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 classic *extended Kalman filter* (EKF) to learn nonlinear stochastic latent dynamics models suitable for long-horizon prediction. Instead of hand-designing an application-specific inference network to estimate the latent state from observations, RO exploits the structure of the EKF to reduce the complexity of the inference problem, improve the training signal for the learned prediction model, and bolster the interpretability of the induced latent manifold. We also build upon overshooting methods commonly used to train other deep prediction models to recover an effective learning objective. We evaluate RO on two experiments: prediction of synthetic video frames of a swinging motorized pendulum and prediction of the planar position of various objects being pushed by a real robotic manipulator (MIT Push Dataset). Our model outperforms several other filter-based dynamics-learning techniques 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 latent 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.

In state estimation theory, the aforementioned observation period prior to prediction is known as *filtering* [2], wherein we compute the distribution $p(z_t \mid u_{1:t-1}, y_{1:t})$ over the latent state at the current time t, z_t, given the observation and control sequences $y_{1:t}$ and $u_{1:t-1}$. If future observations are available, we can use *smoothing* [2] to compute an even better distribution $p(z_t \mid u_{1:T}, y_{1:T})$, where $T > t$. We refer to filtering/smoothing as forms of *posterior inference*, wherein we compute a posterior over z given some observations.

We are instead primarily concerned with *prediction*, where we compute $p(z_T \mid u_{1:T-1}, y_{1:t})$ for $T > t$. However, since the state is unobservable, we must first use posterior inference to compute a belief over past and current states before we can accurately predict future ones. At best, we may have a dynamics and observation model $f(t, z, u)$ and $g(z)$ at our disposal, but often, these are unknown and difficult to analytically derive, motivating a data-driven approach.

[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.**

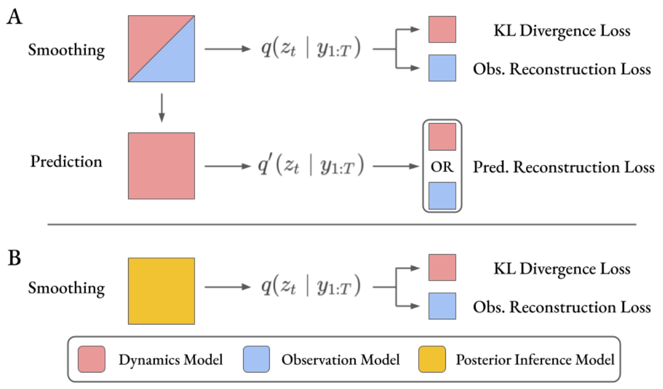

Fig. 1: Comparison of *replay overshooting* (A) versus conventional filter-based dynamics learning models (B). In RO, the dynamics model participates strongly in posterior inference using Kalman updates while in many competing methods, a separate inference model is learned, weakening the dynamics training signal. RO also features a second gradient path that exclusively trains the dynamics.

Many existing works perform posterior inference by introducing a third *inference model* like a recurrent neural network [3], [4], [5]. However, with these methods, the inference model has no intrinsic relation to the dynamics, yet operations through it dominate the training procedure. This detracts from the amount of training signal apportioned to learning f, often failing to produce a good predictor.

To remedy this, we propose *replay overshooting* (RO), an algorithm predicated on the structure of the *extended Kalman filter* (EKF). The EKF is a state estimation algorithm that can perform posterior inference with f and g alone, eliminating the need for a separate inference network (Fig. 1) [6]. We parameterize f and g as neural networks and optimize them using variational inference, extracting f for standalone prediction. RO reduces the complexity of the learning problem and encourages a superior training signal for the dynamics f, permitting accurate long-horizon predictions.

B. Related Work

The use of Bayesian filters to learn dynamical models is well-studied. For example, the decades-old dual extended Kalman filter treats the network weights as additional states and runs two filters in parallel to jointly estimate the weights and states using maximum likelihood estimation [7]. Other works on state-space prediction have used alternative methods like dynamic factor analysis to estimate the posterior [8].

However, these methods suffer from poor scalability in model size or observation dimension, which impedes the training of large neural networks on high-dimensional data.

