model, learning to track via end-to-end training. In particular, the KalmanNet architecture [22] and its variants [23], [24], [25], [26] substitute the Kalman gain (KG) computation of the EKF with a DNN. The resulting algorithm was shown to enable accurate tracking in partially known and complex dynamics, while facilitating combination with pre-processing [30] and downstream state post-processing [31], [32], [33]. However, as such methods learn to estimate the state, they do not naturally provide the error covariance, for which there is rarely "ground-truth".

Uncertainty extraction is typically quite challenging in generic deep learning [34]. Even for classification DNNs, that are designed to output a probability mass function, learned probabilities are often not faithful for uncertainty extraction [35], while in estimation tasks, generic DNNs do not quantify uncertainty. For DNN-aided KFs, several specific extensions were suggested to provide error covariance. For instance, the recurrent Kalman network (RKN) proposed in [9] was trained to output an error estimate in addition to the state estimate, while [36] showed that when applying KalmanNet of [22] in linear SS models with full column rank observation matrix and Gaussian measurement noise, the error covariance can be extracted from internal features of the architecture. These specific studies motivate a unifying study on error covariance extraction in DNN-aided KFs.

In this work we study discriminative DNN-aided tracking with uncertainty quantification for both linear and non-linear partially known SS models. We extensively examine the extraction of error covariance from existing architectures. We categorize DNNs that can learn to track in partially known SS model into model-based deep learning designs, where one can relate internal features to the KG or prior covariance, and black-box architectures, where only the output has an operational meaning. We show how the desired error covariance can be extracted in each setting: for black-box DNNs, error covariance estimation necessitates providing additional output neurons; for architectures whose internal features can be attributed to the prior covariance and the KG, we identify how and in which conditions can its internal features be mapped into error covariance prediction. We present two training methods which encourage such architectures to support error covariance estimation.

Our analysis identifies that reliable extraction of the error covariance from existing DNN-aided algorithms is feasible only in some settings, e.g., when tracking fewer state variables than observed ones. Moreover, doing so requires additional knowledge of the SS model that is not required for state tracking. To overcome this, we propose a novel DNN-aided tracking algorithm that supports uncertainty quantification termed *Bayesian KalmanNet*. Our proposed algorithm draws inspiration from Bayesian deep learning [37], [38], [39], deviating from conventional *frequentist* learning to augment an EKF with a *stochastic DNN* to compute the KG. Doing so enables tracking the state with the same level of partial SS modeling needed by conventional KalmanNet and its variants, while providing reliable uncertainty measures by sampling multiple DNN realizations.

We compare Bayesian KalmanNet and our extension of existing DNN-aided KFs for uncertainty extraction in a qualitative and a quantitative manner. For the latter we consider both a synthetic and a physically compliant navigation setting. Our results show that Bayesian KalmanNet consistently provides reliable tracking of the state along with faithful error covariance prediction under different levels of accuracy in knowledge of the dynamics, and that incorporating domain knowledge in model-based deep learning designs systematically yields the most reliable estimate of the error covariance.

The rest of this paper is organized as follows: Section II briefly reviews SS models and the EKF. Extensions of existing DNN-aided KFs for uncertainty extraction are presented in Section III. Bayesian KalmanNet is derived in Section IV, Section V details our numerical study, and Section VI provides concluding remarks.

Throughout this paper, we use boldface lowercase for vectors, e.g., $\mathbf{x}$, and boldface uppercase letters, e.g., $\boldsymbol{M}$, for matrices. The $i$th element of some vector $\mathbf{x}$ is denoted $[\mathbf{x}]_i$. Similarly, $[\boldsymbol{M}]_{i,j}$ is the $(i,j)$th element of a matrix $\boldsymbol{M}$. We use $\mathbb{N}$ and $\mathbb{R}$ for the sets of natural and real numbers, respectively. The operations $(\cdot)^\top$, $\|\cdot\|$, $\mathbb{E}\{\cdot\}$, and $\mathrm{Var}\{\cdot\}$ are used for transpose, $\ell_2$ norm, stochastic expectation, and variance, respectively.

## II. SYSTEM MODEL AND PRELIMINARIES

In this section we review the system model and recall some preliminaries. We first present the SS model in Subsection II-A and discuss how it is used for model-based and DNN-based tracking in Subsections II-B–II-C, respectively. We formulate the problem of uncertainty-aware tracking in Subsection II-D, and review basics in Bayesian deep learning in Subsection II-E.

### A. SS Model

We consider a dynamic system described as a non-linear SS model in discrete time. This model describes the relationship between a latent state vector $\mathbf{x}_t \in \mathbb{R}^m$ and observation vector $\mathbf{y}_t \in \mathbb{R}^n$ for each time instance $t \in \mathbb{N}$. In a non-linear SS model, this relationship is generally represented as

$$\mathbf{x}_t = \mathbf{f}(\mathbf{x}_{t-1}) + \boldsymbol{w}_t, \tag{1a}$$

$$\mathbf{y}_t = \mathbf{h}(\mathbf{x}_t) + \boldsymbol{v}_t. \tag{1b}$$

In (1), $\mathbf{f} : \mathbb{R}^m \to \mathbb{R}^m$ is the state evolution function; $\mathbf{h} : \mathbb{R}^m \to \mathbb{R}^n$ is the observation function; and $\boldsymbol{w}_t, \boldsymbol{v}_t$ are mutually and temporally independent zero-mean noises with covariances $\mathbf{Q}$ and $\mathbf{R}$, respectively.

While various tasks are associated with SS models, including, e.g., smoothing and imputation, we focus on *tracking*. The tracking (filtering) task refers to estimating the state $\mathbf{x}_t$ from current and past observations $\{\mathbf{y}_\tau\}_{\tau \leq t}$. Performance is

measured via the mean squared error (MSE) between the estimated state and the true one. Namely, for a sequence of length $T$, the time-averaged MSE of an estimate $\{\hat{\mathbf{x}}_t\}_{t=1}^T$ is

$$\text{MSE} = \frac{1}{T} \sum_{t=1}^T \mathbb{E} \left\{ \|\hat{\mathbf{x}}_t - \mathbf{x}_t\|^2 \right\}. \tag{2}$$

### B. Model-Based Tracking

The characterization of dynamic systems via the SS model (1) gives rise to various algorithms for tracking the state. Specifically, given a full and accurate description of the functions $\mathbf{f}(\cdot)$, $\mathbf{h}(\cdot)$, and characterization of the distribution of the state and measurement noises as being Gaussian with known covariances, a suitable tracking algorithm is the EKF [2, Ch. 10].

The EKF is comprised of *prediction* and *measurement update* steps. On each time step $t$, it starts by propagating the first-order statistical moments of the state and observations via

$$\hat{\mathbf{x}}_{t|t-1} = \mathbf{f}(\hat{\mathbf{x}}_{t-1|t-1}), \quad \hat{\mathbf{y}}_{t|t-1} = \mathbf{h}(\hat{\mathbf{x}}_{t|t-1}). \tag{3}$$

Letting $\hat{\mathbf{F}}_t$ and $\hat{\mathbf{H}}_t$ be the Jacobians (partial derivatives) of $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$, evaluated at $\hat{\mathbf{x}}_{t-1|t-1}$ and $\hat{\mathbf{x}}_{t|t-1}$, respectively, the prior covariances are set to

$$\boldsymbol{\Sigma}_{t|t-1} = \hat{\mathbf{F}}_t \boldsymbol{\Sigma}_{t-1|t-1} \hat{\mathbf{F}}_t^\top + \mathbf{Q}, \tag{4a}$$

$$\mathbf{S}_{t|t-1} = \hat{\mathbf{H}}_t \boldsymbol{\Sigma}_{t|t-1} \hat{\mathbf{H}}_t^\top + \mathbf{R}. \tag{4b}$$

The observations are fused with the predictions as

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}), \tag{5}$$

where $\mathbf{K}_t$ is the KG given by

$$\mathbf{K}_t = \boldsymbol{\Sigma}_{t|t-1} \hat{\mathbf{H}}_t^\top \mathbf{S}_{t|t-1}^{-1}. \tag{6}$$

The error covariance is then computed as

$$\boldsymbol{\Sigma}_{t|t} = (\mathbf{I} - \mathbf{K}_t \hat{\mathbf{H}}_t) \boldsymbol{\Sigma}_{t|t-1}. \tag{7}$$

The EKF copes with the non-linearity of the SS model by linearizing the state evolution and measurement functions when propagating the second-order moments in (4). Alternative approaches employ different approximations for propagating these moments in non-linear SS models, including the unscented transform (utilized for that purpose by the unscented KF (UKF) [40]), as well as the cubature [41] and Gauss-Hermite deterministic or stochastic quadrature rules [42], [43]. While these model-based algorithms may not necessarily be MSE optimal (as opposed to linear Gaussian SS models), they $(i)$ inherently provide an estimate of both the state via (5) and the estimation uncertainty via (7); and $(ii)$ rely on accurate description of the dynamics as a known SS model as in (1).

### C. DNN-Aided Tracking

DNNs, and particularly architectures suitable of time sequences such as RNNs, can learn to track in dynamics where the SS model is complex and unknown [5]. DNNs used for tracking take as input the current observation $\mathbf{y}_t$ at each time $t \in \mathbb{N}$. The output of such a DNN-aided filter with trainable parameters $\boldsymbol{\theta}$ at time $t$, denoted $\hat{\mathbf{x}}_t(\mathbf{y}_t, \boldsymbol{\theta}) \in \mathbb{R}^m$, is an estimate of the state $\mathbf{x}_t$, as illustrated in Fig. 1(a).
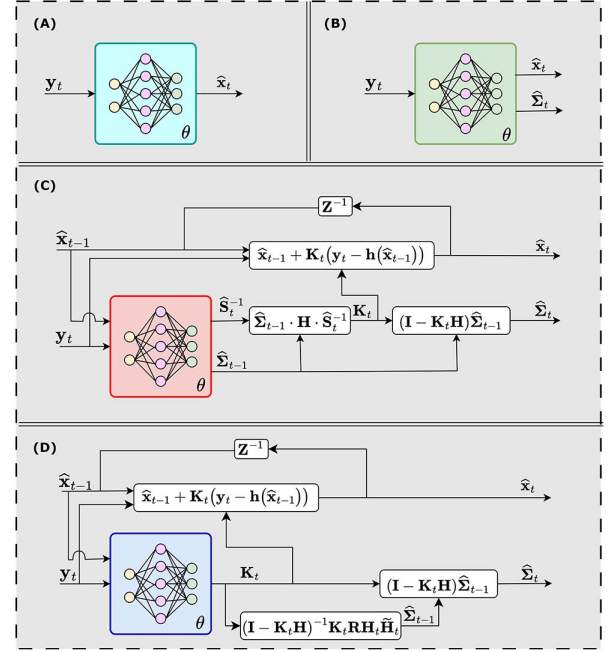


Fig. 1. Considered tracking approaches: $(A)$ standard black-box DNN; $(B)$ DNN with error covariance output (e.g., [9]); $(C)$ DNN-aided KF, estimated posterior covariance and KG (e.g., [23]); and $(D)$ DNN-aided KF, learned KG (e.g., [22]).

We focus on supervised settings, where learning is based on data comprised of pairs of state length $T$ state trajectories and observations, denoted

$$\mathcal{D} = \left\{ \{\mathbf{x}_t^{(i)}, \mathbf{y}_t^{(i)}\}_{t=1}^T \right\}_{i=1}^{|\mathcal{D}|}, \tag{8}$$

where the superscript $(i)$ denotes $i$th data trajectory. In training, the DNN parameters $\boldsymbol{\theta}$ are tuned such that the state estimate best matches the data. Since performance is measured via the MSE (2), a suitable loss function is the empirical $\ell_2$ loss

$$\mathcal{L}_{\mathcal{D}}^{\ell_2}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|T} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^T \left\| \mathbf{x}_t^{(i)} - \hat{\mathbf{x}}_t(\mathbf{y}_t^{(i)}, \boldsymbol{\theta}) \right\|^2. \tag{9}$$

While DNNs can be trained using (9) to track $\mathbf{x}_t$ without requiring characterization of the SS model in (1), they do not provide an estimate of the error covariance, e.g., as the EKF computes $\boldsymbol{\Sigma}_{t|t}$ in (7). Existing approaches for explainability in DNNs, typically involve additional data and conformal prediction tools for boosting verification within confidence regions [44], rather than providing instantaneous error covariance.

