IMPLICIT MAXIMUM A POSTERIORI FILTERING VIA ADAPTIVE OPTIMIZATION

Gianluca Bencomo[1], **Jake Snell**[1] **& Thomas Griffiths**[1,2]
[1]Department of Computer Science, Princeton University
[2]Department of Psychology, Princeton University
{gb5435,js2523,tomg}@princeton.edu

ABSTRACT

Bayesian filtering approximates the true underlying behavior of a time-varying system by inverting an explicit generative model to convert noisy measurements into state estimates. This process typically requires matrix storage, inversion, and multiplication or Monte Carlo estimation, none of which are practical in high-dimensional state spaces such as the weight spaces of artificial neural networks. Here, we consider the standard Bayesian filtering problem as optimization over a time-varying objective. Instead of maintaining matrices for the filtering equations or simulating particles, we specify an optimizer that defines the Bayesian filter *implicitly*. In the linear-Gaussian setting, we show that every Kalman filter has an equivalent formulation using K steps of gradient descent. In the nonlinear setting, our experiments demonstrate that our framework results in filters that are effective, robust, and scalable to high-dimensional systems, comparing well against the standard toolbox of Bayesian filtering solutions. We suggest that it is easier to fine-tune an optimizer than it is to specify the correct filtering equations, making our framework an attractive option for high-dimensional filtering problems.

1 INTRODUCTION

Time-varying systems are ubiquitous in science, engineering, and machine learning. Consider a multielectrode array receiving raw voltage signals from thousands of neurons during a visual perception task. The goal is to infer some underlying neural state that is not directly observable, such that we can draw connections between neural activity and visual perception, but raw voltage signals are a sparse representation of neural activity that is shrouded in noise. To confound the problem further, the underlying neural state changes throughout time in both expected and unexpected ways. This problem, and most time-varying prediction problems, can be formalized as a probablistic state space model where latent variables evolve over time and emit observations (Simon, 2006). One solution to such a problem is to apply a Bayesian filter to infer the values of the latent variables from the observations.

Typically, Bayesian filters require matrix storage, inversion, and multiplication or the storage of *particles*, which are samples from the filtering distribution at every time step. In large state and observation spaces, the computational costs associated with both of these approaches render them impractical (Raitoharju & Piché, 2019). In addition, since most time-varying systems do not have ground-truth states available, accurately estimating the process noise – the variability of the underlying latent variables – is nearly impossible. This problem is exacerbated in continuous systems where we need to perform numerical integration, which introduces additional uncertainty. Existing Bayesian filters are extremely sensitive to process noise and dynamics misspecification, making them hard to use in these settings (Mehra, 1972). Most filters are highly structured, requiring explicit specification of distributional assumptions. To be usable in practical settings, we need a system that remains effective when scaled to large state spaces (being applicable to weather systems, neural recording, or even modeling the evolving weights of a neural network) and is robust to the misspecification of the process noise and dynamics (which is almost guaranteed).

The practical problems that arise when training deep neural networks (DNNs) are not dissimilar from those that make Bayesian filters difficult to work with: high-dimensional state spaces, nonlin-

earities, and nonconvexity. These pressures have made way for adaptive optimizers (Duchi et al., 2011; Tieleman et al., 2012; Zeiler, 2012; Kingma & Ba, 2014) that offer a largely prescriptive solution for training DNNs, succeeding many antiquated approaches to optimizing neural networks that include applying Bayesian filtering equations (Haykin, 2004). To efficiently and effectively overcome the difficulties of training DNNs (Glorot & Bengio, 2010), modern optimizers invariably making some use of (i) the empirical Fisher information to *crudely* incorporate curvature at very little computational cost (Martens, 2020) and (ii) momentum to increase stability and the rate of convergence (Sutskever et al., 2013). We show that these same methods can be applied to Bayesian filtering in an extremely practical and theoretically-motivated framework.

In the following sections, we cast Bayesian filtering as optimization over a time-varying objective and show that much of the information that is *explicitly* defined by structured filters can be *implicitly* internalized by optimizer hyperparameters. In doing so, we can solve many of the scalability challenges associated with Bayesian filters and make Bayesian filtering easier to implement, especially for the well-versed deep learning practitioner. We show that our proposed method, Implicit Maximum a Posteriori (MAP) Filtering, is robust to misspecification and matches or outperforms classical filters on baseline tasks. We also show that it naturally scales up to high-dimensional problems such as adapting the weights of a convolutional neural network, and that it performs particularly well in this setting. We argue that it is easier to specify an optimizer than it is to correctly identify classical filtering equations, and our results illustrate the benefits of this implicit filtering approach.

2 BACKGROUND

We begin with an introduction to the filtering problem. We assume a discrete-time state space model with a Markov process over the states $\mathbf{x}_{0:T} \triangleq \mathbf{x}_0, \mathbf{x}_1, \ldots, \mathbf{x}_T$ and a sequence of observations $\mathbf{y}_{1:T} \triangleq \mathbf{y}_1, \mathbf{y}_2, \ldots, \mathbf{y}_T$ which are conditionally independent given the corresponding state. The joint probability of the states and observations is

$$p(\mathbf{x}_{0:T}, \mathbf{y}_{1:T}) = p(\mathbf{x}_0) \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{y}_t|\mathbf{x}_t), \tag{1}$$

where $p(\mathbf{x}_0)$ is the *initial distribution*, $p(\mathbf{x}_t|\mathbf{x}_{t-1})$ is the *transition distribution*, and $p(\mathbf{y}_t|\mathbf{x}_t)$ is the *likelihood*. The posterior over the joint state $\mathbf{x}_{1:t}$ of the system given all of the observations up until the current time $\mathbf{y}_{1:t}$ is given by:

$$p(\mathbf{x}_{1:t}|\mathbf{y}_{1:t}) = \int p(\mathbf{x}_0) \prod_{s=1}^{t} \frac{p(\mathbf{x}_s|\mathbf{x}_{s-1}) p(\mathbf{y}_s|\mathbf{x}_s)}{p(\mathbf{y}_s|\mathbf{y}_{1:s-1})} \, d\mathbf{x}_0, \tag{2}$$

where $p(\mathbf{y}_1|\mathbf{y}_{1:0}) \triangleq p(\mathbf{y}_1)$. In sequential estimation, we are often only interested in the marginal of the current state $p(\mathbf{x}_t|\mathbf{y}_{1:t})$, known as the *filtering distribution*. We normally want to compute or approximate this distribution such that the requisite computational resources do not depend on the length of the sequence. This can be accomplished by first initializing $p(\mathbf{x}_0|\mathbf{y}_{1:0}) \triangleq p(\mathbf{x}_0)$, and then forming a *predictive distribution* via the Chapman-Kolmogorov equation,

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t|\mathbf{x}_{t-1}) p(\mathbf{x}_{t-1}|\mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1}, \tag{3}$$

and updating that predictive distribution, in light of measurements $\mathbf{y}_t$, by a simple application of Bayes' rule for every time step $t = 1, \ldots, T$,

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \frac{p(\mathbf{y}_t|\mathbf{x}_t) p(\mathbf{x}_t|\mathbf{y}_{1:t-1})}{p(\mathbf{y}_t|\mathbf{y}_{1:t-1})}. \tag{4}$$

2.1 KALMAN FILTER FOR LINEAR-GAUSSIAN SYSTEMS

When the transition distribution and likelihood are linear and Gaussian, we can compute optimal estimates in closed-form via the Kalman filtering equations (Kalman, 1960). Assume the following state space model:

$$p(\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_0 \mid \boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0) \tag{5}$$

$$p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t \mid \boldsymbol{F}_{t-1}\mathbf{x}_{t-1}, \boldsymbol{Q}_{t-1}) \tag{6}$$

$$p(\mathbf{y}_t \mid \mathbf{x}_t) = \mathcal{N}(\mathbf{y}_t \mid \mathbf{H}_t\mathbf{x}_t, \mathbf{R}_t) \tag{7}$$

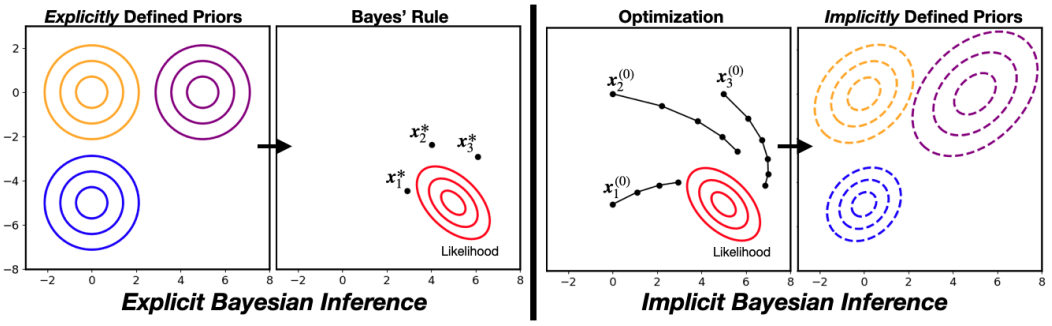

Figure 1: *Explicit Bayesian inference* **(left)** uses Bayes' rule to balance information from the likelihood and explicitly defined priors. x_i^* are MAP estimates of three different posteriors, respective to the priors shown in blue, yellow, and purple. *Implicit Bayesian inference* **(right)** defines an optimization procedure that turns the crank in the opposite direction of Bayes' rule. x_1, x_2, x_3, show 3 different trajectories of gradient descent with a fixed learning rate over 3, 4, and 5 steps, respectively. Taking the solutions produced by gradient descent as MAP estimates, we can use the corresponding gradient steps to determine the prior distributions that they implied (Santos, 1996).

The Kalman filter propagates the filtering distribution from the previous timestep $p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_{t-1} \mid \boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_{t-1})$ with the predict step (see Appendix A.1.1 for full details),

$$p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) = \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^-, \boldsymbol{\Sigma}_t^-), \text{ where } \boldsymbol{\mu}_t^- = \boldsymbol{F}_{t-1}\boldsymbol{\mu}_{t-1}, \boldsymbol{\Sigma}_t^- = \boldsymbol{F}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{F}_{t-1}^\top + \boldsymbol{Q}_{t-1}, \quad (8)$$

and then updates that distribution as follows:

$$p(\mathbf{x}_t|\mathbf{y}_{1:t}) = \mathcal{N}(\mathbf{x}_t \mid \boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t), \text{ where } \boldsymbol{\mu}_t = \boldsymbol{\mu}_t^- + \boldsymbol{K}_t\mathbf{v}_t, \boldsymbol{\Sigma}_t = \left[(\boldsymbol{\Sigma}_t^-)^{-1} + \boldsymbol{H}_t^\top \boldsymbol{R}_t^{-1}\boldsymbol{H}_t\right]^{-1}, \quad (9)$$

$\mathbf{v}_t = \mathbf{y}_t - \boldsymbol{H}_t\boldsymbol{\mu}_t^-$ is the prediction error, and $\boldsymbol{K}_t = \boldsymbol{\Sigma}_t^-\boldsymbol{H}^\top(\boldsymbol{H}_t\boldsymbol{\Sigma}_t^-\boldsymbol{H}_t^\top + \boldsymbol{R}_t)^{-1}$ is known as the Kalman gain.

2.2 Approximate Bayesian Filtering for Nonlinear Systems

Guarantees for optimality break down as we move to real-world systems, which are inherently nonlinear and only approximately Gaussian at best. The broader Bayesian filtering community has been motivated by developing methods that can operate in these settings (Särkkä & Svensson, 2023). Classical solutions include the extended Kalman filter (EKF) and iterated extended Kalman filter (iEKF) that make use of Jacobian matrices to linearize the dynamics and observation models and use the Kalman filtering equations thereafter (Gelb et al., 1974). These are arguably the most popular filters but only work well in modestly nonlinear systems on short timescales (Julier & Uhlmann, 2004). The unscented Kalman filter (UKF) (Julier et al., 1995; Julier & Uhlmann, 2004), like the particle filter (PF) (Doucet et al., 2001), is better equipped to handle highly nonlinear systems but both of these methods suffer from the curse of dimensionality and sensitivity to misspecified dynamics. Nonetheless, EKFs, iEKFs, UKFs, and PFs are the most prominent filtering solutions found throughout industry and research, leveraging an often expensive bag of tricks to treat nonlinearities.

3 Bayesian Filtering as Optimization

The update step (4) in a Bayesian filter is a straightforward application of Bayes' rule with the predictive distribution $p(\mathbf{x}_t|\mathbf{y}_{1:t-1})$ playing the role of the prior. In linear-Gaussian systems, the posterior distribution is available in closed-form (9) but one could alternatively achieve an equivalent result via gradient descent due to its connection to regularized least-squares (Santos, 1996). In this section, we describe this connection and how an optimization procedure can implicitly define a prior distribution, a kind of dual formulation of Bayes' theorem (see Figure 3). Such a formulation reduces Bayesian inference to point estimates, but this is more practical for nonlinear systems whose filtering distributions $p(\mathbf{x}_t|\mathbf{y}_{1:t})$ are not analytically tractable in general.