The later advent of the *variational autoencoder* (VAE) [9] led to a resurgence of filter-based learning methods by allowing tractable sampling-based posterior inference [10]. The *deep Markov model* learns an inference model $q_\psi(z_t \mid y_{1:t})$ parameterized by a bi-RNN in addition to dynamics and observation models [11], [3]. The *deep variational Bayes filter* utilizes a reparameterization of the dynamics to help improve gradient paths at the expense of losing the expressiveness of models parameterized by neural networks [4]. The *Kalman variational autoencoder* (KVAE) uses an image autoencoder to embed observations and associate them with a secondary latent state, learning a dynamics model on both latent states and latent observations to disentangle the dynamics from the higher-dimensional image data [12].

Many similar methods in the deep and reinforcement learning literature exist for planning in latent spaces [13], [14], [15], [5], [16]. One consistent idea presented in these models is training for prediction by *overshooting*, or predicting several steps into the future in order to produce a strong training signal for the dynamics model. Replay overshooting builds upon these ideas to jointly train a good dynamics model as well as a strong inference model.

Finally, recent work also explores learning differentiable filters for the sake of filtering alone [17], [18], [19]. We found that simply training filters does not often yield good predictors, since the learned models tend to strongly rely on observed data rather than the dynamics model to facilitate filtering, disincentivizing dynamics learning.

C. Contributions

The major contributions of this paper are

- replay overshooting, an EKF-based training procedure for learning strong inference and prediction models;
- experiments demonstrating the superiority of replay overshooting over several baselines in the literature;
- and an open-source codebase[1], including scripts for experimental duplication on presented datasets and more.

II. Preliminaries

A. Bayesian Filtering and Smoothing

This section summarizes Bayesian filtering/smoothing. For more detail and filter equations, we refer the reader to [2]. Consider the following discrete-time model with additive noises w_t, v_t drawn from arbitrary distributions:

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

Let the dynamics be Markovian and the measurements be conditionally independent given the state. The discrete-time Bayesian filtering problem consists of recursively computing a distribution over the current state given a history of observations and control inputs $p(z_t \mid y_{1:t}, u_{1:t-1})$.

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

To recover this distribution, we assume knowledge of a distribution over the predicted state given only prior measurements $p(z_t \mid y_{1:t-1}, u_{1:t-1})$. Then, by Bayes' rule, the *measurement update* yields:

$$p(z_t \mid y_{1:t}, u_{1:t-1}) = \frac{p(y_t \mid z_t) p(z_t \mid y_{1:t-1}, u_{1:t-1})}{\int p(y_t \mid z'_t) p(z'_t \mid y_{1:t-1}, u_{1:t-1}) dz'_t}, \tag{2}$$

and given a stochastic dynamics model $p(z_{t+1} \mid z_t, u_t)$, the *prediction update* yields:

$$p(z_{t+1} \mid y_{1:t}, u_{1:t}) = \int p(z_{t+1} \mid z_t, u_t) p(z_t \mid y_{1:t}, u_{1:t-1}) dz_t, \tag{3}$$

which becomes the prior for the next measurement update.

Though this computation is generally intractable, given a prior distribution over the initial state $p(z_1)$, the Kalman filter allows the analytical recovery of a Gaussian distribution over the state given a linear Gaussian system with Gaussian noise

$$z_{t+1} = A_t z_t + B_t u_t + w_t, \quad y_t = C_t z_t + v_t, \tag{4}$$

where $w_t \sim \mathcal{N}(0, Q_t)$ and $v_t \sim \mathcal{N}(0, R_t)$. The resulting dynamics and observation distributions can be written

$$\begin{aligned} p(z_{t+1} \mid z_t, u_t) &= \mathcal{N}(z_{t+1}; A_t z_t + B_t u_t, Q_t), \\ p(y_t \mid z_t) &= \mathcal{N}(y_t; C_t z_t, R_t). \end{aligned} \tag{5}$$

We can also directly recover a Gaussian over the observations given the *distribution* over the latent state instead of a sample, where the mean and covariance over z are denoted μ^z, Σ^z:

$$p(y_t \mid \mu^z_t, \Sigma^z_t) = \mathcal{N}(y_t; C_t \mu^z_t, C_t \Sigma^z_t C_t^\top + R_t). \tag{6}$$

This allows us to compute the likelihood of the observation sequence without sampling-based techniques like the reparameterization trick [2], [9]. Finally, the Kalman smoother can also exactly recover the Gaussian smoothed posterior $p(z_t \mid y_{1:T}, u_{1:T})$ [2].

For the general nonlinear system (1), the filter can at best be made approximate. The most common technique is the EKF, wherein f and g are linearized, yielding modified update equations resembling the standard updates. Since we want to learn nonlinear models, this is our method of choice.

