FORECASTING OBJECT PUSHING WITH THE META-EXTENDED KALMAN FILTER

Albert Li
Department of Mechanical Engineering
Stanford University
Stanford, CA 94305, USA
ahli@stanford.edu

Philipp Wu
Covariant
Emeryville, CA 94608, USA
philipp@covariant.ai

ABSTRACT

Since childhood, humans cultivate a strong intuition of physics such that after just short observation periods, we can relatively accurately predict future trajectories. Our goal is developing a new algorithm to reliably replicate this ability on robots. In this report, we present the *Meta-Extended Kalman Filter* (MEKF), a procedure based on classical state estimation theory that promises a computationally efficient method for posterior inference for stochastic nonlinear latent variable models. Using the MEKF, we can jointly recover dynamics and observation models, jointly optimizing them using a variational inference procedure.

The MEKF algorithm draws significant inspiration from the *Model Agnostic Meta-Learning* (MAML) method, which allows trains a model to maintain a set of adaptable parameters that can easily be fine-tuned to perform well on new tasks. In a similar vein, the MEKF seeks to learn adaptable priors over task-specific parameters such as object mass, geometry, contact parameters, and more such that after filtering on a given observation sequence, the model can recover a confident belief over the parameters relevant for prediction. While MAML backpropagates through gradient steps, the MEKF backpropagates through differentiable filtering, smoothing, and prediction operations.

In addition to the meta-training procedure and a new learning objective, we also introduce a parameterization of the conventional latent dynamics that treats the task-specific parameters of interest as latent states with static dynamics. This segregates the information pertaining to the learned universal dynamics model from the information governing the specific dynamics of the task, and we find that this inductive bias is key to ensuring consistent performance on out-of-distribution and unseen tasks during test time.

The MEKF is evaluated against a single-task EKF trained on the aggregated meta-dataset without the aforementioned dynamics reparameterization. We find that on both quantitative and qualitative metrics, the MEKF significantly outperforms the vanilla EKF. In particular, the MEKF is significantly more confident while being more accurate in its predictions due to the adaptation period and can also perform well even when given less data to filter on than during training, showing the robustness of the method.

Division of Responsibility: Albert was responsible for writing the entire report and also introducing the meta-learning formulation presented (all material relevant to the course). Philipp was responsible for managing the repository structure and the estimation API. We both worked significantly on implementing differentiable versions of the EKF, debugging, and designing experiments to evaluate the success of the EKF for general dynamics learning.

Access to the code is detailed in the appendix.

1 INTRODUCTION

Modern research in robotics seeks to bring the robot out of segregated environments like the factory to more intimate ones like the kitchen, home, or hospital. One consequence of this paradigm shift is that we expect robotic assistants to operate in close proximity to humans and in cluttered, unstructured environments inhabited by them. Thus, the safe deployment of robots depends on their ability to reason about purposeful or incidental contact, whether robot-human, robot-object, or object-object. Since objects differ unboundedly in geometry, mass, chemical makeup, etc., we would prefer to learn a general physics model and adapt it to specific cases rather than learning the physics of each particular scenario from scratch, which is impractical and intractable. Such a setting is appropriate for *meta-learning*, wherein we would like to train a model to perform well across a variety of *tasks* and quickly adapt to new ones with a minimal number of examples.

In this latter regard, humans excel, possessing a remarkable ability to make accurate long-horizon stochastic spatiotemporal predictions based on short observation periods. The natural tendency to develop this intuition throughout childhood 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), which suggests that humans maintain complex internal latent dynamics models to reason about motion (Ungerleider & Haxby, 1994). We are interested in developing an algorithm for robots that replicates this ability in order to facilitate effective decisionmaking in highly dynamical environments.

In this report, we study a structured sub-class of contact modeling and prediction problems: rigid objects being pushed on planar surfaces by a robotic manipulator. The objects can have varying geometry, mass distribution, contact properties, and more. Motivated by human ability, we investigate the one-shot setting, where the model first receives a sequence of observations used to deduce relevant physical properties of an object-surface pair. Then, we provide a second short sequence of observations on the same pair and query the model to predict future elements of the sequence up to some desired horizon.

To that end, we propose the *meta-extended Kalman filter* (MEKF), an algorithm based on classical state estimation theory that can quickly update internal belief distributions by consuming sequences of observations, which makes it well-suited for spatiotemporal prediction. To our knowledge, this is the first use of filtering-related techniques for meta-learning time-series prediction models.

2 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 (Wan & Nelson, 1996). Other works on state-space prediction have used alternative methods like dynamic factor analysis to estimate the posterior (Raiko & Tornio, 2009). 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) (Kingma & Welling, 2014) led to a resurgence of filter-based learning methods by allowing tractable sampling-based posterior inference. 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 (Krishnan et al., 2015; 2016). 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 (Karl et al., 2016). 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 (Fraccaro et al., 2017).