3.1 Finding the Posterior Mode via Truncated Gradient Descent

The mode of the filtering distribution can be expressed as the minimizer of the negative log posterior:

$$\boldsymbol{\mu}_t = \arg\min_{\mathbf{x}_t} \bar{\ell}_t(\mathbf{x}_t), \text{ where } \bar{\ell}_t(\mathbf{x}_t) = -\log p(\mathbf{y}_t|\mathbf{x}_t) - \log p(\mathbf{x}_t|\mathbf{y}_{1:t-1}). \tag{10}$$

In the case of the linear-Gaussian system, the regularized loss function is (up to an additive constant):

$$\bar{\ell}_t(\mathbf{x}_t) = \frac{1}{2}\|\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t\|^2_{\mathbf{R}_t} + \frac{1}{2}\|\mathbf{x}_t - \boldsymbol{\mu}_t^-\|^2_{\boldsymbol{\Sigma}_t^-}. \tag{11}$$

Santos (1996) observed that the minimizer of a regularized least squares problem as in (11) can recovered by performing truncated gradient descent on the likelihood term alone. Specifically, let $\boldsymbol{\mu}_t^{(0)} = \boldsymbol{\mu}_t^-$ and define the recursion

$$\boldsymbol{\mu}_t^{(k+1)} \leftarrow \boldsymbol{\mu}_t^{(k)} + \mathbf{M}_t\mathbf{H}_t^\top\mathbf{R}_t^{-1}(\mathbf{y}_t - \mathbf{H}_t\boldsymbol{\mu}_t^{(k)}), k = 1, \ldots, K-1. \tag{12}$$

Let $\mathbf{B}_t$ simultaneously diagonalize $(\boldsymbol{\Sigma}_t^-)^{-1}$ and $\mathbf{H}_t^\top\mathbf{R}_t^{-1}\mathbf{H}_t$ such that $\mathbf{B}_t^\top(\boldsymbol{\Sigma}_t^-)^{-1}\mathbf{B}_t = \mathbf{I}$ and $\mathbf{B}_t^\top\mathbf{H}_t^\top\mathbf{R}_t^{-1}\mathbf{H}_t\mathbf{B}_t = \text{diag}(r_1, \ldots, r_n)$. Then $\boldsymbol{\mu}_t^{(K)}$ defined in (12) is the minimizer of (11) when $\mathbf{M}_t = \mathbf{B}_t\text{diag}(\lambda_1, \ldots, \lambda_n)\mathbf{B}_t^\top$, where $\lambda_i = (1/r_i)(1 - (1 + r_i)^{-1/K})$ if $r_i \neq 0$ and $\lambda_i = 1$ otherwise. This suggests an optimization wherein $\boldsymbol{\mu}_t$ is computed via K steps of gradient descent on the likelihood, where $\mathbf{M}_t$ is the learning rate matrix implied by $\boldsymbol{\Sigma}_t^-$.

3.2 Truncated Gradient Descent as Prior Specification

Instead of computing the learning rate matrix $\mathbf{M}_t$ for a given $\boldsymbol{\Sigma}_t^-$, consider specifying $\mathbf{M}_t$ directly. For example, $\mathbf{M}_t = \rho\mathbf{I}$ recovers gradient descent on the likelihood with learning rate ρ. Let $\mathbf{C}_t$ be chosen such that $\mathbf{C}_t^\top\mathbf{M}_t^{-1}\mathbf{C}_t = \mathbf{I}$ and $\mathbf{C}_t^\top\mathbf{H}_t^\top\mathbf{R}_t^{-1}\mathbf{H}_t\mathbf{C}_t = \text{diag}(s_1, \ldots, s_n)$. Then, by Theorem 3.2 of Santos (1996), $\boldsymbol{\mu}_t^{(K)}$ as defined by (12) is the minimizer of (11) where $\boldsymbol{\Sigma}_t^- = \mathbf{C}_t\text{diag}(\sigma_1, \ldots, \sigma_n)\mathbf{C}_t^\top$, where $\sigma_i = (1/s_i)((1 - s_i)^{-K} - 1)$ if $s_i \neq 0$ and $\sigma_i = 1$ otherwise. Thus, in a linear-Gaussian system, specifying the learning rate matrix $\mathbf{M}_t$ implicitly defines an equivalent predictive covariance $\boldsymbol{\Sigma}_t^-$ and truncated gradient descent on the likelihood recovers the mode of the filtering distribution.

3.3 Interpretation as Variational Inference

The procedure shown above can be alternatively motivated by positing a variational distribution $q_t(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t \mid \mathbf{m}_t, \mathbf{M}_t)$ with fixed covariance $\mathbf{M}_t$. We show in Appendix A.2 that the truncated optimization of (12) is equivalent to natural gradient ascent on the ELBO. The choice of $\mathbf{M}_t$ then specifies the covariance of the variational approximate to the filtering distribution, the learning rate matrix of the optimization, and the effective prior covariance. More generally, (Khan & Rue, 2021) demonstrate that many popular adaptive optimization techniques such as RMSprop and Adam can be interpreted as performing variational inference with different choices of $q_t(\mathbf{x}_t)$. In this light, the choice of the optimization algorithm when combined with k steps of optimization implicitly defines a corresponding prior distribution. In the next section, we propose a simple yet effective filtering algorithm based on this interpretation of the update step.

4 Implicit Maximum a Posteriori Filtering

In this section, we state the Bayesian filtering steps of the Implicit MAP Filter, which (partially) optimizes the likelihood $p(\mathbf{y}_t|\mathbf{x}_t)$ at every time step. The filtering distribution of the Implicit MAP Filter at each time t is represented by its mean $\boldsymbol{\mu}_t$. In addition to the likelihood, we also assume knowledge of a sequence of transition functions $f_t(\mathbf{x}_{t-1}) \triangleq \mathbb{E}[p(\mathbf{x}_t \mid \mathbf{x}_{t-1})]$. This is a less restrictive assumption than fully specifying the transition distribution as in standard Bayesian filtering. The overall algorithm for the Implicit MAP Filter is summarized in Algorithm 1.

In order to approximate expectations with respect to the filtering distribution, we make use of the delta method (Dorfman, 1938; Ver Hoef, 2012). The first-order delta method approximates an expectation $\mathbb{E}_q[f(\mathbf{x})]$ by a first-order Taylor expansion about the mean $\boldsymbol{\mu}$ of $q(\mathbf{x})$ as follows:

$$\mathbb{E}_q[g(\mathbf{x})] \approx \mathbb{E}_q[g(\boldsymbol{\mu}) + (\mathbf{x} - \boldsymbol{\mu})^\top\nabla_{\mathbf{x}}g(\mathbf{x})|_{\mathbf{x}=\boldsymbol{\mu}}] \approx g(\boldsymbol{\mu}). \tag{13}$$

Algorithm 1 Implicit MAP Filter

Input: Number of timesteps T, initial state estimate $\hat{\boldsymbol{\mu}}_0$, sequence of loss functions $\ell_1, \ldots, \ell_T$ where $\exp(-\ell_t(\mathbf{x})) \propto p(\mathbf{y}_t \mid \mathbf{x}_t = \mathbf{x})$, sequence of transition functions $f_1, \ldots, f_T$ where $f_t(\mathbf{x}) = \mathbb{E}[p(\mathbf{x}_t|\mathbf{x}_{t-1} = \mathbf{x})]$, optimizer $\mathcal{M}$, number of optimization steps K.
Output: Filtering state estimates $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_T$.

for $t = 1$ **to** T **do**
  $\boldsymbol{\mu}_t^- \leftarrow f_t(\hat{\boldsymbol{\mu}}_{t-1})$ {Predict Step}
  $\boldsymbol{\mu}_t \leftarrow$ IMAP-UPDATE$(\boldsymbol{\mu}_t^-, \ell_t, \mathcal{M}, K)$ {Update Step}
end for

4.1 Predict Step

The predictive distribution for the Implicit MAP Filter is obtained by applying the first-order delta method to the Chapman-Kolmogorov equation (3).

$$p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) = \int p(\mathbf{x}_t \mid \mathbf{x}_{t-1}) p(\mathbf{x}_{t-1} \mid \mathbf{y}_{1:t-1}) d\mathbf{x}_{t-1} \approx p(\mathbf{x}_t \mid \mathbf{x}_{t-1} = \boldsymbol{\mu}_{t-1}). \tag{14}$$

The mean of the predictive distribution is thus obtained by applying the transition function to the previous state estimate: $\mathbb{E}[p(\mathbf{x}_t|\mathbf{y}_{1:t-1})] \approx f_t(\boldsymbol{\mu}_{t-1})$. This negates the need for matrix-matrix products as is commonly required for the predict step.

4.2 Update Step

The goal of the update step is to approximate the filtering distribution by using information from the likelihood at the current timestep. By only partially optimizing the likelihood, the Implicit MAP Filter update step produces a state estimate $\boldsymbol{\mu}_t$ that maximizes the posterior with respect to a prior implicitly defined by the choice of optimizer $\mathcal{M}$ and the number of optimizations steps K. Here, $\mathcal{M}$ takes as input the history of gradients and outputs a vector to be added to the current state. This formulation captures many popular optimizers, including SGD, RMSProp, and Adam. Our algorithm for the update step of the Implicit MAP Filter is provided in Algorithm 2.

Algorithm 2 Update step of Implicit MAP Filter: IMAP-UPDATE$(\boldsymbol{\mu}^-, \ell, \mathcal{M}, K)$

Input: Initialization $\boldsymbol{\mu}^-$, loss function ℓ, optimizer $\mathcal{M}$, number of optimization steps K.
Output: State estimate $\boldsymbol{\mu}$.

$\mathbf{m}^{(0)} \leftarrow \boldsymbol{\mu}^-$
for $k = 0$ **to** $K - 1$ **do**
  $\mathbf{g}^{(k)} \leftarrow \nabla_{\mathbf{x}} \ell(\mathbf{x})|_{\mathbf{x}=\mathbf{m}^{(k)}}$
  $\mathbf{m}^{(k+1)} \leftarrow \mathbf{m}^{(k)} + \mathcal{M}(\mathbf{g}^{(k)}; \mathbf{g}^{(0:k-1)})$
end for
$\boldsymbol{\mu} \leftarrow \mathbf{m}^{(K)}$

The choice of optimizer $\mathcal{M}$ will internalize all of the information typically specified by the filtering equations. For example, increasing the number of gradient steps and learning rate increases the Kalman gain (Freitas et al., 2000). The nature of each optimization step and the number of steps taken will define the shape and entropy of the filtering distribution assumed (Duvenaud et al., 2016).

5 Experimental Evaluation

The goal of our experiments is to assess whether the most popular adaptive optimizers can be used to design effective, robust, and scalable Bayesian filters. We begin with a simple low-dimensional system with nonlinear dynamics to establish that our approach is competitive with standard approaches in this simple setting. We then turn to a more challenging system that has been used as a baseline for filtering algorithms, which demonstrates some of the advantages of our approach. Finally, we show how adaptive optimization can allow Implicit MAP filtering to be used effectively in high-dimensional settings, such as adapting the weight space of a convolutional neural network.

5.1 TOY NONLINEAR SYSTEM

The equivalence between Kalman filtering and K steps of gradient descent elucidated in Section 3.2 suggests that the proposed framework should produce reliable estimates in *approximately* linear-Gaussian systems. To better understand filtering amidst nonlinearites, we first consider the one-dimensional example used in Doucet et al. (2001); Gordon et al. (1993); Kitagawa (1996) and originally proposed by Netto et al. (1978), which admits highly nonlinear periodic behavior:

$$\mathbf{x}_t = \frac{1}{2}\mathbf{x}_{t-1} + 25\frac{\mathbf{x}_{t-1}}{1+\mathbf{x}_{t-1}^2} + 8\cos\left(1.2t\Delta t\right) + \boldsymbol{q}_{t-1}, \qquad \mathbf{y}_t = \frac{\mathbf{x}_t^2}{20} + \boldsymbol{r}_t, \tag{15}$$

where $\mathbf{x}_0 \sim \mathcal{N}(0,1)$, $\boldsymbol{q}_{t-1} \sim \mathcal{N}(0,3)$, $\boldsymbol{r}_t \sim \mathcal{N}(0,2)$, and $\Delta t = 0.1$. We simulate ground truth trajectories and their measurements for 200 time steps and evaluate performance using the root mean square errors (RMSEs) with respect to the simulated true trajectories. The means and 95% confidence intervals of the RMSEs are computed from 100 independent Monte Carlo (MC) simulations. For each optimizer, we report the best hyperparameters found by a grid search using 5 separate MC simulations (see Appendix A.4).

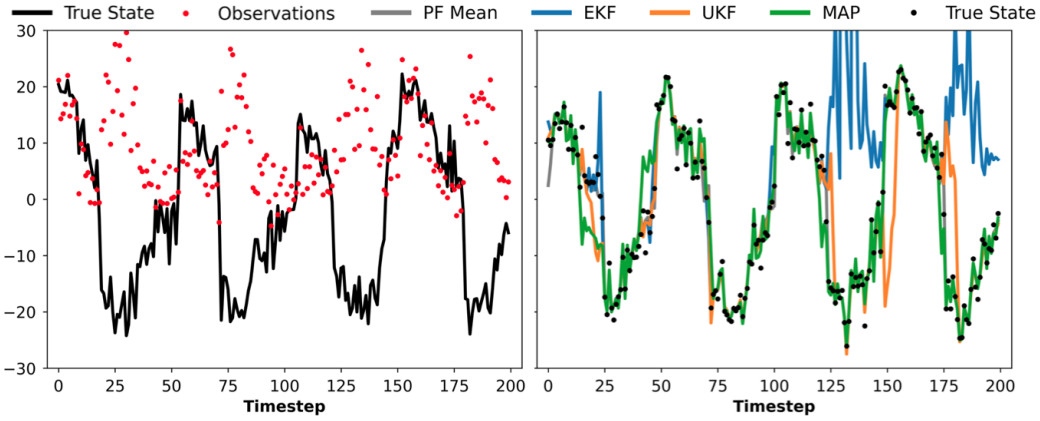

Figure 2: Toy nonlinear system for a single random trajectory **(left)** and the respective filter estimates **(right)**. Implicit MAP Filter shown uses the Adam optimizer with 50 steps at learning rate $\eta = 0.1$ and exponential decay rates $\beta_1, \beta_2 = 0.1$.

We evaluate three different experimental configurations in this system: ideal, misspecified $(+)$, and misspecified $(-)$. These configurations correspond to the process noise matrix, $\boldsymbol{Q}_{t-1}$, being known exactly ($\boldsymbol{Q}_{t-1} = 3$), overestimated ($\boldsymbol{Q}_{t-1} = 5$), and underestimated ($\boldsymbol{Q}_{t-1} = 1$), respectively. Since many filtering applications do not have access to true states, process noise misspecification is a realistic challenge that can significantly distort estimates (Huang et al., 2017; Mehra, 1972).

