SELF-SUPERVISED INFERENCE IN STATE-SPACE MODELS

David Ruhe
AI4Science, AMLab, Anton Pannekoek Institute
University of Amsterdam, The Netherlands
d.ruhe@uva.nl

Patrick Forrée
AI4Science, AMLab
University of Amsterdam, The Netherlands
p.d.forre@uva.nl

ABSTRACT

We perform approximate inference in state-space models with nonlinear state transitions. Without parameterizing a generative model, we apply Bayesian update formulas using a local linearity approximation parameterized by neural networks. This comes accompanied by a maximum likelihood objective that requires no supervision via uncorrupt observations or ground truth latent states. The optimization backpropagates through a recursion similar to the classical Kalman filter and smoother. Additionally, using an approximate conditional independence, we can perform smoothing without having to parameterize a separate model. In scientific applications, domain knowledge can give a linear approximation of the latent transition maps, which we can easily incorporate into our model. Usage of such domain knowledge is reflected in excellent results (despite our model’s simplicity) on the chaotic Lorenz system compared to fully supervised and variational inference methods. Finally, we show competitive results on an audio denoising experiment.

1 INTRODUCTION

Many sequential processes in industry and research involve noisy measurements that describe latent dynamics. A state-space model is a type of graphical model that effectively represents such noise-afflicted data (Bishop, 2006). The joint distribution is assumed to factorize according to a directed graph that encodes the dependency between variables using conditional probabilities. One is usually interested in performing inference, meaning to obtain reasonable estimates of the posterior distribution of the latent states or uncorrupt measurements. Approaches involving sampling (Neal et al., 2011), variational inference (Kingma & Welling, 2013), or belief propagation (Koller & Friedman, 2009) have been proposed before. Assuming a hidden Markov process (Koller & Friedman, 2009), the celebrated Kalman filter and smoother (Kalman, 1960; Rauch et al., 1965) are classical approaches to solving the posterior inference problem. However, the Markov assumption, together with linear Gaussian transition and emission probabilities, limit their flexibility. We present filtering and smoothing methods that are related to the classical Kalman filter updates but are augmented with flexible function estimators without using a constrained graphical model. By noting that the filtering and smoothing recursions can be back-propagated through, these estimators can be trained with a principled maximum-likelihood objective reminiscent of the noise2noise objective (Lehtinen et al., 2018; Laine et al., 2019). By using a locally linear transition distribution, the posterior distribution remains tractable despite the use of non-linear function estimators. Further, we show how a linearized smoothing procedure can be applied directly to the filtering distributions, discarding the need to train a separate model for smoothing.

To verify what is claimed, we perform three experiments. (1) A linear dynamics filtering experiment, where we show how our models approximate the optimal solution with sufficient data. We also report that including expert knowledge can yield better estimates of latent states. (2) A more challenging chaotic Lorenz smoothing experiment that shows how our models perform on par with recently proposed supervised models. (3) An audio denoising experiment that uses real-world noise showing practical applicability of the methods.

Our contributions can be summarized as follows.
1. We show that the posterior inference distribution of a state-space model is tractable while parameter estimation is performed by neural networks. This means that we can apply the classical recursive Bayesian updates, akin to the Kalman filter and smoother, with mild assumptions on the generative process.

2. Our proposed method is optimized using maximum likelihood in a self-supervised manner. That is, ground truth values of states and measurements are not assumed to be available for training. Still, despite our model’s simplicity, our experiments show that it performs better or on par with several baselines.

3. We show that the model can be combined with prior knowledge about the transition and emission probabilities, allowing for better applicability in low data regimes and incentivizing the model to provide more interpretable estimates of the latent states.

4. A linearized smoothing approach is presented that does not require explicit additional parameterization and learning of the smoothing distribution.

2 RELATED WORK

Becker et al. (2019) provide a detailed discussion of recent related work, which we build on here and in table 1. An early method that extends the earlier introduced Kalman filter by allowing nonlinear transitions and emissions is the Extended Kalman filter (Ljung, 1979). It is limited due to the naive approach to locally linearize the transition and emission distributions. Furthermore, the transition and emission mechanisms are usually assumed to be known, or estimated with Expectation Maximization (Moon, 1996). More flexible methods that combine deep learning with variational inference include Black Box Variational Inference (Archer et al., 2015), Structured Inference Networks (Krishnan et al., 2017), Kalman Variational Autoencoder (Fraccaro et al., 2017), Deep Variational Bayes Filters (Karl et al., 2017), Variational Sequential Monte Carlo (Naesseth et al., 2018) and Disentangled Sequential Autoencoder (Yingzhen & Mandt, 2018). However, the lower-bound objective makes the approach less scalable and accurate (see also Becker et al. (2019). Furthermore, all of the above methods explicitly assume a graphical model, imposing a strong but potentially harmful inductive bias. The BackpropKF (Haarnoja et al., 2016) and Recurrent Kalman Network (Becker et al., 2019) move away from variational inference and borrow Bayesian filtering techniques from the Kalman filter. We follow this direction but do not require supervision through ground truth latent states or uncorrupt emissions. Satorras et al. (2019) combine Kalman filters through message passing with graph neural networks to perform hybrid inference. We perform some of their experiments by also incorporating expert knowledge. However, contrary to their approach, we do not need supervision. Finally, concurrently to this work, Revach et al. (2021) develop KalmanNet. It proposes similar techniques but evaluates them in a supervised manner. The authors, however, do suggest that an unsupervised approach can also be feasible. Additionally, we more explicitly state what generative assumptions are required, then target the posterior distribution of interest, and develop the model and objective function from there. Moreover, the current paper includes linearized smoothing (section 6), parameterized smoothing (appendix A), and the recurrent model (appendix C). We also denote theoretical guarantees under the noise2noise objective.
We then put approximated with a linear Gaussian noise model:

\[ y_k = H x_k + e_k \]

Further implicit assumptions about the generative model are imposed via inference model choices (see section 7). Note that this factorization encodes several conditional independences like

\[ y_k \perp \perp (y_{k-1}, x_{k-1}) \mid x_k. \]

Typical models that follow these assumptions are linear dynamical systems, hidden Markov models, but also nonlinear state-space models with higher-order Markov chains in latent space like presented in fig. 2.

In contrast to other approaches (e.g., Krishnan et al. (2015); Johnson et al. (2016); Krishnan et al. (2017)) where one tries to model the latent dynamics with transition probabilities \( p(x_k \mid x_{k-1}) \) and possibly non-linear emission probabilities \( p(y_k \mid x_k) \), we go the other way around. We assume that all the non-linear dynamics are captured inside the latent distribution \( p(x_{0:K}) \), where at this point we make no further assumption about its factorization, and the emission probabilities are (well-approximated with) a linear Gaussian noise model:

\[ p(y_k \mid x_k) = \mathcal{N}(y_k \mid H x_k, R), \]

where the matrix \( H \) represents the measurement device and \( R \) is the covariance matrix of the independent additive noise. We make a brief argument why this assumption is not too restrictive. First, if one is interested in denoising corrupted measurements, any nonlinear emission can be captured directly inside the latent states \( x_k \). To see this, let \( z_k \in \mathbb{R}^N \) denote non-emitted state variables. We then put \( x_k := \begin{bmatrix} y_k \\ z_k \end{bmatrix} \), where \( y_k \) is computed by applying the nonlinear emission to \( z_k \). We thus include the measurements in the modeled latent state \( x_k \). Then we can put \( H := \begin{bmatrix} I_M \\ 0_{M \times (N-M)} \end{bmatrix} \). Second, techniques proposed by Laine et al. (2019) allow for non-Gaussian noise models, relaxing the need for assumption eq. (3). Third, we can locally linearize the
emission (Ljung, 1979). Finally, industrial or academic applications include cases where emissions are (sparse) Gaussian measurements and the challenging nonlinear dynamics occur in latent space. Examples can be found in MRI imaging (Lustig et al., 2007) and radio astronomy (Thompson et al., 2017).

