Replay Overshooting: Learning Stochastic Latent Dynamics with the Extended Kalman Filter

Albert H. Li[1,*], Philipp Wu[2,*], Monroe Kennedy III[1]

***Abstract*— This paper presents *replay overshooting* (RO), an algorithm that uses properties of the *extended Kalman filter* (EKF) to learn nonlinear stochastic latent dynamics models suitable for long-horizon prediction. We build upon overshooting methods used to train other prediction models and recover a novel variational learning objective. Further, we use RO to extend another objective that acts as a surrogate for the true log-likelihood, and show that this objective empirically yields better models than the variational one. We evaluate RO on two tasks: prediction of synthetic video frames of a swinging motorized pendulum and prediction of the planar position of various objects being pushed by a real manipulator (MIT Push Dataset). Our model outperforms several other prediction models on both quantitative and qualitative metrics.**

I. Introduction

A. Motivation

Humans possess a remarkable ability to make accurate long-horizon spatiotemporal predictions based on short observation periods, even in the presence of stochasticity. This ability is so deeply biologically ingrained that the brain processes information on dynamical prediction (e.g. position or velocity) separately from other sensory information about object identity (e.g. color or shape) [1], which suggests that humans maintain complex internal dynamics models to reason about motion. We are interested in an algorithm for robots that replicates this ability in order to facilitate effective decisionmaking in highly dynamical environments. This involves two steps: we first compute the state from observations, then use the dynamics to predict future states.

In step 1, the robot may not directly know the state, but instead must *infer* it from observations. For example, an autonomous vehicle may want to infer the position and velocity of another (state z) from camera images (observations y) and its own actions (control inputs u). Mathematically, the vehicle must compute the *inference model* $p(z_t \mid y_{1:t})$. Since the state is unobserved, we call it *latent*, and presume that the system's evolution is governed by some stochastic latent *dynamics model* $p(z_{t+1} \mid z_t, u_t, \cdot)$ and state information is only indirectly accessible through some *observation model* $p(y_t \mid z_t)$. We may not have exact parametric forms for these models, even in the physics-based regime (for example, when modeling friction), which motivates a learning-based approach. In step 2, we want to predict long-horizon outcomes with the above single-step dynamics, which is naturally subject to cumulative prediction error.

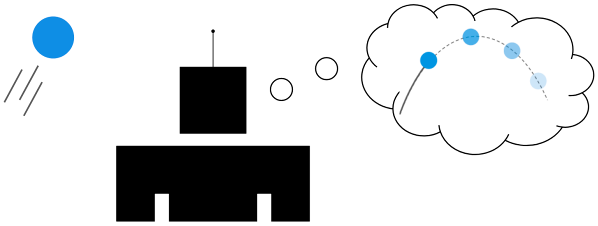

Fig. 1: RO is an algorithm that uses the EKF for inference and learns long-horizon prediction models. We show our models work on a variety of data, both synthetic and real-world.

Accordingly, we are concerned with two questions: (1) what inference algorithm is best for the spatiotemporal setting, and (2) how can we encourage our learning procedure to recover dynamics that perform well even over long horizons? It is worth noting that the inference algorithm is just a means to learn the dynamics, since our ultimate goal is prediction. To that end, we propose *replay overshooting* (RO), an algorithm that uses the *extended Kalman filter and smoother* (EKF/EKS) for inference and a loss function based on *overshooting* methods (discussed in the next section), which encourage learning long-horizon prediction models.

B. Related Work and Extant Problems

This subsection discusses three categories of relevant prior work: conventional variational approaches for learning spatiotemporal inference models, competing Kalman filter-based methods, and overshooting methods.

1) Conventional Inference Models: Recall that the observation and inference models can be expressed as $p(y_t \mid z_t)$ and $p(z_t \mid y_{1:t})$ respectively. These models perform "inverse" roles in the sense that they model the dependence of the observations on the states and vice versa.

Consequently, a large body of work uses the *variational autoencoder* (VAE) [2] to tractably learn some useful latent representation of the data as well as a dynamics model describing their temporal evolution [3], [4], [5], [6], [7], [8]. From this perspective, the observation model acts as the *decoder* and the inference model the *encoder*.

The hallmark of this class of methods is the introduction of an *inference network* $q_\psi(z_t \mid y_{1:t})$ in addition to the dynamics and observation models, which are also typically parameterized as neural nets. As shown in [4] and studied further in [5], in this setting the variational objective depends mainly on the inference and observation models and the dynamics model is relegated to a regularizer for the inference

[1] A. H. Li and M. Kennedy III are with the Department of Mechanical Engineering, Stanford University, Stanford CA, 94305 USA. {ahli, monroek} @stanford.edu.

[2] P. Wu is with Covariant, Emeryville CA, 94608 USA. philipp@covariant.ai.