Table 1 shows the results of these three experimental configurations. Under ideal conditions, we see that there exist RMSprop and Adam hyperparameters that roughly correspond to the performance of an unscented Kalman filter (UKF). Both the extended Kalman filter (EKF) and iterated extended Kalman filter (iEKF) have the tendency to diverge in this system. The particle filter (PF) performs the best under all three configurations as expected, since 1000 particles can capture both uni-modal and bi-modal filtering distributions in one dimension quite well. Nonetheless, our Implicit MAP Filter tends to produce estimates that correspond with the modes of the particle filter, as shown in Appendix A.5.1, only struggling on some of the transition trials. Implementation details for these baseline methods can be found in Appendix A.1.

Since the Implicit MAP Filter does not explicitly use the process noise $\boldsymbol{Q}_{t-1}$, there is no sensitivity to misspecification, as indicated by the empty columns in Table 1. Moreover, our Implicit MAP Filter can always arbitrarily set the measurement noise to the identity matrix, thus reducing the objective to the mean squared error loss (see Appendix A.3 for justification). In our formulation, the choice of optimizer internalizes both the process noise and measurement noise, thus reducing the space and time complexity. Against methods that similarly use gradient information (EKF, iEKF), our Implicit MAP Filters with Adam and RMSprop see a significant performance boost despite the challenge of a non-convex likelihood. We can attribute this boost to the use of momentum

Table 1: RMSEs on the toy nonlinear system. Results show the average RMSE over 100 MC simulations with 95% confidence intervals. The dash $-$ indicates methods that are not depedent on the process noise covariance $\boldsymbol{Q}_{t-1}$.

Method	**RMSE** ($\boldsymbol{Q}_{t-1} = 3$)	**RMSE** ($\boldsymbol{Q}_{t-1} = 1$)	**RMSE** ($\boldsymbol{Q}_{t-1} = 5$)
EKF	34.909 ± 2.732	34.248 ± 3.552	30.861 ± 2.472
iEKF ($n = 5$)*	15.321 ± 0.493	13.640 ± 0.469	15.729 ± 0.565
MAP (Adadelta)*	23.152 ± 3.973	$-$	$-$
MAP (Gradient Descent)*	7.966 ± 0.180	$-$	$-$
MAP (Adagrad)*	6.549 ± 0.223	$-$	$-$
MAP (RMSprop)*	6.000 ± 0.227	$-$	$-$
MAP (Adam)*	5.842 ± 0.231	$-$	$-$
UKF	5.762 ± 0.271	6.767 ± 0.259	5.629 ± 0.311
PF ($n = 1000$)	$\mathbf{2.800 \pm 0.110}$	$\mathbf{4.003 \pm 0.153}$	$\mathbf{3.119 \pm 0.073}$

*Methods where the reported hyperparameters were found via grid search (see Appendix A.4).

(see Appendix A.5.1). Against methods that use sigma points (UKF) or particles (PF), we see competitive performance in a setting where these methods are known to perform exceedingly well, while avoiding any need to maintain noise covariances or many needless copies of the state.

5.2 Stochastic Lorenz Attractor

Consider a stochastic Lorenz '63 system (Lorenz, 1963; Li et al., 2020; Zhao & Särkkä, 2021):

$$\mathrm{d}\begin{pmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{pmatrix} = \begin{pmatrix} \sigma(x_{2,t} - x_{1,t}) \\ x_{1,t}(\rho - x_{3,t}) - x_{2,t} \\ x_{1,t} \cdot x_{2,t} - \beta x_{3,t} \end{pmatrix} \mathrm{d}t + \alpha \mathrm{d}W_t, \qquad \mathbf{y}_t = \begin{pmatrix} x_{1,t} \\ x_{2,t} \\ x_{3,t} \end{pmatrix} + \boldsymbol{r}_t \tag{16}$$

where $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{I}_3)$ for $\boldsymbol{\mu}_0 = (10, 10, 10)$, $\boldsymbol{r}_t \sim \mathcal{N}(\mathbf{0}, 2 \cdot \boldsymbol{I}_3)$, $\sigma = 10$, $\rho = 28$, $\beta = \frac{8}{3}$, $\alpha = 10$, and W_t is a three-dimensional Wiener process with unit spectral density. This system is chaotic, which can make the stochastic differential equation (SDE) unreliable for accurate prediction (Evensen et al., 2009). Similar to Zhao & Särkkä (2021), we simulate numerical ground truth trajectories and their measurements uniformly for 200 time steps with $\mathrm{d}t = 0.02$ using 10000 Euler-Maruyama integration steps between each measurement. As before, we compute RMSEs and 95% confidence intervals from 100 independent MC simulations and report the best hyperparameters for each optimizer found by a separate grid search (see Appendix A.4).

We report three experimental configurations where everything is held constant except for the method used for numerical integration. The first configuration uses the 4th order Runge-Kutta (RK4) method to integrate the Lorenz '63 system, the second configuration uses Euler's method (Euler), and the final configuration performs no integration: the dynamics are assumed to follow a Gaussian random walk (GRW). The purpose of these three experiments is to assess robustness to dynamics misspecification, which is an unavoidable reality wherever numerical integration is required.

Table 2 shows the results of these three experiments. All filters see performance degradation that corresponds to worsening estimates of the SDE. The Implicit MAP Filter with gradient descent shows the best performance in this system and significantly less sensitivity to dynamics misspecification than the UKF and PF. This reflects better adaptation to uncertainty introduced by the numerical approximation, which is a harder to appropriately accommodate in an explicitly defined model. In Table 2, our Implicit MAP Filter with gradient descent uses 3 gradient steps with learning rate $\eta = 0.05$ with RK4, 3 steps with $\eta = 0.1$ for the Euler case, and 10 steps with $\eta = 0.1$ for the GRW case. Increasing the number of steps or learning rate can accommodate worsening approximations, but the same optimizer settings generally shows robustness from RK4 to Euler (see Appendix A.5.2).

The likelihood in this system is *approximately* convex, which explains the EKF performance boost compared to the previous section and the absence of improvement caused by iteratively refining the linearization with the iEKF. Momentum, in this setting, worsens estimates. The performance boost given by Implicit MAP filtering over the EKF can be explained as a better accommodation of

Table 2: RMSEs on the stochastic Lorenz attractor. Results show the average RMSE over 100 MC simulations with 95% confidence intervals. RK4 indicates 4th order Runge-Kutta numerical integration. Euler indicates Euler's method of numerical integration. GRW indicates a Gaussian random walk, meaning no numerical integration.

Method	**RMSE (RK4)**	**RMSE (Euler)**	**RMSE (GRW)**
EKF	0.890 ± 0.094	2.308 ± 0.155	3.057 ± 0.037
iEKF ($n = 5$)	0.890 ± 0.094	2.308 ± 0.158	3.057 ± 0.038
MAP (Adadelta)	6.034 ± 0.634	7.649 ± 0.254	10.754 ± 0.279
MAP (Adam)	0.982 ± 0.135	1.153 ± 0.014	1.885 ± 0.010
MAP (Adagrad)	0.907 ± 0.144	1.096 ± 0.017	1.765 ± 0.013
MAP (RMSprop)	0.881 ± 0.114	1.081 ± 0.015	1.757 ± 0.015
MAP (Gradient Descent)	$\mathbf{0.701 \pm 0.018}$	$\mathbf{0.960 \pm 0.012}$	$\mathbf{1.561 \pm 0.010}$
UKF	2.742 ± 0.059	2.856 ± 0.066	5.628 ± 0.067
PF ($n = 1000$)	1.568 ± 0.027	1.725 ± 0.031	14.346 ± 0.365

*Methods where the reported hyperparameters were found via grid search (see Appendix A.4).

numerical integration uncertainty in an *implicitly* defined model, which is harder to incorporate into an explicitly defined model.

It is important to reiterate that our Implicit MAP Filter does not maintain $2n+1$ sigma points (UKF) or 1000 particles (PF), but only one point estimate. Additionally, it maintains no concept of uncertainty over state estimates for all 200 timesteps and does not use the process noise or measurement noise covariances made available to all the other methods. Yet, we see a significant performance boost whilst only maintaining a point estimate, indicating that our Implicit MAP Filter has the potential to make more with less in settings that are not ideal to filter in. When visualizing trajectories (see Appendix A.5.2), the Implicit MAP Filter does a remarkable job of fitting this system despite every initialization producing a vastly different filtering problem.

5.3 YEARBOOK

Suppose we have a pre-trained neural network that performs inference in a time-varying environment. This could be a recommender system that must evolve with changing seasons or a facial recognition system that must be robust to temporal shift. Let us assume that data curation is expensive, so we are only given a small number of examples at every time step for adaptation. We can formalize this as the following state space model (Haykin, 2004):

$$\mathbf{w}_t = f(\mathbf{x}_{t-1}) + \boldsymbol{q}_{t-1} \tag{17}$$

$$\mathbf{y}_t = h(\mathbf{w}_t, \boldsymbol{X}_t) + \boldsymbol{r}_t \tag{18}$$

where $\mathbf{w}_t$ is the state of the system given by the network's *ideal* weight parameterization, $\boldsymbol{q}_{t-1}$ is some noise vector that corrupts estimates from the transition function f, $\mathbf{y}_t$ is the network's desired response vector after applying the transformation h parameterized by $\mathbf{w}_t$ with inputs $\boldsymbol{X}_t$, and $\boldsymbol{r}_t$ is the measurement noise.

In this experiment, we design a time-varying prediction task as described above using the Yearbook dataset of 37,921 frontal-facing American high school yearbook photos that capture changes in fashion and population demographics from 1905 - 2013 (Ginosar et al., 2015). We use the data from 1905-1930 to pre-train a 4 layer convolutional neural network (CNN) to perform a binary gender classification task. From 1931-2010, we adapt this weight initialization at each year sequentially using five approaches: static weights, direct fit, variational Kalman filtering (VKF), particle filtering (PF), and Implicit MAP filtering. 32 randomly chosen training examples are available at every time step and a held-out test set of roughly 100 images per time step is used validate the approaches (see Appendix A.5.3 for complete details).

Three things complicate Bayesian filtering in this system: (1) a state space of 28,193 parameters, (2) lack of access to weight-space dynamics equations, and (3) lack of process noise and measurement noise specification. To understand our capacity for modeling such a system given a VKF, PF, and Implicit MAP Filter, we test 5 configurations of each method (see Appendix A.5.3).

Table 3: Classification accuracy with 95% confidence intervals over 40 year blocks in the yearbook dataset.

Method	% Correct (Validation Years)	% Correct (Test Years)
Static Weights	82.416 ± 2.270	60.897 ± 1.615
PF	86.430 ± 1.790	66.715 ± 2.390
VKF	93.087 ± 1.038	79.967 ± 2.204
Direct Fit	94.416 ± 0.924	80.449 ± 1.845
MAP	$\mathbf{94.973 \pm 0.837}$	$\mathbf{84.747 \pm 2.030}$

In Table 3, we report classification accuracies for all five methods over 80 years, grouped into two 40 year blocks. Assuming static weights over time yields the worst performance of all five methods due to the problem being fundamentally non-stationary. On the other extreme, attempting to exactly fit the data at each time step underestimates the measurement noise, which proves to be better but suboptimal. The VKF, PF, and Implicit MAP Filter attempt to internalize the system's true state space equations via a small grid search, but this is clearly difficult to do exactly in such a state space. As shown in the previous two experiments, our implicit approach is less sensitive to this inevitable misspecification. The Implicit MAP Filter with 50 steps of Adam optimization not only showed the best performance, but extremely desirable space and time complexity.

6 Related Work

Bayesian filtering, optimization, and neural networks share a long history that dates back to early attempts of using the extended Kalman filter (EKF) algorithm to train multilayer perceptrons (Singhal & Wu, 1988). Since then, Kalman filter theory for the training and use of neural networks saw rich development (Haykin, 2004), but fell out of fashion as gradient descent proved to be the more efficient and scalable option for the training deep neural networks (Bengio et al., 2017). Ruck et al. (1992) realized a connection between optimization and Bayesian filtering, showing that the standard backpropagation algorithm can be seen as a degenerate form of the EKF. Several works have since connected the EKF and the Gauss-Newton method (Bertsekas, 1996; Bell & Cathey, 1993), showing that the EKF can be seen as a single iteration of Newton's method for a specific quadratic form (Humpherys et al., 2012). Ollivier (2018) and Ollivier (2019) proved equivalence between online natural gradient descent and the EKF when the dynamics are stationary or the process noise is proportional to the posterior covariance over states. Aitchison (2020) used the Bayesian filtering formalism to derive adaptive optimizers like Adam and RMSprop. Freitas et al. (2000) showed that the process noise and measurement noise can be viewed as adaptive per-parameter learning rates if treated as free parameters. In the present work, we establish similar connections between optimization and Bayesian filtering but focus on using optimization as a tool for Bayesian filtering rather than using Bayesian filtering as a tool for optimization.

Several works have similarly used popular optimization methods to address challenges with classical Bayesian filters. Auvinen et al. (2010) and Bardsley et al. (2013) used LBFGS and conjugate gradients to derive variational Kalman filters with low storage covariance, and inverse covariance, approximations. They addressed scalability concerns with classical filters using an *explicit* approch, rather than the implicit approach presented here. Chen et al. (2003) describes particle filters that move every particle down their respective gradient prior to sampling from the proposal distribution. This is reminiscent of our approach in that it uses gradients, but differs in that it maintains particles, performs sampling, and again falls within the class of explicit filters.

7 Conclusion

We have shown that Bayesian filtering can be considered as optimization over a time-varying objective and that such a perspective opens the door to effective, robust, and scalable filters built from adaptive optimizers. This framework, however, comes at the cost of interoperability and uncertainty estimates, which limits use in risk-sensitive environments or situations where the filtering equations themselves are used for analysis. Nonetheless, our proposed Implicit MAP Filtering approach is an attractive option for the performance-driven practitioner.

REFERENCES

Laurence Aitchison. Bayesian filtering unifies adaptive and non-adaptive neural network optimization methods. *Advances in Neural Information Processing Systems*, 33:18173–18182, 2020.