B. Learning Generative Time-Series Models

We choose to parameterize the dynamics and observation models as the neural networks $f_\theta(t, z_t, u_t)$ and $g_\phi(z_t)$ respectively. Following from variational inference [9], the log-likelihood of a sequence of observations $y_{1:T}$ conditioned on control inputs $u_{1:T}$ is lower bounded by the expression

$$\begin{aligned} &\log p(y_{1:T} \mid u_{1:T}) \geq \\ &\quad \mathbb{E}_{q(z_{1:T} \mid y_{1:T}, u_{1:T})} \left[\log p(y_{1:T} \mid z_{1:T}, u_{1:T})\right] \\ &\quad - D_{KL}(q(z_{1:T} \mid y_{1:T}, u_{1:T}) \mid\mid p(z_{1:T} \mid u_{1:T})), \end{aligned} \tag{7}$$

where q is a variational distribution that approximately represents the true posterior. Further, [3] proved that this lower bound can be further factorized as

$$
\begin{aligned}
&\log p(y_{1:T}) \geq \sum_{t=1}^{T} \mathbb{E}_{q(z_t|y_{1:T})} \left[\log p(y_t \mid z_t)\right] \\
&- D_{KL}(q(z_1 \mid y_{1:T}) \mid\mid p(z_1)) \\
&- \sum_{t=2}^{T} \mathbb{E}_{q(z_{t-1}|y_{1:T})} \left[D_{KL}(q(z_t \mid z_{t-1}, y_{1:T}) \mid\mid p(z_t \mid z_{t-1}))\right] \\
&= \mathcal{L}_{r,s} + \mathcal{L}_{KL},
\end{aligned}
\tag{8}
$$

where the conditioning on $u_{1:T}$ has been omitted for brevity.

The first summation is the smoothed reconstruction loss $\mathcal{L}_{r,s}$, while the summed KL divergence terms are denoted $\mathcal{L}_{KL}$. These models can be trained by maximizing the lower bound using stochastic backpropagation [9].

III. EKF Training with Replay Overshooting

In this section, we present the replay overshooting method. RO works in two passes: a *smoothing pass* that trains the model's posterior inference by computing $q(z_t \mid u_{1:t-1}, y_{1:t})$ (Sec. III-A) and a prediction pass that computes an alternate factorization of q, denoted $q'(z_t \mid u_{1:t-1}, y_{1:t})$ (Sec. III-B), which provides an exclusive gradient path for the dynamics model. In the following sections, we provide details and compare to other methods.

A. Posterior Inference with the Neural EKF

The *neural EKF* is jointly composed of the dynamics and observation models f_θ and g_ϕ. As stated in Sec. I-A, since the nature of Bayesian filtering allows us to perform inference without a third network, we can use the neural EKF to reduce the number of learnable parameters, eliminate the need to design and tune a new network, and replace the gradient paths for training the inference network with new paths for the dynamics and observation models.

Empirically, we have found that the EKF is an effective learner even when f_θ and g_ϕ are parameterized by shallow multi-layer perceptrons (MLPs) without any additional features like normalization, dropout, etc. Additionally, using a skip connection between the first and last layers of the dynamics such that we model $z_{t+1} = z_t + f_\theta(t, z_t, u_t)$ helps improve learning as in ResNet [20]. Though the EKF requires the Jacobians of f_θ and g_ϕ, since the EKF is differentiable, these are easily computed using autodifferentiation libraries in deep learning frameworks like PyTorch [21], [22].

Further, the architecture does not change even when f_θ instead represents a continuous-time dynamical model. Instead, the Kalman update equations are simply replaced by the *Kalman-Bucy* equations [23] and dynamical rollouts are conducted using differentiable ordinary differential equation (ODE) solver operations rather than looped discrete updates. This allows us to exploit the benefits of neural ODEs [24], including constant-time memory complexity, customizable ODE solvers that balance solution quality vs. computational complexity, and infinite-resolution data imputation. To our knowledge, this is the first framework to unify discrete and continuous-time dynamics learning, unlike other architectures that only operate in continuous-time [24], [25], [26].

One caveat for the EKF is that the distributions over states and observations must be multivariate Gaussians. For data with restricted domains like images, we borrow the KVAE architecture [12], which learns an additional VAE to encode the original observations o into latent observations y. We can then freely choose the distribution over the latent observations y and latent states z to be Gaussian and jointly train the VAE with the dynamics and observation models.