There also exist several meta-learning methods for dynamical prediction. In *modular meta-learning* (Alet et al., 2019), model adaptation occurs with a novel architecture search that combines relevant submodules together to recover new architectures well-suited for new tasks. *Attentive neural processes* (Kim et al., 2019) are a type of attention-based model that does few-shot learning on func-

tions, constructing a predictive distribution given some input/output pairs of said function. This procedure has been successfully used for dynamical meta-learning, in particular in the object-pushing domain (Bauza et al., 2019).

In the reinforcement learning literature, there exist a litany of model-based meta-learning methods, such as *experience-embedded visual foresight* (Yen-Chen et al., 2019), a method that uses a hierarchical Bayes model for few-shot video dynamics adaptation, or the meta-RL method presented by Nagabandi et al. (2019), wherein the model of interest learns to adapt to changing dynamics during online operation by using some sequence from the past to inform the model of the expected performance over some future horizon, and subsequently, how to adapt. In the latter method, planning and execution are governed by a model predictive control scheme. Finally, the general *model-agnostic meta-learning* method (MAML) was derived by Finn et al. (2017), which can be applied to any general gradient-based learning method for model learning.

Despite the litany of successful filter-based learning models and recent works on dynamical meta-learning, there do not exist any works that combine the approaches together. The focus of this work is therefore the intersection between filters and meta-learning, and how benefits of both approaches can be harnessed by the meta-extended Kalman filter.

3 PRELIMINARIES

3.1 THE NEURAL EXTENDED KALMAN FILTER

The information found in this section is discussed in detail by Särkkä (2013), including detailed derivations of the update equations and extensions of Kalman filtering methods. Consider the following 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 $p(z_t \mid y_{1:t}, u_{1:t-1})$. If this distribution can also be conditioned on future information, then it is called the smoothed posterior $p(z_t \mid y_{1:T}, u_{1:T})$, $T > t$. We refer to the filtering and smoothing processes as *posterior inference*.

To recover the filtered posterior, we first assume knowledge of another 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* incorporates the most recent measurement, yielding

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

and given a stochastic dynamics model $p(z_{t+1} \mid z_t, u_t)$, the *prediction update* forecasts the distribution at the next time step, yielding

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

which becomes the prior for the next measurement update. The process then repeats indefinitely, continuously consuming incoming observations.

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

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

While the above equations imply that we can always compute one-step dynamics distributions given the current state, the true strength of the Kalman filter is that it can compute the dynamics of the

distribution parameters themselves *without ever needing intermittent sampling*. In the special case of the linear Gaussian state space model, the Kalman filter recovers an analytical solution to the *Fokker-Planck equation*, also known as the *Kolmogorov forward equation*, a general partial differential equation describing the time evolution of a probability density function (Särkkä, 2013). The prediction and measurement update equations can be written

$$\mu^z_{t|t-1} = A_t \mu^z_{t-1|t-1} + B_t u_t, \tag{6}$$

$$\Sigma^z_{t|t-1} = A_t \Sigma_{t-1|t-1} A_t^\top + Q_t, \tag{7}$$

$$\mu^z_{t|t} = \mu^z_{t|t-1} + K_t(y_t - C_t \mu^z_{t|t-1}), \tag{8}$$

$$\Sigma^z_{t|t} = \Sigma^z_{t|t-1} - K_t C_t \Sigma_{t|t-1}, \tag{9}$$

$$K_t = \Sigma_{t|t-1} C_t^\top (C_t \Sigma_{t|t-1} C_t + R_t)^{-1}, \tag{10}$$

where the notation $(\cdot)_{t_a|t_b}$ indicates a distribution parameter at time t_a conditioned on observations up to time t_b and K_t is called the *Kalman gain*.

We can also directly recover a distribution over the observations given the current latent distribution:

$$p(y_t \mid \mu^z_t, \Sigma^z_t) = \mathcal{N}(y_t; C_t \mu^z_t, C_t \Sigma^z_t C_t^\top + R_t). \tag{11}$$

Crucially, this allows us to compute the likelihood of an observation without sampling-based techniques like the reparameterization trick (Särkkä, 2013; Kingma & Welling, 2014), instead reasoning entirely in terms of probability densities.