Harri Auvinen, Johnathan M Bardsley, Heikki Haario, and T Kauranne. The variational kalman filter and an efficient implementation using limited memory bfgs. *International Journal for Numerical Methods in Fluids*, 64(3):314–335, 2010.

Johnathan M Bardsley, Albert Parker, Antti Solonen, and Marylesa Howard. Krylov space approximate kalman filtering. *Numerical Linear Algebra with Applications*, 20(2):171–184, 2013.

Bradley M Bell and Frederick W Cathey. The iterated kalman filter update as a gauss-newton method. *IEEE Transactions on Automatic Control*, 38(2):294–297, 1993.

Yoshua Bengio, Ian Goodfellow, and Aaron Courville. *Deep learning*, volume 1. MIT press Cambridge, MA, USA, 2017.

Dimitri P Bertsekas. Incremental least squares methods and the extended kalman filter. *SIAM Journal on Optimization*, 6(3):807–822, 1996.

Zhe Chen et al. Bayesian filtering: From kalman filters to particle filters, and beyond. *Statistics*, 182 (1):1–69, 2003.

RA Dorfman. A note on the delta-method for finding variance formulae. *Biometric Bulletin*, 1938.

Arnaud Doucet, Nando De Freitas, Neil James Gordon, et al. *Sequential Monte Carlo methods in practice*, volume 1. Springer, 2001.

John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

David Duvenaud, Dougal Maclaurin, and Ryan Adams. Early stopping as nonparametric variational inference. In *Artificial intelligence and statistics*, pp. 1070–1077. PMLR, 2016.

Geir Evensen et al. *Data assimilation: the ensemble Kalman filter*, volume 2. Springer, 2009.

JFG de Freitas, Mahesan Niranjan, and Andrew H. Gee. Hierarchical bayesian models for regularization in sequential learning. *Neural computation*, 12(4):933–953, 2000.

Arthur Gelb et al. *Applied optimal estimation*. MIT press, 1974.

Shiry Ginosar, Kate Rakelly, Sarah Sachs, Brian Yin, and Alexei A Efros. A century of portraits: A visual historical record of american high school yearbooks. In *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 1–7, 2015.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pp. 249–256. JMLR Workshop and Conference Proceedings, 2010.

Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE proceedings F (radar and signal processing)*, volume 140, pp. 107–113. IET, 1993.

Simon Haykin. *Kalman filtering and neural networks*. John Wiley & Sons, 2004.

Yulong Huang, Yonggang Zhang, Zhemin Wu, Ning Li, and Jonathon Chambers. A novel adaptive kalman filter with inaccurate process and measurement noise covariance matrices. *IEEE transactions on Automatic Control*, 63(2):594–601, 2017.

Jeffrey Humpherys, Preston Redd, and Jeremy West. A fresh look at the kalman filter. *SIAM review*, 54(4):801–823, 2012.

Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004.

Simon J Julier, Jeffrey K Uhlmann, and Hugh F Durrant-Whyte. A new approach for filtering nonlinear systems. In *Proceedings of 1995 American Control Conference-ACC'95*, volume 3, pp. 1628–1632. IEEE, 1995.

Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. 1960.

Mohammad Emtiyaz Khan and Håvard Rue. The bayesian learning rule. *arXiv preprint arXiv:2107.04562*, 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Genshiro Kitagawa. Monte carlo filter and smoother for non-gaussian nonlinear state space models. *Journal of computational and graphical statistics*, 5(1):1–25, 1996.

Xuechen Li, Ting-Kam Leonard Wong, Ricky TQ Chen, and David Duvenaud. Scalable gradients for stochastic differential equations. In *International Conference on Artificial Intelligence and Statistics*, pp. 3870–3882. PMLR, 2020.

Edward N Lorenz. Deterministic nonperiodic flow. *Journal of atmospheric sciences*, 20(2):130–141, 1963.

James Martens. New insights and perspectives on the natural gradient method. *The Journal of Machine Learning Research*, 21(1):5776–5851, 2020.

Raman Mehra. Approaches to adaptive filtering. *IEEE Transactions on automatic control*, 17(5): 693–698, 1972.

ML Andrade Netto, L Gimeno, and MJ Mendes. A new spline algorithm for non-linear filtering of discrete time systems. *IFAC Proceedings Volumes*, 11(1):2123–2130, 1978.

Yann Ollivier. Online natural gradient as a kalman filter. *Electronic Journal of Statistics*, 12:2930–2961, 2018.

Yann Ollivier. The extended kalman filter is a natural gradient descent in trajectory space. *arXiv e-prints*, 2019.

Matti Raitoharju and Robert Piché. On computational complexity reduction methods for kalman filter extensions. *IEEE Aerospace and Electronic Systems Magazine*, 34(10):2–19, 2019.

Dennis W. Ruck, Steven K. Rogers, Matthew Kabrisky, Peter S. Maybeck, and Mark E. Oxley. Comparative analysis of backpropagation and the extended kalman filter for training multilayer perceptrons. *IEEE transactions on pattern analysis & machine intelligence*, 14(06):686–691, 1992.

Reginaldo J Santos. Equivalence of regularization and truncated iteration for general ill-posed problems. *Linear algebra and its applications*, 236:25–33, 1996.

Simo Särkkä and Lennart Svensson. *Bayesian filtering and smoothing*, volume 17. Cambridge university press, 2023.

Dan Simon. *Optimal state estimation: Kalman, H infinity, and nonlinear approaches*. John Wiley & Sons, 2006.

Sharad Singhal and Lance Wu. Training multilayer perceptrons with the extended kalman algorithm. *Advances in neural information processing systems*, 1, 1988.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pp. 1139–1147. PMLR, 2013.

Tijmen Tieleman, Geoffrey Hinton, et al. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31, 2012.

Jay M. Ver Hoef. Who Invented the Delta Method? *The American Statistician*, 66(2):124–127, May 2012. ISSN 0003-1305, 1537-2731. doi: 10.1080/00031305.2012.687494.

Matthew D Zeiler. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701*, 2012.

Zheng Zhao and Simo Särkkä. Non-linear gaussian smoothing with taylor moment expansion. *IEEE Signal Processing Letters*, 29:80–84, 2021.

A Appendix

A.1 Baseline Bayesian Filter Methods and Implementation Details

In this section we define the Kalman filter (KF), Extended Kalman filter (EKF), iterated extended Kalman filter (iEKF), unscented Kalman filter (UKF), particle filter (PF), and variational Kalman filter (VKF) as implemented in the experimental section of this paper. We closely follow the definitions and notation of Särkkä & Svensson (2023) and refer the reader to that text for proofs and a more extensive treatment of these topics.

We use this section to state design choices made for each baseline filter when it comes to the experiments reported in the main section.

A.1.1 Kalman Filter

Assume the following state space model:

$$\begin{aligned}\mathbf{x}_t &= \boldsymbol{F}_{t-1}\mathbf{x}_{t-1} + \boldsymbol{q}_{t-1}\\ \mathbf{y}_t &= \boldsymbol{H}_t\mathbf{x}_t + \boldsymbol{r}_t\end{aligned}$$

where $\mathbf{x}_t \in \mathbb{R}^n$ is the state, $\mathbf{y}_t \in \mathbb{R}^m$ is the measurement, $\boldsymbol{F}_{t-1} \in \mathbb{R}^{n\times n}$ is the transition matrix, $\boldsymbol{H}_t \in \mathbb{R}^{m\times n}$ is the measurement model matrix, $\boldsymbol{q}_{t-1} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{Q}_{t-1})$ is the process noise and $\boldsymbol{r}_t \sim \mathcal{N}(\mathbf{0}, \boldsymbol{R}_t)$ is the measurement noise.

The Kalman filter (Kalman, 1960) defines the predictive distribution, filtering distribution, and marginal likelihood at time step t as

$$\begin{aligned}p(\mathbf{x}_t|\mathbf{y}_{1:t-1}) &= \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t^-, \boldsymbol{\Sigma}_t^-),\\ p(\mathbf{x}_t|\mathbf{y}_{1:t}) &= \mathcal{N}(\mathbf{x}_t|\boldsymbol{\mu}_t, \boldsymbol{\Sigma}_t),\\ p(\mathbf{y}_t|\mathbf{y}_{1:t-1}) &= \mathcal{N}(\mathbf{x}_t|\boldsymbol{H}_t\boldsymbol{\mu}_t^-, \boldsymbol{S}_t),\end{aligned}$$

with predict step:

$$\begin{aligned}\boldsymbol{\mu}_t^- &= \boldsymbol{F}_{t-1}\boldsymbol{\mu}_{t-1},\\ \boldsymbol{\Sigma}_t^- &= \boldsymbol{F}_{t-1}\boldsymbol{\Sigma}_{t-1}\boldsymbol{F}_{t-1}^\top + \boldsymbol{Q}_{t-1},\end{aligned}$$

and update step:

$$\begin{aligned}\boldsymbol{v}_t &= \mathbf{y}_t - \boldsymbol{H}_t\boldsymbol{\mu}_t^-,\\ \boldsymbol{S}_t, &= \boldsymbol{H}_t\boldsymbol{\Sigma}_t^-\boldsymbol{H}_t^\top + \boldsymbol{R}_t,\\ \boldsymbol{K}_t &= \boldsymbol{\Sigma}_t^-\boldsymbol{H}_t^\top\boldsymbol{S}_t^{-1},\\ \boldsymbol{\mu}_t &= \boldsymbol{\mu}_t^- + \boldsymbol{K}_t\boldsymbol{v}_t,\\ \boldsymbol{\Sigma}_t &= \boldsymbol{\Sigma}_t^- - \boldsymbol{K}_t\boldsymbol{S}_t\boldsymbol{K}_t^\top.\end{aligned}$$

Recursion begins from some $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$. In the stochastic Lorenz Attractor, the Gaussian random walk (GRW) case is equivalent to a Kalman filter where $\boldsymbol{F}_{t-1} = \boldsymbol{I}_3$ and $\boldsymbol{H}_t = \boldsymbol{I}_3$ at every time step t. We do not see the linear-Gaussian Kalman filter in any other experiments.

A.1.2 Extended Kalman Filter

In the extended Kalman filter (EKF), we use first-order Taylor approximations of the nonlinear transition function f and measurement model h where appropriate. We denote the Jacobians of these functions as $\boldsymbol{F}_{\mathbf{x}}(\cdot)$ and $\boldsymbol{H}_{\mathbf{x}}(\cdot)$. The predict step is now

$$\begin{aligned}\boldsymbol{\mu}_t^- &= f(\boldsymbol{\mu}_{t-1}),\\ \boldsymbol{\Sigma}_t^- &= \boldsymbol{F}_{\mathbf{x}}(\boldsymbol{\mu}_{t-1})\boldsymbol{\Sigma}_{t-1}\boldsymbol{F}_{\mathbf{x}}(\boldsymbol{\mu}_{t-1})^\top + \boldsymbol{Q}_{t-1},\end{aligned}$$

and the update step is

$$\begin{aligned}
\boldsymbol{v}_t &= \mathbf{y}_t - h(\boldsymbol{\mu}_t^-), \\
\boldsymbol{S}_t, &= \boldsymbol{H}_{\mathbf{x}}(\boldsymbol{\mu}_t^-)\boldsymbol{\Sigma}_t^- \boldsymbol{H}_{\mathbf{x}}(\boldsymbol{\mu}_t^-)^\top + \boldsymbol{R}_t, \\
\boldsymbol{K}_t &= \boldsymbol{\Sigma}_t^- \boldsymbol{H}_{\mathbf{x}}(\boldsymbol{\mu}_t^-)^\top \boldsymbol{S}_t^{-1}, \\
\boldsymbol{\mu}_t &= \boldsymbol{\mu}_t^- + \boldsymbol{K}_t \boldsymbol{v}_t, \\
\boldsymbol{\Sigma}_t &= \boldsymbol{\Sigma}_t^- - \boldsymbol{K}_t \boldsymbol{S}_t \boldsymbol{K}_t^\top,
\end{aligned}$$

with recursion starting from some $\mathbf{x}_0 \sim \mathcal{N}(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ as before.

A.1.3 Iterated Extended Kalman Filter

The iterated extended Kalman filter (iEKF) (Gelb et al., 1974) shares an identical predict step with the EKF. The update step is redefined as the following iterative procedure, starting from $\mathbf{x}_t^{(0)} = \boldsymbol{\mu}_t^-$ from the predict step:

- For $i = 1, 2, \dots K$:

$$\begin{aligned}
\boldsymbol{v}_t^{(i)} &= \mathbf{y}_t - h(\mathbf{x}_t^{(i-1)}) - \boldsymbol{H}_{\mathbf{x}}(\mathbf{x}_t^{(i-1)})(\boldsymbol{\mu}^- - \mathbf{x}_t^{(i-1)}), \\
\boldsymbol{S}_t^{(i)}, &= \boldsymbol{H}_{\mathbf{x}}(\mathbf{x}_t^{(i-1)})\boldsymbol{\Sigma}_t^- \boldsymbol{H}_{\mathbf{x}}(\mathbf{x}_t^{(i-1)})^\top + \boldsymbol{R}_t, \\
\boldsymbol{K}_t^{(i)} &= \boldsymbol{\Sigma}_t^- \boldsymbol{H}_{\mathbf{x}}(\mathbf{x}_t^{(i-1)})^\top \left[\boldsymbol{S}_t^{(i)}\right]^{-1}, \\
\mathbf{x}_t^{(i)} &= \boldsymbol{\mu}_t^- + \boldsymbol{K}_t^{(i)} \boldsymbol{v}_t^{(i)}.
\end{aligned}$$

- Set $\boldsymbol{\mu}_t = \mathbf{x}_t^{(K)}$.
- Set $\boldsymbol{\Sigma}_t = \boldsymbol{\Sigma}_t^- - \boldsymbol{K}_t^{(K)} \boldsymbol{S}_t^{(K)} \left[\boldsymbol{K}_t^{(K)}\right]^\top$.

For both the stochastic Lorenz Attractor and the toy nonlinear system, we ran iEKFs for iterations $K \in \{1, 3, 5, 10, 25, 50, 100\}$. We consistently report the iEKF that showed the highest average RMSE over 100 Monte Carlo experiments.