Finally, some related methods propose the use of standard rather than extended Kalman updates [4], [12]. These methods learn an ensemble of linear models and use a neural network to compute the weights between them at every time step, yielding an *approximate linear dynamical model* that enjoys the benefits of the standard Kalman filter's *exact posterior inference*. Conversely, the EKF instead learns an *exact nonlinear dynamical model* but can only perform *approximate posterior inference* using the extended Kalman updates. We show experimentally that the latter approach is superior on the tasks examined in Sec. IV.

B. Replay Overshooting

The objective in eqn. (8) provides a strong training signal for posterior inference through the reconstruction loss term $\mathcal{L}_{r,s}$. However, the dynamics distribution only appears in the KL divergence terms. For methods that do not utilize the dynamics model during posterior inference (e.g. [3], [4], [5]), this severely weakens the training signal for prediction. We observe that this often overtrains the observation model to compensate for dynamic model inaccuracy, which precludes learning an effective standalone predictor f_θ.

In contrast, many methods propose the idea of overshooting, or rolling out their dynamics model over several steps to provide a training signal that backpropagates exclusively through f_θ. *Observation overshooting* [15], [5] consists of rolling out "branched" predictions at every state in a length-n latent trajectory for an additional $k < n$ steps, then computing the losses on the corresponding predicted future observations. While this method enriches the training signal, it is inefficient for large k or high-dimensional observations like images [5]. Alternative methods first filter on n steps then compute a reconstruction prediction loss over an additional k steps from only the last state of the filtered sequence, which is generally less computationally intensive [13], [14]. We refer to these two classes of overshooting as *branched* and *sequential* overshooting.

In *replay overshooting*, we perform sequential overshooting from the smoothed initial condition instead of the end of the filtered sequence, which "replays" the distributions we have already computed in the smoothing pass but without intermittent measurement updates. This allows us to optimize our models using (8), wherein the KL loss terms regularize the smoothed posterior with the transition model $p(z_t \mid z_{t-1}, u_{t-1})$. These terms also encourage the transition distributions to resemble the smoothed posteriors, which

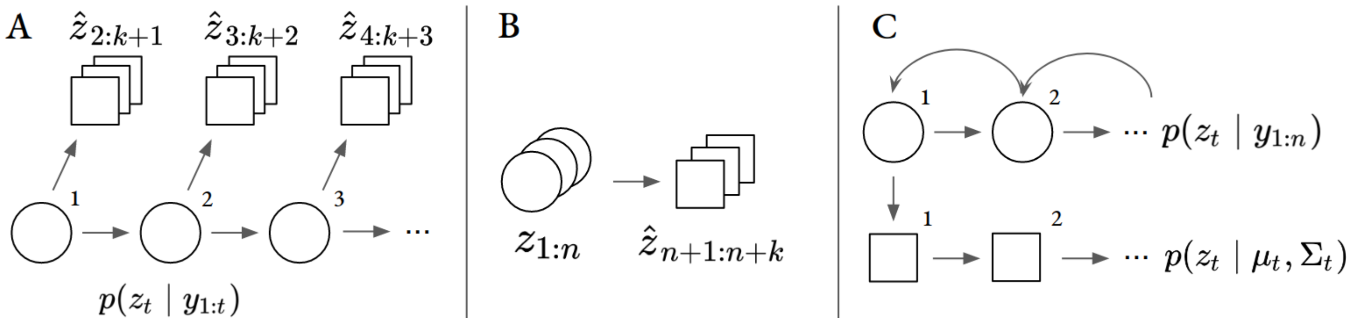

Fig. 2: Overshooting methods. Circles indicate filtering or smoothing and squares prediction. Superscripts indicate time step. Predicted samples are indicated by $\hat{(\cdot)}$. (A) *Branched* overshoot [15], [5]. k predicted states are rolled out from each of n filtered latent distributions. (B) *Sequential* overshoot [13], [14]. n states are filtered and then k more states are predicted afterwards. (C) Replay overshoot (ours). n smoothing steps are executed (top). The smoothed prior is used to sequentially roll out n prediction distributions (bottom).

enjoy the information from the full observation sequence. Fig. 2 summarizes the various overshooting methods.