4 PARAMETERIZATION

In this section, we show how we parameterize the inference model. A lot of the paper’s work relies on established Bayesian filtering machinery. However, for completeness, we like to prove how all the update steps remain valid while using neural networks for function estimation.

Given our noisy measurements \( y_{0:K} = (y_0, \ldots, y_K) \) we want to find good estimates for the latent states \( x_{0:K} = (x_0, \ldots, x_K) \), which generated \( y_{0:K} \). For this, we want to infer the marginal conditional distributions \( p(x_k \mid y_{\leq k}) \) or \( p(x_k \mid y_{<k}) \) (for forecasting), for an online inference approach (filtering); and \( p(x_k \mid y_{0:K}) \) or \( p(x_k \mid y_{<k}) \), for a full inference approach (smoothing). In the main body of the paper, we only consider filtering. Smoothing can be performed similarly, which is detailed in the supplementary material (appendix A).

We start with the following advantageous parameterization:

\[
p(x_k \mid x_{k-1}, y_{<k}) = \mathcal{N} \left( x_k \mid \hat{F}_{k} < k x_{k-1} + \hat{e}_{k < k}, \hat{Q}_{k < k} \right),
\]

where \( \hat{F}_{k < k} := \hat{F}_{k < k} (y_{<k}) \), \( \hat{e}_{k < k} := \hat{e}_{k < k} (y_{<k}) \) and \( \hat{Q}_{k < k} := \hat{Q}_{k < k} (y_{<k}) \) are parameterized with neural networks. Next, we have available

\[
p(x_{k-1} \mid y_{\leq (k-1)}) = \mathcal{N} \left( x_{k-1} \mid \hat{x}_{(k-1) \mid (k-1)}, \hat{P}_{(k-1) \mid (k-1)} \right),
\]

i.e., the previous time-step’s conditional marginal distribution of interest. For \( k = 1 \), this is some initialization. Otherwise, it is the result of the procedure we are currently describing. We use this distribution to evaluate the marginalization

\[
p(x_k \mid y_{<k}) = \int p(x_k \mid x_{k-1}, y_{<k}) p(x_{k-1} \mid y_{<k}) \, dx_{k-1}
\]

\[
= \mathcal{N} \left( x_k \mid \hat{x}_{k < k}, \hat{P}_{k < k} \right),
\]

with

\[
\hat{x}_{k < k} (y_{<k}) = \hat{F}_{k < k} \hat{x}_{k-1 \mid k-1} + \hat{e}_{k < k}, \quad \hat{P}_{k < k} (y_{<k}) = \hat{F}_{k < k} \hat{P}_{k-1 \mid k-1} \hat{F}_{k < k}^\top + \hat{Q}_{k < k}.
\]

Note that the distributions under the integral eq. (6) are jointly Gaussian only because of the parameterization eq. (4). Hence, we can evaluate the integral analytically.

Finally, to obtain the conditional \( p(x_k \mid y_{\leq k}) = p(x_k \mid y_k, y_{<k}) \) we use Bayes’ rule:

\[
p(x_k \mid y_k, y_{<k}) = \frac{p(y_k \mid x_k, y_{<k}) \cdot p(x_k \mid y_{<k})}{p(y_k \mid y_{<k})} \quad \overset{\text{eq. (2)}}{=} \quad \frac{p(y_k \mid x_k)}{p(y_k \mid y_{<k})} \cdot p(x_k \mid y_{<k}).
\]

Equation (3) and the result eq. (6) allow us to also get an analytic expression for

\[
p(x_k \mid y_{\leq k}) = \mathcal{N} \left( x_k \mid \hat{x}_{k \mid \leq k}, \hat{P}_{k \mid \leq k} \right)
\]

with the followingabbreviations:

\[
\hat{x}_{k \mid \leq k} (y_{\leq k}) := \hat{x}_{k \mid < k} + \hat{K}_k (y_k - H \hat{x}_{k \mid < k}), \quad \hat{P}_{k \mid \leq k} (y_{\leq k}) := \hat{P}_{k \mid < k} - \hat{K}_k H \hat{P}_{k \mid < k}.
\]

We introduce the Kalman gain matrix similar to the classical formulas:

\[
\hat{K}_k := \hat{P}_{k \mid < k} H^\top \left( H \hat{P}_{k \mid < k} H^\top + R \right)^{-1}.
\]

Note that taking the matrix inverse at this place in eq. (11) is more efficient than in the standard Gaussian formulas (for reference presented in appendix B) if \( M \leq N \), which holds for our experiments.
This completes the recursion: we can use eq. (9) for a new time-step \( k + 1 \) by plugging it back into eq. (6). We have shown how estimating a local linear transition using neural networks in eq. (4) ensures that all the recursive update steps from the Kalman filter analytically hold without specifying and estimating a generative model.

We note that we could also have parameterized

\[
p(x_k \mid y_{<k}) = \mathcal{N}(x_k \mid \hat{x}_{k\mid < k}(y_{<k}), \hat{P}_{k\mid < k}(y_{<k}))
\]

(12)

with \( \hat{x}_{k\mid < k}(y_{<k}) \) and \( \hat{P}_{k\mid < k}(y_{<k}) \) directly estimated by a neural network. This has the advantage that we do not rely on a local linear transition model. However, it also means that we are estimating \( x_k \) without any form of temporal regularization. Additionally, it is harder to incorporate prior knowledge about the transitions maps into such a model. Nonetheless, we detail this parameterization further in the supplementary material (appendix C) and include its performance in our experiments in section 8.

To conclude the section, we like to discuss some of the limitations of the approach. 1. The Gaussianity assumption of eq. (4) ensures but also restricts eq. (8) and eq. (9) to these forms. That is, we make a direct assumption about the form of the posterior \( p(x_k \mid y_{<k}) \). Defending our case, we like to point out that methods such as variational inference (Krishnan et al., 2017) or posterior regularization (Ganchev et al., 2010) also make assumptions (e.g., mean-field Gaussian) about the posterior. 2. Since we did not explicitly specify a factorization of \( p(x_{0:K}) \), we cannot ensure that the distributions we obtain from the above procedure form a posterior to the ground truth generative process. This does not mean, however, that we cannot perform accurate inference. Arguably, not making explicit assumptions about the generative process is preferred to making wrong assumptions and using those for modeling, which can be the case for variational auto-encoders. 3. The local linearity assumption is justifiable if the length between time-steps is sufficiently small. However, note that the model is more flexible than directly parameterizing eq. (12) (see appendix C) since it reduces to that case by putting \( F_k := 0 \) for all \( k \).