[*] **Equal contribution. Authors listed in alphabetical order.**

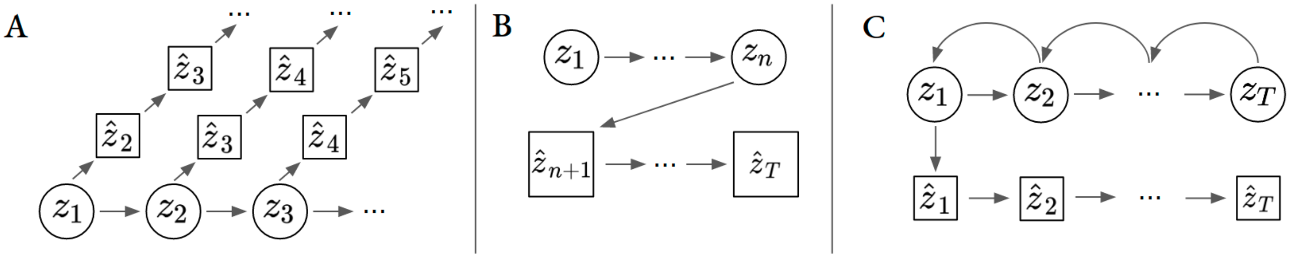

Fig. 2: Overshooting methods. Circles indicate filtering/smoothing and squares prediction. Predicted samples are indicated by $\hat{(\cdot)}$. The overshot latents are observed upon and losses are taken in observation space. **(A)** *Branched*. k predicted states are rolled out from each of T filtered latent distributions. **(B)** *Sequential*. n states are filtered and then k more states are predicted afterwards. **(C)** *Replay* (ours). T smoothing steps are executed (top). The smoothed prior is used to sequentially roll out T prediction distributions (bottom).

network. Often, this makes the learned model unsuitable for prediction because the quality of VAE reconstructions depend only weakly on the dynamics [5].

2) Kalman Inference Models: The *Kalman filter* (KF) is a type of Bayesian filter that performs efficient exact inference for *linear Gaussian state-space models*, which have linear dynamics and observation models with additive white Gaussian noise [9]. Further, the EKF allows approximate inference for nonlinear models by linearizing them and using the regular Kalman filter updates. Most importantly, KFs conduct inference using only the dynamics and observation models, which eliminates the need for an inference network.

Many works study the use of differentiable filters trained solely for state estimation, especially for high-dimensional data like images [10], [11], [12], [13]. These methods do not expressly train for prediction, but rather jointly train the dynamics and observation models to perform good inference when used in a Bayesian filtering algorithm. Further, they are supervised, requiring true labels of all states for each observation. In cases where the state is not well-defined (e.g. the state of a human), this limits their utility, motivating an unsupervised approach (and one designed for prediction).

The caveat is that in latent variable models, the states may not have ready interpretations like “position” or “velocity,” but instead represent some arbitrary features that the model finds useful for reconstruction [2]. Thus, the scope of our approach and others is restricted to scenarios where the goal is planning or predicting in the observation space, possibly aided by latent dynamics models [6], [7], [14], [15], [16].

One such approach is the *Kalman variational autoencoder* (KVAE) [17], which optimizes the desired models with variational inference and performing posterior inference using the standard KF. One limitation of the KVAE is the use of linear ensembles to approximate all models, which allows exact inference. RO does the reverse, learning exact nonlinear models and using the EKF to perform approximate inference.

3) Overshooting Methods: Parallel to the study of filtering methods are methods for learning long-horizon prediction models, which we refer to as *overshooting* methods (coined in the PlaNet paper [6]) [18], [19], [20]. These algorithms train the dynamics model to predict multiple steps into the future, which means the model must be accurate enough to endure cumulative error from long rollouts.

To our knowledge, there are two types of overshooting methods. We call the first *sequential overshooting*, which uses some inference procedure on n observations to recover a confident belief over the state, then predicts (overshoots) k more steps, taking a loss on the overshot steps (where $n+k = T$, the length of training trajectories) [18], [19]. The other we call *branched overshoot*, where we infer now on T steps, but for *each* of these roll out $k < T$ prediction steps [20], [6]. See Fig. 2 to compare these with replay overshooting.

In the case of sequential overshooting, the length-T inference period is sample-inefficient, since we only compute losses over the k overshot steps and discard the rest. On the other hand, branched overshooting is computationally expensive, since we must roll out k predictions for each of T points. Further, to train the model on longer horizons, every branch increases in length, increasing the cost. We would like a method that does not discard useful training data and can inexpensively train for long-horizon predictions.

C. Contributions

The contributions of this paper are (1) RO, an EKF-based overshooting algorithm for learning strong inference and long-horizon prediction models; (2) a surrogate objective derived from KF properties that exceeds the performance of models trained with a variational objective; and (3) an open-source codebase[1] with scripts for experimental duplication.

II. Preliminaries

A. Kalman Filtering and Smoothing

This section presents the mathematics of Kalman filtering/smoothing, summarized from [9]. Consider the following (Markovian) dynamics and observation models:

$$z_{t+1} = f(t, z_t, u_t) + w_t, \quad y_t = g(z_t) + v_t. \tag{1}$$

In the KF formulation, f and g represent deterministic dynamics and observation models and $w_t \sim \mathcal{N}(0, Q_t), v_t \sim \mathcal{N}(0, R_t)$ represent additive white Gaussian noises that introduce stochasticity. In the standard KF, we have

$$f(z_t, u_t) = Az_t + Bu_t, \quad g(z_t) = Cz_t, \tag{2}$$

while in the EKF, f and g can be nonlinear. This gives the following transition distribution, which will be useful later:

$$p(z_{t+1} \mid z_t, u_t) = \mathcal{N}(f(z_t, u_t), Q_t). \tag{3}$$

[1] www.github.com/wuphilipp/replay-overshooting

The goal of the KF is to compute $p(z_t \mid y_{1:t}, u_{1:t}) = \mathcal{N}(\mu_{t|t}, \Sigma_{t|t})$ for $t = 1, \ldots, T$ given some choice of prior $\mathcal{N}(\mu_{0|0}, \Sigma_{0|0})$, which is *exact* for linear models and *approximate* for nonlinear ones. For brevity, we denote $\Omega_F := \{(\mu_{t|t-1}, \Sigma_{t|t-1})\}_{t=1}^T$. This is done iteratively using two updates. In the *prediction update*, we first compute $p(z_t \mid y_{1:t-1}, u_{1:t}) = \mathcal{N}(\mu_{t|t-1}, \Sigma_{t|t-1})$:

$$\mu_{t|t-1} = f(\mu_{t-1|t-1}, u_{t-1}), \tag{4}$$

$$\Sigma_{t|t-1} = A_{t-1}\Sigma_{t-1|t-1}A_{t-1}^\top + Q_{t-1}, \tag{5}$$

where for nonlinear systems $A_t = \frac{\partial f(\mu_{t|t}, u_t)}{\partial \mu_{t|t}}$. In the *measurement update*, we use the previous values to compute

$$K_t = \Sigma_{t|t-1}C_t^\top (C_t\Sigma_{t|t-1}C_t^\top + R_t)^{-1}, \tag{6}$$

$$\mu_{t|t} = \mu_{t|t-1} + K_t(y_t - g(\mu_{t|t-1})), \tag{7}$$

$$\Sigma_{t|t} = (I - K_tC_t)\Sigma_{t|t-1}, \tag{8}$$

where for nonlinear systems, $C_t = \frac{\partial g(\mu_{t|t-1})}{\partial \mu_{t|t-1}}$.

In the learning setting, we have access to trajectories of length T, so we may want a stronger form of inference, i.e. computing $p(z_t \mid y_{1:T}, u_{1:T}) = \mathcal{N}(\mu_{t|T}, \Sigma_{t|T})$, which depends on observations *after* time t as well. This is known as *smoothing*, and there are corresponding Kalman smoothing algorithms. In *Rauch-Tung-Striebel* smoothing, we first perform the filtering pass above, caching certain values, then execute an iterative backwards pass starting from time T:

$$K_t^s = \Sigma_{t|t}A_t^\top(\Sigma_{t+1|t})^{-1}, \tag{9}$$

$$\mu_{t|T} = \mu_{t|t} + K_t^s(\mu_{t+1|T} - \mu_{t+1|t}), \tag{10}$$

$$\Sigma_{t|T} = \Sigma_{t|t} + K_t^s(\Sigma_{t+1|T} - \Sigma_{t+1|t})(K_t^s)^\top. \tag{11}$$

Finally, the most likely latent trajectory can be computed with *maximum a posteriori* (MAP) estimation, or solving $\max_{z_{0:T}} p(z_{0:T} \mid y_{1:T})$. Since the KF operates in the linear Gaussian regime, the joint distribution over the states is a multivariate Gaussian, so the solution to the MAP problem is given by the solutions to $T + 1$ smaller MAP problems computable with the KF: $\max_{z_t} p(z_t \mid y_{1:T})$.

B. Learning Generative Time-Series Models

Using Kalman inference, the primary learnable quantities are $f_\theta, g_\phi, \mu_{0|0}$, and $\Sigma_{0|0}$, where θ, ϕ represent neural network parameters. If we assume fixed noise distributions, we may also directly learn covariances Q and R. If there are additional networks used, for example, to perform intermediate operations on observations as in [10], [17], then they may also be trained end-to-end in tandem.

As discussed in Sec. I-B, we can use the EKS in a variational inference setting to train our models. In particular, [6] showed that the general variational lower bound given any time-series inference model $q_\psi(z_t)$ and Markovian dynamics can be written in a factorized form as

$$\begin{aligned}&\sum_{t=1}^T \Bigg(\mathbb{E}_{q_\psi(z_t)}\left[\log p_\phi(y_t \mid z_t)\right] \\ &\quad - \mathbb{E}_{q_\psi(z_{t-1})}\left[D_{KL}(q_\psi(z_t) \mid\mid p_\theta(z_t \mid z_{t-1}, u_{t-1}))\right]\Bigg) \\ &=: \mathcal{L}_r + \mathcal{L}_{KL},\end{aligned} \tag{12}$$

where for RO, $q_\psi(z_t) = q_{(\theta,\phi)}(z_t \mid y_{1:T}, u_{1:T})$. $\mathcal{L}_r$ represents the *negative reconstruction loss*, which is the sum of the expectations over the observation likelihoods and similarly for $\mathcal{L}_{KL}$, the *negative regularization loss*.

$\mathcal{L}_{KL}$ attempts to make the single-step dynamics transition similar to the variational posterior. For RO, this means that the single-step prediction of the next state without observations should resemble the distribution of the next state given all observations (including future ones).

The use of the KF imposes a curious architecture: the inference pipeline (which includes f_θ and g_ϕ) represents the encoder $q_{(\theta,\phi)}$, while the observation model g_ϕ represents the decoder. In the conventional inference methods discussed in Sec. I-B, q_ψ is some separate inference network that has no relation to f_θ, weakening the training signal, since f_θ only participates in the computation of $\mathcal{L}_{KL}$.