A.1.4 Unscented Kalman Filter

The predict step for the Unscented Kalman filter (UKF) is as follows:

- Form $2n + 1$ sigma points where

$$\begin{aligned}
\chi_{t-1}^{(0)} &= \boldsymbol{\mu}_{t-1}, \\
\chi_{t-1}^{(i)} &= \boldsymbol{\mu}_{t-1} + \sqrt{n + \lambda} \left[\sqrt{\boldsymbol{\Sigma}_{t-1}}\right]_i, \\
\chi_{t-1}^{(i+n)} &= \boldsymbol{\mu}_{t-1} - \sqrt{n + \lambda} \left[\sqrt{\boldsymbol{\Sigma}_{t-1}}\right]_i,
\end{aligned}$$

  for $i = 1, \dots, n$ where n is the dimension of the state space and λ is a tuneable parameter defined below.
- Apply the dynamics model to the sigma points:

$$\hat{\chi}_t^{(i)} = f(\chi_{t-1}^{(i)})$$

  for $i = 0, \dots, 2n$.
- Compute the predict step mean and covariance:

$$\begin{aligned}
\boldsymbol{\mu}_t^- &= \sum_{i=0}^{2n} w_i^{(m)} \hat{\chi}_t^{(i)}, \\
\boldsymbol{\Sigma}_t^- &= \sum_{i=0}^{2n} w_i^{(c)} \left(\hat{\chi}_t^{(i)} - \boldsymbol{\mu}_t^-\right)\left(\hat{\chi}_t^{(i)} - \boldsymbol{\mu}_t^-\right)^\top + \boldsymbol{Q}_{t-1},
\end{aligned}$$

where

$$
\begin{aligned}
w_0^{(m)} &= \frac{\lambda}{n+\lambda}, \\
w_0^{(c)} &= \frac{\lambda}{n+\lambda} + (1-\alpha^2+\beta), \\
w_i^{(m)} &= \frac{\lambda}{2(n+\lambda)}, \\
w_i^{(c)} &= \frac{\lambda}{2(n+\lambda)},
\end{aligned}
$$

where α, β are tuneable parameters.

The update step:

- Form $2n+1$ sigma points as before except replace $\boldsymbol{\mu}_{t-1}$ and $\boldsymbol{\Sigma}_{t-1}$ with $\boldsymbol{\mu}_t^-$ and $\boldsymbol{\Sigma}_t^-$ from the predict step. Denote these sigma points as $\hat{\chi}_t^{-(i)}$ for $i = 0, \ldots, 2n$.
- Apply the measurment model to the sigma points:

  $$\hat{\mathcal{Y}}_t^{(i)} = h(\chi_{t-1}^{(i)}),$$

  for $i = 0, \ldots, 2n$.
- Compute:

$$
\begin{aligned}
\boldsymbol{m}_t &= \sum_{i=0}^{2n} w_i^{(m)} \hat{\mathcal{Y}}_t^{(i)}, \\
\boldsymbol{S}_t &= \sum_{i=0}^{2n} w_i^{(c)} \left(\hat{\mathcal{Y}}_t^{(i)} - \boldsymbol{m}_t\right) \left(\hat{\mathcal{Y}}_t^{(i)} - \boldsymbol{m}_t\right)^\top + \boldsymbol{R}_t, \\
\boldsymbol{C}_t &= \sum_{i=0}^{2n} w_i^{(c)} \left(\hat{\chi}_t^{-(i)} - \boldsymbol{\mu}_t^-\right) \left(\hat{\mathcal{Y}}_t^{(i)} - \boldsymbol{m}_t\right)^\top.
\end{aligned}
$$

- Compute the Kalman gain and perform the update:

$$
\begin{aligned}
\boldsymbol{K}_t &= \boldsymbol{C}_t \boldsymbol{S}_t^{-1}, \\
\boldsymbol{\mu}_t &= \boldsymbol{\mu}_t^- + \boldsymbol{K}_t \left(\mathbf{y}_t - \boldsymbol{m}_t\right), \\
\boldsymbol{\Sigma}_t &= \boldsymbol{\Sigma}_t^- - \boldsymbol{K}_t \boldsymbol{S}_t \boldsymbol{K}_t^\top.
\end{aligned}
$$

For both the stochastic Lorenz Attractor and the toy nonlinear system, we set $\alpha = 1$, $\beta = 3 - n$, $\lambda = \alpha^2 \cdot (n + \beta) - n$.

A.1.5 PARTICLE FILTER

We ran a Bootstrap filter (BF) with 1000 particles for every experiment reported in the main section. The BF algorithm at every time step is as follows:

- Sample

  $$\mathbf{x}_t^{(i)} \sim p(\mathbf{x}_t | \mathbf{x}_{t-1}^{(i)})$$

  for every ith particle.
- Compute weights

  $$w_t^{(i)} \propto p(\mathbf{y}_t | \mathbf{x}_t^{(i)})$$

  for every ith particle and normalize.

- Perform resampling.

We did multinomial resampling at every time step t. For the yearbook dataset, since the transition distribution was unknown, we used the following transition distribution:

$$p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}) = \mathcal{N}(p(\mathbf{x}_t|\mathbf{x}_{t-1}^{(i)}, \sigma^2\boldsymbol{I}) \tag{19}$$

where $\sigma^2 \in \{0.1, 0.05, 0.01, 0.005, 0.001\}$. Selecting σ^2 requires a grid search, which is described in A.4.

A.1.6 VARIATIONAL KALMAN FILTER

A variational Kalman filter (VKF) was used in the Yearbook experiments due to the storage complexity of the EKF and UKF. Similar to the Implicit MAP Filter, the VKF we implemented propagates a point-mass forward in time but differs in that it models the prior uncertainty *explicitly* rather than *implicitly*.

Assume that the initial distribution is $p(\mathbf{w}_0) = \mathcal{N}(\mathbf{w}_0|\mathbf{0}, \sigma_0^2\boldsymbol{I})$, the transition distribution is $p(\mathbf{w}_t|\mathbf{w}_{t-1}) = \mathcal{N}(\mathbf{w}_t|\mathbf{w}_{t-1}, \sigma^2\boldsymbol{I})$, and the likelihood is $p(\mathbf{y}_t|\boldsymbol{X}_t) = \prod_{i=1}^{n_t} p_{\mathbf{w}_t}(y_{ti}|\boldsymbol{X}_{ti})$, where $p_{\mathbf{w}}(y|\boldsymbol{X})$ is the output of a neural network with parameters $\mathbf{w}$ that produces a distribution over target y given input $\boldsymbol{X}$. We assume that $p(\mathbf{w}_t|\boldsymbol{X}_{1:t}, \mathbf{y}_{1:t}) \approx \delta(\hat{\mathbf{w}}_t - \mathbf{w}_t)$ where $\delta(\cdot)$ denotes a Dirac-delta function centered at $\mathbf{w}_t$.

The VKF algorithm at every time step is as follows:

- Predict step:

$$\begin{aligned} p(\mathbf{w}_t|\boldsymbol{X}_{1:t-1}, \mathbf{y}_{1:t-1}) &= \int \delta(\hat{\mathbf{w}}_{t-1} - \mathbf{w}_{t-1})\mathcal{N}(\mathbf{w}_t|\mathbf{w}_{t-1}, \sigma^2\boldsymbol{I})\mathrm{d}\mathbf{w}_{t-1} \\ &= \mathcal{N}(\mathbf{w}_t|\hat{\mathbf{w}}_{t-1}, \sigma^2\boldsymbol{I}) \end{aligned}$$

- Update step:

$$p(\mathbf{w}_t|\boldsymbol{X}_{1:t}, \mathbf{y}_{1:t}) \propto \mathcal{N}(\mathbf{w}_t|\hat{\mathbf{w}}_{t-1}, \sigma^2\boldsymbol{I}) \prod_{i=1}^{n_t} p_{\mathbf{w}_t}(y_{ti}|\boldsymbol{X}_{ti})$$

Now, we approximate $p(\mathbf{w}_t|\boldsymbol{X}_{1:t}, \mathbf{y}_{1:t})$ with a point mass $\hat{\mathbf{w}}_t$ such that

$$\begin{aligned} \hat{\mathbf{w}}_t &\triangleq \arg\max_{\mathbf{w}_t} \left[\log \mathcal{N}(\mathbf{w}_t|\hat{\mathbf{w}}_{t-1}, \sigma^2\boldsymbol{I}) + \sum_{i=1}^{n_t} \log p_{\mathbf{w}_t}(y_{ti}|\boldsymbol{X}_{ti}) \right] \\ &= \arg\min_{\mathbf{w}_t} \left[\underbrace{-\frac{1}{n_t} \sum_{i=1}^{n_t} \log p_{\mathbf{w}_t}(y_{ti}|\boldsymbol{X}_{ti})}_{\text{mean binary cross entropy}} + \underbrace{\frac{1}{2n_t\sigma^2} \sum_{d=1}^{D} (w_{td} - \hat{w}_{t-1,d})^2}_{\text{weight decay centered at } \hat{\mathbf{w}}_{t-1}} \right] \end{aligned}$$

In the yearbook experiment, we run 5 configurations with $\sigma^2 \in \{0.1, 0.05, 0.01, 0.005, 0.001\}$. We optimize using the same procedure as the direct fit case, with the exception that the objective has explicit regularization.

A.2 VARIATIONAL INFERENCE INTERPRETATION OF THE UPDATE STEP

In this section, we show how truncated gradient descent on the likelihood (12) can be interpreted as variational inference with the variational distribution $q_t(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t \mid \mathbf{m}_t, \mathbf{M}_t)$ where $\mathbf{M}_t^{-1} \succ \mathbf{0}$ is fixed. The derivation in this section is based on Section 4.1 of (Khan & Rue, 2021), but adapts it to the setting of the update step of a Bayesian filter and further applies the equivalence due to (Santos, 1996) in order to deduce the implied filtering covariance $\boldsymbol{\Sigma}_t^-$. The variational lower bound on the log marginal likelihood for time step t is:

$$\log p(\mathbf{y}_t \mid \mathbf{y}_{1:t-1}) \geq \mathbb{E}_{q_t}\left[\log \frac{p(\mathbf{x}_t, \mathbf{y}_t \mid \mathbf{y}_{1:t-1})}{q_t(\mathbf{x}_t)}\right] \tag{20}$$

$$= \mathbb{E}_{q_t}[\log p(\mathbf{y}_t \mid \mathbf{x}_t) + \log p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1})] + \mathcal{H}(q_t) \tag{21}$$

Then the optimal variational distribution is:

$$q_t^*(\mathbf{x}_t) = \arg\min_{q_t} \mathbb{E}_{q_t}[\bar{\ell}_t(\mathbf{x}_t)] - \mathcal{H}(q_t), \text{ where} \tag{22}$$

$$\bar{\ell}_t(\mathbf{x}_t) \triangleq -\log p(\mathbf{y}_t \mid \mathbf{x}_t) - \log p(\mathbf{x}_t \mid \mathbf{y}_{1:t-1}) \tag{23}$$

$$= -\log \mathcal{N}(\mathbf{y}_t \mid \mathbf{H}_t\mathbf{x}_t, \mathbf{R}_t) - \log \mathcal{N}(\mathbf{x}_t \mid \boldsymbol{\mu}_t^-, \boldsymbol{\Sigma}_t^-) \tag{24}$$

We take the approach of (Khan & Rue, 2021) where the variational optimization is performed using natural gradient descent. In particular, assume the variational distribution to be of the form

$$q_{\boldsymbol{\lambda}_t}(\mathbf{x}_t) = h(\mathbf{x}_t)\exp[\langle \boldsymbol{\lambda}_t, \mathbf{T}(\mathbf{x}_t)\rangle - A(\boldsymbol{\lambda}_t)] \tag{25}$$

Natural gradient descent on the variational objective can then be written as:

$$\boldsymbol{\lambda}_t^{(k+1)} \leftarrow (1-\rho_t^{(k)})\boldsymbol{\lambda}_t^{(k)} - \rho_t^{(k)}\nabla_{\boldsymbol{\mu}}\mathbb{E}_{q_t^{(k)}}[\bar{\ell}_t(\mathbf{x}_t) + \log h(\mathbf{x}_t)] \tag{26}$$

With the choice $q_t^{(k)}(\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_t \mid \mathbf{m}_t^{(k)}, \mathbf{M}_t)$, we have $\boldsymbol{\lambda}_t^{(k)} = \mathbf{M}_t^{-1}\mathbf{m}_t^{(k)}$, $\boldsymbol{\mu}_t^{(k)} = \mathbf{m}_t^{(k)}$ and $2\log h(\mathbf{x}_t) = P\log|2\pi\mathbf{M}_t^{-1}| - \mathbf{x}_t^\top\mathbf{M}_t^{-1}\mathbf{x}_t$. After substituing, we find that the updates can be written as follows:

$$\mathbf{x}_t^{(k+1)} \leftarrow \mathbf{x}_t^{(k)} - \rho_t^{(k)}\mathbf{S}_t^{-1}\nabla_{\mathbf{x}_t}\bar{\ell}_t(\mathbf{x}_t)|_{\mathbf{x}_t=\mathbf{m}_t^{(k)}}. \tag{27}$$

The gradient $\nabla_{\mathbf{x}_t}\bar{\ell}(\mathbf{x}_t)$ takes the following form:

$$\nabla_{\mathbf{x}_t}\bar{\ell}(\mathbf{x}_t) = \nabla_{\mathbf{x}_t}\left[\frac{1}{2}\|\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t\|^2_{\mathbf{R}_t} + \frac{1}{2}\|\mathbf{x}_t - \boldsymbol{\mu}_t^-\|^2_{\boldsymbol{\Sigma}_t^-}\right] \tag{28}$$

$$= -\mathbf{H}_t^\top\mathbf{R}_t^{-1}(\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t) + (\boldsymbol{\Sigma}_t^-)^{-1}(\mathbf{x}_t - \boldsymbol{\mu}_t^-) \tag{29}$$

Now if we optimize this to convergence, we would recover the setting $\mathbf{m}_t^*$ such that $\mathcal{N}(\mathbf{x}_t \mid \mathbf{m}_t^*, \mathbf{M}_t) \approx p(\mathbf{x}_t \mid \mathbf{y}_{1:t})$. This requires knowledge of both $\boldsymbol{\mu}_t^-$ and $\boldsymbol{\Sigma}_t^-$.