If n is large, then replay overshooting alone can lead to unstable learning, since before the dynamics model is near convergence, a long sequence of inaccurate predictions can easily lead to exploding gradients, which also occurs when rolling out time-series predictions with RNNs [27]. We have also observed that this effect practically constrains the value of k chosen in standard overshooting. To remedy this, we institute a ramped *learning curriculum*, where the observation trajectories are lengthened during training from length 2 to n. Learning is easier on short sequences, so 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.

C. Mixed Learning Objectives

Since the Kalman update equations allow direct propagation of the prediction distributions, we can directly recover an alternate joint distribution over the latent states:

$$q'(z_{1:T} \mid y_{1:T}, u_{1:T}) = p(z_1 \mid y_{1:T}) \prod_{t=2}^{T} p(z_t \mid z_{t-1}, u_{t-1}), \tag{9}$$

where $p(z_1 \mid y_{1:T})$ is computed from the smoothing pass and the distributions in the product are computed from Kalman prediction updates. Like in observation overshooting, this yields a *prediction reconstruction objective*:

$$\mathcal{L}^{y}_{r,p} = \mathbb{E}_{q'(z_{1:T} \mid y_{1:T}, u_{1:T})} \left[\log p(y_{1:T} \mid z_{1:T}, u_{1:T}) \right]. \tag{10}$$

We can now augment the lower bound in (8) and consider a new training objective derived from RO:

$$\mathcal{L}^{1}_{mixed} = \alpha \mathcal{L}_{r,s} + (1-\alpha) \mathcal{L}^{y}_{r,p} + \mathcal{L}_{KL}, \tag{11}$$

where $0 \leq \alpha \leq 1$. We refer to this as *observation replay overshooting* (ORO).

Alternatively, consider the scenario where an oracle provides true latent states $\bar{z}_{1:T}$ corresponding to observations $y_{1:T}$. In this scenario, we could train the dynamics directly by maximizing the likelihood of the true states. One approach is to use the means of the smoothed latent distributions $\bar{z}_{1:T} = \mu^{z}_{1:T}$ as latent targets when rolling out the dynamics. Practically, this works well since the smoother often converges much earlier than the dynamics alone. This perspective is simply the replayed version of the *latent overshooting* method presented in [5], so we refer to it as *latent replay overshooting* (LRO).

Formally, we can maximize the *latent prediction reconstruction objective* $\mathcal{L}^{z}_{r,p}$:

$$\mathcal{L}^{z}_{r,p} = p(\bar{z}_1 \mid y_{1:T}) \prod_{t=2}^{T} \log p(\bar{z}_t \mid \bar{z}_{t-1}, u_{t-1}). \tag{12}$$

This yields a second modified learning objective:

$$\mathcal{L}^{2}_{mixed} = \alpha \mathcal{L}_{r,s} + (1-\alpha) \mathcal{L}^{z}_{r,p} + \mathcal{L}_{KL}. \tag{13}$$

Finally, we found that annealing both the KL terms and the prediction loss terms often helped, since first learning a good smoother helps stabilize the learning for the dynamics model by recovering more accurate distributions over the initial condition $p(z_1 \mid y_{1:T})$, leading to prediction rollouts with less cascading error. As in [28], we found that adding a coefficient $\beta > 1$ to $\mathcal{L}_{KL}$ sometimes improved performance.

IV. Experiments

A. Setup

Our model is evaluated on two different datasets. The first is the pendulum example from [4], 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 objects of varying geometry being pushed on surfaces of varying material by a real ABB IRB 120 robot arm. The data were collected by Vicon motion capture cameras [29], [30]. The quality of predictions was evaluated with two metrics: the negative log-likelihood (NLL) as well as the average L2 loss over 100 samples.

We compare our models to the branched latent overshoot, branched observation overshoot, and no overshoot recurrent state space models (L-RSSM, O-RSSM, and N-RSSM) from [5]. We also compare to the linear ensemble EKF dynamics model from [12] trained with ORO (O-LE).