III. Replay Overshooting

This section discusses our main contribution: the RO algorithm, training objectives, and implementation details.

A. The RO Algorithm and the Variational Objective

Training with replay overshooting happens in two passes: the *inference pass* and the *replay overshooting pass*. The inference pass is simply the execution of the smoothing algorithm described in Sec. II-A, which gives smoothed distribution parameters $\Omega_S := \{(\mu_{t|T}, \Sigma_{t|T})\}_{t=0}^T$. Note that this is sufficient to compute $\mathcal{L}_r$ and $\mathcal{L}_{KL}$.

In the RO pass, we "replay" the computation of the same distributions starting from the smoothed prior $\mathcal{N}(\mu_{0|T}, \Sigma_{0|T})$ using *only* the prediction update, i.e. eqns. (4)-(5), and *without* the measurement update, which we refer to as the *extended Kalman predictor* (EKP). This yields a new sequence of replayed distribution parameters $\Omega_P := \{(\bar{\mu}_t, \bar{\Sigma}_t)\}_{t=0}^T$, where $(\bar{\mu}_0, \bar{\Sigma}_0) = (\mu_{0|T}, \Sigma_{0|T})$ and

$$\bar{\mu}_t = f_\theta(\bar{\mu}_{t-1}, u_{t-1}), \tag{13}$$

$$\bar{\Sigma}_t = A_{t-1}\bar{\Sigma}_{t-1}A_{t-1}^\top + Q_{t-1}. \tag{14}$$

Using this procedure, we now recover a set of new variational distributions $q'(z_t) = \mathcal{N}(\bar{\mu}_t, \bar{\Sigma}_t)$ that in turn are used to compute a replayed reconstruction loss:

$$\bar{\mathcal{L}}_r = \sum_{t=1}^T \mathbb{E}_{q'_{(\theta,\phi)}(z_t)}\left[\log p_\phi(y_t \mid z_t)\right]. \tag{15}$$

Now, we can construct the *variational RO* learning objective:

$$\mathcal{L}_{VRO} = \alpha\mathcal{L}_r + (1 - \alpha)\bar{\mathcal{L}}_r + \mathcal{L}_{KL}, \tag{16}$$

where $\alpha \in [0, 1]$ weights the reconstruction loss terms.

Because the RO pass has no measurement updates, dynamics errors propagate freely and the resulting training signal depends strongly on f_θ. Unlike sequential overshooting, all data are used in the RO pass and unlike branched overshooting, we train for length T horizons in every pass.

Online deployment is simple: at time t, the robot computes $p(z_t \mid y_{1:t}, u_{1:t-1})$ with the EKF until $t = k$ (the number of observations). Then, for $t > k$, using just prediction

Algorithm 1 Replay Overshooting

```
Init. parameters $\xi := (f_\theta, g_\phi, \mu_{0|0}, \Sigma_{0|0}, Q, R)$
while $\xi$ not converged do
  for batch $b = 1, \dots, B$ do
    $\Omega_F = EKF(\xi)$                              ▷ filtering, (4)-(8)
    $\Omega_S = EKS(\xi)$                              ▷ smoothing, (9)-(11)
    $\Omega_P = EKP(\xi, \mu_{0|T}, \Sigma_{0|T})$     ▷ RO, (13)-(14)
    $\mathcal{L} = \mathcal{L}_{SRO}(\Omega_F, \Omega_P)$ or $\mathcal{L}_{VRO}(\Omega_S, \Omega_P)$   ▷ (16)/(19)
    Update $\xi$ with stochastic gradient ascent on $\mathcal{L}$
```

updates, the robot computes $p(z_{t+n} \mid y_{1:t}, u_{t+n-1})$, some n-step prediction distribution that can be used for planning (e.g. CEM [21]). The algorithm is summarized in Alg. 1.

B. The Surrogate Objective

In time-series analysis, it is well known that for linear Gaussian models, the exact likelihood over an observation sequence can be computed using the Kalman filter [22]:

$$\begin{aligned} \tilde{p}(y_t) &= \mathcal{N}(C_t \mu_{t|t-1}, C_t \Sigma_{t|t-1} C_t^\top + R_t), \\ \log p(y_{1:T}) &= \sum_{t=1}^{T} \log \tilde{p}(y_t), \end{aligned} \tag{17}$$

where $\tilde{p}$ is a Gaussian density function and we denote eqn. (17) as the *surrogate loss*, $\mathcal{L}_{sl}$, since for nonlinear models, this likelihood expression is approximate. Similarly, we can also construct a replayed surrogate loss term

$$\begin{aligned} \bar{p}(y_t) &= \mathcal{N}(C_t \bar{\mu}_t, C_t \bar{\Sigma}_t C_t^\top + R_t), \\ \bar{\mathcal{L}}_{sl} &= \sum_{t=1}^{T} \log \bar{p}(y_t), \end{aligned} \tag{18}$$

from which we can derive the *surrogate RO* objective:

$$\mathcal{L}_{SRO} = \alpha \mathcal{L}_{sl} + (1 - \alpha) \bar{\mathcal{L}}_{sl}. \tag{19}$$

Note that $\mathcal{L}_{sl}$ depends on the filtered rather than smoothed distributions, in contrast with the computation of $\mathcal{L}_r$.