### D. Problem Formulation

We consider the problem of *uncertainty-aware tracking*, namely, the design of an algorithm that maps the observed $\mathbf{y}_t$ into estimates of both the state $\hat{\mathbf{x}}_t$ and its *estimation uncertainty* $\hat{\boldsymbol{\Sigma}}_t$. The former is evaluated via the MSE in (2), while the latter is measured via the accuracy of the estimated variances, i.e.,

$$\frac{1}{T} \sum_{t=1}^T \sum_{j=1}^m \mathbb{E} \left\{ \left| \left[\hat{\boldsymbol{\Sigma}}_t\right]_{j,j} - \text{Var} \left\{ [\mathbf{x}_t]_j \right\} \right| \right\}. \tag{10}$$

Model-based tracking algorithms estimate both the state and the error covariance when the dynamics are characterized by a known SS model. However, we focus on setting where the SS is only partially known, and one must cope with the following requirements:

- R1 The state evolution and observation models, $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$, are *approximations of the true dynamics*, that are possibly mismatched.
- R2 The distribution of the noises $\boldsymbol{v}_t$ and $\boldsymbol{w}_t$, which capture the inherent stochasticity in the dynamics, is unknown, and can be complex and non-Gaussian.
- R3 As error covariance is a statistical quantity, it is unlikely to be provided in a dataset $\mathcal{D}$. Accordingly, we assume access to data that is labeled as in (8), namely, it is comprised of trajectories of observations and states but does not include the error covariance.

These requirements motivate exploring data-driven approaches for tackling tasks associated with SS models, allowing to both recover the state and the error covariance.

### E. Bayesian Deep Learning

While the formulation of the SS model in Subsection II-A and the model-based tracking in Subsection II-B focuses on a Bayesian setting, from a deep learning perspective, the term Bayesian deep learning refers to a framework in which a stochastic modeling is employed on the DNN itself. Accordingly, we henceforth use the term *frequentist* within the context of DNNs to refer to the standard deep learning approach in which the weights of a DNN are deterministic weights, whereas a *Bayesian* DNN models its weights as random variables governed by a learned posterior distribution [37].

Specifically, Bayesian deep learning offers a framework to quantify uncertainty in DNN predictions. It treats the DNN parameters as random variables, and learns their posterior distribution given the data. This probabilistic approach provides a measure of uncertainty by sampling multiple realizations of the (random) DNN during inference [45].

In Bayesian deep learning, instead of learning a single set of parameters $\boldsymbol{\theta}$, we aim to learn the posterior distribution $p(\boldsymbol{\theta}|\mathcal{D})$ over the parameters given the data $\mathcal{D}$. In the considered context of estimating a state $\mathbf{x}_t$ from observations $\mathbf{y}_t$, the state estimate $\hat{\mathbf{x}}_t$ and the associated uncertainty can then be obtained by marginalizing over this posterior distribution, i.e.,

$$p(\hat{\mathbf{x}}_t|\mathbf{y}_t, \mathcal{D}) = \int p(\hat{\mathbf{x}}_t|\mathbf{y}_t, \boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta}. \quad (11)$$

To learn a stochastic DNN, one typically seeks to approximate a posterior distribution on the weights, i.e., $p(\boldsymbol{\theta}|\mathcal{D})$, that is dictated by some parameters $\boldsymbol{\phi}$. This can result in either an explicit formulation of the distribution, as in variational inference [46], or via alternative induced trainable stochasticity, as in Monte Carlo dropout [47]. Once dictated, the parameters of the DNN distribution, $\boldsymbol{\phi}$, are trained using the evidence lower bound (ELBO) objective, which regularizes the expected loss with a term minimizing the Kullback-Leibler (KL) divergence between the induced distribution $q(\boldsymbol{\theta}|\boldsymbol{\phi})$ and some pre-determined prior $p(\boldsymbol{\theta})$, i.e.,

$$\text{ELBO}_{\mathcal{D}}(\boldsymbol{\phi}) = \mathbb{E}_{q(\boldsymbol{\theta}|\boldsymbol{\phi})}\left[\log p(\boldsymbol{\theta}|\mathcal{D})\right] - \text{KL}(q(\boldsymbol{\theta}|\boldsymbol{\phi})|p(\boldsymbol{\theta})). \quad (12)$$

Once trained, the stochastic DNN is applied via ensembling method. Particularly, during inference $J$ realizations, denoted $\{\boldsymbol{\theta}_j\}_{j=1}^J$, are drawn in an i.i.d. manner from $q(\boldsymbol{\theta}|\boldsymbol{\phi})$. Then, inference is carried out by applying these $J$ DNNs to the input, and combining their results. For instance, in our context of state tracking, the resulting estimate becomes

$$\hat{\mathbf{x}}_t(\mathbf{y}_t, \boldsymbol{\phi}) = \frac{1}{J}\sum_{j=1}^J \hat{\mathbf{x}}_t^{(j)}(\mathbf{y}_t, \boldsymbol{\theta}_j), \quad \{\boldsymbol{\theta}_j\} \overset{\text{i.i.d.}}{\sim} q(\boldsymbol{\theta}|\boldsymbol{\phi}). \quad (13)$$

## III. ERROR COVARIANCE EXTRACTION IN FREQUENTIST DNN-AIDED TRACKING

The first part of our study focuses on how one can extract error covariance from existing DNN-based tracking algorithms, based on conventional frequentist training (i.e., learning of deterministic DNNs). We divide existing architectures by their interpretable features that can be related to error covariance in Subsection III-A. Then, we propose dedicated training objectives to encourage these architectures to allow extraction of faithful uncertainty measures in Subsection III-B, respectively, and provide a discussion in Subsection III-C.

### A. Error Covariance Indicative Features

Different DNN-aided algorithms provide different features that can be used to extract the error covariance. We divide DNN-aided systems accordingly, considering black-box methods, where only the outputs are observed, as well as model-based deep learning algorithms following the EKF operation, where one can possibly relate internal features to the prior covariance and KG.

*1) Output Features:* In conventional black-box DNNs, one can only assign an interpretable meaning to the input and output of the system. In such cases, in order to estimate the error covariance, additional outputs are required (see Fig. 1(b)); these outputs should represent the estimated covariance denoted $\hat{\boldsymbol{\Sigma}}_t$. This can be achieved by having the DNN output (in addition to $\hat{\mathbf{x}}_t(\mathbf{y}_t, \boldsymbol{\theta})$), a vector $\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t, \boldsymbol{\theta}) \in \mathbb{R}_+^m$, as done in the RKN method [9], setting the estimated error covariance to be

$$\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t, \boldsymbol{\theta}) = \text{diag}(\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t, \boldsymbol{\theta})). \quad (14)$$

*2) KG and Prior Covariance Features:* Several hybrid model-based deep learning learn to track in partially known SS models by augmenting the EKF with DNNs. Some algorithms, such as Split-KalmanNet (SKN) [23], learn to compute the KG, denoted $\hat{\mathbf{K}}_t(\mathbf{y}_t, \boldsymbol{\theta})$, by treating an internal feature as an estimate of the prior covariance, denoted $\hat{\boldsymbol{\Sigma}}_{t|t-1}(\mathbf{y}_t, \boldsymbol{\theta})$. Following the EKF (4)–(7), these features can be combined to estimate the error covariance as illustrated in Fig. 1(c) via

$$\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t, \boldsymbol{\theta}) = (\mathbf{I} - \hat{\mathbf{K}}_t(\mathbf{y}_t, \boldsymbol{\theta})\hat{\mathbf{H}}_t)\hat{\boldsymbol{\Sigma}}_{t|t-1}(\mathbf{y}_t, \boldsymbol{\theta}). \quad (15)$$

The formulation in (15) is based on the update step of the EKF in (7), which implicitly assumes that the KG is the MSE optimal linear estimator. As there is no guarantee that this indeed holds for a learned DNN-aided KG, one can still estimate the error covariance via the Joseph form

$$\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t, \boldsymbol{\theta}) = \hat{\mathbf{K}}_t(\mathbf{y}_t, \boldsymbol{\theta}) \mathbf{R} \hat{\mathbf{K}}_t^\top(\mathbf{y}_t, \boldsymbol{\theta}) + (\mathbf{I} - \hat{\mathbf{K}}_t(\mathbf{y}_t, \boldsymbol{\theta}) \hat{\mathbf{H}}_t)$$
$$\times \hat{\boldsymbol{\Sigma}}_{t|t-1}(\mathbf{y}_t, \boldsymbol{\theta})(\mathbf{I} - \hat{\mathbf{K}}_t(\mathbf{y}_t, \boldsymbol{\theta}) \hat{\mathbf{H}}_t)^\top. \quad (16)$$

*3) KG Features:* The formulations in (15)-(16) rely on the ability to extract both the prior covariance and the KG as internal features. DNN-aided systems such as KalmanNet [22] learn to compute the KG from data without recovering the prior covariance. There, if $\tilde{\mathbf{H}}_t \triangleq \left(\hat{\mathbf{H}}_t^\top \hat{\mathbf{H}}_t\right)^{-1}$ exists, then, by combining (4b)-(6), the prior covariance can be recovered via

$$\hat{\boldsymbol{\Sigma}}_{t|t-1}(\mathbf{y}_t, \boldsymbol{\theta}) = (\mathbf{I} - \hat{\mathbf{K}}_t(\mathbf{y}_t, \boldsymbol{\theta}) \hat{\mathbf{H}}_t)^{-1} \hat{\mathbf{K}}_t(\mathbf{y}_t, \boldsymbol{\theta}) \mathbf{R} \hat{\mathbf{H}}_t \tilde{\mathbf{H}}_t. \quad (17)$$

Using the recovered prior covariance in (17), the desired error covariance is obtained by either (15) or (16). This approach is illustrated in Fig. 1(d).

*Remark 1:* Converting the KG into the prior covariance in (17) requires $\tilde{\mathbf{H}}_t$ to exist, i.e., $\hat{\mathbf{H}}_t$ has full column rank. Yet, even when some of the columns of $\hat{\mathbf{H}}_t$ are zero or $n < m$, error covariance matrix elements with respect to the same state variables can still be recovered by considering the sub-matrices of $\hat{\mathbf{H}}_t$, $\hat{\boldsymbol{\Sigma}}_{t|t-1}$, and $\hat{\boldsymbol{\Sigma}}_t$ corresponding to these variables. For instance, in navigation scenarios with noisy position measurements, one can still extract the position error covariance while also tracking velocity.

### B. Training to Predict the Error Covariance

The indicative features detailed in Subsection III-A allow DNN-aided systems to provide $\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t, \boldsymbol{\theta})$, viewed as an estimate of the error covariance. However, training via (9), which solely observes the state estimate, does not explicitly encourage $\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t, \boldsymbol{\theta})$ to accurately match the error covariance. Accordingly, we consider two alternative loss measures that overcome the absence of a ground truth covariance, one that is based on evaluating the empirically averaged error variance, and one that utilizes a Gaussian prior of the estimated state.

*1) Empirical Error Variance:* A straight-forward approach to account for the error covariance in the loss is to introduce an additional term that compares $\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t, \boldsymbol{\theta})$ to the instantaneous empirical estimate. Specifically, by defining the error vector

$$\boldsymbol{e}_t(\mathbf{y}_t, \mathbf{x}_t, \boldsymbol{\theta}) \triangleq \mathbf{x}_t - \hat{\mathbf{x}}_t(\mathbf{y}_t, \boldsymbol{\theta}), \quad (18)$$

the loss includes an additive term that compares the predicted error with its empirical value obtained from (18). By defining the second moment loss over the whole dataset as:

$$\mathcal{L}_{\mathcal{D}}^{\mathrm{M2}}(\boldsymbol{\theta}) = \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{T} \sum_{j=1}^{m} \left| [\boldsymbol{e}_t(\mathbf{y}_t^{(i)}, \mathbf{x}_t^{(i)}, \boldsymbol{\theta})]_j^2 - [\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t^{(i)}, \boldsymbol{\theta})]_{j,j} \right|. \quad (19)$$

The resulting empirical loss used for training is

$$\mathcal{L}_{\mathcal{D}}^{\mathrm{Emp}}(\boldsymbol{\theta}) = (1 - \beta) \cdot \mathcal{L}_{\mathcal{D}}^{\ell_2}(\boldsymbol{\theta}) + \beta \cdot \mathcal{L}_{\mathcal{D}}^{\mathrm{M2}}(\boldsymbol{\theta}). \quad (20)$$

The hyperparameter $\beta \in [0, 1]$ in (20) is used to balance the trade-off between minimizing the tracking error and encouraging accurate prediction of the error covariance. In our numerical implementation detailed in Section V, $\beta$ is initialized with a small value at the beginning of training, so that the network initially focuses on learning to accurately track the state. As training progresses, $\beta$ is gradually increased, thereby placing more emphasis on the accuracy of the predicted uncertainty. Specifically, when setting $\beta = 0$, then training with (20) coincides with training with the standard $\ell_2$ loss.