5 Fitting

We have shown in the previous section how parameterization of a local linear transition model leads to recursive estimation of \( p(x_k \mid y_{<k}) \) and \( p(x_k \mid y_{<k}) \) for all \( k \) using classical Bayesian filtering formulas. The inference is only effective if the estimates \( \hat{F}_{k\mid < k}, \hat{e}_{k\mid < k} \) and \( \hat{Q}_{k\mid < k} \) from eq. (4) are accurate. We can use the parameterization \( p(x_k \mid y_{<k}) \) of eq. (6), the emission model \( p(y_k \mid x_k) = \mathcal{N}(y_k \mid H x_k, R) \) from eq. (3), and the factorization from eq. (1) to see that an analytic form of the log-likelihood of the data emerges:

\[
p(y_k \mid y_{<k}) = \int p(y_k \mid x_k) p(x_k \mid y_{<k}) d x_k
\]

(13)

\[
= \mathcal{N}(y_k \mid H \hat{x}_{k\mid < k}(y_{<k}), H \hat{P}_{k\mid < k}(y_{<k}) H^T + R),
\]
\[
\log p(y_{0:K}) = \sum_{k=0}^{K} \log \mathcal{N} \left( y_k \mid H \hat{x}_{k|<k}(y_{<k}), H \hat{P}_{k|<k}(y_{<k}) H^\top + R \right). \tag{14}
\]

If we put \( \hat{y}_{k|<k} := H \hat{x}_{k|<k}(y_{<k}) \) and \( \hat{M}_{k|<k} := H \hat{P}_{k|<k}(y_{<k}) H^\top + R \), then the maximum-likelihood objective leads to the following loss function, which we can minimize using gradient descent methods w.r.t. all model parameters:

\[
\mathcal{L} := \sum_{k=0}^{K} (\hat{y}_{k|<k} - y_k)^\top \hat{M}_{k|<k}^{-1} (\hat{y}_{k|<k} - y_k) + \log \det \hat{M}_{k|<k}.
\tag{15}
\]

Note that each term in the sum above represents a one-step-ahead self-supervised error term. We thus minimize the prediction residuals \( \hat{y}_{k|<k}(y_{<k}) - y_k \) in a norm that is inversely scaled with the above covariance matrix, plus a regularizing determinant term, which prevents the covariance matrix from diverging. The arisen loss function is similar to the noise2noise (Lehtinen et al., 2018; Krull et al., 2019; Batson & Royer, 2019; Laine et al., 2019) objective from computer vision literature, combined with a locally linear transition model like Becker et al. (2019). We show in appendix D that this objective will yield correct results (meaning estimating the ground-truth \( x_k \)) if the noise is independent with \( \mathbb{E}[y_k \mid H x_k] = H x_k \). A similar procedure in the causality literature is given by Schölkopf et al. (2016). An algorithmic presentation of performing inference and fitting is presented in appendix E.

Note that after fitting the parameters to the data, eq. (6) can directly be used to do one-step ahead forecasting. Forecasting an arbitrary number of time steps is possible by plugging the new value \( \hat{x}_{K+1|K} \) via \( y_{K+1} := H \hat{x}_{K+1|K} \) back into the recurrent model, and so on. This is not a generative model but merely a convenience that we deemed worth mentioning.

6 LINEARIZED SMOOTHING

So far, we have only discussed how to perform filtering. Recall that for smoothing, we are instead interested in the quantity \( p(x_k \mid y_{0:K}) \). A smoothing strategy highly similar to the methods described earlier can be obtained by explicitly parameterizing such a model, which we detail in the supplementary material. Here, we introduce a linearized smoothing procedure. The essential advantage is that no additional model has to be trained, which can be costly. Several algorithms stemming from the Kalman filter literature can be applied, such as the RTS smoother (Rauch et al., 1965) and the two-filter smoother (Kitagawa, 1994). To enable this, we need to assume that the conditional mutual information \( I(x_{k-1}; y_{k|K} \mid x_k, y_{0:k-1}) \) is small for all \( k = 1, \ldots, K \). In other words, we assume that we approximately have the following conditional independences:

\[
x_{k-1} \perp \perp y_{k:K} \mid (x_k, y_{0:k-1}). \tag{16}
\]

To explain the motivation for this requirement, consider the model in fig. 2. If the states of \( y_{0:k-1} \) and \( x_k \) are known, then the additional information that \( y_{k|K} \) has about the latent variable \( x_{k-1} \) would need to be passed along the unblocked deeper paths like \( y_{k+1} \leftarrow x_{k+1} \leftarrow e_{k+1} \leftarrow e_k \leftarrow x_{k-1} \). Then the assumption of small \( I(x_{k-1}; y_{k|K} \mid x_k, y_{0:k-1}) \) can be interpreted as that the deeper paths transport less information than the lower direct paths. If we consider all edges to the \( x_k \)‘s as linear and the edges to the \( e_k \)‘s as non-linear maps, the above could be interpreted as an information-theoretic version of expressing that the non-linear correction terms are small compared to the linear parts in the functional relations between the variables.

We will now show that under the earlier assumptions and eq. (16) we get a Gaussian approximation:

\[
p(x_k \mid y_{0:K}) \approx \mathcal{N} \left( x_k \mid \hat{z}_k, \hat{G}_k \right). \tag{17}
\]

We will do backward induction with \( \hat{z}_K := \hat{x}_{K|\leq K} \) and \( \hat{G}_K := \hat{P}_{K|\leq K} \). To propagate this to previous time steps \( k - 1 \) we use the chain rule:

\[
p(x_{k-1} \mid y_{0:K}) = \int p(x_{k-1} \mid x_k, y_{0:K}) p(x_k \mid y_{0:K}) \, dx_k,
\tag{17}
\]

where the second term is known by backward induction and for the first term we make use of the approximate conditional independence from eq. (16) to get

\[
p(x_{k-1} \mid x_k, y_{0:K}) = p(x_{k-1} \mid y_{k|K}, x_k, y_{0:k-1}) \overset{\text{eq. (16)}}{=} p(x_{k-1} \mid x_k, y_{0:k-1}). \tag{18}
\]
The latter was shown to be Gaussian in section 4:

\[
p(x_{k-1}, x_k \mid y_{0:k-1}) = \mathcal{N}\left(\begin{bmatrix} x_{k-1} \\ \hat{x}_{k-1} \end{bmatrix}, \begin{bmatrix} P_{k-1} & \hat{P}_{k-1} \\ \hat{x}_{k-1} & P_{k-1} \end{bmatrix}\right)
\]

(19)

By use of the usual formulas for Gaussians and the reverse Kalman gain matrix \( \hat{J}_{k-1|k} \) we arrive at the following update formulas, \( k = K, \ldots, 1 \), with \( \hat{z}_k := \hat{x}_{K|\leq k} \) and \( \hat{G}_k := P_{K|\leq k} \):

\[
\hat{J}_{k-1|k} := \hat{P}_{k-1|\leq k-1} F_{k|< k}^{-1} \hat{P}_{k|< k},
\]

(20)

\[
\hat{G}_{k-1} := \hat{P}_{k-1|\leq k-1} + \hat{J}_{k-1|k} \left( \hat{P}_{k|\leq k} - \hat{P}_{k|< k} \right) \hat{J}_{k-1|k}^\top,
\]

(21)

\[
\hat{z}_{k-1} := \hat{x}_{k-1|\leq k-1} + \hat{J}_{k-1|k} (\hat{z}_k - \hat{x}_{k|\leq k}).
\]

(22)

As such, we can perform inference for all \( k = 0, \ldots, K \) with \( p(x_k \mid y_{0:k}) \approx \mathcal{N}\left( x_k \mid \hat{z}_k, \hat{G}_k \right) \).