Since the surrogate objective acts as an approximation of the true log-likelihood, we propose directly optimizing on it rather than the variational objective whenever possible. This avoids the use of high-variance sampling-based techniques like the reparameterization trick, and further, since we are not optimizing a lower bound, we expect to often recover superior models with the surrogate. An experiment on a toy example comparing both objectives is shown in Fig. 3, but a formal mathematical analysis is left for future work.

C. Other Implementation Details

In practice, we parameterize f_θ and g_ϕ as shallow multilayer perceptrons and use a skip connection [23] between the first and last layers of the dynamics network such that we model $z_{t+1} = z_t + f_\theta(z_t, u_t)$. To compute the Jacobians of f_θ and g_ϕ, we use PyTorch [24].

One caveat for the EKF is that all distributions must be multivariate Gaussians. For data with restricted domains like images, we borrow the KVAE architecture [17], which learns an additional VAE to encode the original observations o into latent observations y. We then choose the distribution over the latent observations y and states z to be Gaussian and jointly train the VAE with f_θ and g_ϕ. In this case, we combine a variational objective to train the image autoencoder with the surrogate to train the filter.

If T is large, RO can lead to unstable learning, since before the dynamics converge, a long sequence of inaccurate predictions can lead to exploding gradients [25]. Thus, we institute a ramped *learning curriculum*, where trajectories are lengthened during training from length 2 to T. By truncating the data early on, the training remains stable and learning on increasingly longer sequences becomes easier. This ultimately allows stable learning on the full trajectories.

Finally, we found that in the variational setting, annealing both the KL terms and the RO loss terms often improved performance, since first learning a good EKF/EKS helps stabilize the learning for the dynamics by recovering better smoothed priors, leading to prediction rollouts with less cascading error. As in [26], we found that adding a coefficient β to $\mathcal{L}_{KL}$ sometimes improved performance, with lower values ($\beta < 1$) improving reconstruction quality.

IV. Experiments

A. Overview

1) Datasets: Our model is evaluated on two different datasets. The first is the pendulum example from [5], which consists of sequences of synthetic noisy video frames showing the motion of a swinging motorized pendulum. The second is a subset of the MIT Push Dataset, which consists of planar position trajectories of various objects being pushed on surfaces of varying material by a real ABB IRB 120 robot arm. The data were collected by Vicon motion capture cameras [27], [28]. Both datasets include fixed control inputs for each trajectory.

2) Models: We consider four EKF-based models. Our methods are denoted *surrogate RO* (SRO-EKF), *variational*

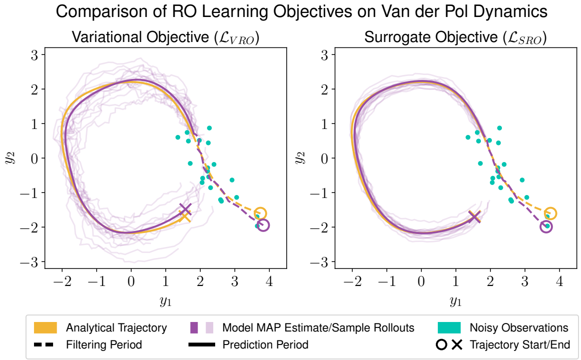

Fig. 3: Typical MAP estimates for EKF models trained with the variational vs. surrogate objective on Van der Pol dynamics. A random (analytical) true trajectory is plotted in gold and noisy initial observations of it in teal. The model's best guess is in dark purple while prediction samples are transparent. The surrogate-trained model filters (dashed lines) without overfitting to noise, predicts (solid lines) the true dynamics accurately, and has low-variance samples. Over 1000 samples, the average L2 error and 95% CI was 0.518 ± 0.022 for VRO and 0.241 ± 0.010 for SRO.

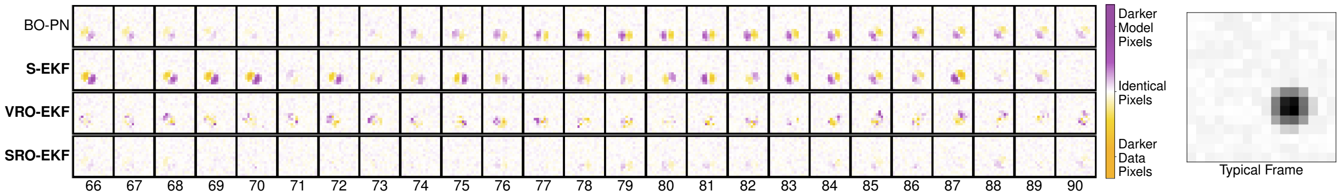

Fig. 4: Pendulum sample errors. Models (ours bold) were given 5 video frames and predicted the next 85. **Left**: differences between the true and predicted pixels for last 25 frames. Colors denote error: gold indicates areas where data have dark and predictions have light pixels (vice versa for purple). White indicates similar values. Note how the SRO-EKF (ours) produces low-error reconstructions while other models exhibit persistent errors. **Right**: a typical frame from the dataset. We only visualize the pendulum mass for simplicity.