*2) Gaussian Prior:* An alternative loss that accounts for error covariance without explicitly comparing predicted and empirical values leverages Gaussian priors. Particularly, one can train to both track and provide error covariance by imposing a Gaussian prior on the tracked state, as done in [9]. Under this assumption, a suitable loss is the log-likelihood of a Gaussian distribution with mean $\hat{\mathbf{x}}_t(\mathbf{y}_t^{(i)}, \boldsymbol{\theta})$ and covariance $\hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t^{(i)}, \boldsymbol{\theta})$, i.e.,

$$\mathcal{L}_{\mathcal{D}}^{\mathrm{GP}}(\boldsymbol{\theta}) = \frac{1}{|\mathcal{D}|T} \sum_{i=1}^{|\mathcal{D}|} \sum_{t=1}^{T} \boldsymbol{e}_t^\top(\mathbf{y}_t^{(i)}, \mathbf{x}_t^{(i)}, \boldsymbol{\theta}) \hat{\boldsymbol{\Sigma}}_t^{-1}(\mathbf{y}_t^{(i)}, \boldsymbol{\theta})$$
$$\times \boldsymbol{e}_t(\mathbf{y}_t^{(i)}, \mathbf{x}_t^{(i)}, \boldsymbol{\theta})$$
$$+ \log \det \left( \hat{\boldsymbol{\Sigma}}_t(\mathbf{y}_t^{(i)}, \boldsymbol{\theta}) \right). \quad (21)$$

We note that when the error covariance $\hat{\boldsymbol{\Sigma}}_t$ is not estimated, but is instead assumed to be a scaled identity, then training with the loss in (21) is equivalent to training with the standard $\ell_2$ loss.

### C. Discussion

We identified three different approaches to extract error covariance in discriminative DNN-aided tracking in non-linear SS models. These approaches vary based on the architecture employed and the level of interpretability it provides. It is emphasized though that for all considered approaches, one must still alter the learning procedure for the extracted features to be considered as uncertainty measures. To encourage learning to accurately predict the error, we propose two training objectives. This examination of error covariance extraction in DNN-aided KFs combines new approaches, while encompassing techniques used for specific schemes (e.g., (21) used by RKN [9]), and extending previous findings (e.g., (17) extends [36] to non-linear models).

The main distinction between the approaches lies in their domain knowledge, i.e., which parts of the SS model are needed to track and provide error covariance, as summarized in Table I. Specifically, black-box DNNs do not require any model knowledge. Nonetheless, as empirically demonstrated in Section V, using black-box architectures with additional outputs regarded as the error covariance often leads to inaccurate tracking and unreliable error prediction, even when trained with a covariance-oriented loss as the ones proposed in Subsection III-B. Model-based deep architectures that learn the KG require only approximate knowledge of $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ to track [22], [23]. However, to extract covariance, $\mathbf{h}(\cdot)$ should be accurately known (as, e.g., (15) relies on $\hat{\mathbf{H}}_t$), where $\mathbf{R}$ is also needed

TABLE I
DOMAIN KNOWLEDGE REQUIREMENTS COMPARISON

| Method | Feature | $\mathbf{F}(\cdot)$ | $\mathbf{H}(\cdot)$ | $\mathbf{R}$ | $\mathbf{Q}$ |
|---|---|---|---|---|---|
| Output features | State | None | None | None | None |
| e.g., RKN | Error cov | None | None | None | None |
| **Bayesian KalmanNet** | State | Partial | Partial | None | None |
|  | Error cov | Partial | Partial | None | None |
| Learn KG and Prior | State | Partial | Partial | None | None |
| e.g., SKN | Error cov | Partial | Full | None | None |
| Learn KG | State | Partial | Partial | None | None |
| e.g., KalmanNet | Error cov | Partial | Full | Full | None |
| Model-based | State | Full | Full | Full | Full |
| e.g., EKF | Error cov | Full | Full | Full | Full |

when the prior covariance is not tracked, as in KalmanNet. Accordingly, requirements R1 and R2 are not fully met by these methods. Moreover, the latter also requires the linearized observation model (i.e., $\tilde{\mathbf{H}}_t$) to have full column rank. When these requirements are met, it is empirically shown in Section V that, when training via the losses proposed in Subsection III-B, one can both accurately track the latent state and reliably predict its covariance from the internal features of the DNN-aided algorithm.

## IV. BAYESIAN KALMANNET

As detailed in the previous section, and empirically demonstrated in Section V, model-based deep learning architectures based on the KalmanNet algorithm can simultaneously provide accurate state tracking alongside reliable error prediction. Nonetheless, achieving the latter requires deviating from how these DNN-aided tracking algorithms are designed (e.g., in [22] and [23]) in the following aspects:

A1 Additional domain knowledge that is not required merely for state tracking is needed for covariance extraction (e.g., full characterization of the observations model (1b)), and the family of dynamic systems where one can extract uncertainty is limited (e.g., $\tilde{\mathbf{H}}_t$ has to be non-singular).

A2 The training objective should be altered to encourage reliable uncertainty extraction.

In this section we propose a DNN-aided learning algorithm that preserves the ability of model-based deep learning algorithms such as KalmanNet to accurately track, in partially known SS models, while providing uncertainty measures without being subject to the limitations in A1, thus fully meeting requirements R1–R3. We build upon the observation in A2, namely, that error covariance extraction already necessitates altering the learning objective. Instead of just altering the learning objective, we modify the learning *paradigm*, deviating from conventional frequentist learning into Bayesian learning of stochastic DNNs. Our proposed algorithm, coined *Bayesian KalmanNet*, is detailed in Subsection IV-A, with its training procedure formulated in Subsection IV-B, while Subsection IV-C provides a discussion.

## A. Bayesian KalmanNet Tracking Algorithm

As the name suggests, Bayesian KalmanNet is based on the KalmanNet algorithm of [22], while employing Bayesian DNNs. Accordingly, the algorithm uses a stochastic DNN to compute the KG of the EKF. The usage of a Bayesian DNNs results in Monte Carlo sampling of $J$ different KalmanNet realizations during inference, that are used to recover both the state and its predicted error via ensemble averaging.

**Architecture:** We employ a Bayesian DNN with parameters $\boldsymbol{\theta}$ whose distribution is parameterized by $\boldsymbol{\phi}$. Each DNN realization maps the indicative features $\Delta\mathbf{x}_{t-1} \triangleq \hat{\mathbf{x}}_{t-1} - \hat{\mathbf{x}}_{t-2}$, $\Delta\mathbf{y}_t \triangleq \mathbf{y}_t - \mathbf{y}_{t|t-1}$ and $\Delta\tilde{\mathbf{y}}_t \triangleq \mathbf{y}_t - \mathbf{y}_{t-1}$ into the KG $\hat{\mathbf{K}}_t(\mathbf{y}_t, \boldsymbol{\theta})$.

In our numerical study, we employ *Architecture 1* of [22], comprised of gated recurrent unit (GRU) cells with input and output fully connected (FC) layers. Stochasticity is achieved by integrating dropout into the FC layers [47], with $\boldsymbol{p}$ denoting the Bernoulli distribution parameter assigned with each neuron of these layers. The trainable parameters dictating the DNN distribution, denoted $\boldsymbol{\theta}$, thus encompass a setting of the DNN parameters, denoted $\boldsymbol{\theta}_0$, as well as the dropout neuron-wise probability $\boldsymbol{p}$, i.e., $\boldsymbol{\phi} = [\boldsymbol{\theta}_0, \boldsymbol{p}]$. In each realization, the DNN is applied with weights $\boldsymbol{\theta}_0$ with the FC layers employing a dropout mask randomized with per-element distribution $\boldsymbol{p}$.

**Uncertainty-Aware Tracking:** The proposed augmentation of the EKF with Bayesian DNNs allows to track as in Kalman-Net, while leveraging ensembling to predict the estimation error. When tracking is launched, we sample $J$ i.i.d. realizations of the KG computations DNN from $q(\boldsymbol{\theta}|\boldsymbol{\phi})$, denoted $\{\boldsymbol{\theta}_j\}_{j=1}^J$, with $J$ being a hyperparameter. The type of distribution represented by $q(\boldsymbol{\theta}|\boldsymbol{\phi})$ is a design hyperparameter, as different families of distributions are considered in the Bayesian deep learning literature [46]. In our numerical study reported in Section V we employ Monte Carlo dropout [47], where each DNN realization is obtained from the initial fixed weight $\boldsymbol{\theta}_j$ and a realization of the dropout masks sampled with multivariate Bernoulli distribution $\boldsymbol{p}$. However, the formulation of Bayesian KalmanNet is not restricted to Monte Carlo dropout, and can be combined with alternative distributions, e.g., Gaussian weights whose parameters are the first- and second-order moments.

On each time instance $t$, the first-order moments of the state and the observation are predicted based on the (possibly approximated) knowledge of $\mathbf{f}(\cdot)$ and $\mathbf{h}(\cdot)$ via (3). These estimated moments are used to form the input features to each of the $J$ DNN realizations, obtaining $J$ KGs, which are in turn used to produce $J$ separate state estimates, denoted $\{\hat{\mathbf{x}}_t^j\}_{j=1}^J$. The mean and variance of these estimates provide the state estimate and its associated uncertainty, respectively, via

$$\hat{\mathbf{x}}_t = \frac{1}{J}\sum_{j=1}^J \hat{\mathbf{x}}_t^{(j)}(\mathbf{y}_t, \boldsymbol{\theta}_j), \tag{22a}$$

$$\hat{\boldsymbol{\Sigma}}_t = \frac{1}{J}\sum_{j=1}^J (\hat{\mathbf{x}}_t^{(j)} - \hat{\mathbf{x}}_t)(\hat{\mathbf{x}}_t^{(j)} - \hat{\mathbf{x}}_t)^\top. \tag{22b}$$

From a Bayesian deep learning perspective, the matrix $\hat{\boldsymbol{\Sigma}}_t$ in (22b) represents an empirical estimate of the *predictive*

**Algorithm 1:** Bayesian KalmanNet

**Init:** DNN distribution $\phi$; ensemble size $J$
**Input:** Initial state $\mathbf{x}_0$

1   Set $\hat{\mathbf{x}}_0^j \equiv \mathbf{x}_0$;
2   **for** $t = 1, 2, \ldots$ **do**
3      Receive new observation $\mathbf{y}_t$;
4      Sample $\{\boldsymbol{\theta}_j\} \overset{\text{i.i.d.}}{\sim} q(\boldsymbol{\theta}|\phi)$;
5      **for** $j = 1, 2, \ldots, J$ **do**
6          Predict $\hat{\mathbf{x}}_{t|t-1}^j$ and $\hat{\mathbf{y}}_{t|t-1}^j$ from $\hat{\mathbf{x}}_{t-1}$ via (3);
7          Compute $kg$ $\hat{\mathbf{K}}_t^j$ via DNN $\boldsymbol{\theta}_j$;
8          Update $\hat{\mathbf{x}}_t^j = \hat{\mathbf{x}}_{t|t-1}^j + \hat{\mathbf{K}}_t^j(\mathbf{y}_t - \hat{\mathbf{y}}_{t|t-1}^j)$
9      Compute state estimate $\hat{\mathbf{x}}_t$ via (22a);
10     Predict estimation error $\hat{\boldsymbol{\Sigma}}_t$ via (22b);
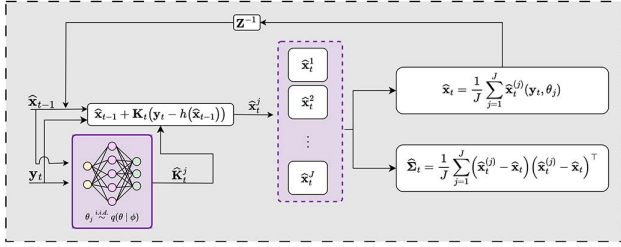


Fig. 2.    Bayesian KalmanNet - Architecture.