To compute the smoothed distributions, we use the iterative *Rauch-Tung-Striebel smoother*, an algorithm that first performs filtering over the entire observation sequence, caching particular values over the forward pass. Then, a backwards pass is conducted to adjust the filtered posteriors to be conditioned on future observations:

$$\begin{aligned} \mu_{t|T} &= \mu_{t|t} + K^s_t(\mu_{t+1|T} - \mu_{t+1|t}), \\ \Sigma_{t|T} &= \Sigma_{t|t} + K^s_t(\Sigma_{t+1|T} - \Sigma_{t+1|t})(K^s_t)^\top, \\ K^s_t &= \Sigma_{t|t} A_t^\top \Sigma^{-1}_{t+1|t}. \end{aligned} \tag{12}$$

Now, consider once again the general nonlinear system (1). The *extended Kalman filter* (EKF) uses slightly modified update equations to perform *approximate* posterior inference, estimating all distributions as Gaussians. With some abuse of notation, let

$$A_t = \left.\frac{\partial f}{\partial z_t}\right|_{z_t = \mu^z_{t-1|t-1}}, \quad C_t = \left.\frac{\partial g}{\partial z_t}\right|_{z_t = \mu^z_{t|t-1}} \tag{13}$$

in the nonlinear case. Then, the update equations for the covariances and Kalman gain take the same form, and we use new update equations for the distribution means:

$$\mu^z_{t|t-1} = f(t, \mu^z_{t-1|t-1}, u_{t-1}), \tag{14}$$

$$\mu^z_{t|t} = \mu^z_{t|t-1} + K_t(y_t - g(\mu^z_{t|t-1})). \tag{15}$$

In general, the convergence of the EKF is not guaranteed. We require that the prior $p(z_1)$ relatively accurately describes the true latent initial condition, the noises in the system are not too large, and that error covariances remain positive-definite and bounded during operation (this last condition is often violated by numerical error, and many stable filters have been developed in response) (Reif et al., 1999). While we cannot control the noise characteristics of the dynamics or observation model, we have full control over the choice of prior. Learning a sensible choice is central to the meta-EKF formulation presented in the sequel.

Finally, we note that posterior inference with the EKF can be conducted exclusively with the dynamics and observation models f and g. We choose to parameterize these models as neural networks f_θ and g_ϕ, which are typically shallow multi-layer perceptrons with a modest number of hidden units. Additionally, we can also directly learn the prior $p(z_1)$ as well as the dynamics and observation covavariances Q, R, which we define collectively as $\beta := \{p(z_1), Q, R\}$. We refer to the tuple $(f_\theta, g_\phi, \beta)$ as the neural EKF.

In contrast, the dominant paradigm in the learning literature is the use of a third *inference network*, which we denote $q_\psi(z_t \mid y_{1:T})$ (Krishnan et al., 2015; Watter et al., 2015; Krishnan et al., 2016; Karl et al., 2016; Chiappa et al., 2017; Villegas et al., 2017; Hafner et al., 2019). In these methods, the inference model is typically hand-designed for some specific application and, crucially, separates the inference procedure from the dynamics and observation model. This choice increases architectural complexity, reduces parameter efficiency, and weakens the training signal afforded to f_θ and g_ϕ during training. The neural EKF simply outsources the computations performed by the inference network to the Kalman update equations.

3.2 A Factorized Variational Lower Bound for Filter Optimization

To train the neural EKF on sequences of a single task, we turn to variational inference. Consider the problem of maximizing the likelihood of an observation sequence conditioned on a control sequence, $p(y_{1:T} \mid u_{1:T})$. We can derive a variational lower bound for this objective:

$$\begin{aligned}
\log p(y_{1:T} \mid u_{1:T}) &= \log \int p(y_{1:T} \mid z_{1:T}, u_{1:T}) p(z_{1:T} \mid u_{1:T}) dz_{1:T} \\
&= \log \int p(y_{1:T} \mid z_{1:T}, u_{1:T}) p(z_{1:T} \mid u_{1:T}) \frac{q(z_{1:T} \mid y_{1:T}, u_{1:T})}{q(z_{1:T} \mid y_{1:T}, u_{1:T})} dz_{1:T} \\
&\geq \mathbb{E}_{q(z_{1:T} \mid y_{1:T}, u_{1:T})} \left[\log p(y_{1:T} \mid z_{1:T}, u_{1:T}) \right] \\
&\quad - D_{KL}(q(z_{1:T} \mid y_{1:T}, u_{1:T}) \mid\mid p(z_{1:T} \mid u_{1:T})).
\end{aligned} \tag{16}$$