RO (VRO-EKF), and *surrogate only* (S-EKF). We also consider the *variational without RO* (V-EKF) case, which consists of the simple extension of conventional variational methods to EKFs with no other modifications. All EKF models directly learn a diagonal-covariance prior $p(z_0)$ and diagonal time-invariant noise covariances Q and R.

We further compare these EKF models to the *PlaNet model with and without branched overshooting* (NO-PN and BO-PN respectively) [6]. All models are summarized here.

TABLE I: Experimental configurations (ours bold).

Model	Description
NO-PN	PlaNet model without overshooting [6]
BO-PN	PlaNet model + branched overshooting [6]
V-EKF	EKF + variational objective only (12)
S-EKF	EKF + surrogate objective only (17)
VRO-EKF	EKF + VRO objective (16)
SRO-EKF	EKF + SRO objective (19)

To allow fair comparisons, we parameterize the dynamics and observation networks as MLPs with three hidden layers and 64 hidden units for all models. We also use the ramped curriculum for all models to stabilize training and share hyperparameters whenever possible.

B. Pendulum Experiments

1) Setup: The pendulum model was as follows:

$$\ddot{\theta} = -\frac{g}{l}\sin(\theta) - \frac{b}{ml^2}\dot{\theta} + \frac{1}{ml^2}u, \tag{20}$$

where m, b, l, u are mass, damping, length, and input torque.

We use a synthetic training dataset of 10,000 noisy grayscale image trajectories with a pixel resolution of 256 along with corresponding random sequences of input torques. The control signals are subjected to a zero-order hold discretization at the sampling frequency of the data and the trajectories are converted into 16 × 16 grayscale video frames of the pendulum mass. Like in [29], we represent the distribution over pixels as a discrete softmax distribution for all models.

We evaluate our models on two metrics normalized by batch size and trajectory length. First, because the joint likelihood over image trajectories is intractable, we instead use the *negative sum of marginal log-likelihoods* (NSMLL), which computes the marginal log-likelihood of each point in time using the EKF and sums over the trajectory. NSMLL generally correlates well with good qualitative performance. Second, we also take an averaged pixelwise L2 loss on the models' mean predictions.

2) Hyperparameters: All models use a latent dimension of 3 and an exponential learning rate schedule with base rate 1e-3 and decay rate 0.975 per 100 steps. Surrogate-trained models use $\alpha = 0.5$ and variational models $\beta = 2.0$. PlaNet models have an overshoot horizon of $k = 2$.

3) Results: Fig. 4 shows that the SRO-EKF is the most resistant configuration to long-horizon prediction error, whereas other models degrade at similar scales. Additionally, Fig. 5 compares learned latent embeddings for SRO-EKF vs. S-EKF. Adding RO yields smooth and structured latent trajectories that correlate strongly to angular position and velocity, as in [5]. In contrast, the S-EKF produces unpredictable latent trajectories, suggesting that RO helps regularize the dynamics.

Our three EKF models also outperform all baselines on quantitative metrics. In particular, adding RO improves otherwise identical configurations. We find that the PlaNet models output overconfident predictions over pixel values, which is heavily penalized by the NSMLL metric. However, the predicted pixel values are still close to the true values, yielding comparable L2 performance. In other words, our EKF models prioritize uncertainty quantification, while methods like PlaNet prioritize mean accuracy, which may be sufficient for good performance in deterministic physics simulators often used in reinforcement learning.

Finally, since the observation model is extremely nonlinear, only optimizing on the surrogate is demonstrably poorer

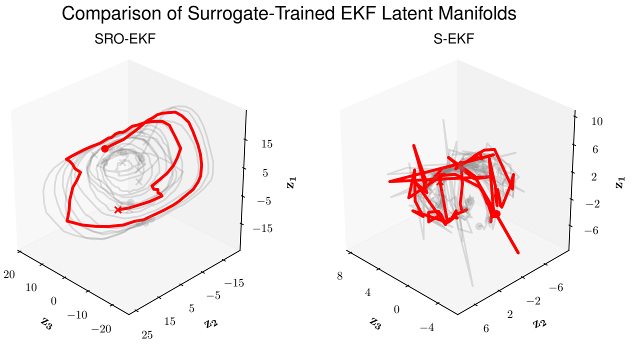

Fig. 5: Predicted latent trajectories (gray) for the pendulum experiment generated from a surrogate-trained EKF with RO (left) and without (right). A sample was chosen and its embedding in both spaces highlighted in red. The EKF with RO learned a smooth saddle-shaped latent manifold that correlates well to sinusoidal motion, resembling the manifolds from [5]. Without RO, the manifold is less structured with noisy and unstable trajectories, suggesting that RO helps regularize the latent dynamics.

TABLE II: Quantitative metrics for all models (lower is better). Probabilistic and L2 losses were estimated by rolling out 100 samples and were normalized by batch size and trajectory length. F#/P# indicates the number of provided points to filter on and the following prediction horizon. Model size refers to the number of parameters of the base dynamics and observation model. The size of the image VAE (5,252,065 parameters) was subtracted from the reported PEND models since they used the same architecture for fair comparison.