Algorithmically, the above is presented in appendix E.

7 Recurrent Neural Network

Before going into the experiments section, we briefly explain how we specifically estimate the functions \( F_{k|< k}(y_{< k}) \), \( \hat{e}_{k|< k}(y_{< k}) \) and \( \hat{Q}_{k|< k}(y_{< k}) \) that parameterize the transition probability \( p(x_k \mid y_{< k}, x_{k-1}) \) (eq. (4)). The choice of the model here implicitly makes further assumptions about the generative model. If we consider neural networks, the temporal nature of the data suggests recurrent neural networks (Graves et al., 2013), convolutional neural networks (Kalchbrenner et al., 2014), or transformer architectures (Vaswani et al., 2017). Additionally, if the data is image-based, one might further make use of convolutions. For our experiments, we use a Gated Recurrent Unit (GRU) network (Cho et al., 2014), that recursively encodes hidden representations. Therefore, we put

\[
\hat{Q}_{k|< k} := L_k L_k^\top,
\]

\[
\hat{F}_k := f(h_k^-), \quad \hat{e}_k := \text{GRU}(y_{k-1}^-, h_k^-),
\]

(23)

where \( \hat{L}_k \) is a Cholesky factor and \( f \) is a multi-layer perceptron decoder.

8 Experiments

We perform three experiments, as motivated in section 1. Technical details on the setup of the experiments can be found in the supplementary material (appendix F). We refer to the model detailed in section 4 as the recursive filter, as it uses the Bayesian update recursion. For smoothing experiments, we use recursive smoother. The model obtained by parameterizing \( p(x_k \mid y_{< k}) \) directly (eq. (12)) is referred to as the recursive filter or recursive smoother, as it only employs recurrent neural networks (and no Bayesian recursion) to estimate said density directly.

8.1 Linear Dynamics

In the linear Gaussian case, it is known that the Kalman filter will give the optimal solution. Thus, we can get a lower bound on the test loss. In this toy experiment, we simulate particle tracking under linear dynamics and noisy measurements of the location. We use Newtonian physics equations as prior knowledge. We generate trajectories \( \mathcal{T} = \{x_{0:K}, y_{0:K}\} \) with \( x_k \in \mathbb{R}^6 \) and \( y_k \in \mathbb{R}^2 \) according to the differential equations:

\[
\dot{x} = Ax = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -c & 1 \\ 0 & -\tau c & 0 \end{bmatrix} \begin{bmatrix} p \\ v \\ \alpha \end{bmatrix},
\]

(24)

We obtain sparse, noisy measurements \( y_k = H x_k + r \) with \( r \sim \mathcal{N}(0, R) \). \( H \) is a selection matrix that returns a two-dimensional position vector. We run this experiment in a filtering setting, i.e.,
we only use past observations. We compare against (1) the raw, noisy measurements which inherently deviate from the clean measurements, (2) the Kalman filter solution where we optimized the transition covariance matrix using clean data (hence supervised), (3) the optimal solution, which is a Kalman filter with ground truth parameters performing exact inference. To estimate the transition maps for the Kalman filter, we use the standard Taylor series of $e^{A \Delta t}$ up to the first order.

Additionally, we use this expert knowledge as an inductive bias for the recursive filter’s transition maps.

In fig. 4 we depict the test mean squared error (MSE, lower is better) as a function of the number of training samples. Given enough data, the self-supervised models approximate the optimal solution arbitrarily well. Our recursive model significantly outperforms both the Kalman filter and the inference model in the low-data regime by using incorporated expert knowledge. Additionally, we report that the recursive model’s distance to the ground truth latent states is much closer to the optimal solution than both the inference model and Kalman filter. Specifically, we report average mean squared errors of 0.685 for the inference model, 0.241 for the Kalman filter, 0.161 for the recursive model compared to 0.135 for the optimal Kalman filter. Finally, it is worth noting that the recursive model has much less variance as a function of its initialization.

8.2 Lorenz Equations

We simulate a Lorenz system according to

$$\dot{x} = Ax = \begin{bmatrix} -\sigma & \sigma & 0 \\ \rho - x_1 & -1 & 0 \\ x_2 & 0 & -\beta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. \tag{25}$$

We have $H = I$, $x \in \mathbb{R}^3$ and $y \in \mathbb{R}^3$. The Lorenz equations model atmospheric convection and form a classic example of chaos. Therefore, performing inference is much more complex than in the linear case. This time, we perform smoothing (see appendix A) and compare against (1) the raw measurements, (2) a supervised Extended Kalman smoother (Ljung, 1979), (3) the variational inference approach of Krishnan et al. (2017), (4) the supervised recursive model of Satorras et al. (2019). Our models include a recurrent smoother, a recursive smoother, and the recursive filter with linearized smoothing (section 6). Transition maps for the Extended Kalman smoother and the recursive models are obtained by taking a second-order Taylor series of $e^{A \Delta t}$. For the supervised extended Kalman filter, we again optimize its covariance estimate using ground truth data.

In fig. 5 we plot the test mean squared error (MSE, lower is better) as a function of the number of examples available for training. It is clear that our methods approach the ground truth states with
Table 2: Considers the audio denoising experiment (section 8.3). Test mean squared error (MSE, lower is better) per model and noise subset. Blue numbers indicate second-best performing models.


10 Ethics Statement

The paper presents a simple method to perform inference using noisy sequential data. Applications can be found throughout society, e.g., tracking particles, denoising images or audio, or estimating system states. While many such examples are for good, there are applications with ethically debatable motivations. Among these could be tracking humans or denoising purposefully corrupted data (e.g., to hide one’s identity).

11 Reproducibility Statement

We are in the process of releasing code for the current work. For clarity and reproducibility, the presented methods are available as algorithms in the supplementary material. Furthermore, we made explicit wherever we needed to make an approximation or an assumption.

References


A PARAMETERIZED SMOOTHING

In the main body of the paper, we showed how to parameterize the model for recursive estimation of \( p(x_k \mid y_{\leq k}), k = 0, \ldots, K \). Additionally, we provided a linearized smoothing procedure that yields \( p(x_k \mid y_{1:K}) \). The disadvantage clearly is the linearization. Here, we show how we can recursively estimate \( p(x_k \mid y_{1:K}) \) in a similar sense to the filtering setting.