Due to the iterative nature of the Kalman filter, we would prefer to compute the bound in terms of factorized distributions which are output by the filter one at a time rather than joint distributions, which are harder to construct. Such a factorized bound was derived by Krishnan et al. (2016):

$$\begin{aligned}
\log p(y_{1:T} \mid u_{1:T}) &\geq \sum_{t=1}^{T} \left(\mathbb{E}_{q(z_t \mid y_{1:T}, u_{1:T})} \left[\log p(y_t \mid z_t) \right] \right) - D_{KL}(q(z_1 \mid y_{1:T}, u_{1:T}) \mid\mid p(z_1)) \\
&\quad - \sum_{t=2}^{T} \mathbb{E}_{q(z_{t-1} \mid y_{1:T}, u_{1:T})} \left[D_{KL}(q(z_t \mid z_{t-1}, y_{1:T}, u_{t-1}) \mid\mid p(z_t \mid z_{t-1}, u_{t-1})) \right] \\
&= \mathcal{L}_r(y_{1:T}, \{\mu_i^y, \Sigma_i^y\}_{1:T}^s, \beta) + \mathcal{L}_{KL}(y_{1:T}, \{\mu_i^z, \Sigma_i^z\}_{1:T}^s, \{\mu_i^z, \Sigma_i^z\}_{1:T}^p, \beta).
\end{aligned} \tag{17}$$

We can interpret this bound in terms of reconstruction and regularization as in standard variational inference, where we define the expectations over the observation distributions as $\mathcal{L}_r$ and the KL terms as $\mathcal{L}_{KL}$. Each term is a function of the observation sequence, but also sets of smoothed distributions and predicted distributions, denoted by sets with superscripts s or p respectively. The reason we use the smoothed distributions for reconstruction is because conditioning on future information should allow for the best possible estimate of the latent state (compared to just filtering). This not only improves the reconstruction accuracy, it also forces the best possible inferred states to be regularized by the standalone dynamics.

Typically, this objective would be maximized using stochastic optimization and the reparameterization trick, jointly learning f_θ, g_ϕ, and β. As noted in the previous section, with the EKF we can compute distributions over observations without sampling-based approximations using equation (11). Additionally, we can also directly compute the distributions in the KL terms without intermittent sampling by using the Kalman update equations. We hypothesize that this reduces the variances of parameter updates during the learning procedure, but have not derived any formal results comparing this method to the reparameterization trick.

To use the resulting dynamics model, we consider the test-time setting described in Section 1 where we receive a short sequence of data $y_{1:T'}$ and are asked to predict future elements in the sequence. First, we filter on the data to recover a posterior distribution over the final latent state $z_{T'}$. Then, given a desired prediction horizon H, we deploy the Kalman prediction update equations without the intermittent measurement updates to propagate a predicted belief over the states $z_{T':T'+H}$. Finally, we can convert these beliefs into predicted distributions over observations $y_{T':T'+H}$ using equation (11). At this point, we can sample any predicted observation along the trajectory or evaluate the model using metrics like negative log-likelihood (NLL).

4 THE META-EKF

4.1 META-LEARNING PROBLEM SETUP

We would now like to adapt the single-task EKF algorithm to the meta-learning setting. First, we must formalize the problem and introduce the necessary vocabulary for the resulting discussion.

We define each task $\mathcal{T}$ as a tuple $(\mathcal{L}, H, \bar{f}(t, z_t, u_t), \bar{g}(z_t))$, where $\mathcal{L}$ is a loss function over an observation trajectory $y_{1:T}$, H is the desired prediction horizon, $\bar{f}$ is the true latent dynamics governing the evolution of a particular object-surface pair under robot control inputs u_t, and $\bar{g}$ is the observation model relating latent states to observations. Given a particular type of data (e.g. video frames or positional motion capture data), the loss function will be the same across all tasks, and we also choose to fix H during evaluation. Therefore, the tasks we consider are distinguished entirely by the true dynamics and observation models induced by varying object-surface pairs. It is precisely these models we seek to learn.

We would like the meta-learner to be able to quickly adapt to some task $\mathcal{T}_i$ drawn from task distribution $p(\mathcal{T})$ after observing on a single trajectory of length T generated from $\mathcal{T}_i$. We call this the *adaptive pass*. The model is tested by predicting H future elements of a partial sequence of observations of length T', and we update the learned models f_θ and g_ϕ based on the test error. We call this second stage the *evaluative pass*. The same procedure is repeated during meta-test time on a held out set of trajectories from unseen and/or out-of-distribution tasks.

4.2 A DYNAMICS MODEL FOR INFORMATION SEGREGATION

There are three types of information in which we are interested: (a) universal dynamical relations governed by the laws of physics (e.g. the general relation between force and position); (b) task-specific parameters that characterize the unique physical interactions of a particular object-surface pair (e.g. object mass, contact parameters, etc.); and (c) transient states that evolve according to an unknown latent dynamics model (e.g. object position, velocity, etc.).

In the canonical single-task model-learning setting, there exists no separation between information types (a) and (b), which are captured by the parameters θ in our model f_θ. Information of type (c) is captured by the latent state z_t. In the meta-learning setting, we would like to encourage these information types to be segregated during learning such that the task-specific parameters can change upon observing some exemplary sequence while leaving the more general physics model intact.