*uncertainty*, namely, the second-order moment of the posterior predictive distribution over the state $\mathbf{x}_t$ given the observations and training data [48]. Specifically, each estimate $\hat{\mathbf{x}}_t^{(j)}$ in (22a) corresponds to a sample from a different model instantiation, i.e., from the distribution $q(\boldsymbol{\theta}|\phi)$ over the neural network weights. This ensemble of predictions approximates the posterior predictive distribution, and the sample covariance $\hat{\boldsymbol{\Sigma}}_t$ estimates the total predictive uncertainty. While there is no general guarantee that individual estimators $\hat{\mathbf{x}}_t^{(j)}$ are realizations of an unbiased estimator, the usage of its sample covariance for quantifying uncertainty is a standard and widely accepted approach in Bayesian deep learning. Accordingly, our proposed Bayesian KalmanNet uses $\hat{\boldsymbol{\Sigma}}_t$ in (22b) as an estimate of the error covariance. The resulting operation is summarized as Algorithm 1 (in which, for brevity, we write the output of the $j$th KG DNN realization at time $t$ as $\hat{\mathbf{K}}_t^j$), and illustrated in Fig. 2.

### B. Training Bayesian KalmanNet

The trainable parameters of Bayesian KalmanNet are the distribution parameters $\phi$. In principle, one can train Bayesian KalmanNet as a standard Bayesian regression DNN, evaluating it based on the augmented EKF output using the ELBO objective in (12). Nonetheless, as we are particularly interested in accurate error covariance recovery (which is not explicitly accounted for in the ELBO formulation), we next propose a

dedicated training loss, which is also geared towards our usage of Monte Carlo dropout for inducing stochasticity.

We follow the specialization of the ELBO objective to concrete Monte Carlo based DNNs proposed in [47], while altering the data matching term to incorporate explicit encouraging of reliable error variance recovery, as proposed in Subsection III-B for frequentist learning. To formulate this, let $L$ denote the number of layers in the DNN in which dropout is incorporated, and let $\{\lambda_l\}_{l=1}^L$ denote their indices. In the implementation detailed in Subsection IV-A, these indices correspond to the FC layer of KalmanNet's Architecture 1. We write the parameters of layer $\lambda$ of $\boldsymbol{\theta}$ as $\boldsymbol{\theta}^\lambda$ and its dropout coefficient as $p^{(\lambda)}$. Accordingly, the trainable parameters of Bayesian KalmanNet in such settings, denoted $\phi$, include the weights of each of the layers $\boldsymbol{\theta}_0$, as well as the distribution parameters for the layers in which dropout is incorporated, denoted by $\boldsymbol{p} = [p^{\lambda_1}, \ldots, p^{\lambda_L}]$. The proposed loss function, based on the labeled data in (8), is formulated as

$$\mathcal{L}_{\mathcal{D}}^{\text{BKN}}(\phi = [\boldsymbol{\theta}_0, \boldsymbol{p}])$$
$$= \mathbb{E}_{\boldsymbol{\theta} \sim q(\boldsymbol{\theta}|\phi)}\{\mathcal{L}_{\mathcal{D}}^{\text{Emp}}(\boldsymbol{\theta})\} + \sum_{l=1}^L \mathcal{L}_{\lambda_l}^{\text{KL}}(\boldsymbol{\theta}_0^{\lambda_l}, p^{(\lambda_l)}), \quad (23)$$

where $\mathcal{L}_{\lambda_l}^{\text{KL}}(\phi)$ computes the KL-based regularization term of the layer $\lambda_l$ based on the learned multivariate Bernoulli distribution, with pre-determined prior in (12) being a maximal entropy Bernoulli distribution. The empirical loss term and the KL loss term in (23) are balanced through the non-negative hyperparameters $c_1, c_2$, that appear in the layer-wise loss term, which is given by [47]

$$\mathcal{L}_\lambda^{\text{KL}}(\boldsymbol{\theta}, p) = c_1 \cdot \frac{\|\boldsymbol{\theta}\|^2}{1-p} - c_2 \cdot \mathcal{H}(p) \cdot n^{(\lambda)}. \quad (24)$$

In (24), $n^{(\lambda)}$ is the number of input features to the $\lambda$th layer; and $\mathcal{H}(p)$ is the entropy of a Bernoulli random variable with probability $p$, defined as

$$\mathcal{H}(p) = -p \log(p) - (1-p) \log(1-p). \quad (25)$$

This loss in (23) guides the tuning of $\phi$ when training Bayesian KalmanNet to both extract an accurate state estimation and error covariance (by taking the expectation of our proposed empirical error variance as the data matching term), while encouraging randomness in the prior on the dropout probability $\boldsymbol{p}$. Having formulated the loss in (23), the training of Bayesian KalmanNet follows the conventional stochastic gradient based optimization of DNNs, only with a Bayesian loss term. Accordingly, the overall training complexity is comparable to that of frequentist KalmanNet [22].

### C. Discussion

Bayesian KalmanNet is designed to allow DNN-aided tracking in partially known SS models with error prediction. The extensions to existing frequentist KalmanNet-type methods proposed in Section III, which map the interpretable internal features into the error covariance, follow the mathematical steps used by the EKF in such computations, and thus necessitate additional domain knowledge not required by KalmanNet for

state tracking. Bayesian KalmanNet bypasses the need for such excessive knowledge by adopting SS model-agnostic Bayesian deep learning techniques for error covariance extraction, combined with the SS model-aware KalmanNet for state tracking. This results in Bayesian KalmanNet requiring the same level of partial SS model knowledge as needed by KalmanNet and SKN for state estimation, while also providing error prediction, thus meeting R1–R3. As numerically reported in Section V, Bayesian KalmanNet achieves reliable state tracking and error prediction in various settings with different levels of domain knowledge.

The uncertainty measure extracted by Bayesian KalmanNet encapsulates the overall estimation error covariance. This uncertainty measure inherently comprises two components [48]: *aleatoric uncertainty*, resulting from the stochasticity of the system (e.g., process and measurement noises); and *epistemic uncertainty*, arising due to limited or mismatched knowledge within the estimator, particularly when the underlying dynamics are only partially known or when trained with a limited dataset. Classical model-based methods, such as the EKF, can reliably quantify aleatoric uncertainty when operating with an accurate SS model and well-characterized noise statistics, they underperform when the model is misspecified. Conversely, existing frequentist DNN-aided KFs struggle to provide any form of uncertainty quantification, even with abundant training data. The approaches proposed for extending frequentist DNN-aided KFs introduced in Section III are geared towards quantifying aleatoric uncertainty, aiming to recreate the covariance computations of the model-based EKF without explicitly propagating second-order moments. Accordingly, the resulting characterization, which does not encompass epistemic uncertainty, is expected to be sensitive to the amount of training data. Bayesian KalmanNet overcomes these limitations by providing uncertainty estimates that reflect both sources of uncertainty, thereby yielding more reliable and calibrated error covariances under data scarcity and model mismatch. This dual capability makes Bayesian KalmanNet particularly suitable for practical scenarios where both the dynamics and noise statistics are only partially known.

The incorporation of Bayesian deep learning techniques can be viewed as trading computational complexity for uncertainty measures. This induced complexity also translates into additional inference latency, as reported in Section V. The hyperparameter $J$ plays a key role in affecting both the performance, as increasing $J$ enhances the approximation of the posterior predictive distribution, while also increasing complexity. Specifically the application of Bayesian KalmanNet can be viewed as $J$ realizations of KalmanNet with different parameterizations, thus increasing the computational burden by a factor of $J$ at each time instance. The induced latency of Bayesian KalmanNet can be notably reduced using parallel computing, as each of the $J$ realizations can be applied simultaneously. We also note that this operation of Bayesian KalmanNet bears similarity to alternative tracking algorithm that implement multiple Kalman-type filters in parallel, e.g., [49], though the conventional approach uses a different SS model with deterministic parameters for each filter, while Bayesian KalmanNet uses

multiple stochastic realizations of the filter obtained for the same (partially known) SS model, following the task-oriented discriminative learning paradigm [13].

Our design of Bayesian KalmanNet gives rise to numerous possible avenues for extension. Our design, as well as that considered for frequentist KalmanNet [22], assumes that the SS model in (1) does not change. Thus, our approach is not naturally transferable to time-varying dynamics, for which adaptive KFs, often based on variational inference [50], [51], [52], are required. Previous works that considered the KalmanNet methodology in time-varying dynamics, which proposed adaptation based on hypernetworks [53] and unsupervised learning [54], focused on block-wise variations in the SS model. The extension of the KalmanNet methodology to time-varying dynamics is the area of ongoing research, which, inspired by recent advances in continual Bayesian learning [55], [56], can potentially benefit from the incorporation of Bayesian deep learning. Moreover, while our formulation accommodates a generic Bayesian DNN, our implementation detailed in Section V is based on an architecture proposed in [22], and training objective focus on the usage of Monte Carlo dropout. Our formulation differs than standard Bayesian DNNs as all realizations are based on the same prior. One can possibly consider alternative Bayesian parameterizations, based on, e.g., variational inference with explicit priors [46], as well as its incorporation with different KalmanNet-type architectures, e.g., [23], [24], [25], [26]. Moreover, the combination of Bayesian deep learning and KalmanNet indicates that such methodologies can also be employed in alternative forms of model-based deep learning [12]. We leave these extensions for future work.

## V. NUMERICAL EVALUATION

In this section we numerically compare the above techniques to extract error covariance in DNN-aided KFs. We first detail the considered algorithms implemented and our utilized performance measures in Subsection V-A. Then, we detail a set of experiments dedicated to evaluating state and covariance estimation in different SS models in Subsections V-B–V-D. We conclude by comparing inference time in Subsection V-E.

### A. Experimental Setup

*1) Tracking Algorithms:* We henceforth compare the following tracking algorithms[1]:

- **RKN**, using the code provided in [9], representing DNNs with dedicated output features.
- **SKN**, using the code provided in [23], representing model-based deep learning with learned KG and prior covariance.
- KalmanNet, using both Architecture 1 (**KNetV1**) and Architecture 2 (**KNetV2**) of [22].
- Bayesian KalmanNet (termed **BKN** for brevity), based on KNetV1 with dropout incorporated into its FC layers, taking $J = 20$ realizations.

[1]The source code and all hyperparameters used can be found online at https://github.com/yonatandn/Uncertainty-extraction-in-Model-Based-DL

- The fully model-based **EKF**.

Unless stated otherwise, we use (15) to convert the KG and prior covariance into error covariance. We train RKN and SKN with (21); KalmanNet is trained with (9); while Bayesian KalmanNet is trained using (23). All data-driven algorithms are trained with the same data comprising of $|\mathcal{D}| = 300$ trajectories. Training commences with the DNN weights initialized via standard Xavier initialization, and the Monte Carlo dropout parameters of BKN randomized uniformly in $[0.5, 0.8]$, following the common practice in Bayesian DNNs [47]. The remaining learning hyperparameters were selected based on standard cross validation.

*2) Data Generation:* Unless stated otherwise, throughout the following experimental study, we set the observation noise covariance $\mathbf{R}$ to take the form $r^2\mathbf{I}$, with $1/r^2 = n/\mathrm{Tr}(\mathbf{R})$ denoting the signal-to-noise ratio (SNR). The ratio between the state evolution and the observation noise variances is fixed to 0.01, such that changes in the SNR affect both the state evolution and the observation noises. All DNN-aided algorithms were trained on a range of SNRs rather than for each SNR separately, to emphasize their robustness and ability to operate across different noise levels. To perform statistical tests and evaluation, Monte-Carlo simulation was performed, with $N = 100$ trajectories per scenario.

*3) Performance Measures:* The tracking algorithms are evaluated in terms of the following performance measures: $(i)$ The empirical estimation error, i.e., state estimate MSE (2); and $(ii)$ The predicted error covariance, namely, $\mathrm{Tr}(\hat{\boldsymbol{\Sigma}}_t)$. Performance is thus measured not only in state estimation error, but also on the ability to predict this error from the recovered covariance.

To assess the credibility of the estimators, we also evaluate the average normalized estimation error squared (ANEES) measure [3], [57], [58], [59]. This measure combines state estimation and error covariance prediction in a single measure. For an estimator outputting state estimates and covariances $\{\hat{\mathbf{x}}_t, \hat{\boldsymbol{\Sigma}}_t\}_{t=1}^T$, the ANEES is given by

$$\mathrm{ANEES} = \frac{1}{T}\sum_{t=1}^T \mathbb{E}\left\{(\mathbf{x}_t - \hat{\mathbf{x}}_t)^\top \hat{\boldsymbol{\Sigma}}_t^{-1}(\mathbf{x}_t - \hat{\mathbf{x}}_t)\right\}, \quad (26)$$

where the stochastic expectation is computed by empirical averaging over the test data. The closer to the ANEES is to being 1, the more credible the estimator, with values surpassing 1 representing overconfidence, and lower values corresponding to conservative predictions.