First, we put
\[
p(x_k \mid x_{k-1}, y_{-k}) = \mathcal{N}(x_k \mid \hat{F}_{k|k} x_{k-1} + \hat{e}_{k|k}, \hat{Q}_{k|k}),
\]
(26)

where \( \hat{F}_{k|k}(y_{-k}) \) and \( \hat{e}_{k|k}(y_{-k}) \) and \( \hat{Q}_{k|k}(y_{-k}) \) are two-sided recurrent neural network outputs similar to section 7. We compute the distribution of interest as follows:
\[
p(x_k \mid y_{0:K}) = \frac{p(y_k \mid x_k)p(x_k \mid y_{-k})}{p(y_k \mid y_{-k})}
\]
(27)

\[
\propto p(y_k \mid x_k) \int p(x_k, x_{k-1} \mid y_{-k}) dx_{k-1}
\]
(28)

\[
= p(y_k \mid x_k) \int p(x_k \mid x_{k-1}, y_{-k}) p(x_{k-1} \mid y_{-k}) dx_{k-1}
\]
(29)

\[
x_{k-1} \approx \int p(y_k \mid x_k) \int p(x_k \mid x_{k-1}, y_{-k}) p(x_{k-1} \mid y_{0:K}) dx_{k-1}
\]
(30)

where we make the approximation to compute eq. (27) efficiently and recursively. It is justified if \( I(x_{k-1}; y_k \mid y_{-k}) < \epsilon \) for small \( \epsilon \). That is, the additional information that \( y_k \) conveys about \( x_{k-1} \) is marginal if we have all other data. Let
\[
p(x_{k-1} \mid y_{0:K}) := \mathcal{N}(x_{k-1} \mid \bar{x}_{k-1|0:K}, \bar{P}_{k-1|0:K})
\]
(31)

be previous time-step’s posterior. Then put
\[
p(x_k \mid x_{k-1}, y_{-k}) := \mathcal{N}(x_k \mid \hat{F}_{k|k-1} x_{k-1} + \hat{e}_{k|k-1}, \hat{Q}_{k|k-1})
\]
(32)

where we left out the arguments for the following quantities estimated by an RNN.
\[
\begin{bmatrix}
\hat{F}_{k|k-1}(y_{-k}) \\
\hat{e}_{k|k-1}(y_{-k}) \\
\hat{Q}_{k|k-1}(y_{-k})
\end{bmatrix}
:= f(h_k^+, h_k^-)
\]
\[
\begin{bmatrix}
h_k^- \\
h_k^+
\end{bmatrix}
:= \text{GRU}(h_{k-1}^+, y_{k-1})
\]
(33)
Applying the integral in eq. (30) we get
\[ \int p(x_k| x_{k-1}, y_{-k})p(x_{k-1}| y_{0:K})dx_{k-1} = \mathcal{N}\left(x_k| \hat{x}_{k|\cdot-k}, \hat{P}_{k|\cdot-k}\right) \] (34)
where we put
\[ \hat{x}_{k|\cdot-k} := \hat{F}_{k|\cdot-k}\hat{x}_{k-1|0:K} + \hat{e}_{k|\cdot-k}, \quad \hat{P}_{k|\cdot-k} := \hat{F}_{k|\cdot-k}\hat{P}_{k-1|0:K}\hat{F}_{k|\cdot-k}^\top + \hat{Q}_{k|\cdot-k} \] (35)
A data likelihood can be computed as follows
\[ p(y_k| y_{-k}) = \int p(y_k| x_k)p(x_k| y_{-k})dx_k \]
\[ = \int p(y_k| x_k)\int p(x_k| x_{k-1}, y_{-k})dx_kdx_{k-1} \]
\[ = \int p(y_k| x_k)\int p(x_k| x_{k-1}, y_{-k})p(x_{k-1}| y_{-k})dx_kdx_{k-1} \]
where we made the same approximation eq. (30) as before. It evaluates to
\[ \int p(y_k| x_k)p(x_k| y_{-k})dx_k = \mathcal{N}\left(y_k| \hat{y}_{k|\cdot-k}, \hat{M}_{k|\cdot-k}\right) \] (40)
with
\[ \hat{y}_{k|\cdot-k} := H\hat{x}_{k|\cdot-k}, \quad \hat{M}_{k|\cdot-k} := H\hat{P}_{k|\cdot-k}H^\top + R \] (41)
For fitting to the data we now would use a maximum-pseudo-likelihood (Gourieroux et al., 1984) approach by maximizing:
\[ \sum_{k=0}^{K} \log p(y_k| y_{-k}) = \sum_{k=0}^{K} \log \mathcal{N}\left(y_k| H\hat{x}_{k|\cdot-k}(y_{-k}), H\hat{P}_{k|\cdot-k}(y_{-k})H^\top + R\right), \] (42)
leading to minimizing the following self-supervised loss function:
\[ \mathcal{L} := \sum_{k=0}^{K} \left[ (\hat{y}_{k|\cdot-k} - y_k)^\top \hat{M}_{k|\cdot-k}^{-1} (\hat{y}_{k|\cdot-k} - y_k) + \log \det \hat{M}_{k|\cdot-k} \right], \] (43)
where \( \hat{y}_{k|\cdot-k} := H\hat{x}_{k|\cdot-k}(y_{-k}) \) and \( \hat{M}_{k|\cdot-k} := H\hat{P}_{k|\cdot-k}(y_{-k})H^\top + R \).

B GAUSSIAN CONDITIONING FORMULAS
Since many of the calculations used in this work are based on the Gaussian conditioning formulas, we provide them here. If
\[ p(x) = \mathcal{N}(x| \mu, P), \]
\[ p(y| x) = \mathcal{N}(y| Hx + b, R), \] (44) (45)
then
\[ p(y) = \mathcal{N}(y| H\mu + b, R + HPH^\top), \]
\[ p(x| y) = \mathcal{N}(x| \Sigma[H^\top R^{-1}(y - b) + P^{-1}\mu], \Sigma), \] (46) (47)
with
\[ \Sigma = (P^{-1} + H^\top R^{-1}H)^{-1}. \] (48)
C Direct Parameterization of \( p(x \mid y_{<k}) \)

We show here how to directly parameterize \( p(x \mid y_{<k}) \) and \( p(x \mid y_{-k}) \). This parameterization is referred to as the recurrent model in our experiments. The procedure is rather straightforward. For filtering, we put \( p(x_k \mid y_{<k}) = \mathcal{N}\left(x_k \mid \hat{x}_{k-1}(y_{<k}), P_{k|<k}(y_{<k})\right) \). We model:

\[
\hat{x}_{k|<k} := e_k, \quad \hat{P}_{k|<k} := L_k L_k^\top, \quad \begin{bmatrix} e_k \\ L_k \end{bmatrix} := f(h_k^-),
\]

where \( L_k \) is a cholesky factor and \( f \) is a multi-layer perceptron. The argument \( h_k^- \) is recursively given by

\[
h_k^- := \text{GRU}(y_{k-1}, h_{k-1}^-),
\]

where we employ a Gated Recurrent Unit (GRU) network (Cho et al., 2014). Note that this parameterization is equivalent to the model described in the main paper with \( F_k := 0 \).

For smoothing, \( p(x_k \mid y_{-k}) = \mathcal{N}\left(x_k \mid \hat{x}_{k|k}(y_{-k}), P_{k|k}(y_{-k})\right) \),

\[
\hat{x}_{k|k} := e_k, \quad \hat{P}_{k|k} := L_k L_k^\top, \quad \begin{bmatrix} e_k \\ L_k \end{bmatrix} := f(h_k^+),
\]

where \( L_k \) is a cholesky factor and \( f \) is a multi-layer perceptron.

\[
h_k^+ := \text{GRU}(y_{k+1}, h_{k+1}^-), \quad h_k^- := \text{GRU}(y_{k-1}, h_{k-1}^-).
\]

Once \( p(x_k \mid y_{<k}) \) (filtering) or \( p(x_k \mid y_{-k}) \) (smoothing) is obtained, all the procedures for inference and optimization described in the main paper and appendix A remain the same.