	PEND					PUSH				
Model	NSMLL		L2 × 1e-2 (avg. pixel)		# of Model Parameters	NLL		L2 (cm)		# of Model Parameters
	F5/P20	F5/P40	F5/P20	F5/P40		F5/P20	F5/P40	F5/P20	F5/P40	
NO-PN	1024	1072	2.70	2.78	91,392	-7.27	-7.32	0.40	0.62	147,412
BO-PN	957	963	2.57	2.59	91,392	-6.29	-6.55	0.34	0.52	147,412
V-EKF	1148	1150	4.95	5.18	**21,726**	-5.58	-4.397	1.09	2.98	**23,126**
S-EKF	818	870	2.03	2.16	**21,726**	**-9.83**	**-9.65**	0.21	**0.25**	**23,126**
VRO-EKF	698	**699**	2.17	2.20	**21,726**	-6.07	-4.02	0.38	0.48	**23,126**
SRO-EKF	**689**	700	**1.83**	**1.89**	**21,726**	-9.66	-9.32	**0.20**	0.27	**23,126**

than with RO, since the systems is not well-approximated as linear Gaussian. In contrast, the MIT Push experiments show that for simpler data, the surrogate alone may be sufficient for good performance.

C. MIT Push Experiments

1) Setup: We use a whitened curated subset of the MIT Push Dataset from [28] consisting of around 1,600 trajectories. In each trajectory, the object geometry varies as well as the material of the surface. No models condition on shape or material type. We use custom control inputs consisting of the commanded xy position and velocity of the robot end-effector and a contact boolean value indicating whether the end-effector should touch the object.

Since the observation trajectory can be modeled by a multivariate Gaussian, we estimate the NLL by rolling out 100 trajectories and computing the sample mean and covariance over trajectories $y_{1:T}$. We also compute an L2 loss of model prediction means vs. the data.

2) Hyperparameters: All models use a latent dimension of 8 and an exponential learning rate schedule with base rate 5e-3 and decay rate 0.95 per 200 steps. Surrogate-trained models use $\alpha = 0.9$ and variational models $\beta = 0.1$. PlaNet models have an overshoot horizon of $k = 2$. Gradient norms are clipped at 100.

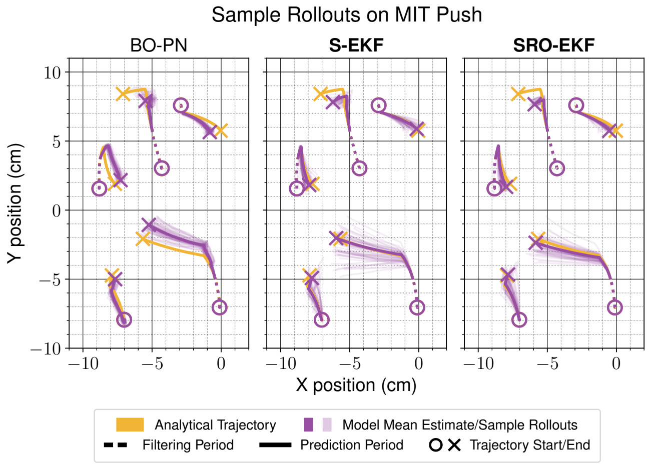

Fig. 6: Example model predictions comparing BO-PN (left), S-EKF (middle), and SRO-EKF (right) on a curated set of object trajectories on the MIT Push dataset (ours bold). Note that BO-PN overshoots on observations y, not latent states z like in [6]. The models' mean prediction is in dark purple and sample rollouts are transparent. The EKF models tend to more accurately predict the object state, especially during sharp transitions in the trajectory, while retaining reasonable uncertainty compared to BO-PN.

3) Results: Fig. 6 compares the branched overshoot PlaNet model vs. our surrogate-trained EKF models. We find that our models yield more accurate mean predictions while also retaining reasonable uncertainty as the prediction horizon increases, in contrast with BO-PN, which clearly exhibits overconfidence (this conclusion is also supported by the metrics in Table II).

We also find that our models tend to degrade much less as the prediction horizon increases by both metrics (with the exception of the VRO-EKF model on the NLL metric, which we attribute to high variance from reparameterization). On the other hand, the PlaNet models exhibit the interesting behavior that the NLL performance slightly increases upon lengthening the prediction horizon. We conjecture that because these models are overconfident, the increase in uncertainty with time improves the loss more than the error in mean prediction worsens it.

We also observe that the variational models improve L2 performance after adding overshooting to the detriment of likelihood metrics. This suggests that any type of overshooting may trade off mean accuracy with overconfidence. However, we see that the use of the surrogate objective significantly diminishes this effect, suggesting that there may exist conditions where the addition of overshooting harmonizes, rather than trades off, with reconstruction objectives. We leave this study for future work.

V. Conclusion

This paper introduced replay overshooting (RO), a method for learning inference and long-horizon prediction models from any type of time-series data using the EKF. By construction, RO provides a strong training signal for the learned dynamics and performs well versus baselines on both synthetic and real prediction tasks on both quantitative and qualitative metrics. We explored model performance using various training objectives and generally find that the EKF and RO provide quantitative and qualitative benefits for prediction models. In the future, we plan to study the integration of RO with planning-based methods for control.

VI. Acknowledgements

We thank Brent Yi for writing the crucial `fannypack` library, for debugging, and for helpful discussions. We thank Mayee Chen, Rachel Thomasson, and Brent Yi for feedback.