On the other hand, consider truncated gradient descent on the negative log-likelihood $\ell_t(\mathbf{x}_t)$:

$$\mathbf{x}_t^{(k+1)} \leftarrow \mathbf{x}_t^{(k)} - \rho_t^{(k)}\mathbf{M}_t\nabla_{\mathbf{x}_t}\ell_t(\mathbf{x}_t)|_{\mathbf{x}_t=\mathbf{m}_t^{(k)}}, \text{ where} \tag{30}$$

$$\ell(\mathbf{x}_t) \triangleq -\log \mathcal{N}(\mathbf{y}_t \mid \mathbf{H}_t\mathbf{x}_t, \mathbf{R}_t) \tag{31}$$

$$\nabla_{\mathbf{x}_t}\ell(\mathbf{x}_t) = \nabla_{\mathbf{x}_t}\left[\frac{1}{2}\|\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t\|^2_{\mathbf{R}_t}\right] \tag{32}$$

$$= -\mathbf{H}_t^\top\mathbf{R}_t^{-1}(\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t) \Rightarrow \tag{33}$$

$$\mathbf{x}_t^{(k+1)} \leftarrow \mathbf{x}_t^{(k)} + \rho_t^{(k)}\mathbf{M}_t\mathbf{H}_t^\top\mathbf{R}_t^{-1}(\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t) \tag{34}$$

Assume that K such steps of gradient descent are taken where $\mathbf{x}_t^{(0)} = \boldsymbol{\mu}_t^-$ and $\rho_t^{(k)} = \rho_t$ for $k = 0, 1, \ldots, K-1$. Then by (Santos, 1996),

$$\mathbf{x}_t^{(K)} = \arg\min_{\mathbf{x}_t}\left[\frac{1}{2}\|\mathbf{y}_t - \mathbf{H}_t\mathbf{x}_t\|^2_{\mathbf{R}_t} + \frac{1}{2}\|\mathbf{x}_t - \boldsymbol{\mu}_t^-\|^2_{\boldsymbol{\Sigma}_t^-}\right], \tag{35}$$

where $\boldsymbol{\Sigma}_t^-$ is defined implicitly by the choice of ρ_t, $\mathbf{M}_t$, and K, in the sense we make explicit here. Let $\mathbf{C}_t$ simultaneously diagonalize $(1/\rho_t)\mathbf{M}_t^{-1}$ and $\mathbf{H}_t^\top\mathbf{R}_t^{-1}\mathbf{H}_t$ (this is possible since both are symmetric and $(1/\rho_t)\mathbf{M}_t^{-1}$ is positive definite as assumed above):

$$\mathbf{C}_t^\top(1/\rho_t)\mathbf{M}_t^{-1}\mathbf{C} = \mathbf{I} \tag{36}$$

$$\mathbf{C}_t^\top\mathbf{H}_t^\top\mathbf{R}_t^{-1}\mathbf{H}_t\mathbf{C}_t = \boldsymbol{\Lambda}_t \tag{37}$$

Then the equivalent $\boldsymbol{\Sigma}_t^-$ is defined to be:

$$\boldsymbol{\Sigma}_t^- = \mathbf{C}_t\text{diag}(\sigma_i)\mathbf{C}, \text{ where} \tag{38}$$

$$\sigma_i = (1/\lambda_i)[(1-\lambda_i)^{-k} - 1] \text{ if } \lambda_i \neq 0 \text{ and 1 otherwise.} \tag{39}$$

A.3 JUSTIFICATION SETTING MEASUREMENT NOISE TO IDENTITY

To make things simpler, consider the arbitrary specification of $\boldsymbol{R}_t = \boldsymbol{I}$. In an optimization scheme over a Gaussian likelihood, this is preferable because it reduces the objective function to the mean squared error loss, which is simple to implement and fast to compute. Such an arbitrary choice implies that compensated predict step covariance in the Kalman filter $\boldsymbol{\Sigma}_t^- = \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top \left(\boldsymbol{R}_t^*\right)^{-1} \left(\boldsymbol{H}_t^\top\right)^+$, where $\boldsymbol{\Sigma}_t^*$ is the true predict step covariance and $\boldsymbol{R}_t^*$ is the true measurement noise at timestep t. The Moore–Penrose inverse of matrix $\boldsymbol{A}$ is denoted $\boldsymbol{A}^+$.

Proof: Let $\boldsymbol{\Sigma}_t^*$ be the true predict step covariance and $\boldsymbol{R}_t^*$ be the true measurement noise at timestep t. Suppose we wish to identify the quantity of $\boldsymbol{X}$ that makes the following expression true:

$$\boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top \left(\boldsymbol{H}_t \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top + \boldsymbol{R}_t^*\right)^{-1} = \boldsymbol{X} \boldsymbol{H}_t^\top \left(\boldsymbol{H}_t \boldsymbol{X} \boldsymbol{H}_t^\top + \boldsymbol{I}_M\right)^{-1}.$$

This is exactly our Kalman gain expression where the L.H.S represents the optimal estimate and the R.H.S shows an arbitrary matrix $\boldsymbol{X}$ given $\boldsymbol{R}_t = \boldsymbol{I}_M$.

$$\begin{aligned}
\boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top \left(\boldsymbol{H}_t \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top + \boldsymbol{R}_t^*\right)^{-1} &= \boldsymbol{X} \boldsymbol{H}_t^\top \left(\boldsymbol{H}_t \boldsymbol{X} \boldsymbol{H}_t^\top + \boldsymbol{I}_M\right)^{-1} \\
\boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top \left(\boldsymbol{H}_t \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top + \boldsymbol{R}_t^*\right)^{-1} \boldsymbol{H}_t \boldsymbol{X} \boldsymbol{H}_t^\top + \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top \left(\boldsymbol{H}_t \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top + \boldsymbol{R}_t^*\right)^{-1} &= \boldsymbol{X} \boldsymbol{H}_t^\top \\
\left(\boldsymbol{H}_t \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top + \boldsymbol{R}_t^*\right)^{-1} \boldsymbol{H}_t \boldsymbol{X} \boldsymbol{H}_t^\top + \left(\boldsymbol{H}_t \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top + \boldsymbol{R}_t^*\right)^{-1} &= \left(\boldsymbol{H}_t^\top\right)^+ \left(\boldsymbol{\Sigma}_t^*\right)^{-1} \boldsymbol{X} \boldsymbol{H}_t^\top \\
\boldsymbol{H}_t \boldsymbol{X} \boldsymbol{H}_t^\top + \boldsymbol{I}_M = \boldsymbol{H}_t \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top \left(\boldsymbol{H}_t^\top\right)^+ \left(\boldsymbol{\Sigma}_t^*\right)^{-1} \boldsymbol{X} \boldsymbol{H}_t^\top + \boldsymbol{R}_t^* \left(\boldsymbol{H}_t^\top\right)^+ \left(\boldsymbol{\Sigma}_t^*\right)^{-1} \boldsymbol{X} \boldsymbol{H}_t^\top \\
\boldsymbol{H}_t \boldsymbol{X} \boldsymbol{H}_t^\top + \boldsymbol{I}_M = \boldsymbol{H}_t \boldsymbol{X} \boldsymbol{H}_t^\top + \boldsymbol{R}_t^* \left(\boldsymbol{H}_t^\top\right)^+ \left(\boldsymbol{\Sigma}_t^*\right)^{-1} \boldsymbol{X} \boldsymbol{H}_t^\top \\
\boldsymbol{X} = \boldsymbol{\Sigma}_t^* \boldsymbol{H}_t^\top \left(\boldsymbol{R}_t^*\right)^{-1} \left(\boldsymbol{H}_t^\top\right)^+ = \boldsymbol{\Sigma}_t^-
\end{aligned}$$

□

Such a specification considerably simplifies computation in an *implicit* scheme. By setting $\boldsymbol{R}_t = \boldsymbol{I}$, we do not need to store $\boldsymbol{R}_t$ or perform any computations with it. We just need to pick an optimizer that correctly specifies this *compensated* predict step covariance. The argument extended to the nonlinear case with only minor adjustments.

A.4 OPTIMIZER GRID SEARCH DETAILS

The framework proposed in this paper attempts to define Bayesian filtering equations *implicitly* via specifying an appropriate optimizer. This requires a small validation set and a hyperparameter search, which we will now describe in the context of our experiments.

A.4.1 TOY NONLINEAR SYSTEM & STOCHASTIC LORENZ ATTRACTOR

In both the toy nonlinear system and the stochastic Lorenz attactor system, we performed a grid search for Adam (Kingma & Ba, 2014), RMSprop (Tieleman et al., 2012), Adagrad (Duchi et al., 2011), Adadelta (Zeiler, 2012), and gradient descent. For 5 separate Monte Carlo (MC) experiments, we tested every combination of step size $n \in \{1, 3, 5, 10, 25, 50, 100\}$, learning rate $\eta \in \{1.0, 0.5, 0.1, 0.05, 0.01\}$, and decay terms $\gamma, \beta_1, \beta_2 \in \{0.1, 0.5, 0.9\}$ where applicable. Adam's decay terms are β_1 and β_2 whereas γ only applies to RMSprop. For Adam, we always set $\beta_2 = \beta_1$ to simplify the search. In total, this corresponds to 7 test configurations for Adadelta, 35 test configurations each for gradient descent and Adagad, and 105 test configurations each for RMSprop and Adam.

A.4.2 YEARBOOK

For this experiment, we reduced the grid search to only five configurations of Adam where all hyperparameters were set to standard settings ($\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$) and the number of steps n was selected from the set $\{1, 10, 25, 50, 100\}$. As described in A.5.3, we divided up the 80 filtering years of the yearbook dataset into 40 tuning years and 40 testing years. During the tuning years, we had a small held-out validation set of 16 examples for each year that we used to calculate the classification accuracy for the parameters found by each optimizer. In the main paper, we report the optimizer that performed the best on this validation set.

We similarly performed a grid search for the variational Kalman filter (VKF) and particle filter (PF) for the σ^2 term in the transition distribution. Since the transition distribution is unknown, a grid search is required in order to approximate this transition distribution. We report the results for the VKF and PF that performed the best on the same validation set, with $\sigma^2 \in \{0.1, 0.05, 0.01, 0.005, 0.001\}$.

A.5 Extended Experimental Results and Details

We report additional results, tables, and figures from all three experiments in this section. For the yearbook dataset, we give a complete description of the experiment performed.

A.5.1 Toy nonlinear System

In the main paper, we reported results for 100 Monte Carlo (MC) experiments from the *best* optimizer hyperparameters found using a grid search over 5 separate MC experiments. To give the reader some intuition about good hyperparameters in this system, we report the top 20 performing optimizers from the grid search in Table 4. We tested 287 configurations in total across Adam (Kingma & Ba, 2014), RMSprop Tieleman et al. (2012), Adadelta Zeiler (2012), Adagrad Duchi et al. (2011), and gradient descent. Of those 287 configurations, the top 20 performing configurations were exclusively between Adam and RMSprop.

Table 4: Top 20 optimizers found in grid search for the toy nonlinear system. The RMSprop and Adam configurations reported in the main paper are shown in **bold**. n denotes the number of steps used at every time step. η is the learning rate. γ, β_1, β_2 are the decay terms specific to each optimizer.

Method	**RMSE**
RMSprop ($n = 5, \eta = 1.0, \gamma = 0.5$)	6.391 ± 0.230
RMSprop ($n = 50, \eta = 0.1, \gamma = 0.9$)	6.380 ± 0.230
RMSprop ($n = 10, \eta = 0.5, \gamma = 0.1$)	6.293 ± 0.222
RMSprop ($n = 10, \eta = 0.5, \gamma = 0.5$)	6.248 ± 0.240
Adam ($n = 10, \eta = 0.5, \beta_1, \beta_2 = 0.9$)	6.205 ± 0.241
RMSprop ($n = 100, \eta = 0.05, \gamma = 0.9$)	6.201 ± 0.236
Adam ($n = 5, \eta = 1.0, \beta_1, \beta_2 = 0.9$)	6.112 ± 0.239
Adam ($n = 10, \eta = 0.5, \beta_1, \beta_2 = 0.1$)	6.059 ± 0.221
Adam ($n = 25, \eta = 0.1, \beta_1, \beta_2 = 0.9$)	6.027 ± 0.238
RMSprop ($n = 50, \eta = 0.1, \gamma = 0.5$)	6.018 ± 0.223
RMSprop ($n = 100, \eta = 0.05, \gamma = 0.5$)	6.010 ± 0.229
RMSprop ($n = 50, \eta = 0.1, \gamma = 0.1$)	$\mathbf{6.000 \pm 0.227}$
RMSprop ($n = 100, \eta = 0.05, \gamma = 0.1$)	5.987 ± 0.225
Adam ($n = 5, \eta = 1.0, \beta_1, \beta_2 = 0.5$)	5.973 ± 0.218
Adam ($n = 50, \eta = 0.05, \beta_1, \beta_2 = 0.9$)	5.963 ± 0.224
Adam ($n = 10, \eta = 0.5, \beta_1, \beta_2 = 0.5$)	5.953 ± 0.219
Adam ($n = 50, \eta = 0.1, \beta_1, \beta_2 = 0.1$)	$\mathbf{5.842 \pm 0.231}$
Adam ($n = 100, \eta = 0.05, \beta_1, \beta_2 = 0.1$)	5.830 ± 0.232
Adam ($n = 50, \eta = 0.1, \beta_1, \beta_2 = 0.5$)	5.794 ± 0.216
Adam ($n = 100, \eta = 0.05, \beta_1, \beta_2 = 0.5$)	5.780 ± 0.220

From Table 4, it is clear that there is a diverse set of optimizer hyperparameters that can produce good results. The goal of hyperparameter selection in this context is to strike a balance between overfitting and underfitting the likelihood such that it reflects a similar balance between the prior and measurement noise in the explicit filtering sense. It is clear that momentum plays a beneficial role here, since gradient descent and the EKF did not perform as well as optimizers with momentum.