Additional metrics used to evaluate the different methods at a given time instance $t$ over a test set $\mathcal{D}$ are the average predicted error covariance (APEC) and the empirical error covariance (EEC). The APEC is given by

$$\mathrm{APEC}_t(\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} \hat{\boldsymbol{\Sigma}}_t^{(i)}, \quad (27)$$

while the EEC is given by

$$\mathrm{EEC}_t(\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{i=1}^{|\mathcal{D}|} (\mathbf{x}_t - \hat{\mathbf{x}}_t^{(i)})(\mathbf{x}_t - \hat{\mathbf{x}}_t^{(i)})^\top. \quad (28)$$

We say that the algorithm is a better uncertainty estimator as its uncertainty prediction (APEC along time) is closer to the true error covariance (EEC along time).

*4) Mismatch Setup:* We consider the following forms of mismatches in the model knowledge:

- *Process noise mismatch*: a dataset generated with a process noise covariance matrix $\mathbf{Q}_{\mathrm{data}} = 100 \cdot \mathbf{Q}_{\mathrm{model}}$ where $\mathbf{Q}_{\mathrm{model}}$ is the process noise covariance matrix known to the algorithm.
- *Measurement noise mismatch*: a dataset generated with a measurement noise covariance matrix $\mathbf{R}_{\mathrm{data}} = 100 \cdot \mathbf{R}_{\mathrm{model}}$ where $\mathbf{R}_{\mathrm{model}}$ is the measurement noise covariance matrix known to the algorithm.
- *Model mismatch*: in the constant velocity (CV) SS, data was generated using a constant acceleration (CA) model, while testing the different methods with a CV SS.

Our motivation for employing the considered mismatches, imposed on all algorithms that require knowledge of the SS model, is to evaluate sensitivity to incorrect modeling assumptions. Specifically, the noise mismatches (which affect just the EKF) serve to highlight the behavior of uncertainty quantification under severe mismatches, while the last model mismatch represents a realistic setup in which a target follows a more complicated state evolution model compared to the one used by the tracking algorithm.

### B. Linear Canonical SS Model

We first simulate a synthetic two-dimensional linear SS model with Gaussian process and measurement noises governed by a controllable canonical state transition matrix, defined in [22, Sec. IV.B]. Here, the state transition matrix $\mathbf{F}$ and measurement matrix $\mathbf{H}$ are

$$\mathbf{F} = \frac{1}{\sigma_{\max}}\begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}, \quad \mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix},$$

where $\sigma_{\max}$ is the largest singular value of $\hat{\mathbf{F}}_t$ ensuring stability.

The considered performance measures versus the observations noise variance $r^2$ are reported in Fig. 3(b). When examining the MSE in Fig. 3(a), we can see that all the model aware methods achieved a smaller estimation error compared to the RKN which is model agnostic. In Fig. 3(b) we observe that (frequentist) KalmanNet provides the most accurate error covariance estimates compared to the other methods when introducing a mismatch in the process noise covariance $\mathbf{Q}$. This is evidenced by the lowest absolute value of $\log(\mathrm{ANEES})$, suggesting that KalmanNet closely matches the true uncertainty in the system. This stems from the fact that in this relatively simple setup, one can translate the KG into an error covariance estimate as detailed in Section III. The SKN produces either too conservative or too confident predictions, meaning it brutally overestimates or underestimates the uncertainty, leading to larger error covariances. Bayesian KalmanNet demonstrates minor overconfidence in its predictions, underestimating the uncertainty. Both RKN and EKF exhibit significant overconfidence, with their predictions being excessively optimistic. This means their error covariances are underestimated and are not aligned with the true uncertainties.
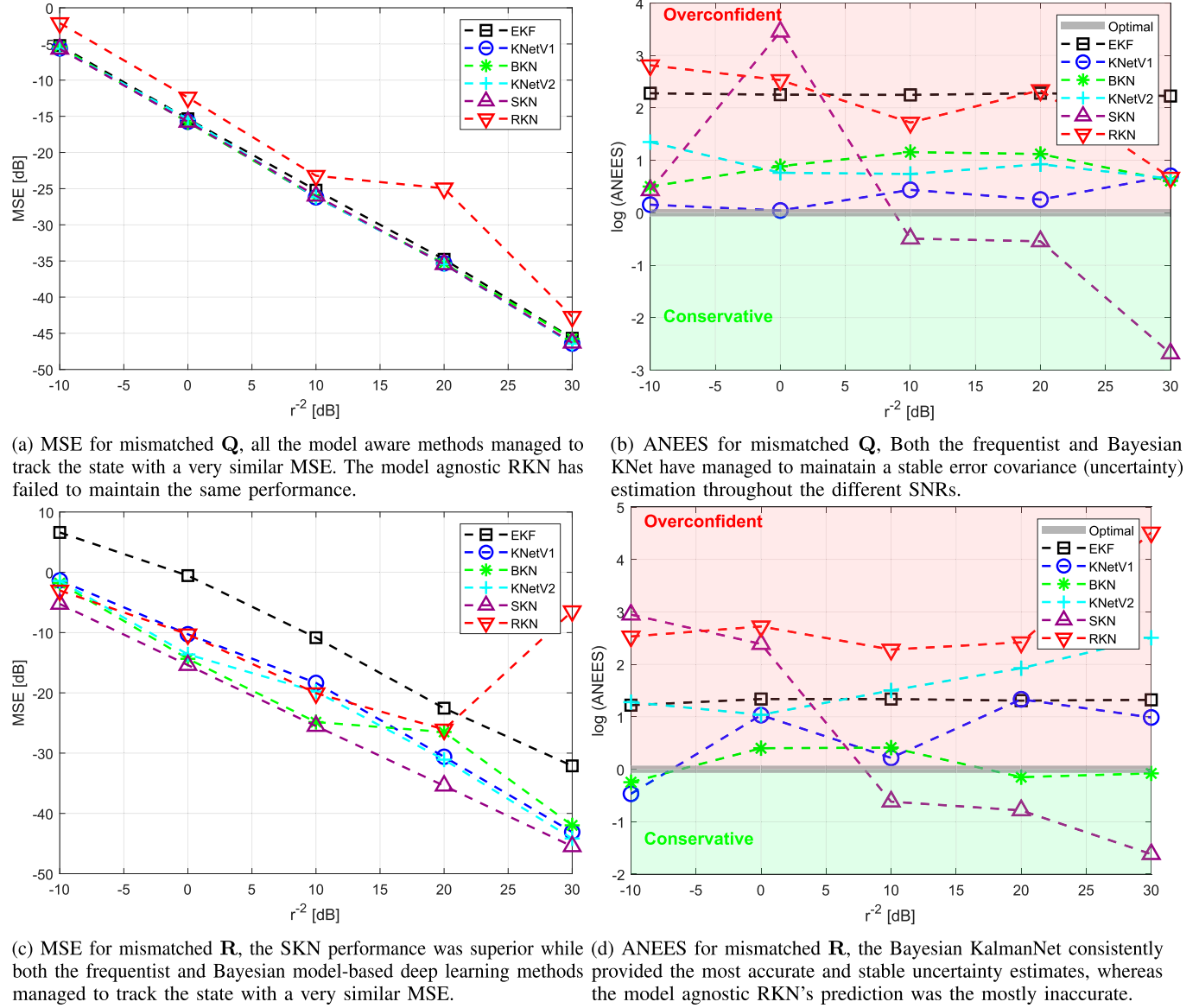
(a) MSE for mismatched $\mathbf{Q}$, all the model aware methods managed to track the state with a very similar MSE. The model agnostic RKN has failed to maintain the same performance.

(b) ANEES for mismatched $\mathbf{Q}$, Both the frequentist and Bayesian KNet have managed to maintain a stable error covariance (uncertainty) estimation throughout the different SNRs.

(c) MSE for mismatched $\mathbf{R}$, the SKN performance was superior while both the frequentist and Bayesian model-based deep learning methods managed to track the state with a very similar MSE.

(d) ANEES for mismatched $\mathbf{R}$, the Bayesian KalmanNet consistently provided the most accurate and stable uncertainty estimates, whereas the model agnostic RKN's prediction was the mostly inaccurate.

Fig. 3. Canonical linear SS model - mismatched process noise $\mathbf{Q}$ and measurement noise $\mathbf{R}$ covariances.

When introducing a mismatch in the measurement noise covariance $\mathbf{R}$ and evaluating the filter performance using the ANEES metric, as shown in Fig. 3(d), it is evident that Bayesian KalmanNet achieves the most favorable results. Bayesian KalmanNet provides a credible error covariance estimation, meaning it maintained reliability. The EKF and KalmanNet produce comparable results, indicating that both methods offer similar levels of credibility in their error covariance predictions. In contrast, the RKN struggles to provide credible estimates, leading to unreliable uncertainty predictions. Comparing Fig. 3(b) and 3(d) shows that frequentist methods can also achieve strong performance, particularly in scenarios where uncertainty can be reliably estimated. However, in cases where uncertainty estimation is inherently more challenging, e.g., where inaccuracies directly impact the observed data—frequentist methods relying on internal features may struggle.

In such settings, Bayesian KalmanNet provides more robust and calibrated uncertainty estimates by explicitly modeling parameter uncertainty through ensemble predictions. Overall, both variants exhibit stable and credible uncertainty estimates across different SNR levels, and the observed differences are relatively minor.

Turning to the MSE results, illustrated in Fig. 3(c), it is clear that the SKN, which is particularly designed for settings with imbalances between state evolution and measurement noises [23], outperforms other methods by achieving the lowest estimation error across various measurement noise covariance levels. This indicates that SKN consistently provided the most accurate state estimates, regardless of the noise magnitude. The RKN, however, only performed well at higher $r^2$ values, suggesting that it struggles in high SNRs with limited data sets, where accurate uncertainty modeling is more critical. All
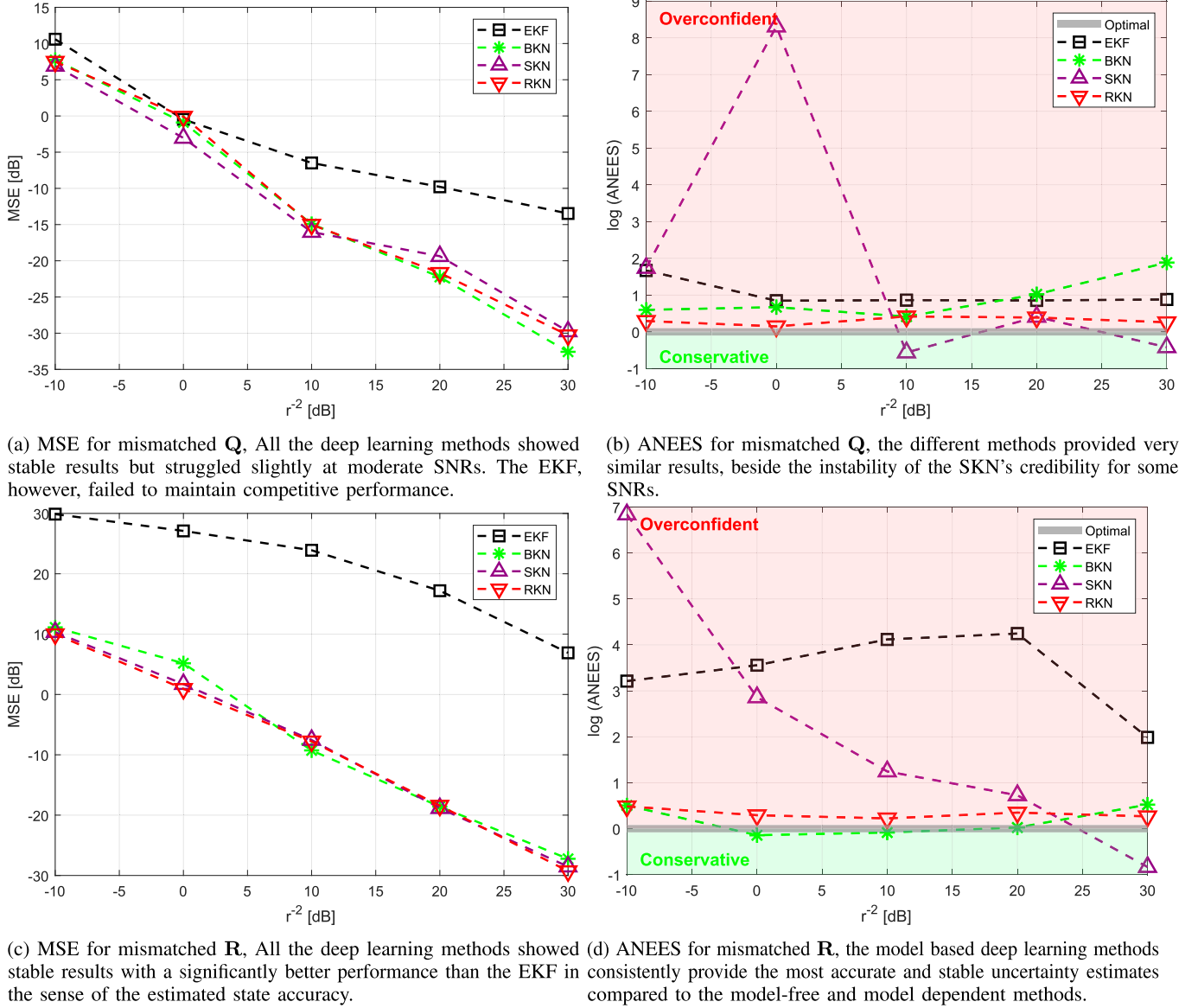
(a) MSE for mismatched $\mathbf{Q}$, All the deep learning methods showed stable results but struggled slightly at moderate SNRs. The EKF, however, failed to maintain competitive performance.