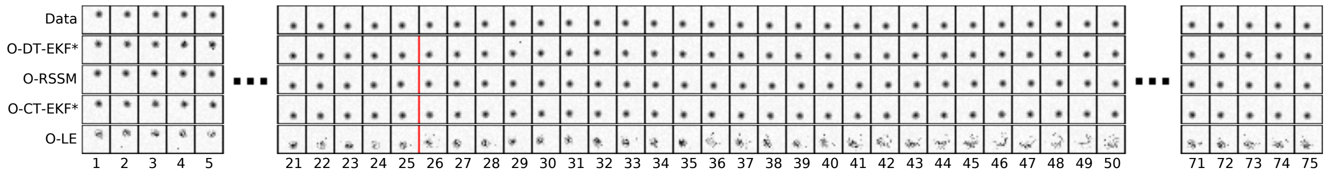

Fig. 3: Curated predictions for pendulum experiments. The top row is the ground truth data. Our methods are denoted with an asterisk. The red line ($t = 25$) denotes the end of filtering and start of prediction. The two best models quantitatively (O-DT-EKF and O-RSSM) yield accurate sampled predictions over the whole sequence. The overconfidence of the RSSM model can be observed in frames 35-40, where the predictions are slightly offset from the data. Additionally, the continuous-time model (O-CT-EKF) also generates high-quality predictions despite early stopping. Finally, the linear ensemble model (O-LE) suffers from degrading reconstructions over time. The distribution over pixels appears wide, which prevents the NLL loss from becoming too large, but causes the L2 loss to increase substantially.

We test four versions of our model: the discrete-time model with LRO, ORO, and no overshooting (L-DT-EKF, O-DT-EKF, N-DT-EKF) as well as a continuous-time model trained with ORO (O-CT-EKF). All EKF models directly learn a diagonal-covariance Gaussian prior $p(z_1)$ and diagonal time-invariant noise covariances Q and R. To make comparisons across models fair, we parameterized the dynamics and observation networks as MLPs with three hidden layers for all models. We also used the ramped curriculum for all models to stabilize training.

The continuous-time models trained more slowly by 1 to 2 orders than the discrete-time models and were early-stopped after about one day of training instead of to convergence to roughly normalize model performance with wall clock time. Still, the continuous-time models remain competitive. In the future, recent speedups for neural ODEs may be applied to improve the runtime and performance [31], [32].

B. Pendulum Experiments

The pendulum data were generated from the model

$$\ddot{\theta} = -\frac{g}{l}\sin(\theta) - \frac{b}{ml^2}\dot{\theta} + \frac{1}{ml^2}u, \tag{14}$$

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

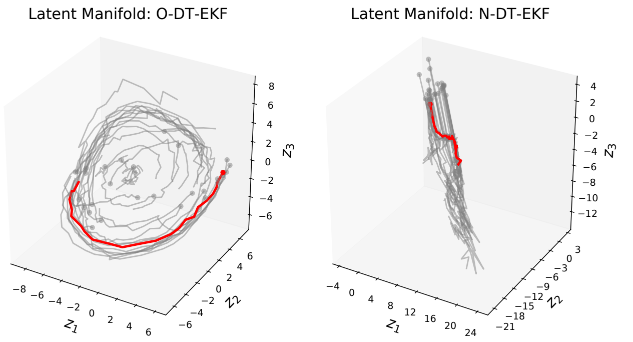

Fig. 4: Predicted latent manifolds (gray) for the pendulum experiment generated from using observation replay overshooting (left) and no overshooting (right). A sample was chosen and its embedding in both spaces highlighted in red. The figure suggests replay overshooting learns interpretable manifolds that correlate well to sinusoidal motion, resembling the manifolds from [4]. Without overshooting, the manifold does not form a meaningful geometry (all states go from one corner to the other regardless of initial condition). This poor qualitative result accompanies equally poor performance on quantitative metrics in Sec. IV.

We constructed a training dataset of 10000 trajectories injected with Gaussian noise along with corresponding random sequences of input torques. The control signals were subjected to a zero-order hold discretization at the sampling frequency of the data and the trajectories were converted into 16×16 grayscale video frames. Like in [33], we represented the distribution over pixels as a discrete softmax distribution instead of a density over continuous pixel values.

The RO models all used $\alpha = 0.5$ for the joint reconstruction weight, the RSSM models used overshooting length $k = 2$, and the O-LE model used an ensemble of size 10. All models used the same image VAE to recover latent embeddings.

Fig. 4 illustrates the learned latent embedding of the O-DT-EKF versus the N-DT-EKF. We found that the resulting latent manifold was well-correlated with the pendulum's angle and angular velocity, consistent with findings from [4]. We suspect the improved training signal from RO helps shape the manifold, whereas when training for posterior inference alone, the latent space remains less structured as the measurement update strongly corrects dynamics errors.