In Figure 3, we take a random experiment and show the filtering distributions from every other time step produced by a Bootstrap filter with $100,000$ particles. On top of the filtering distributions, we show the maximum a posteriori (MAP) estimates produced by Adam with gradient steps $n = 50$, learning rate $\eta = 0.1$, and momentum terms $\beta_1, \beta_2 = 0.1$ (red). On most time steps, our Implicit MAP Filter exactly maximizes the filtering distribution maintained by the particle filter, despite all time steps being non-convex optimization problems. The trials where the Implicit MAP Filter

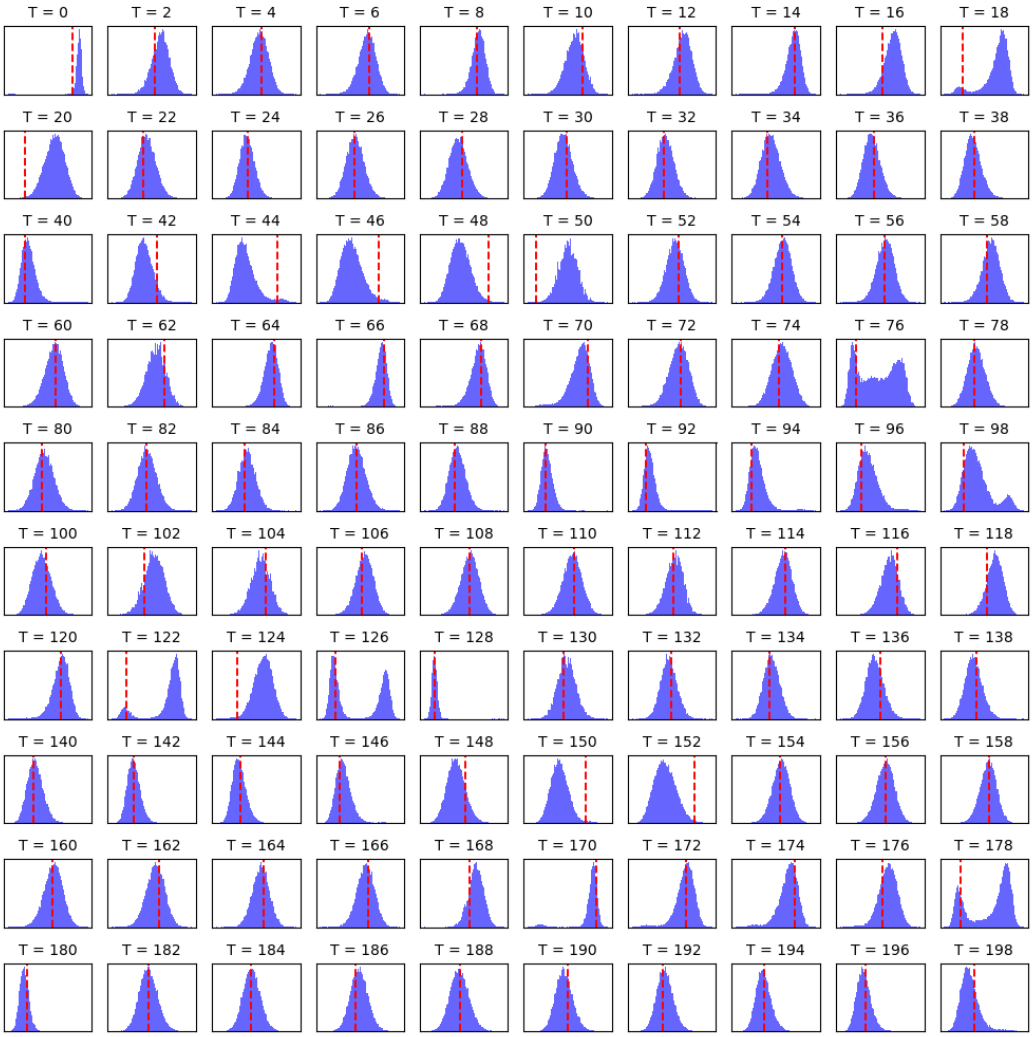

Figure 3: Filtering distribution from the Bootstrap filter (blue) for 100000 particles and the MAP estimates from the Implicit MAP Filter (red) using a single random experiment.

estimates do not correspond to the true MAP most often come on transition trials, where the state is making an aggressive crossing over the barrier in the double-well. These are the trials where the EKF tended to diverge and the UKF equivalently struggled to produce accurate estimates.

In Figure 4, we visualize the update steps of both an EKF and an Implicit MAP Filter with an Adam optimizer ($n = 25, \eta = 0.1, \beta_1 = 0.9, \beta_2 = 0.9$). Both of these methods rely on similar gradient information so it is important to understand why the EKF fails catastrophically and diverges but adaptive optimizers, such as Adam, do not suffer the same effect in this system. This figure sheds some light, showing that the toy nonlinear system can be broken into three different modes: pre-transition, transition, and post-transition. At time step $t = 5$, we are in the pre-transition stage where the objective is approximately convex within the local optimization region. Both Adam and the EKF produce similar estimates in this pre-transition stage. At time steps $t = 12$ and $t = 15$, we are entering the transition phase where the double well shape of the likelihood begins to flatten out. Time step $t = 16$ is the most important step. Here, the update step must cross the barrier at 0 in order to avoid divergence. Adam crosses the barrier at 0 because its incorporation of momentum, which continues to move the estimates despite there being a lack of gradient. The EKF update step does not have momentum. At time step $t = 17$, post-transition, the EKF finds itself on the wrong side of the double well. Since measurements occur only on one side of the double well, as shown at time step $T = 27$, the EKF cannot recover, hence the divergence.

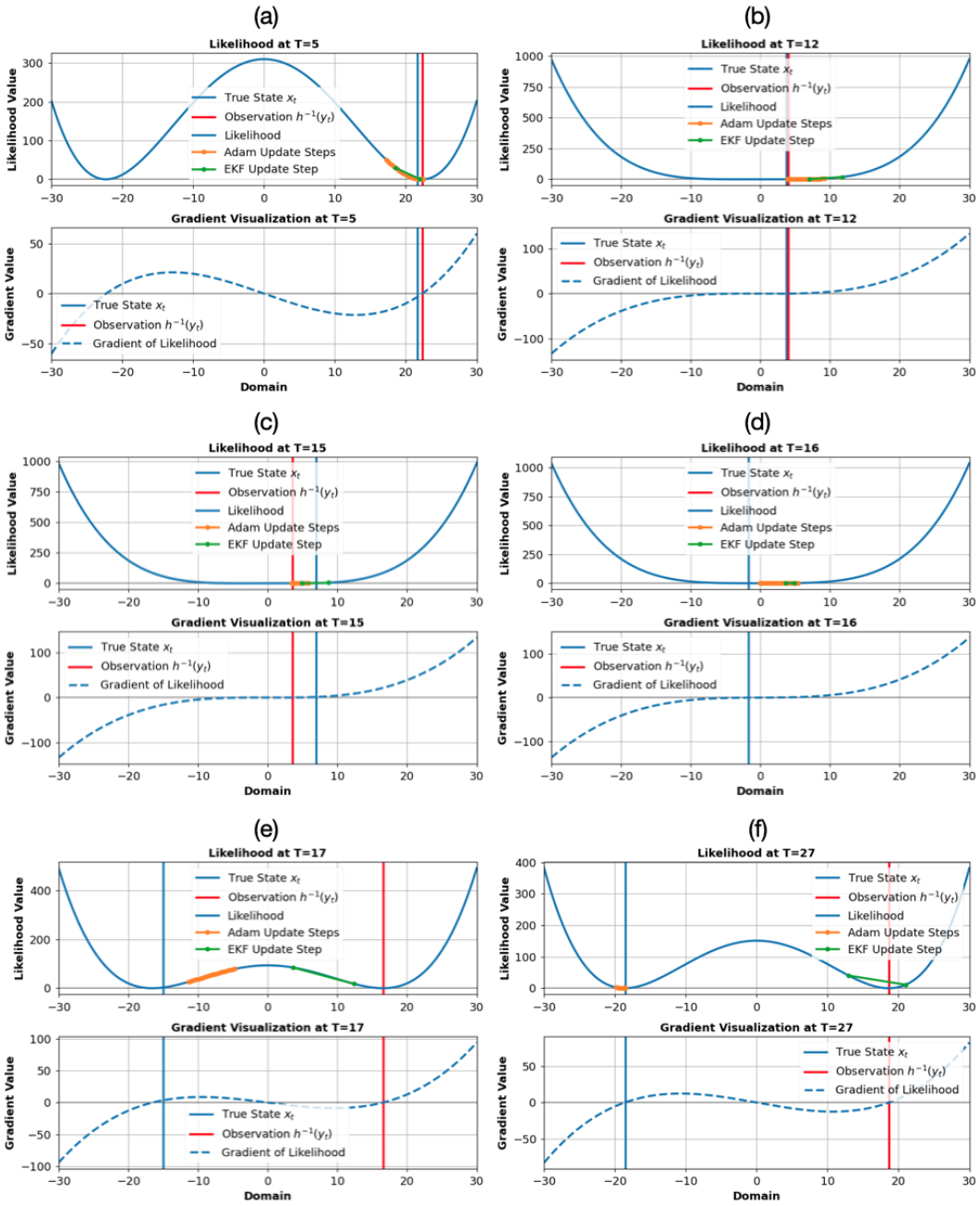

Figure 4: Likelihood and gradient visualization for the toy nonlinear system. (a) pre-transition time steps, (b,c,d) transition time steps. (e, f) post-transition time steps. In (a), the objective function is locally convex and gradient information can be used effectively. The likelihood takes the shape of a double-well. In (b, c, d), gradients approach zero as the double-well flattens out. In (e,f), the double-well picks up again, requiring estimates to be on the correct side of the double-well in order to perform close to optimal.

In Table 5, we show the effect of changing the number of gradient steps for the Adam optimizer reported in the main section, holding all other hyperparameters fixed. The number of gradient steps, n, is the most natural hyperparameter to adjust, and Table 5 demonstrates that the specification of n plays a strong role in performance when attempting to implicitly filter in a time-varying setting.

Table 5: The effect of modifying the number of gradient steps using the optimizer configuration reported in the main section for Adam on the toy nonlinear system.

Method	**RMSE**
Adam ($n = 1, \eta = 0.1, \beta_1, \beta_2 = 0.1$)	10.510 ± 0.263
Adam ($n = 25, \eta = 0.1, \beta_1, \beta_2 = 0.1$)	10.478 ± 0.409
Adam ($n = 3, \eta = 0.1, \beta_1, \beta_2 = 0.1$)	9.749 ± 0.288
Adam ($n = 5, \eta = 0.1, \beta_1, \beta_2 = 0.1$)	9.244 ± 0.266
Adam ($n = 10, \eta = 0.1, \beta_1, \beta_2 = 0.1$)	9.218 ± 0.291
Adam ($n = 100, \eta = 0.1, \beta_1, \beta_2 = 0.1$)	6.575 ± 0.347
Adam ($n = 50, \eta = 0.1, \beta_1, \beta_2 = 0.1$)	5.842 ± 0.231

A.5.2 Stochastic Lorenz Attractor

Similar to the previous section, we report the top 20 optimizers found from the grid search in Table 6 using the ideal condition (RK4). Exactly as before, we tested 287 configurations in total across Adam, RMSprop, Adadelta, Adagrad, and gradient descent. Of those 287 configurations, we achieved a more diverse set of optimizers that performed well compared to the previous system.

Table 6: Top 20 optimizers found in grid search for the stochastic Lorenz attractor with respect to the RK4 case. We also show performance of the same optimizer configurations for Euler and GRW cases. RMSEs that outperform the EKF are shown in **bold**. The same configuration is robust from RK4 $\rightarrow$ Euler. GRW requires different hyperparameters for the optimizer since the top RK4 optimizers do not correspond to top GRW optimizers.