One way to do this is to view the task-specific parameters as states which do not evolve over time. To enforce this, we can explicitly divide the state space into task-specific parameters z^{task} and transient states z_t^{tran}. Then, we explicitly define the dynamics for the task-specific parameters as static and parameterize the state dynamics with a neural network:

$$f_\theta(t, z_t, u_t) = \begin{bmatrix} z^{task} \\ NN_\theta(t, z_t, u_t) \end{bmatrix}, \quad z_t = \begin{bmatrix} z^{task} \\ z_t^{tran} \end{bmatrix}. \tag{18}$$

This small modification allows us to use state estimation techniques like the EKF to do task adaptation by filtering on observation sequences to infer the task-specific parameters while allowing the gradient updates to learn the generalized pushing model. We note that treating unknown parameters as static states is a relatively common technique, leveraged in methods like EKF simultaneous localization and mapping (Durrant-Whyte & Bailey, 2006).

4.3 META-LEARNING WITH THE EKF

The MEKF algorithm draws inspiration from MAML. While MAML seeks to learn pre-adaptation model parameters θ that, after fine-tuning, can perform well on some new task, the MEKF seeks to also learn a strong prior $p(z_1)$ such that the adaptive pass can recover a confident belief distribution over the task-specific parameters and subsequently maximize the accuracy of predictions during the evaluative pass. MAML backpropagates through gradient descent operations and the MEKF analogously backpropagates through Kalman filter updates.

Algorithm 1: Meta-EKF Algorithm

Require: Task distribution $p(\mathcal{T})$
Require: Learning rate γ
Initialize learnable models/parameters f_θ, g_ϕ, $\beta := \{p(z_1), Q, R\}$
while *not converged* **do**
  Sample a batch of tasks $\mathcal{T}_i \sim p(\mathcal{T})$
  for *all* $\mathcal{T}_i$ **do**
    Sample trajectories $y^{adapt}_{1:T}$, $y^{eval}_{1:T'+H}$ from $\mathcal{T}_i$
    Execute adaptive pass with the Kalman filter:
    $\mu^z_{1:T}, \Sigma^z_{1:T} = FILTER(y^{adapt}_{1:T}; f_\theta, g_\phi, p(z_1), Q, R)$
    Construct the evaluative prior $\tilde{p}(z_1)$ as in equations (19) and (20)
    Execute smoothing stage of the evaluative pass with the Kalman smoother:
    $\tilde{\mu}^z_{1:T'}, \tilde{\Sigma}^z_{1:T'} = SMOOTH(y^{eval}_{1:T'}; f_\theta, g_\phi, \tilde{p}(z_1), Q, R)$
    Execute predictive stage of the evaluative pass with the Kalman prediction equations:
    $\tilde{\mu}^z_{T':T'+H}, \tilde{\Sigma}^z_{T':T'+H} = PREDICT(y^{eval}_{T':T'+H}; f_\theta, g_\phi, \tilde{p}(z_{T'} \mid y^{eval}_{1:T'}), Q, R)$
  Update learnable parameters using objective from equation (21) and any optimization algorithm (vanilla gradient ascent shown here):
  $\theta \leftarrow \theta + \gamma \nabla_\theta \sum_i \mathcal{L}^{MEKF}_{\mathcal{T}_i}(f_\theta, g_\phi, \beta)$
  $\phi \leftarrow \phi + \gamma \nabla_\phi \sum_i \mathcal{L}^{MEKF}_{\mathcal{T}_i}(f_\theta, g_\phi, \beta)$
  $\beta \leftarrow \beta + \gamma \nabla_\beta \sum_i \mathcal{L}^{MEKF}_{\mathcal{T}_i}(f_\theta, g_\phi, \beta)$

After the adaptive pass is complete, the filter will output some distribution over the latent states

$$\mu^z_T = \begin{bmatrix} \mu^{task}_T \\ \mu^{tran}_T \end{bmatrix}, \quad \Sigma^z_T = \begin{bmatrix} \Sigma^{task}_T & (\cdot) \\ (\cdot) & \Sigma^{tran}_T \end{bmatrix}. \tag{19}$$

We assume that the covariance between task-specific parameters and initial transient states is 0, since, for example, the mass should not have any effect on the object's initial position. Therefore, we can extract the distribution parameters corresponding to the task-specific parameters and re-initialize the prior during the evaluative pass as

$$\begin{aligned} \tilde{p}(z_1) &= \mathcal{N}(z_1; \tilde{\mu}^z_1, \tilde{\Sigma}^z_1), \\ \tilde{\mu}^z_1 &= \begin{bmatrix} \mu^{task}_T \\ \mu^{tran}_1 \end{bmatrix}, \\ \tilde{\Sigma}^z_1 &= \begin{bmatrix} \Sigma^{task}_T & 0 \\ 0 & \Sigma^{tran}_1 \end{bmatrix}, \end{aligned} \tag{20}$$

where the tilde denotes the initial distribution for the evaluative pass. We can reuse the learned prior for the transient states, since the initial transient states for one trajectory cannot be inferred by looking at another, assuming the trajectories are sampled independently from each other.