D Bias-Variance-Noise Decomposition of the Self-Supervised Generalization Error

Any estimate \( \hat{x}_k = \hat{x}_k(y_{-k}) \) for \( x_k \) that is not dependent on \( y_k \) will give us a bias-variance-noise decomposition of the generalization error. Note that this setting covers both the filtering and smoothing case. Define "optimal model" \( \hat{y}_k^* := \mathbb{E}[y_k \mid y_{-k}] \), then under the specified generative model (section 3) we have

\[
\mathbb{E} \left[ \| \hat{y}_k - y_k \|^2_2 \mid y_{-k} \right] = \mathbb{E} \left[ \| \hat{y}_k - \hat{y}_k^* + \hat{y}_k^* - y_k \|^2_2 \mid y_{-k} \right]
\]

\[
= \mathbb{E} \left[ \| \hat{y}_k - \hat{y}_k^* \|^2_2 \mid y_{-k} \right] + \text{Var} \left( y_k \mid y_{-k} \right) + 2 \mathbb{E} \left[ (\hat{y}_k - \hat{y}_k^*)^\top (\hat{y}_k - y_k) \mid y_{-k} \right] + 2 \mathbb{E} \left[ (\hat{y}_k - y_k)^\top (\hat{y}_k - y_k) \mid y_{-k} \right]
\]

\[
= \| \hat{y}_k - \hat{y}_k^* \|^2_2 + \text{Var} \left( y_k \mid y_{-k} \right) + 2 \mathbb{E} \left[ (\hat{y}_k - y_k)^\top (\hat{y}_k - y_k) \mid y_{-k} \right]
\]

\[
\text{Var} \left( y_k \mid y_{-k} \right) = \mathbb{E} \left[ \| \hat{y}_k^* - H x_k - n_k \|^2_2 \mid y_{-k} \right]
\]

\[
= \mathbb{E} \left[ \| \hat{y}_k^* - H x_k \|^2_2 \mid y_{-k} \right] + \mathbb{E} \left[ \| n_k \|^2_2 \mid y_{-k} \right] + 2 \mathbb{E} \left[ (H x_k - \hat{y}_k^*)^\top n_k \mid y_{-k} \right]
\]

\[
= \mathbb{E} \left[ \| \hat{y}_k^* - H x_k \|^2_2 \mid y_{-k} \right] + \text{tr}(R)
\]

Thus,

\[
\mathbb{E} \left[ \| \hat{y}_k - y_k \|^2_2 \mid y_{-k} \right] = \| \hat{y}_k - \hat{y}_k^* \|^2_2 + \text{Var} \left( y_k \mid y_{-k} \right) + 2 \mathbb{E} \left[ (\hat{y}_k^* - H x_k)^\top n_k \mid y_{-k} \right] + \text{tr}(R)
\]

Note that \( \hat{y}_k^* = \mathbb{E}[Hx_k \mid y_{-k}] = H \mathbb{E}[x_k \mid y_{-k}] \). Then, define our model \( \hat{y}_k := H \hat{x}_k \). The reducible part of the error becomes

\[
\| \hat{y}_k - \hat{y}_k^* \|^2_2 = \| H (\mathbb{E}[x_k \mid y_{-k}] - \hat{x}_k) \|^2_2
\]
For this reason, the model output $\hat{x}_k$ approaches the optimal model under minimization of the self-supervised error (perturbed by $H$).

Additionally, the model $\hat{y}_k$ approaches $Hx_k$ (the uncorrupted measurement) under this criterion.

\begin{align}
E \left[ \| \hat{y}_k - y_k \|^2 \right] &= E \left[ \| \hat{y}_k - Hx_k + Hx_k - y_k \|^2 \right] \\
&= E \left[ \| \hat{y}_k - Hx_k \|^2 \right] + E \left[ \| Hx_k - y_k \|^2 \right] + 2 E \left[ (\hat{y}_k - Hx_k)^\top (Hx_k - y_k) \right] \\
&= E \left[ \| \hat{y}_k - Hx_k \|^2 \right] + E \left[ \| Hx_k - y_k \|^2 \right] + 2 E \left[ (\hat{y}_k - Hx_k)^\top E \left[ Hx_k - y_k \right] \right] \\
&= E \left[ \| \hat{y}_k - Hx_k \|^2 \right] + \text{tr}(R) + E \left[ \| Hx_k - y_k \|^2 \right] \tag{64}
\end{align}

If we then take $\hat{y}_k := H\hat{x}_k$ then the reducible part of the error approximates the true $x_k$ (perturbed by $H$).

\begin{align}
E \left[ \| \hat{y}_k - Hx_k \|^2 \right] &= E \left[ \| H(\hat{x}_k - x_k) \|^2 \right] \\
&= E \left[ \| H(\hat{x}_k - x_k) \|^2 \right] \tag{65}
\end{align}
E  ALGORITHMS

Algorithm 1: Recursive Filter (Inference)

\textbf{input} : Data (time-series) \( y_{0:K} = (y_0, \ldots, y_K) \), emission matrices \( H \) and \( R \), parameters \( \phi \).

\textbf{output}:
1. For training: Loss value \( L_K \) and its gradient \( \nabla_{\phi} L_K \).
2. For inference: \( \hat{x}_{k|k} \) and \( \hat{P}_{k|k} \) for \( k = 0, \ldots, K \). Inference is done via:
   \[ p(x_k \mid y_{\leq k}) = \mathcal{N}(x_k \mid \hat{x}_{k|k}, \hat{P}_{k|k}) \]
3. For linearized smoothing (section 6): \( \hat{F}_{k|k} \), \( \hat{P}_{k|k} \), \( \hat{x}_{k|k} \) and \( \hat{P}_{k|\leq k} \), for \( k = 0, \ldots, K \).
4. For forecasting: \( \hat{x}_{K+1|<(K+1)} \) and \( \hat{P}_{K+1|<(K+1)} \). Forecasting is done via:
   \[ p(x_{K+1} \mid y_{0:K}) = \mathcal{N}(x_{K+1} \mid \hat{x}_{K+1|<(K+1)}, \hat{P}_{K+1|<(K+1)}) \]

\begin{algorithm}[h]
\SetAlgoLined
\KwData{\( h_0 := 0 \), \( L_{-1} := 0 \), \( P_{0|0} := Q_0 \), \( \bar{x}_{0|0} := \bar{e}_0 \)}
\For{\( k \in 0, \ldots, K \)}{
  \( \hat{B}_{k|k} := (H \hat{P}_{k|k} H^\top + R)^{-1} \)
  \( \hat{y}_{k|k} := H \hat{x}_{k|k} \)
  \( L_k := L_{k-1} + (y_k - \hat{y}_{k|k})^\top \hat{B}_{k|k} (y_k - \hat{y}_{k|k}) - \log \det \hat{B}_{k|k} \)
  \( \hat{K}_k := P_{k|k} H^\top \hat{B}_{k|k} \)
  \( \hat{P}_{k|\leq k} := \hat{P}_{k|k} - \hat{K}_k H \hat{P}_{k|k} \)
  \( \hat{x}_{k|\leq k} := \hat{x}_{k|k} + \hat{K}_k (y_k - \hat{y}_{k|k}) \)
  \( h_{k+1} := \text{GRU}_\phi(h_k, y_k) \)
  \[ \begin{bmatrix} \hat{e}_{k+1|<(k+1)} \\ \hat{F}_{k+1|<(k+1)} \\ \hat{L}_{k+1|<(k+1)} \end{bmatrix} := f_\phi(h_{k+1}) \]
  \( Q_{k+1|<(k+1)} := L_{k+1|<(k+1)} \hat{L}_{k+1|<(k+1)}^\top \)
  \( P_{k+1|<(k+1)} := \hat{F}_{k+1|<(k+1)} \hat{P}_{k|\leq k} \hat{F}_{k+1|<(k+1)}^\top + Q_{k+1|<(k+1)} \)
  \( \hat{x}_{k+1|<(k+1)} := \hat{F}_{k+1|<(k+1)} \hat{x}_{k|\leq k} + \hat{e}_{k+1|<(k+1)} \)
}
\end{algorithm}

For the training case we use backpropagation through the above loop to compute \( \nabla_{\phi} L_K \).