Quantitatively, the three ORO-trained models significantly outperformed all others in the NLL criterion. ORO models tended to develop more conservative predictions over the image reconstructions while the RSSM models and no-overshoot EKF developed overconfident estimates with very peaked histogram distributions over pixels values. This peakedness produced poor NLL performance, but since the peaks were often close enough to the true values, models like O-RSSM or L-RSSM performed well according to the L2 criterion, with O-RSSM outperforming O-DT-EKF.

In other words, ORO helps learn more realistic measures of uncertainty while the other methods prioritize computing accurate mean predictions. In a reinforcement learning environment with deterministic physics, this may be sufficient to achieve good performance with the RSSM models [5], [16]. We also found the L-DT-EKF performed as poorly as the N-DT-EKF. We hypothesize that since the latent space itself evolves during training, the targets used in LRO are not stable for dynamics learning. In contrast, the space of observations and the data remain static, which naturally encourages stable manifold learning with ORO. The EKF models achieved at least as good performance as the baselines with about 4.5 times fewer parameters. For a summary, see Table I.

	PEND (NLL)		PEND (L2 $\times$ 1e-2)		PEND	PUSH (NLL)		PUSH (L2 $\times$ 1e-2)		PUSH
	F5/P50	F25/P50	F5/P50	F25/P50	Model Size	F5/P25	F25/P25	F5/P25	F25/P25	Model Size
O-LE	788.6	762.0	5.303	4.663	**597**	2.400	4.017	23.55	27.05	39,342
N-RSSM	1158	1055	3.194	2.806	91,392	3.662	1.891	14.46	13.26	147,412
L-RSSM	1197	853.1	3.325	2.14	91,392	1.636	1.289	16.94	15.74	147,412
O-RSSM	1034	758.2	**2.981**	**1.997**	91,392	4.023	2.199	26.68	21.97	147,412
O-CT-EKF	832.9	769.2	3.362	2.272	21,726	-0.565	-0.832	18.49	17.25	**23,126**
N-DT-EKF	1180	1202	4.087	4.157	21,726	-2.261	**-2.482**	14.40	13.26	**23,126**
L-DT-EKF	1108	1102	4.592	4.553	21,726	-1.632	-2.226	12.28	10.70	**23,126**
O-DT-EKF	**781.7**	**744.6**	3.102	2.605	21,726	**-3.360**	-2.389	**8.810**	**10.172**	**23,126**

TABLE I: The full quantitative results for all models using NLL and L2 loss normalized by batch size and trajectory length (lower is better). F#/P# indicates the number of provided points to filter on and the number of subsequent points to predict. 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 all PEND models used the exact same image autoencoder architecture for equal comparison. We observe that our EKF model trained with ORO performs the best overall, especially when the number of filter points is low.

As in [5], we found that using standard overshooting did not guarantee good performance, and oftentimes seemed to yield worse models. Additionally, models trained with an overshoot of $k > 2$ generally experienced instability during training and did not satisfactorily converge, and if they did, significant amounts of hyperparameter tuning were required. We suspect that the exploding gradient problem motivating our ramped curriculum strongly influences this instability.

C. MIT Push Experiments

We use a whitened curated subset of the MIT Push Dataset from [30] consisting of around 1600 trajectories. In each trajectory, the object geometry varied as well as the material of the surface. We did not train any models to condition on shape or material type. We constructed 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. Again, we used $\alpha = 0.5$ for RO weighting and $k = 2$ for the standard overshoot horizon. The O-LE model used 100 linear models.

The extreme nonlinearity of the friction and contact dynamics is apparent upon analyzing the poor O-LE predictions, which supports our method of learning exact nonlinear models over approximate linear ones. Table I indicates that all EKF models trained using RO are more accurate by almost all measures and over 6 times more parameter-efficient than the baseline models. LRO also achieved worse NLL results than no overshooting, but better L2 results. This suggests that the LRO model also suffers from overconfident predictions. We note that the O-DT-EKF has *decreasing* performance with more filter points, which we suspect is also due to the aforementioned overconfidence effect induced by an abundance of observations.