The "outer loop" of the MEKF procedure remains similar to that of MAML in that the universal model parameters θ and ϕ are still updated with any gradient-based optimization procedure like Adam. The main distinction is therefore in the "inner loop," where the adaptation occurs.

One advantage of this scheme is that the universal dynamics model represented by f_θ cannot be destabilized during the adaptive pass due to the information segregation enforced by the structure of the dynamics, whereas in MAML, a poor choice of inner learning rate can unlearn the entire model. However, while the adaptive pass only needs to infer a small subset of the dynamics parameters, the operations executed during filtering are more expensive than a simple gradient step in MAML.

We now formally introduce the meta-learning training objective. As described, the evaluative pass is comprised of two distinct stages: the filtering phase and the the prediction phase. Consequently,

the meta-learning objective can be expressed in a way similar to equation (17):

$$\begin{aligned}\max_{\theta,\phi,\beta} \Bigg\{ & \mathcal{L}_r(y^{eval}_{1:T'}, \{\tilde{\mu}^y_i, \tilde{\Sigma}^y_i\}^s_{1:T'}, \beta) + \mathcal{L}_{KL}(y^{eval}_{1:T'}, \{\tilde{\mu}^z_i, \tilde{\Sigma}^z_i\}^s_{1:T'}, \{\tilde{\mu}^z_i, \tilde{\Sigma}^z_i\}^p_{1:T'}, \beta) \\ & + \mathcal{L}_r(y^{eval}_{T':T'+H}, \{\tilde{\mu}^y_i, \tilde{\Sigma}^y_i\}^p_{T':T'+H}, \beta) \Bigg\} \\ = \max_{\theta,\phi,\beta} & \mathcal{L}^{MEKF}_{\mathcal{T}}(f_\theta, g_\phi, \beta).\end{aligned} \tag{21}$$

The first two terms of this objective are the same as in equation (17) but only over the filtering phase. The third term is a reconstruction term over the predicted distributions computed during the prediction phase. The algorithm is summarized in Algorithm 1.

5 Experiments

5.1 Setup

We conducted experiments using a curated (Kloss et al., 2020) subset of the *MIT Push* dataset (Yu et al., 2016), which consisted of a few thousand trajectories of 12 different object-surface pairs. We held out one surface type for evaluation and used the remaining trajectories for training.

The data are represented by the planar position and rotation (x, y, θ) of the object, measured using Vicon motion capture markers. The control inputs are the commanded planar position and velocity of the manipulator as well as a contact boolean indicating whether the object would be touched.

Since the EKF is naturally adaptive any time the filter is in operation, we evaluated the MEKF versus a regular EKF that was only trained on the aggregated training dataset and without the modification to the dynamics model presented in Section 4.2. The main question we sought to answer was: how much better was the adaptive ability of the meta-model when allowed to operate in a one-shot setting rather than an aggregated single-task ("zero-shot") setting?

5.2 Hyperparameters and Implementation Details

For both methods, the dynamics and observation models were parameterized as three-layer MLPs with 64 hidden units per layer and softplus nonlinearities. In practice, we found that adding a skip connection between the first and last layer of the dynamics model such that we instead learned the dynamics $z_{t+1} = z_t + f_\theta(t, z_t, u_t)$ was helpful for encouraging learning, as in ResNets (He et al., 2016).

The dimensions of the transient latent state and task-specific parameters were 8 and 10 respectively, chosen after some mild hyperparameter tuning. The optimization algorithm used was Adam (Kingma & Ba, 2015) with an initial learning rate of 5e-3, an exponential learning rate decay of 0.9 every 100 iterations, and a minimum learning rate of 1e-6. The gradients were clipped at a maximum value of 100.

Each trajectory was 50 steps long, and we chose $T' = H$ such that the filtering period was the same length as the prediction period during the evaluative pass. Additionally, we found that instituting a learning curriculum where the the length of the training trajectories were slowly increased was helpful to stabilize learning, since before the learned models are near convergence, the numerical stability of the EKF may be suspect over long horizons. Therefore, first learning on shorter trajectories typically facilitated more consistent learning over long trajectories.

5.3 Results

We evaluated our models using two numerical metrics: the negative log-likelihood (NLL) of only the predicted portion of the held out evaluative trajectories as well as the average displacement error (ADE) of samples of the trajectories.