(b) ANEES for mismatched $\mathbf{Q}$, the different methods provided very similar results, beside the instability of the SKN's credibility for some SNRs.

(c) MSE for mismatched $\mathbf{R}$, All the deep learning methods showed stable results with a significantly better performance than the EKF in the sense of the estimated state accuracy.

(d) ANEES for mismatched $\mathbf{R}$, the model based deep learning methods consistently provide the most accurate and stable uncertainty estimates compared to the model-free and model dependent methods.

Fig. 4. Pendulum non-linear SS model - mismatched process noise $\mathbf{Q}$ and measurement noise $\mathbf{R}$ covariances.

KalmanNet based methods, including Bayesian KalmanNet, outperform the EKF. Comparing the MSE results in Fig. 3(c) and the uncertainty results in Fig. 3(d) indicates on the trade-off emerging by the need to balance both objectives during training. In such multi-objective settings, improving one aspect—e.g., achieving lower MSE—may slightly compromise the precision of uncertainty estimates, sometimes somewhat compromise the precision of uncertainty estimates, and in some cases hardly affect the latter.
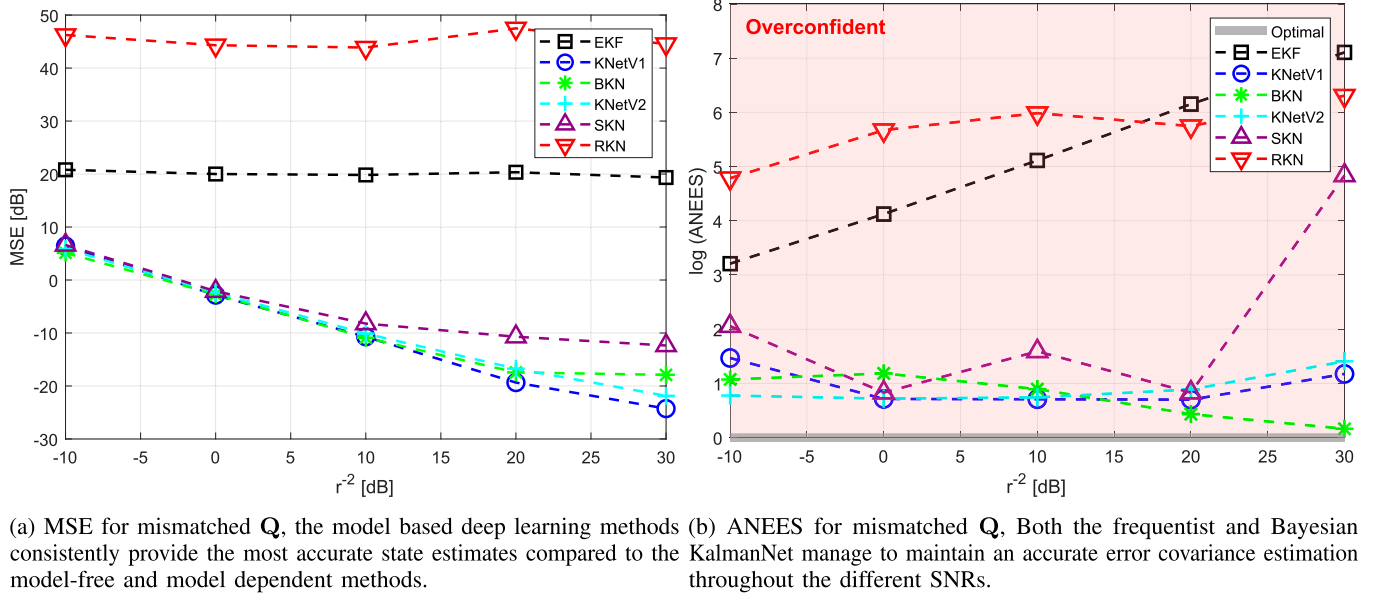
The results in this SS model emphasize that for a variety of SNRs, even for a linear state transition and measurement functions, both the frequentist and the Bayesian methods combining model knowledge and deep learning achieve better results than both model agnostic deep learning and model based without deep learning.

TABLE II
PERFORMANCE METRICS AT SNR $1/r^2 = 10$ [dB] WITH MISMATCHED $\mathbf{R}$, PENDULUM NON-LINEAR SS MODEL

| Metric | EKF | SKN | RKN | BKN |
|---|---|---|---|---|
| MSE [dB] | 23.8933 | -7.5536 | -7.7836 | -9.2741 |
| log(ANEES) | 4.1182 | 1.2452 | 0.2234 | -0.0834 |

### C. Pendulum SS Model

Next, we consider a pendulum with mass $m = 1$ and length $\ell = 1$. The pendulum exhibits nonlinear behavior due to the gravitational force $g$ acting on it. We describe the system using the angle $\theta$ and angular velocity $\dot{\theta}$ as the state variables, sampled at interval of $\Delta t = 0.01$. Let the state vector $\mathbf{x}_t$ be

(a) MSE for mismatched $\mathbf{Q}$, the model based deep learning methods consistently provide the most accurate state estimates compared to the model-free and model dependent methods.

(b) ANEES for mismatched $\mathbf{Q}$, Both the frequentist and Bayesian KalmanNet manage to maintain an accurate error covariance estimation throughout the different SNRs.

Fig. 5.   CV SS model - mismatched process noise $\mathbf{Q}$.

defined as:

$$\mathbf{x}_t = \begin{bmatrix} \theta_t \\ \dot{\theta}_t \end{bmatrix}.$$

The dynamics of the pendulum can be modeled using Newton's second law. For a pendulum subject to gravity, we can express this second-order equation as a system of first-order differential equations such that the discrete nonlinear state transition function $\mathbf{F}(\mathbf{x}_t)$ can be written as:

$$\mathbf{f}(\mathbf{x}_t) = \begin{bmatrix} \theta_t + \dot{\theta}_t \Delta t \\ \dot{\theta}_t - \frac{g}{\ell} \sin(\theta_t) \Delta t \end{bmatrix}.$$

Assuming that $\mathbf{x}_t$ at each time step is small enough to allow a local linearization, the Jacobian matrix of $\mathbf{F}(\mathbf{x}_t)$ is

$$\hat{\mathbf{F}}_t = \begin{bmatrix} 1 & \Delta t \\ -\frac{g}{\ell} \cos(\theta_t) \Delta t & 1 \end{bmatrix}.$$

We assume the pendulum's position in Cartesian coordinates, such that the observation matrix $\mathbf{h}(\mathbf{x}_t)$ for converting state to measurement is given by:

$$\mathbf{h}(\mathbf{x}_t) = \begin{bmatrix} \ell \cos(\theta_t) \\ \ell \sin(\theta_t) \end{bmatrix}.$$

Here, $\hat{\mathbf{H}}(\mathbf{x}_t)$ is non-full columns rank. This means that the frequentist model-based methods are unable to extract an error covariance matrix as explained in Remark 1, and thus are not tested throughout this scenario.

The achieved performance measures are reported in Fig. 4. When examining the MSE in Fig. 4(a), it can be seen that the EKF fails to produce an accurate state estimation compared to all other methods, especially at higher SNRs. The ANEES results in Fig. 4(b) highlight that beside the SKN, all algorithms have a stable credibility with a minor overconfidence throughout different SNRs, whereas the SKN shows some instability in
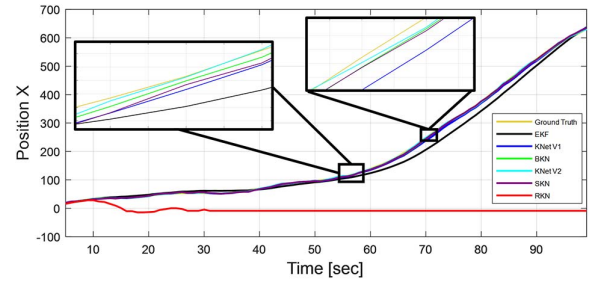


Fig. 6.   x-axis estimate in a single run, CV SS model.

its credibility. The RKN provides the most accurate error covariance estimates compared to the other methods when introducing a mismatch in the process noise covariance $\mathbf{Q}$.

Under mismatched $\mathbf{R}$, the MSE results in Fig. 4(c) show that in terms of estimation error the EKF performs the worst, as its MSE remains significantly higher than the other methods across the entire range of $1/r^2$. This indicates that the EKF struggles to maintain accurate state estimates, particularly in the presence of higher measurement noise. Its performance gap widens considerably as the noise mismatch increases, showing that the EKF is less robust under such conditions. The DNN-aided Bayesian KalmanNet, SKN, and RKN all perform much better, achieving comparable results for most of the range.

Fig. 4(d) shows that the SKN is the most overconfident filter only at higher noise levels, though it improves compared to the other methods as the noise level is lower. The EKF, demonstrates a large overconfidence, throughout all noise levels. In contrast, Bayesian KalmanNet and the RKN consistently yield a very accurate (credible) error covariance estimates as it stays closest to the optimal $\log(\text{ANEES}) = 0$ for much of the range. To highlight the dual gains of Bayesian KalmanNet in accuracy and uncertainty, we also include a tabular comparison of the

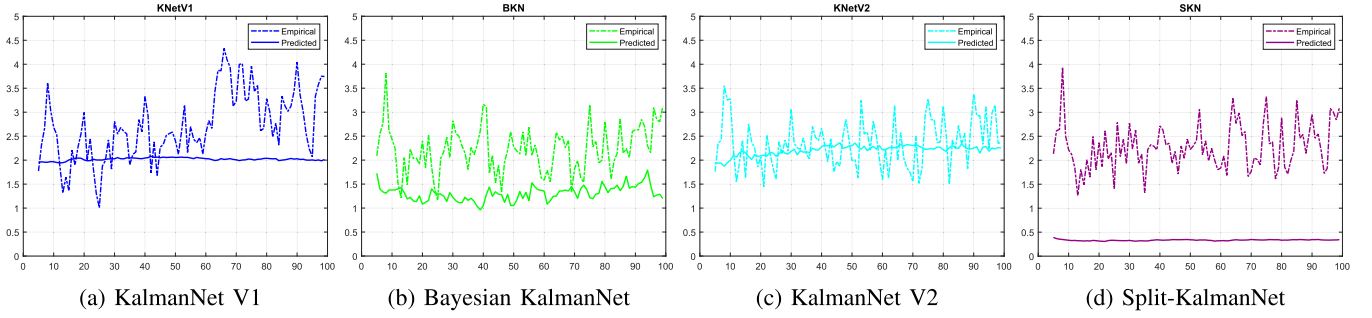(a) KalmanNet V1  (b) Bayesian KalmanNet  (c) KalmanNet V2  (d) Split-KalmanNet

Fig. 7. The EEC and APEC (as described in (27) and (28)) Vs. time for the model-based deep learning methods.

MSE and ANEES of the considered algorithms for the setting with mismatched $\mathbf{R}$ for fixed SNR of $1/r^2 = 10$ [dB]. The results, reported in Table II, clearly indicate on the ability of Bayesian KalmanNet to simultaneously provide accurate tracking alongside faithful uncertainty estimation, notably outperforming both frequentist DNN-based methods as well as the model-based EKF.

### D. Navigation SS Model

Our third scenario focuses on a more realistic SS as we track the location and velocity of a vehicle moving in a 2D plane. We assume its movement is in a CV manner. The system is modeled using the vehicle's position and velocity in the $x$- and $y$-directions as state variables.