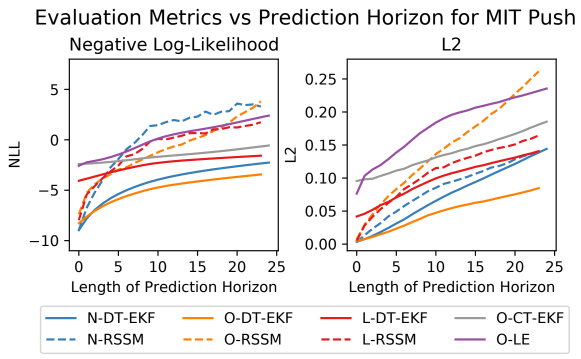

Fig. 5: The evaluated metrics plotted over the prediction horizon given 5 filter points (lower is better). The RO models (solid) are compared to the baselines (dashed). EKF and RSSM methods with the same type of overshooting (none, observation, or latent) are plotted in the same color. The EKF models scale better as the prediction horizon increases except for the O-LE model, which poorly approximates the nonlinear contact dynamics. The RSSM models tend to learn overconfident predictions, often scaling competitively with respect to L2 but performing relatively worse on NLL. This is usually a result of very peaked prediction distributions near the true value but with low variance, causing large decreases in likelihood.

Fig. 5 shows that as the prediction horizon increases, our EKF-based models retain much more accurate probability distributions than their RSSM counterparts. The excellent performance of O-DT-EKF across all prediction horizons suggests that our hypothesis that the EKF structure provides a strong training signal for learning a robust stochastic dynamics is correct. Meanwhile, once the RSSM models are applied to the real-world push data rather than synthetic data generated from simulation environments, the performance decreases substantially. Again, ORO produced significantly better performance than LRO, matching the pendulum results, suggesting that independent of the data, using the observations as targets for training is more effective than using the smoothed latent states.

V. Conclusion

This paper introduced replay overshooting, a method that capitalizes on the structure of the EKF to learn stochastic nonlinear latent dynamics models from any type of time-series data. The EKF enjoys a much simpler architecture than competing methods with improved gradient paths that strongly encourage learning good latent dynamics models and meaningful latent manifolds. We demonstrated that models trained using RO can accurately generate predictions over long horizons and outperform many other overshooting-based models both quantitatively and qualitatively. We also demonstrate that RO can learn competitive continuous-time dynamics models without modifying the architecture.

VI. Acknowledgments

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

[3] R. G. Krishnan, U. Shalit, and D. A. Sontag, "Structured inference networks for nonlinear state space models," *ArXiv*, vol. abs/1609.09869, 2016.

[4] M. Karl, M. Sölch, J. Bayer, and P. van der Smagt, "Deep variational bayes filters: Unsupervised learning of state space models from raw data," *ArXiv*, vol. abs/1605.06432, 2016.

[5] 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.

[6] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, 1960.

[7] E. A. Wan and A. T. Nelson, "Dual kalman filtering methods for nonlinear prediction, smoothing and estimation," in *NIPS*, 1996.

[8] T. Raiko and M. Tornio, "Variational bayesian learning of nonlinear hidden state-space models for model predictive control," *Neurocomputing*, vol. 72, pp. 3704–3712, 2009.

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

[10] 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.

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

[12] 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.

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

[14] 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.

[15] 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.

[16] 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.

[17] R. Jonschkowski, D. Rastogi, and O. Brock, "Differentiable particle filters: End-to-end learning with algorithmic priors," in *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018.

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

[19] 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.

[20] 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.

[21] 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.

[22] 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.

[23] S. Särkkä and J. Sarmavuori, "Gaussian filtering and smoothing for continuous-discrete dynamic systems," *Signal Process.*, vol. 93, pp. 500–510, 2013.

[24] T. Q. Chen, Y. Rubanova, J. Bettencourt, and D. Duvenaud, "Neural ordinary differential equations," *CoRR*, 2018.

[25] C. Yildiz, M. Heinonen, and H. Lahdesmaki, "Ode2vae: Deep generative second order odes with bayesian neural networks," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 13 412–13 421.

[26] E. De Brouwer, J. Simm, A. Arany, and Y. Moreau, "Gru-ode-bayes: Continuous modeling of sporadically-observed time series," in *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 7379–7390.

[27] 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.

[28] 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.

[29] 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.

[30] 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.

[31] M. Poli, S. Massaroli, A. Yamashita, H. Asama, and J. Park, "Hypersolvers: Toward fast continuous-depth models," *ArXiv*, vol. abs/2007.09601, 2020.

[32] C. Finlay, J. Jacobsen, L. Nurbekyan, and A. M. Oberman, "How to train your neural ODE: the world of Jacobian and Kinetic regularization," *CoRR*, vol. abs/2002.02798, 2020.

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