We found that the MEKF significantly outperformed the EKF on both metrics, which was expected. The EKF may in theory be able to encode cross-task information in the latent states, but the inductive bias of the MEKF's dynamics model should be much better at encouraging the latent space to encode

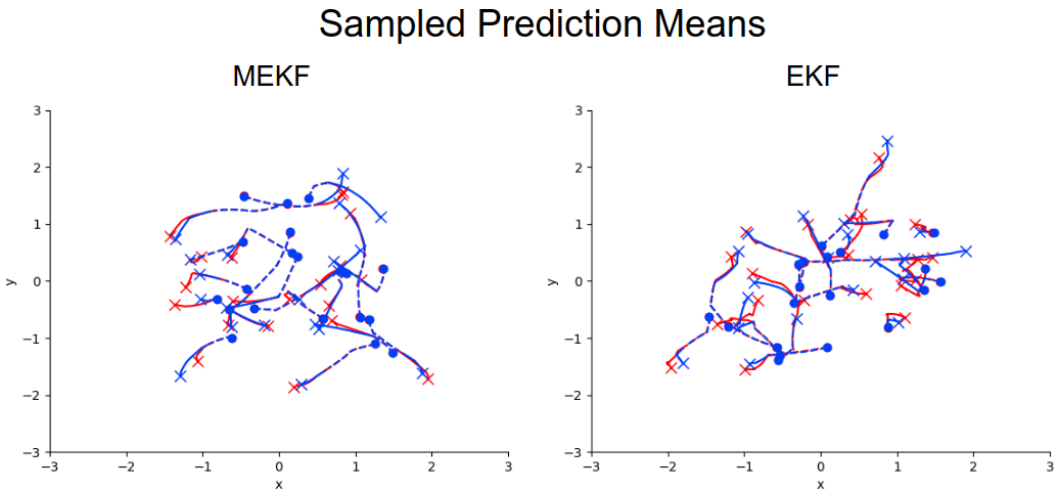

Figure 1: Randomly chosen prediction means for the meta-EKF (left) and the vanilla EKF (right). Dashed lines represent the filtering period and solid lines the prediction period. Blue lines are the ground truth data and red lines the model outputs. We observe that the output means are similar in quality, though the quantitative performance of the MEKF is significantly better than the vanilla EKF. This suggests that our adaptation procedure recovers much better confidence over the predictions and slightly better accuracy.

task-specific information. Additionally, the adaptive pass of the MEKF allows more computations to be executed for the purpose of adaptation compared with the EKF.

Additionally, we not only tested both models on their ability to filter on 25 points and predict on 25 points, we also evaluated their ability to adapt when only allowed to filter on 5 points to predict 25. In this secondary test, we also found that the MEKF yielded superior performance, which suggests that the procedure for learning the task-specific parameters prior to the evaluative pass recovers crucial information. The numerical results are summarized in Table 1.

Qualitatively, we observe that the prediction means of both methods are actually quite similar. Figure 1 shows randomly sampled true planar positions (blue) of pushed objects versus the smoothed (dashed) and predicted (solid) model outputs (red). The means are typically aware of important outcomes like direction changes, sudden stoppages, or cessation of applied force. This suggests that one of the primary benefits of the new adaptation scheme presented in this report is increasing the confidence of the prediction, which is directly related to better initial estimates of task-specific parameters. The results in Table 1 are consistent with this viewpoint.

6 Conclusion and Future Work

This report introduced the Meta-Extended Kalman Filter, a novel algorithm based on classical state estimation theory that can be effectively applied to learn deep stochastic latent dynamics models, and in particular, to adapt to new tasks, both in or out of the training distribution. We evaluated the model on object pushing tasks using real motion capture data from a robot manipulator and a collection of objects with varying mass and geometry on different surfaces.

The MEKF leverages the Kalman update equations to execute parameter-efficient posterior inference, which improves the training signal afforded to learning the dynamics model and significantly decreases the complexity of the model architecture. Additionally, the structure of the algorithm allows us to directly learn a malleable prior over both the task-specific parameters and transient states such that after observing just a single trajectory drawn from an unseen task, it can effectively predict a second trajectory. We observe significantly improved performance of our adaptive model over one which tries to learn an aggregated single-task pushing model, both in quantitative and qualitative metrics.

	NLL (F25/P25)	NLL (F5/P25)	ADE (F25/P25)	ADE (F5/P25)
EKF	-1.541	-1.281	0.199	0.211
MEKF (ours)	-3.716	-3.322	0.139	0.133

Table 1: Numerical results comparing the Meta-EKF to the single-task vanilla EKF (lower is better). F#/P# indicates the number of provided filter points and queried prediction points on evaluative trajectories. We observe superior performance from the meta-model on all metrics and all tasks in the held-out test set, which suggests that the adaptation period for recovering accurate beliefs over task-specific parameters is crucial. The reported values are normalized by both trajectory length and batch size.