Algorithm 2: Recursive Filter (Training)

\textbf{input} : Data (time-series) \( y_{0:K} = (y_0, \ldots, y_K) \), emission matrices \( H \) and \( R \), initialized parameters \( \phi_0 \), number of training rounds \( I \).

\textbf{output}: Model parameters \( \phi^* \) for inference at test-time.

\For{\( i \in 0, \ldots, I \)}{
  Obtain \( L_K^{(i)} \) and \( \nabla_{\phi} L_K^{(i)} \) from algorithm 1.
  Run preferred optimizer step to update parameters \( \phi \) with \( \nabla_{\phi} L_K^{(i)} \) (and \( L_K^{(i)} \)).
}
Algorithm 3: Parameterized Recursive Smoother (Inference)

\[ h_{0}^{\rightarrow} := 0 \]
\[ h_{K+1}^{\rightarrow} := 0 \]
\[ L_{-1}^{(i)} := 0 \]
\[ P_{0:0} := \hat{Q}_{0} \]
\[ x_{0:0} := \hat{e}_{0} \]
for \( k \) in \( 0, \ldots, K \)
do
\[ \hat{B}_{k|-k} := \left( H \hat{P}_{k|-k} H^{\top} + R \right)^{-1} \]
\[ \hat{y}_{k|-k} := H \hat{x}_{k|-k} \]
\[ L_{k}^{(i)} := L_{k-1}^{(i)} + (y_{k} - \hat{y}_{k|-k})^{\top} \hat{B}_{k|-k} (y_{k} - \hat{y}_{k|-k}) - \log \det \hat{B}_{k|-k} \]
\[ \hat{K}_{k} := \hat{P}_{k|-k} H^{\top} \hat{B}_{k|-k} \]
\[ \hat{P}_{k|-k} := \hat{P}_{k|-k} - \hat{K}_{k} H \hat{P}_{k|-k} \]
\[ \hat{x}_{k|-k} := \hat{x}_{k|-k} + \hat{K}_{k} (y_{k} - \hat{y}_{k|-k}) \]
\[ h_{k+1}^{\rightarrow} := \text{GRU}_{\phi}(h_{k}^{\rightarrow}, y_{k}) \]
\[ h_{k+1}^{\leftarrow} := \text{GRU}_{\phi}(h_{k}^{\rightarrow}, y_{k+1}) \]
\[ \left[ \hat{e}_{k+1|-k+1}^{(k+1)} \right] := f_{\phi}(h_{k+1}^{\rightarrow}, h_{k+1}^{\leftarrow}) \]
\[ \hat{F}_{k+1|-k+1}^{(k+1)} := \hat{F}_{k+1|-k+1}^{(k+1)} \]
\[ \hat{L}_{k+1|-k+1} := \hat{L}_{k+1|-k+1}^{(k+1)} \]
end

For training case we also use backpropagation through the above loop to compute \( \nabla_{\phi} L_{K} \)

Algorithm 4: Parameterized Recursive Smoother (Training)

\[ h_{0}^{\rightarrow} := 0 \]
\[ h_{0}^{\leftarrow} := 0 \]
\[ L_{-1}^{(i)} := 0 \]
\[ P_{0:0} := \hat{Q}_{0} \]
\[ x_{0:0} := \hat{e}_{0} \]
for \( i \) in \( 1, \ldots, I \)
do
Obtain \( \nabla_{\phi} L_{K}^{(i)} \) from algorithm 3.
Run preferred optimizer step.
end
Algorithm 5: Linearized Smoother

input : Values of: $\hat{F}_k|<k$, $\hat{P}_k|<k$, $\hat{P}_k|\leq k$, $\hat{x}_k|\leq k$ for all $k = 0, \ldots, K$, obtained from the recursive filter algorithm.

output: Linearly smoothed distributions $p(x_k \mid y_{0:K}) = N(\hat{x}_k, \hat{G}_k)$ for all $k = 0, \ldots, K$.

$\hat{z}_K := \hat{x}_K|\leq K$

$\hat{G}_K := \hat{P}_K|\leq K$

for $k = K, \ldots, 1$ do

$\hat{J}_{k-1|k} := \hat{P}_{k-1|\leq k-1} \hat{F}_{k|k}^\top \hat{P}_{k|\leq k}^{-1}$

$\hat{G}_{k-1} := \hat{P}_{k-1|\leq k-1} + \hat{J}_{k-1|k} \left( \hat{P}_{k|\leq k} - \hat{P}_{k|\leq k} \right) \hat{J}_{k-1|k}^\top$

$\hat{z}_{k-1} := \hat{x}_{k-1|\leq k-1} + \hat{J}_{k-1|k} (\hat{z}_k - \hat{x}_k|\leq k)$

end

Algorithm 6: Recurrent Smoother (Training)

input : Training data (time-series) $y_{0:K} = (y_0, \ldots, y_K)$, emission function $N(Hx_k, R)$, initialized parameters $\phi_0$, number of training iterations $n$.

output: Optimized parameters $\phi^*$

for $i$ in $1$ to $n$ do

$h_0 := 0$

$h_{K+1} := 0$

$L^{(i)} := 0$

for $k$ in $0$ to $K$ do

$h_k^+ := \text{GRU}_\phi(h_{k-1}^+, y_k)$

$h_k^- := \text{GRU}_\phi(h_{k-1}^-, y_k)$

$[\hat{x}_k|<k, \hat{L}_k|<k] := f_\phi(h_k^+, h_k^-)$

$\hat{P}_{k|<k} := \hat{L}_{k|<k} \hat{L}_{k|<k}^\top$

$\hat{y}_{k|<k} := H\hat{x}_{k|<k}$

$\hat{B}_{k|<k} := \left( H\hat{P}_{k|<k} H^\top + R \right)^{-1}$

$L^{(i)}_k := L^{(i)}_{k-1} + (y_k - \hat{y}_{k|<k})^\top \hat{B}_{k|<k} (y_k - \hat{y}_{k|<k}) - \log \det \hat{B}_{k|<k}$

end

Compute $\nabla_\phi L^{(i)}_K$ and apply SGD step with respect to all model parameters, which amounts to backpropagation through the above calculations.

end

For the filter variant, the steps that involve the backward direction ($\leftarrow$) are left out.
Algorithm 7: Recurrent Smoother (Inference)

**input**: Test data $y_{0:K} = (y_0, \ldots, y_K)$, trained parameters $\phi^*$, emission function $\mathcal{N}(Hx, R)$.