Let the state vector $\mathbf{x}_t$ and state transition matrix $\mathbf{F}$ be defined as:

$$\mathbf{x}_t = \begin{bmatrix} x_t \\ y_t \\ \dot{x}_t \\ \dot{y}_t \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where:
- $x_t$ and $y_t$ represent the position of the vehicle in the $x$- and $y$-directions.
- $\dot{x}_t$ and $\dot{y}_t$ represent the velocity of the vehicle in the $x$- and $y$-directions.

Assume we can observe the position of the vehicle in the $x$- and $y$-directions, but not its velocity. The observation matrix $\mathbf{H}$ maps the state vector to the measurements:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix}$$

We generate a set of 100 trajectories, with 5 used for validation and 10 reserved for testing. The sampling interval parameter $\Delta t$ is normalized to unity.

The results for this setting are reported in Fig. 5. When examining the MSE in Fig. 5(a), it can be seen that both the EKF and the RKN not only fail to estimate the uncertainty, but also fail to produce an accurate state estimation compared to all other methods, especially at higher SNRs. The results in Fig. 5(b) shows that all methods resulted in an overconfident covariance prediction, although the both the frequentist and Bayesian model based deep learning methods manage to



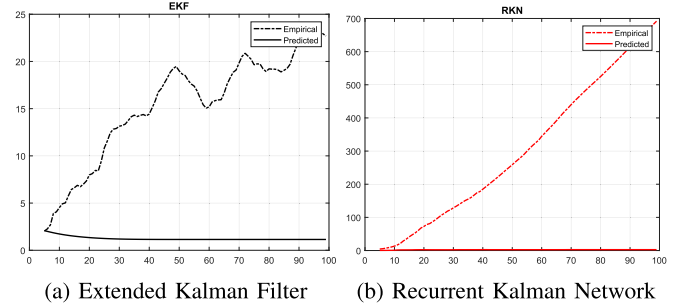(a) Extended Kalman Filter  (b) Recurrent Kalman Network

Fig. 8. The EEC and APEC (as described in 27 and 28) for the model-based EKF and model-agnostic RKN.

have a much better credibility with a more accurate uncertainty predictions.

In this section, we describe a single run of our constant velocity model under specific conditions. The true process noise covariance $\mathbf{Q}_{\mathrm{data}}$ was set such that the given process noise covariance $\mathbf{Q}_{\mathrm{model}}$ is underestimated by a factor of 100, i.e., $\mathbf{Q}_{\mathrm{data}} = 100 \cdot \mathbf{Q}_{\mathrm{model}}$. The noise covariance $\mathbf{R}$ was set to 10, which is $100\times$ greater than $\mathbf{Q}_{\mathrm{model}}$. This setup aims to analyze the system's behavior under significant model-process mismatch. Fig. 6 shows the predicted position x track of each method compared to the ground truth trajectory. This figure is a qualitative measure that shows the ability to track the trajectory. As can be seen, the model agnostic RKN failed to track the state along time, whereas all the model aware methods provide a reasonable state estimation.

To compare the different methods' performance in prediction the uncertainty, Figs. 7 and 8 show x-axis position APEC compared to its EEC. These metrics are achieved by performing a simulation with $N = 100$ trajectories, and using (27) and (28) to calculate APEC and EEC respectively. As can be seen, the model-based deep learning methods achieve better results than the model-based EKF and the model-agnostic RKN.

The results under model mismatch due to sampling mismatch are reported in Fig. 9. Fig. 9(b) shows clearly that Bayesian KalmanNet has supplied the most reliable error covariance estimation when given a mismatched model. The model based deep learning frequentist methods also results in a reliable error covariance prediction, whereas the EKF fails to extract the uncertainty, outperforming only the model agnostic RKN. The
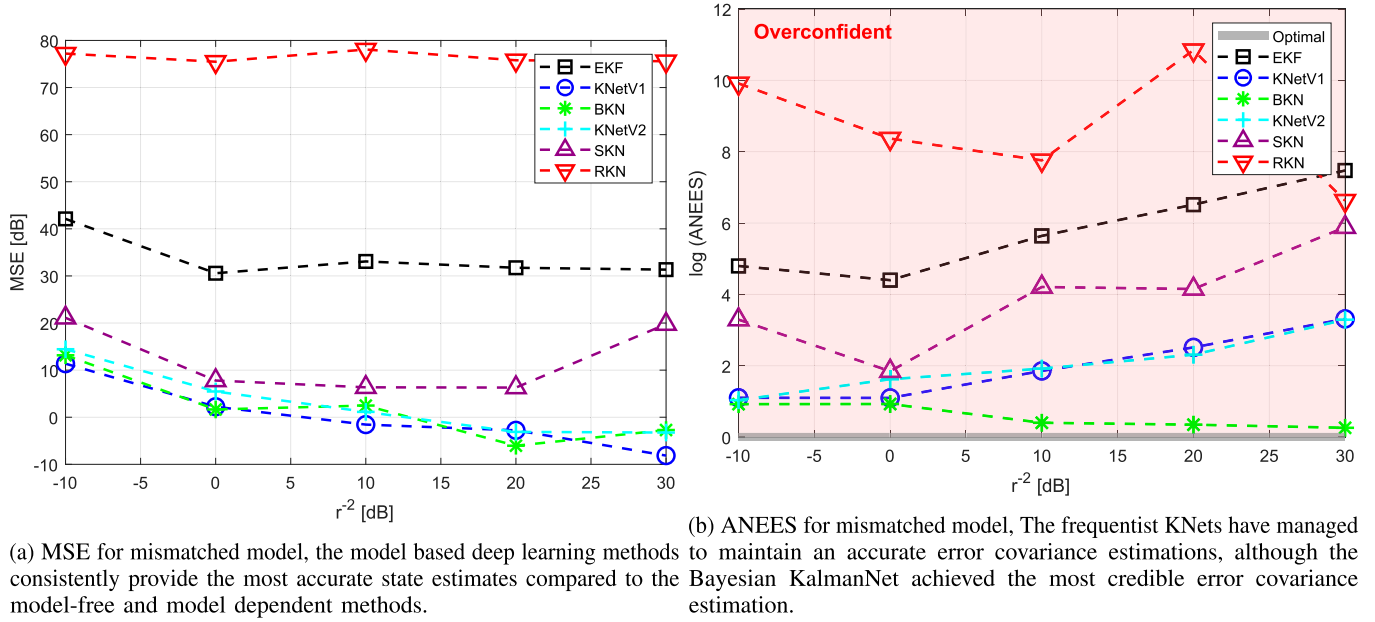
(a) MSE for mismatched model, the model based deep learning methods consistently provide the most accurate state estimates compared to the model-free and model dependent methods.

(b) ANEES for mismatched model, The frequentist KNets have managed to maintain an accurate error covariance estimations, although the Bayesian KalmanNet achieved the most credible error covariance estimation.

Fig. 9.   CV SS model - mismatched model.

results for the state estimation when given a mismatched model are very similar to the results in the error covariance estimation. Fig. 9(a) shows that the Bayesian KalmanNet has supplied the most reliable state estimation, the frequentist methods resulted in a reasonable state prediction, whereas the EKF and RKN failed to estimate an accurate state estimation.

### E. Inference Time Comparison

The results presented so far evaluate the considered methods in terms of their accuracy and uncertainty estimation performance. As Bayesian deep learning induces excessive latency in inference compared with frequentist models due to its inherent ensembling, we next evaluate inference time. The results in Table III report the inference latency of the different filtering methods across both a linear (canonical) and non-linear (pendulum) scenario. The platform on which all algorithms were tested and runtimes were computed is an Intel(R) Core(TM) i7-10510U CPU @ 1.80GHz 2.30 GHz. For Bayesian KalmanNet (BKN), The latency values reflect the total execution time required to process all $J$ realizations sequentially, i.e., the actual inference latency experienced in a serial execution setting. In the canonical SS model, the EKF demonstrates the fastest performance of all model aware methods, with the lowest average inference time of 5.279 milliseconds. The SKN, KNet V1, and V2 methods exhibit comparable speeds, with marginal differences around 5.3 milliseconds. These results suggest that these methods maintain efficient performance close to the traditional EKF, making them suitable for real-time applications where speed is critical.

The RKN achieves an average inference time of just 0.194 milliseconds. This presumably stems from the efficiency of

TABLE III
AVERAGE INFERENCE TIME [MSEC]

| Type | Model | EKF | KNet V1 | KNet V2 | SKN | RKN | BKN |
|---|---|---|---|---|---|---|---|
| Linear | Canonical | 5.279 | 5.303 | 5.279 | 5.315 | 0.194 | 104.88 |
| Non-Linear | Pendulum | 13.842 | N/A | N/A | 13.906 | 0.198 | 112.461 |

RNN software implementations in PyTorch. Bayesian Kalman-Net requires substantially more time, with an inference time of 104.88 milliseconds for the canonical model. This reflects the computational cost associated with more advanced uncertainty quantification and Bayesian learning techniques. While these methods enhance credibility and robustness, they introduce a significant computational burden, making them less suited for time-sensitive tasks.

In the pendulum scenario, the inference times increase across all methods due to the model's non-linearity. The EKF inference time rises to 13.842 milliseconds, which is still faster than Bayesian KalmanNet at 112.461 milliseconds. Notably, the KalmanNet versions do not have available inference times for the pendulum case, as the measurement matrix $\mathbf{H}$ in that SS is not full column rank, preventing the calculation of the error covariance. The SKN retains its relatively low inference time of 13.906 milliseconds, maintaining its appeal for applications requiring both computational efficiency and accuracy.

Overall, while the Bayesian KalmanNet offers substantial benefits in terms of credibility and robustness, these come at the cost of longer inference times. For real-time applications, a parallelization could be used to solve this issue, as the different inferences in the Bayesian KalmanNet are independent and thus could be computed in parallel. While there is no consistent algorithm that achieves the best performance in both accuracy and uncertainty throughout the various settings considered in our numerical study, it is clearly demonstrated that incorporating

domain knowledge notably facilitates uncertainty extraction. Moreover, it is systematically shown that Bayesian KalmanNet enables DNN-aided KFs to both extract the state and predict the error accurately in nonlinear, possibly mismatched SS models, and that its reliability is maintained across all the considered settings.

## VI. Conclusion

In this work, we studied uncertainty extraction in DNN-aided KFs. We identified three techniques based on different indicative features, and proposed two learning schemes that encourage learning to predict the error. Moreover, we have suggested a new method termed Bayesian KalmanNet incorporating Bayesian deep learning to a DNN augmented KF. Our numerical results show that incorporating domain knowledge notably facilitates uncertainty extraction, and that Bayesian KalmanNet allows DNN-aided KFs to both extract the state and predict the error accurately in non-linear possibly mismatched SS models.

## References

[1] Y. Dahan, G. Revach, J. Dunik, and Shlezinger, N., "Uncertainty quantification in deep learning based Kalman filters," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2024, pp 13121–13125.

[2] J. Durbin and S. J. Koopman, *Time Series Analysis by State Space Methods*. London, U.K.: Oxford Univ. Press, 2012.

[3] Y. Bar-Shalom, X. R. Li, and T. Kirubarajan, *Estimation With Applications to Tracking and Navigation: Theory Algorithms and Software*. Hoboken, NJ, USA: Wiley, 2004.

[4] S. Gannot and A. Yeredor, "The Kalman filter," *Springer Handbook of Speech Processing*, Berlin, Germany: Springer, 2008, pp. 135–160.

[5] B. Lim and S. Zohren, "Time-series forecasting with deep learning: A survey," *Philos. Trans. Roy. Soc. A*, vol. 379, no. 2194, 2021.

[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016

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

[8] A. Gu et al., "Combining recurrent, convolutional, and continuous-time models with linear state space layers," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 572–585.

[9] 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 *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 544–552.

[10] N. Shlezinger, J. Whang, Y. C. Eldar, and A. G. Dimakis, "Model-based deep learning," *Proc. IEEE*, vol. 111, no. 5, pp. 465–499, May 2023.

[11] N. Shlezinger, Y. C. Eldar, and S. P. Boyd, "Model-based deep learning: On the intersection of deep learning and optimization," *IEEE Access*, vol. 10, pp. 115384–115398, 2022.

[12] N. Shlezinger and Y. C. Eldar, "Model-based deep learning," *Found. Trends® Signal Process.*, vol. 17, no. 4, pp. 291–416, 2023.

[13] N. Shlezinger and T. Routtenberg, "Discriminative and generative learning for linear estimation of random signals [lecture notes]," *IEEE Signal Process. Mag.*, vol. 40, no. 6, pp. 75–82, Sep. 2023.