Finally, due to time constraints, we were unable to evaluate our model against other time-series prediction meta-learning models. In the future, we would like to evaluate against a larger set of baselines and on more tasks, as well as using more comprehensive meta-learning datasets such as the Omnipush dataset (Bauza et al., 2019). Additionally, we may explore other filter-based techniques like the unscented Kalman filter or the particle filter, which may have other desirable properties for filter-based dynamical prediction.

REFERENCES

Ferran Alet, Tomás Lozano-Pérez, and Leslie P. Kaelbling. Modular meta-learning, 2019.

Maria Bauza, Ferran Alet, Yen-Chen Lin, Tomas Lozano-Perez, Leslie P. Kaelbling, Phillip Isola, and Alberto Rodriguez. Omnipush: accurate, diverse, real-world dataset of pushing dynamics with rgb-d video, 2019.

Silvia Chiappa, Sébastien Racanière, Daan Wierstra, and Shakir Mohamed. Recurrent environment simulators. In *Proceedings of the 2014 International Conference on Learning Representations*, 04 2017.

H. Durrant-Whyte and T. Bailey. Simultaneous localization and mapping: part i. *IEEE Robotics Automation Magazine*, 13(2):99–110, 2006. doi: 10.1109/MRA.2006.1638022.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks, 2017.

Marco Fraccaro, Simon Kamronn, Ulrich Paquet, and Ole Winther. A disentangled recognition and nonlinear dynamics model for unsupervised learning. In *Advances in Neural Information Processing Systems 30*, pp. 3601–3610. Curran Associates, Inc., 2017.

Danijar Hafner, T. Lillicrap, Ian S. Fischer, R. Villegas, David R Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *ICML*, 2019.

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)*, pp. 770–778, 6 2016. doi: 10.1109/CVPR.2016.90.

Maximilian Karl, Maximilian Sölch, Justin Bayer, and Patrick van der Smagt. Deep variational bayes filters: Unsupervised learning of state space models from raw data. *ArXiv*, abs/1605.06432, 2016.

Hyunjik Kim, A. Mnih, Jonathan Schwarz, Marta Garnelo, S. Eslami, Dan Rosenbaum, Oriol Vinyals, and Y. Teh. Attentive neural processes. *ArXiv*, abs/1901.05761, 2019.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015.

Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. *2nd International Conference on Learning Representations, ICLR 2014*, 2014.

Alina Kloss, Stefan Schaal, and Jeannette Bohg. Combining learned and analytical models for predicting action effects from sensory data. *The International Journal of Robotics Research*, 2020.

Rahul G. Krishnan, Uri Shalit, and David Sontag. Deep kalman filters, 2015.

Rahul G. Krishnan, Uri Shalit, and David A Sontag. Structured inference networks for nonlinear state space models. *ArXiv*, abs/1609.09869, 2016.

Anusha Nagabandi, Ignasi Clavera, Simin Liu, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning, 2019.

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

K. Reif, S. Gunther, E. Yaz, and R. Unbehauen. Stochastic stability of the discrete-time extended kalman filter. *IEEE Transactions on Automatic Control*, 44(4):714–728, 1999. doi: 10.1109/9.754809.

Simo Särkkä. *Bayesian Filtering and Smoothing*. Cambridge University Press, USA, 2013. ISBN 1107619289.

Leslie G. Ungerleider and James V. Haxby. 'What' and 'where' in the human brain. *Current Opinion in Neurobiology*, 4:157–165, 1994.

R. Villegas, Jimei Yang, Y. Zou, Sungryull Sohn, Xunyu Lin, and H. Lee. Learning to generate long-term future via hierarchical prediction. In *ICML*, 2017.

Eric A. Wan and Alex T. Nelson. Dual kalman filtering methods for nonlinear prediction, smoothing and estimation. In *NIPS*, 1996.

Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Advances in Neural Information Processing Systems 28*, pp. 2746–2754. Curran Associates, Inc., 2015.

Lin Yen-Chen, Maria Bauza, and Phillip Isola. Experience-embedded visual foresight, 2019.

Kuan-Ting 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.

APPENDIX

The stripped code can be accessed at
`https://drive.google.com/file/d/1GfK5YaqcHCGK00ipik-MMQlrpk8giURD/view?usp=sharing`.
Note that a large portion of the codebase not relevant to this project has been removed, so some infrastructure in the file organization may seem unnecessary. The main part of the code is in `dynamics_learning/networks`. This contains the top-level parent classes for the estimation API as well as the relevant child classes. In the subdirectory `kalman` are the default implementations of the neural EKF while the subdirectory `metalearning_models` contains the modifications for the meta-EKF. Training and evaluation scripts are located in the top-level `scripts` folder. Note that both training and evaluation are stochastic, so results may not exactly match up with the reported values even though the exact checkpoint is provided for evaluation.