**output**: Inferred posteriors $p(x_k \mid y_{0:K}) = \mathcal{N}(\hat{x}_k, \hat{P}_k)$ for all $k$

$h_0 := 0$

$h_{K+1} := 0$

for $k$ in 0 to $K$

\[ h_{k}^\rightarrow := \text{GRU}_\phi(h_{k-1}^\rightarrow, y_{k-1}) \]
\[ h_{k}^\leftarrow := \text{GRU}_\phi(h_{k-1}^\leftarrow, y_{k+1}) \]

\[
\begin{bmatrix}
\hat{x}_{k|-k} \\
\hat{L}_{k|-k}
\end{bmatrix} := f_\phi(h_{k}^\rightarrow, h_{k}^\leftarrow)
\]

\[ \hat{P}_{k|-k} := \hat{L}_{k|-k} \hat{L}_{k|-k}^\top \]

\[ K_k := \hat{P}_{k|-k} H^\top \left( H \hat{P}_{k|-k} H^\top + R \right)^{-1} \]

\[ \hat{x}_{k|0:K} := \hat{x}_{k|-k} + K_k(y_k - H \hat{x}_{k|-k}) \]

\[ \hat{P}_{k|0:K} := \hat{P}_{k|-k} - K_k H \hat{P}_{k|-k} \]

end

*For the filter variant, the steps that involve the backward direction ($\leftarrow$) are left out.*
F EXPERIMENTS: DETAILS

F.1 LINEAR DYNAMICS

As specified in the main paper, the dynamics are according to

\[
\dot{x} = Ax = \begin{bmatrix} 0 & 1 & 0 \\ 0 & -c & 1 \\ 0 & -\tau c & 0 \end{bmatrix} \begin{bmatrix} p \\ v \\ a \end{bmatrix}. 
\] (66)

Since these are linear transitions, we can calculate any transition directly using

\[
x(t + \Delta t) = e^{A\Delta t}x(t).
\]

We used \(c = 0.06\), \(\tau = 0.17\), \(\Delta t := 1\) and covariance

\[
\bar{Q} = 0.1^2 \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 3 \end{bmatrix}. 
\] (68)

The matrix exponential is computed using Bader et al. (2019). The parameters for the emission distribution:

\[
H := \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad R := 0.5^2 \cdot \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}. 
\] (69)

We simulate a \(K := 131,072\) trajectory for training, \(K := 16,384\) trajectory for validation and \(K := 32,768\) for testing. The \(\tilde{F}\) that is used in the (non-optimal) Kalman filter and recursive model is computed as follows:

\[
e^{A\Delta t} \approx \tilde{F} := \sum_{n=0}^{1} (\Delta t A^n) / n! 
\] (70)

F.2 LORENZ EQUATIONS

We simulate a Lorenz system according to

\[
\dot{x} = Ax = \begin{bmatrix} -\sigma & \sigma & 0 \\ \rho - x_1 & -1 & 0 \\ x_2 & 0 & -\beta \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}. 
\] (71)

We integrate the system using \(dt = 0.00001\) and sample it uniformly at \(\Delta t = 0.05\). We use \(\rho = 28\), \(\sigma = 10\), \(\beta = 8/3\). The transition in \(\Delta t\) arbitrary time-steps is linearly approximated by a Taylor expansion and used in the Kalman smoother and recursive models.

\[
e^{A|x_k,\Delta t} \approx \tilde{F}_k := \sum_{n=0}^{2} (\Delta t A|x_k)^n / n! 
\] (72)

We simulate \(K := 131,072\) steps for training, \(K := 32,768\) for testing and \(K := 16,384\) for validation. We have \(H := I\) and thus \(x \in \mathbb{R}^3\) and \(y \in \mathbb{R}^3\). We use \(R := 0.5^2 I\).

G PARAMETERIZED SMOOTHING: ALTERNATIVE POSTERIOR EVALUATION

We would like to point out that the distribution \(p(x_k \mid y_{0:k})\) can be obtained without making assumption eq. (30), which we stretch out here. Initial experiments showed that using these calculations the model did not converge as smoothly as when using the ones stated before. However, it could be of interest to further investigate. Returning to the posterior of interest

\[
p(x_k \mid y_{0:k}) = \int p(x_k \mid x_{k-1}, y_{0:k}) p(x_{k-1} \mid y_{0:k}) \, dx_{k-1} 
\] (73)

\[
= \int \frac{p(y_k \mid x_k)}{p(y_k \mid x_{k-1}, y_{-k})} p(x_k \mid x_{k-1}, y_{-k}) p(x_{k-1} \mid y_{0:k}) \, dx_{k-1}. 
\] (74)
We have
\[
p(x_{k-1} \mid y_{0:K}) = \mathcal{N}(x_{k-1} \mid \hat{x}_{k-1:0:K}(y_{0:K}), \hat{P}_{k-1:0:K}(y_{0:K}))
\]  
(75)

as the previous time-step’s posterior.
\[
p(x_k \mid x_{k-1}, y_{-k}) = \mathcal{N}(x_k \mid \hat{F}_{k-1} x_{k-1} + \hat{e}_{k|\bar{y}_k}, \hat{Q}_{k|\bar{y}_k})
\]  
(76)

where \(\hat{F}_{k|\bar{y}_k}(y_{-k}), \hat{e}_{k|\bar{y}_k}(y_{-k})\) and \(\hat{Q}_{k|\bar{y}_k}(y_{-k})\) are estimated by a neural network. Combining this with noise model eq. (3) we get:
\[
p(x_k \mid x_{k-1}, y_{0:K}) = \mathcal{N}(x_k \mid \hat{F}_{k|\bar{y}_k} x_{k-1} + \hat{e}_{k|\bar{y}_k} + K_k (y_k - H (\hat{F}_{k|\bar{y}_k} x_{k-1} + \hat{e}_{k|\bar{y}_k})), \hat{Q}_{k|\bar{y}_k} - K_k H \hat{Q}_{k|\bar{y}_k})
\]  
(77)

where we directly applied the Woodbury matrix identity to obtain Kalman gain matrix
\[
K_k := \hat{Q}_{k|\bar{y}_k} H^\top \left( H \hat{Q}_{k|\bar{y}_k} H^\top + R \right)^{-1}
\]  
(79)

Then, applying the integral we get:
\[
p(x_k \mid y_{0:K}) = \mathcal{N}(x_k \mid \hat{x}_{k|0:K}, \hat{P}_{k|0:K}),
\]  
(80)

with:
\[
\hat{P}_{k|0:K} = (I - K_n H) \hat{F}_{k|\bar{y}_k} \hat{P}_{k-1|0:K} \hat{F}_{k|\bar{y}_k}^\top (I - K_n H) + (I - K_n H) \hat{Q}_{k|\bar{y}_k},
\]  
(81)

and
\[
\hat{x}_{k|0:K} = \hat{F}_{k|\bar{y}_k} \hat{x}_{k-1|0:K} + \hat{e}_{k|\bar{y}_k} + K_k (y_k - H (\hat{F}_{k|\bar{y}_k} \hat{x}_{k-1|0:K} + \hat{e}_{k|\bar{y}_k}))
\]  
(82)

\[
= (I - K_n H) \left( \hat{F}_{k|\bar{y}_k} \hat{x}_{k-1|0:K} + \hat{e}_{k|\bar{y}_k} \right) + K_k y_k.
\]  
(83)