[14] T. Imbiriba, A. Demirkaya, J. Duník, O. Straka, D. Erdoğmuş, and P. Closas, "Hybrid neural network augmented physics-based models for nonlinear filtering," in *Proc. IEEE Int. Conf. Inf. Fusion (FUSION)*, 2022, pp. 1–6.

[15] T. Imbiriba, O. Straka, J. Duník, and P. Closas, "Augmented physics-based machine learning for navigation and tracking," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 60, no. 3, pp. 2692–2704, Jun. 2024.

[16] A. Chiuso and G. Pillonetto, "System identification: A machine learning perspective," *Annu. Rev. Control, Robot., Auton. Syst.*, vol. 2, pp. 281–304, May 2019.

[17] D. Gedon, N. Wahlström, T. B. Schön, and L. Ljung, "Deep state space models for nonlinear system identification," *IFAC-PapersOnLine*, vol. 54, no. 7, pp. 481–486, 2021.

[18] L. Ljung, C. Andersson, K. Tiels, and T. B. Schön, "Deep learning and system identification," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1175–1181, 2020.

[19] N. Shlezinger et al., "AI-aided Kalman filters," 2024, *arXiv:2410.12289*.

[20] 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.

[21] W. He, N. Williard, C. Chen, and M. Pecht, "State of charge estimation for Li-ion batteries using neural network modeling and unscented Kalman filter-based error cancellation," *Int. J. Electr. Power Energy Syst.*, vol. 62, pp. 783–791, Jun. 2014.

[22] 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.

[23] G. Choi, J. Park, N. Shlezinger, Y. C. Eldar, and N. Lee, "Split-KalmanNet: A robust model-based deep learning approach for state estimation," *IEEE Trans. Veh. Technol.*, vol. 72, no. 9, pp. 12326–12331, Sep. 2023.

[24] G. Revach, X. Ni, N. Shlezinger, R. J. van Sloun, and Y. C. Eldar, "RTSNet: Learning to smooth in partially known state-space models," *IEEE Trans. Signal Process.*, vol. 71, pp. 4441–4456, 2023.

[25] J. Song, W. Mei, Y. Xu, Q. Fu, and L. Bu, "Practical implementation of KalmanNet for accurate data fusion in integrated navigation," *IEEE Signal Process. Lett.*, vol. 31, pp. 1890–1894, 2024.

[26] J. Wang, X. Geng, and J. Xu, "Nonlinear Kalman filtering based on self-attention mechanism and lattice trajectory piecewise linear approximation," 2024, *arXiv:2404.03915*.

[27] A. Ghosh, A. Honoré, and S. Chatterjee, "DANSE: Data-driven nonlinear state estimation of model-free process in unsupervised learning setup," *IEEE Trans. Signal Process.*, vol. 72, pp. 1824–1838, 2024.

[28] D. Yang, F. Jiang, W. Wu, X. Fang, and M. Cao, "Low-complexity acoustic echo cancellation with neural Kalman filtering," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2023, pp. 1–5.

[29] A. Juárez-Lora, L. M. García-Sebastián, V. H. Ponce-Ponce, E. Rubio-Espino, H. Molina-Lozano, and H. Sossa, "Implementation of Kalman filtering with spiking neural networks," *Sensors*, vol. 22, no. 22, 2022, Art. no. 8845.

[30] I. Buchnik, G. Revach, D. Steger, R. J. Van Sloun, T. Routtenberg, and N. Shlezinger, "Latent-KalmanNet: Learned Kalman filtering for tracking from high-dimensional signals," *IEEE Trans. Signal Process.*, vol. 72, pp. 352–367, 2023.

[31] S. G. Casspi, O. Hüsser, G. Revach, and N. Shlezinger, "LQGNet: Hybrid model-based and data-driven linear quadratic stochastic control," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2023, pp. 1–5.

[32] A. N. Putri, C. Machbub, D. Mahayana, and E. Hidayat, "Data driven linear quadratic Gaussian control design," *IEEE Access*, vol. 11, pp. 24227–24237, 2023.

[33] A. Milstein, G. Revach, H. Deng, H. Morgenstern, and N. Shlezinger, "Neural augmented Kalman filtering with Bollinger bands for pairs trading," *IEEE Trans. Signal Process.*, vol. 72, pp. 1974–1988, 2024.

[34] R. L. Russell and C. Reale, "Multivariate uncertainty in deep learning," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 12, pp. 7937–7943, Dec. 2022.

[35] H. Wei, R. Xie, H. Cheng, L. Feng, B. An, and Y. Li, "Mitigating neural network overconfidence with logit normalization," in *Proc. Int. Conf. Mach. Learn. PMLR*, 2022, pp. 23631–23644.

[36] I. Klein, G. Revach, N. Shlezinger, J. E. Mehr, R. J. G. van Sloun, and Y. C. Eldar, "Uncertainty in data-driven Kalman filtering for partially known state-space models," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2022, pp. 3194–3198.

[37] A. Kendall and Y. Gal, "What uncertainties do we need in Bayesian deep learning for computer vision?" in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[38] L. V. Jospin, W. Buntine, F. Boussaid, H. Laga, and M. Bennamoun, "Hands-on Bayesian neural networks–a tutorial for deep learning users," *IEEE Comput. Intell. Mag.*, vol. 17, no. 2, pp. 29–48, May 2022.

[39] T. Raviv, S. Park, O. Simeone, and N. Shlezinger, "Modular model-based Bayesian learning for uncertainty-aware and reliable deep MIMO receivers," 2023, *arXiv:2302.02436*.

[40] S. J. Julier and J. K. Uhlmann, "Unscented filtering and nonlinear estimation," *Proc. IEEE*, vol. 92, no. 3, pp. 401–422, Mar. 2004.

[41] I. Arasaratnam and S. Haykin, "Cubature Kalman filters," *IEEE Trans. Autom. Control*, vol. 54, no. 6, pp. 1254–1269, Jun. 2009.

[42] I. Arasaratnam, S. Haykin, and R. J. Elliott, "Discrete-time nonlinear filtering algorithms using Gauss–Hermite quadrature," *Proc. IEEE*, vol. 95, no. 5, pp. 953–977, May 2007.

[43] J. Dunik, O. Straka, M. Simandl, and E. Blasch, "Random-point-based filters: Analysis and comparison in target tracking," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 51, no. 2, pp. 1403–1421, Apr. 2015.

[44] L. Lindemann, Y. Zhao, X. Yu, G. J. Pappas, and J. V. Deshmukh, "Formal verification and control with conformal prediction," 2024, *arXiv:2409.00536*.

[45] A. G. Wilson and P. Izmailov, "Bayesian deep learning and a probabilistic perspective of generalization," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 33, 2020, pp. 4697–4708.

[46] V. Fortuin, "Priors in Bayesian deep learning: A review," *Int. Statist. Rev.*, vol. 90, no. 3, pp. 563–591, 2022.

[47] Y. Gal, J. Hron, and A. Kendall, "Concrete dropout," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, 2017.

[48] J. Gawlikowski et al., "A survey of uncertainty in deep neural networks," *Artif. Intell. Rev.*, vol. 56, pp. 1513–1589, Jul. 2023.

[49] V. Lefkopoulos, M. Menner, A. Domahidi, and M. N. Zeilinger, "Interaction-aware motion prediction for autonomous driving: A multiple model Kalman filtering scheme," *IEEE Robot. Autom. Lett.*, vol. 6, no. 1, pp. 80–87, Jan. 2021.

[50] Y. Huang, Y. Zhang, Z. Wu, N. Li, and J. Chambers, "A novel adaptive Kalman filter with inaccurate process and measurement noise covariance matrices," *IEEE Trans. Autom. Control*, vol. 63, no. 2, pp. 594–601, Feb. 2017.

[51] H. Lan, J. Hu, Z. Wang, and Q. Cheng, "Variational nonlinear Kalman filtering with unknown process noise covariance," *IEEE Trans. Aerosp. Electron. Syst.*, vol. 59, no. 6, pp. 9177–9190, Dec. 2023.

[52] H. Lan, S. Zhao, J. Hu, Z. Wang, and J. Fu, "Joint state estimation and noise identification based on variational optimization," *IEEE Trans. Autom. Control*, early access, 2025, doi: 10.1109/TAC.2024.3524270.

[53] X. Ni, G. Revach, and N. Shlezinger, "Adaptive KalmanNet: Data-driven Kalman filter with fast adaptation," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2024, pp. 5970–5974.

[54] G. Revach, N. Shlezinger, T. Locher, X. Ni, R. J. van Sloun, and Y. C. Eldar, "Unsupervised learned Kalman filtering," in *Eur. Signal Process. Conf. (EUSIPCO)*. IEEE, 2022, pp. 1571–1575.

[55] M. Jones, P. Chang, and K. P. Murphy, "Bayesian online natural gradient (BONG)," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 37, 2024, pp. 131104–131153.

[56] Y. Gusakov, O. Simeone, T. Routtenberg, and N. Shlezinger, "Rapid online Bayesian learning for deep receivers," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2025, pp. 1–5.

[57] Y. Bar-Shalom and K. Birmiwal, "Consistency and robustness of PDAF for target tracking in cluttered environments," *Automatica*, vol. 19, no. 4, pp. 431–437, 1983.

[58] O. E. Drummond, X. R. Li, and C. He, "Comparison of various static multiple-model estimation algorithms," in *Proc. Defense, Secur., Sens.*, 1998, pp. 510–527.

[59] X. R. Li, Z. Zhao, and V. P. Jilkov, "Estimator's credibility and its measures," in *Proc. IFAC 15th World Congr.*, 2002.

**Yehonatan Dahan** (Student Member, IEEE) received the B.Sc. degrees (double degree with distinction) in mechanical engineering and materials engineering and the M.Sc. degree in electrical and computer engineering from Ben-Gurion University of the Negev, Israel, in 2023 and 2025, respectively. His research under the supervision of Dr. Nir Shlezinger include model-based deep learning, Bayesian filtering, state estimation, and their applications in navigation and situational awareness.

**Guy Revach** (Senior Member, IEEE) received the B.Sc. (*cum laude*) and M.Sc. degrees from Andrew and Erna Viterbi Department of Electrical & Computer Engineering, Technion – Israel Institute of Technology, Haifa, in 2008 and 2017, respectively. He completed his master's thesis under the supervision of Prof. Nahum Shimkin on planning for cooperative multiagents. Since 2019, he has been a Ph.D. student with the Institute for Signal and Information Processing (ISI), ETH Zürich, Switzerland, supervised by Prof. Dr. Hans-Andrea Loeliger. He is a Researcher with a proven industry track record as an Innovator and System Engineer. His main research focuses on the intersection of machine learning with signal processing, explicitly combining sound theoretical principles from classical signal processing with state-of-the-art machine learning algorithms for tracking and detection problems. Before joining ETH Zürich, he worked in the Israeli wireless communication industry for over ten years, initially as a real-time embedded software engineer and later as a software manager. He was the leading innovator behind state-of-the-art, software-defined radio (SDR) for wireless communication, which was game-changing and groundbreaking in size, weight, and power. As a System Engineer, he defined significant aspects of SDR requirements and architecture, including hardware, software, network, cyber defense, signal processing, data analysis, and control algorithms.

**Jindrich Dunik** (Senior Member, IEEE) is an Associate Professor with the Department of Cybernetics, University of West Bohemia, and a Senior Scientist with the Honeywell International, Aerospace Advanced Technology Europe. He focuses on state estimation, system identification, inertial, and satellite-based navigation systems, and integrity monitoring methods. He is a co-author of more than 80 technical papers (both journal and conference) and patents. He is a member of the IEEE AESS Navigation System Panel and serves as an Associate Editor of IEEE TRANSACTIONS ON AEROSPACE AND ELECTRONIC SYSTEMS. He received the Honeywell Technology Achievement Awards, the Werner von Siemens Excellence Award, and the IEEE AESS Harry Rowe Mimno Award.

**Nir Shlezinger** (Senior Member, IEEE) received the B.Sc., M.Sc., and Ph.D. degrees from Ben-Gurion University, Israel, in 2011, 2013, and 2017, respectively. He is an Assistant Professor with the School of Electrical and Computer Engineering, Ben-Gurion University, Israel. From 2017 to 2019, he was a Postdoctoral Researcher with Technion, and from 2019 to 2020 he was a Postdoctoral Researcher with Weizmann Institute of Science. His research interests include signal processing, machine learning, and communications. He was awarded the FGS prize for Outstanding Postdoctoral Research Achievements, the 2024 Krill Award for Young Researchers, the Toronto Early Career Award, and the 2024 IEEE Communications Society Fred W. Ellersick Prize.