References

[1] L. G. Ungerleider and J. V. Haxby, "'What' and 'where' in the human brain," *Current Opinion in Neurobiology*, vol. 4, pp. 157–165, 1994.

[2] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

[3] R. G. Krishnan, U. Shalit, and D. Sontag, "Deep kalman filters," 2015.

[4] R. Krishnan, U. Shalit, and D. Sontag, "Structured inference networks for nonlinear state space models," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, Feb. 2017.

[5] M. Karl, M. Soelch, J. Bayer, and P. van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," in *Proceedings of the 2017 International Conference on Learning Representations*, 04 2017.

[6] D. Hafner, T. Lillicrap, I. S. Fischer, R. Villegas, D. R. Ha, H. Lee, and J. Davidson, "Learning latent dynamics for planning from pixels," in *ICML*, 2019.

[7] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," in *Proceedings of the 2020 International Conference on Learning Representations*, 12 2019.

[8] M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller, "Embed to control: A locally linear latent dynamics model for control from raw images," in *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc., 2015, pp. 2746–2754.

[9] S. Särkkä, *Bayesian Filtering and Smoothing*. USA: Cambridge University Press, 2013.

[10] T. Haarnoja, A. Ajay, S. Levine, and P. Abbeel, "Backprop kf: Learning discriminative deterministic state estimators," in *Proceedings of the 30th International Conference on Neural Information Processing Systems*, ser. NIPS'16. Red Hook, NY, USA: Curran Associates Inc., 2016, p. 4383–4391.

[11] M. A. Lee, B. Yi, R. Martín-Martín, S. Savarese, and J. Bohg, "Multimodal sensor fusion with differentiable filters," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2020.

[12] J. B. Alina Kloss, Georg Martius, "How to train your differentiable filter," in *Proceedings of Robotics: Science and Systems*, Jul. 2020.

[13] P. Karkus, D. Hsu, and W. S. Lee, "Particle filter networks with application to visual localization," in *Proceedings of The 2nd Conference on Robot Learning*, ser. Proceedings of Machine Learning Research, A. Billard, A. Dragan, J. Peters, and J. Morimoto, Eds., vol. 87. PMLR, 29–31 Oct 2018, pp. 169–178.

[14] S. Tian, S. Nair, F. Ebert, S. Dasari, B. Eysenbach, C. Finn, and S. Levine, "Model-based visual planning with self-supervised functional distances," in *Proceedings of the 2021 International Conference on Learning Representations*, 2021.

[15] M. Babaeizadeh, M. Saffar, D. Hafner, H. Kannan, C. Finn, S. Levine, and D. Erhan, "Models, pixels, and rewards: Evaluating design trade-offs in visual model-based reinforcement learning," *ArXiv*, vol. abs/2012.04603, 2020.

[16] R. Hoque, D. Seita, A. Balakrishna, A. Ganapathi, A. Tanwani, N. Jamali, K. Yamane, S. Iba, and K. Goldberg, "Visuospatial foresight for multi-step, multi-task fabric manipulation," in *Proceedings of Robotics: Science and Systems Conference (RSS)*, 07 2020.

[17] M. Fraccaro, S. Kamronn, U. Paquet, and O. Winther, "A disentangled recognition and nonlinear dynamics model for unsupervised learning," in *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc., 2017, pp. 3601–3610.

[18] S. Chiappa, S. Racanière, D. Wierstra, and S. Mohamed, "Recurrent environment simulators," in *Proceedings of the 2014 International Conference on Learning Representations*, 04 2017.

[19] R. Villegas, J. Yang, Y. Zou, S. Sohn, X. Lin, and H. Lee, "Learning to generate long-term future via hierarchical prediction," in *ICML*, 2017.

[20] B. Amos, L. Dinh, S. Cabi, T. Rothörl, S. G. Colmenarejo, A. Muldal, T. Erez, Y. Tassa, N. de Freitas, and M. Denil, "Learning awareness models," in *Proceedings of the 2018 International Conference on Learning Representations*, 2018.

[21] P.-T. de Boer, D. Kroese, S. Mannor, and R. Rubinstein, "A Tutorial on the Cross-Entropy Method," *Annals of Operations Research*, vol. 134, no. 1, pp. 19–67, February 2005.

[22] H. Lütkepohl, *New Introduction to Multiple Time Series Analysis*. Springer Publishing Company, Incorporated, 2007.

[23] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 6 2016, pp. 770–778.

[24] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035.

[25] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *Proceedings of Machine Learning Research*, vol. 28, no. 3. PMLR, 17–19 Jun 2013, pp. 1310–1318.

[26] I. Higgins, L. Matthey, A. Pal, C. Burgess, X. Glorot, M. Botvinick, S. Mohamed, and A. Lerchner, "beta-vae: Learning basic visual concepts with a constrained variational framework," in *Proceedings of the International Conference on Learning Representations*, 2017.

[27] K.-T. Yu, M. Bauzá, N. Fazeli, and A. Rodríguez, "More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing," *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 30–37, 2016.

[28] A. Kloss, S. Schaal, and J. Bohg, "Combining learned and analytical models for predicting action effects from sensory data," *The International Journal of Robotics Research*, 2020.

[29] A. van den Oord and N. Kalchbrenner, "Pixel rnn," in *ICML*, 2016.