Method	**RMSE (RK4)**	**RMSE (Euler)**	**RMSE (GRW)**
Adam ($n = 10, \eta = 0.05, \beta_1, \beta_2 = 0.5$)	0.903 ± 0.014	$\mathbf{1.189 \pm 0.025}$	6.728 ± 0.101
RMSprop ($n = 10, \eta = 0.05, \gamma = 0.5$)	0.903 ± 0.112	$\mathbf{1.525 \pm 0.009}$	7.526 ± 0.105
RMSprop ($n = 50, \eta = 0.01, \gamma = 0.1$)	0.900 ± 0.079	$\mathbf{1.470 \pm 0.111}$	7.516 ± 0.105
RMSprop ($n = 50, \eta = 0.01, \gamma = 0.5$)	0.896 ± 0.077	$\mathbf{1.473 \pm 0.111}$	7.518 ± 0.105
Adam ($n = 25, \eta = 0.01, \beta_1, \beta_2 = 0.9$)	0.895 ± 0.015	$\mathbf{1.221 \pm 0.026}$	8.335 ± 0.108
Adam ($n = 5, \eta = 0.1, \beta_1, \beta_2 = 0.5$)	0.893 ± 0.014	$\mathbf{1.219 \pm 0.026}$	6.896 ± 0.102
RMSprop ($n = 5, \eta = 0.1, \gamma = 0.5$)	0.893 ± 0.102	$\mathbf{1.118 \pm 0.015}$	7.531 ± 0.105
RMSprop ($n = 10, \eta = 0.05, \gamma = 0.1$)	0.890 ± 0.064	$\mathbf{1.472 \pm 0.112}$	7.517 ± 0.105
Gradient Descent ($n = 10, \eta = 0.01$)	$\mathbf{0.888 \pm 0.090}$	$\mathbf{2.286 \pm 0.148}$	5.962 ± 0.088
Adam ($n = 5, \eta = 0.1, \beta_1, \beta_2 = 0.1$)	$\mathbf{0.882 \pm 0.049}$	$\mathbf{1.416 \pm 0.105}$	7.411 ± 0.104
RMSprop ($n = 10, \eta = 0.05, \gamma = 0.9$)	$\mathbf{0.881 \pm 0.114}$	$\mathbf{1.540 \pm 0.123}$	7.554 ± 0.104
RMSprop ($n = 5, \eta = 0.1, \gamma = 0.9$)	$\mathbf{0.881 \pm 0.113}$	$\mathbf{1.545 \pm 0.124}$	7.555 ± 0.104
Adam ($n = 10, \eta = 0.05, \beta_1, \beta_2 = 0.1$)	$\mathbf{0.877 \pm 0.039}$	$\mathbf{1.417 \pm 0.122}$	7.395 ± 0.104
Adam ($n = 50, \eta = 0.01, \beta_1, \beta_2 = 0.1$)	$\mathbf{0.876 \pm 0.037}$	$\mathbf{1.391 \pm 0.083}$	7.384 ± 0.104
Gradient Descent ($n = 1, \eta = 0.1$)	$\mathbf{0.849 \pm 0.077}$	$\mathbf{2.114 \pm 0.125}$	5.852 ± 0.086
Adagrad ($n = 50, \eta = 0.05$)	$\mathbf{0.846 \pm 0.014}$	$\mathbf{1.213 \pm 0.027}$	6.905 ± 0.102
Gradient Descent ($n = 3, \eta = 0.1$)	$\mathbf{0.799 \pm 0.009}$	$\mathbf{0.960 \pm 0.012}$	$\mathbf{3.061 \pm 0.038}$
Gradient Descent ($n = 5, \eta = 0.05$)	$\mathbf{0.743 \pm 0.010}$	$\mathbf{0.996 \pm 0.014}$	3.575 ± 0.046
Gradient Descent ($n = 25, \eta = 0.01$)	$\mathbf{0.738 \pm 0.010}$	$\mathbf{1.002 \pm 0.015}$	3.627 ± 0.047
Gradient Descent ($n = 3, \eta = 0.05$)	$\mathbf{0.701 \pm 0.018}$	$\mathbf{1.316 \pm 0.027}$	4.911 ± 0.068

It is clear that the same optimizers perform well when the accuracy of the numerical integration degrades. The number of optimizers that outperform the EKF increases as we move from 4th order Runge-Kutta (RK4) to Euler's Method. This reflects a robustness property that is desirable in this implicit formulation. When numerical integration is completely removed, as it is for the Gaussian random walk, most of the optimizers that work well with RK4 do not work well here, demonstrating that the Implicit MAP Filter is not perfectly robust to dynamics misspecification. This result is sensible, as the transition distribution itself changed, not just the numerical approximation of it.

In Figure 5, we show the fitted trajectories for 8 different stochastic Lorenz attractor systems using gradient descent with 3 steps and learning rate $\eta = 0.05$. Two things are important to note: (i) the stochastic Lorenz attractor has extreme variation in the trajectories it produces and (ii) gradient descent does a remarkably good job of fitting this system despite the stochastic differential equation

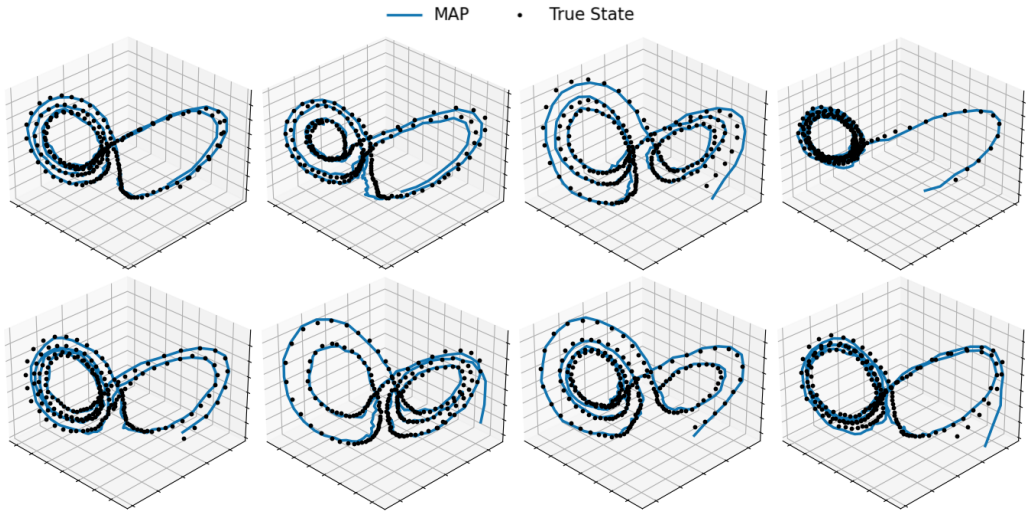

Figure 5: Eight random trajectories for the stochastic Lorenz attractor with 4th order Runge-Kutta and the filter estimates for MAP with gradient descent. 3 steps with learning rate $\eta = 0.05$.

being chaotic. The extreme variation in the trajectories makes every random seed a completely different filtering problem.

Table 7: The effect of modifying the number of gradient steps using the optimizer configuration reported in the main section for Gradient Descent on the stochastic Lorenz attractor (RK4).

Method	**RMSE**
Gradient Descent ($n = 100, \eta = 0.05$)	1.985 ± 0.011
Gradient Descent ($n = 1, \eta = 0.05$)	1.937 ± 0.282
Gradient Descent ($n = 50, \eta = 0.05$)	1.847 ± 0.010
Gradient Descent ($n = 25, \eta = 0.05$)	1.493 ± 0.009
Gradient Descent ($n = 10, \eta = 0.05$)	0.987 ± 0.008
Gradient Descent ($n = 5, \eta = 0.05$)	0.743 ± 0.010
Gradient Descent ($n = 3, \eta = 0.05$)	0.701 ± 0.018

A.5.3 YEARBOOK

The yearbook dataset consists of 37,921 frontal-facing American high school yearbook photos from 1905 - 2013 from 128 high schools in 27 states. Each image is resized to a $32 \times 32 \times 1$ grayscale image and paired with a binary label y, representing the student's gender. The years 1905 - 1930 only have 20 years with images available and years with as little as one photo available, making them not ideal for training and testing. Thus, we use these 20 years as a single pre-training dataset of 886 images.

For all five methods we test, we use a 4-layer convolutional neural network (CNN). Each convolutional layer has a kernel size of 3×3, stride of 1×1, same padding, 32 output channels, ReLU activation, and a 2D max pool layer with kernel size 2×2. We use stochastic gradient descent (SGD) with a fixed learning rate of 10^{-3}, the pre-training dataset of 886 images, and a batch size of 64 to train the initial network for 200 steps. Since the results we report are the average over 10 random seeds, we pre-train the initial network 10 different times in exactly this fashion. This is supposed to resemble samples from some approximate initial starting distribution over neural network weights.

From 1931 - 2010, we take 32 randomly chosen images as a training set and 100 randomly chosen images as a test set at every time step. From 1931 - 1970, we take an additional 16 images as a validation set. In the main paper, we report test accuracy of the configurations with the best average validation performance from 1931 - 1970. The years 1941 and 2006 only had 93 and 70 images

available, respectively. Thus, 1941 has a test set size of 45 and 2006 has a test set size of 38. All other training sets, validation sets, and test sets are exactly as described.

For the static weights approach, we do not do any additional training after the 200 steps of SGD over the pre-training dataset. We report the accuracy of those weights from 1931 - 2010 without any adaptation.

The direct fit approach uses 1000 full batch steps of the Adam optimizer at a fixed learning rate of $\mathbf{1}0^{-3}$ at every time step. This resembles what a practitioners might do to fit the training data. Instead of re-training from scratch, they would likely use the weight initialization from the previous time step. This inherently resembles the Implicit MAP filtering approach we propose here, but we are misspecifying the number of gradient steps by using 1000. 1000 gradient steps, or optimization to near convergence, assumes that the measurement noise is close to $\mathbf{0}$ at every time step. By using this overfitting Implicit MAP Filter, which we call direct fit, we are trying to show why it is important to appropriately define the *implicit* filtering equations. It is naive to not consider the number of gradient steps as an important hyperparameter when working with time-varying objectives.

For the particle filter (PF) and variational Kalman filter (VKF), we explicitly specify a transition distribution $p(\mathbf{w}_t|\mathbf{w}_{t-1}) = \mathcal{N}(\mathbf{w}_t|\mathbf{w}_{t-1}, \sigma^2\boldsymbol{I})$ where $\sigma^2 \in \{0.1, 0.05, 0.01, 0.005, 0.001\}$. We run both of these methods exactly as described in A.1.

Finally, we test five configurations of Adam for the Implicit MAP Filter where all hyperparameters were set to standard settings ($\eta = 0.001, \beta_1 = 0.9, \beta_2 = 0.999$) and the number of steps n was selected from the set $\{1, 10, 25, 50, 100\}$. We use the validation set from the first 40 years to select the number of steps to report in the main paper.

Figures 6, 7, and 8 show the classification accuracy of every MAP filter, PF, and VKF, respectively. For Adam, performance was maximized by $n = 50$ and $n = 100$. The static weights case can be seen as an Implicit MAP Filter with $\mathbf{0}$ process noise over a transition function that is the identity function. This approach is a clear misspecification.

The particle filters were generally only marginally better than the static weights case. This is due to a poorly defined proposal distribution, since it is hard to imagine that we could find the true transition distribution via grid search.

The VKF saw better performance than the PF for nearly all configurations tested. This is due to the fact that we are explicitly optimizing the network weights. However, by explicitly regularizing the network instead of implicitly regularizing via early stopping, we do not match the performance of the Implicit MAP Filter or even the direct fit approach. This is likely due to the fact that the transition distribution is far from correctly specified. Again, we cannot imagine that we could find the correct transition distribution via grid search.

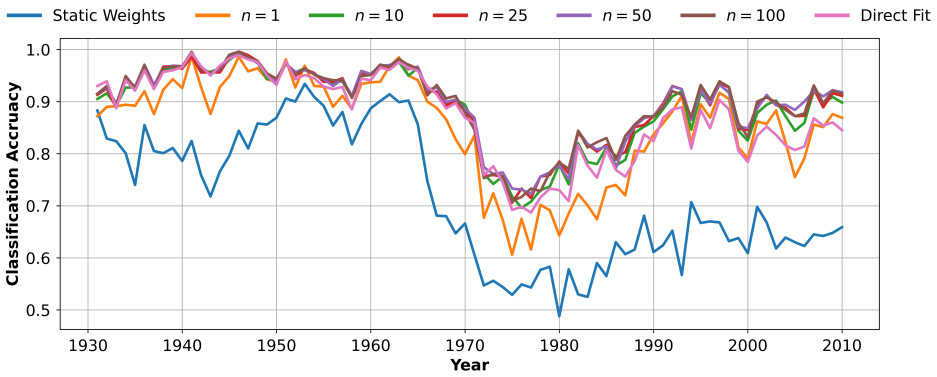

Figure 6: Yearbook dataset filtering results using standard Adam hyperparameters over number of steps $n \in \{1, 10, 25, 50, 100\}$. We also plot the static weights case and direct fit case as comparative baselines.

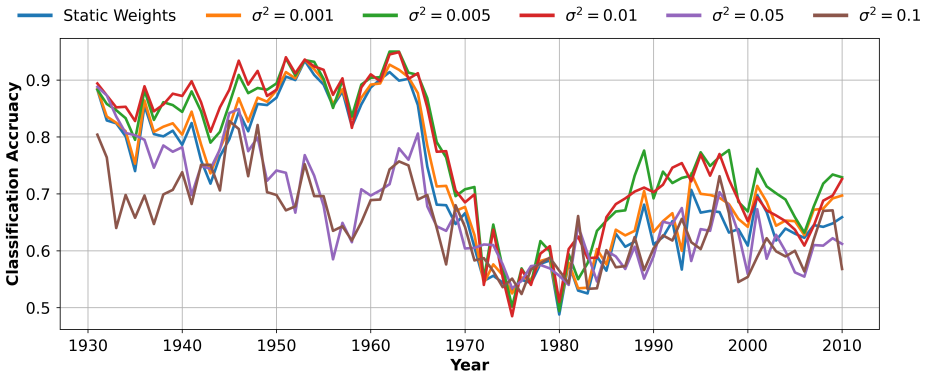

Figure 7: Yearbook dataset filtering results using a particle filter with transition $\sigma^2 \in \{0.1, 0.05, 0.01, 0.005, 0.001\}$. We plot the static weights case as a comparative baselines.

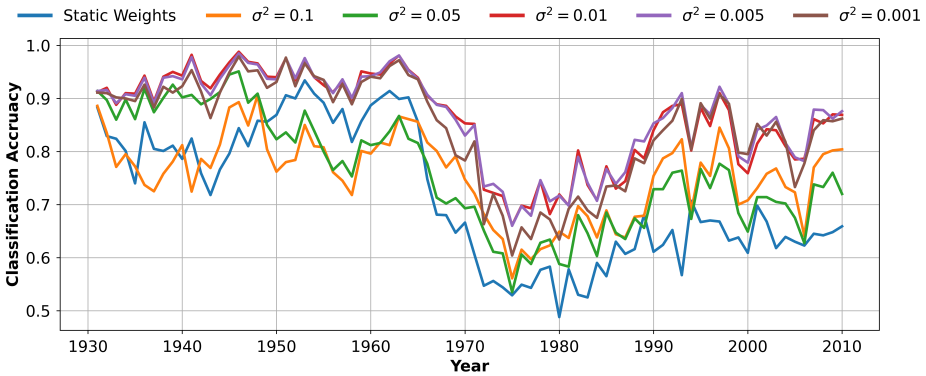

Figure 8: Yearbook dataset filtering results using a variational Kalman filter with transition $\sigma^2 \in \{0.1, 0.05, 0.01, 0.005, 0.001\}$. We plot the static weights case as a comparative baselines.