# Efficient transfer learning and online adaptation with latent variable models for continuous control

**Christian F. Perez**
Uber AI Labs
San Francisco, CA 94105
cfp@uber.com

**Felipe Petroski Such**
Uber AI Labs
San Francisco, CA 94105
felipe.such@uber.com

**Theofanis Karaletsos**
Uber AI Labs
San Francisco, CA 94105
theofanis@uber.com

## Abstract

Traditional model-based RL relies on hand-specified or learned models of transition dynamics of the environment. These methods are sample efficient but exhibit performance losses under model mis-specification and ill model fitting resulting in differences between modeled and true dynamics. In addition, agents typically have to collect significant buffers of data to update their beliefs about the system dynamics. We propose using variational inference to learn an explicit latent representation of hidden dynamics that accelerates learning and facilitates generalization on novel environments at test time by allowing the model to align to current conditions. We combine this approach with Online Bayesian Inference to rapidly adapt to changes in environments as they happen.We model uncertainty by using neural network ensembles to parametrize environment dynamics separately from unknown environment properties and demonstrate our model on a continuous control task in the Mujoco domain.

## 1 Introduction

The ideal reinforcement learning algorithm learns efficiently with little data, generalizes well to new environments, and readily adapts to changing conditions. Model-free methods have achieved impressive results, but may require millions of observations during training [13]. Model-based methods are sample efficient, but often perform worse than model-free policies [5, 6, 8]. Both methods can suffer from over-fitting to training conditions, yielding agents that perform poorly when test conditions differ.

We explore learning across environments in which transition dynamics can be expected to vary in systematic ways. The rules of physics do not change, but unknown physical parameters (e.g., friction, mass, motor actuator gain) can easily vary. Hierarchical probabilistic models facilitate incorporation of prior knowledge about such information and can help transfer knowledge between tasks (for an early example see [9] and more recently [11, 7, 10].) We propose hierarchical probabilistic model-based control with dynamics models that (1) use auxiliary latent variables to learn more efficiently by transferring learned dynamics across environments, and (2) generalize to novel environments through inference of the latent variables. The key characteristic of our model is an explict shared dynamics factor over all instances (the network) and private latent variables which model instance-specific variability. Dynamics can change over time on a real system. We thus employ online Bayesian inference to equip the agent with the ability to infer changing environments on the fly. The resulting

Continual Learning Workshop at NIPS 2018, Montréal, Canada.

high-performing adaptive agents learn efficiently within tens of episodes across related continuous control environments and generalize to novel and changing dynamics.

In contrast to previous work on latent variable MDPs with small discrete action spaces [7, 11, 15], our controllers achieve high reward on challenging robotic environments with continuous control tasks and rely on models in contrast to [10]. As a pragmatic alternative to Bayesian neural networks, *deep ensembles* [12] are a promising approach that can address *model bias* and uncertainty quantification in model-based methods, which can close the gap with model-free performance [2, 8, 14]. Furthermore, we leverage variational inference to quickly learn the hidden latent environment variables within a single episode. This allows test-time adaptation to dynamically changing conditions, similar to [3].

## 2 Dynamics models with auxiliary latent variables

For environments with different dynamics, we define $\mathbf{e}_k \in \mathbb{R}^{d_e}$ as a latent variable representation of the relevant degrees of freedom in the dynamics. In general, we do not have access to the parameters of the transition function, and so treat them as hidden latent variables. We consider $p(\mathbf{e}_k) = \mathcal{N}(0, I)$ and initial state distribution $p(\mathbf{s}_0)$ given by the environment.

For an agent acting in an unknown/new environment (specifically with an unknown transition function), a robust dynamics model explicitly accounts for beliefs about the environment, actor-properties and environment-conditional dynamics and can marginalize oover them:

$$p(\mathbf{s}_{0:T+1}, \mathbf{a}_{0:T}, \mathbf{e}_k) = p(\mathbf{e}_k)p(\mathbf{s}_0)\prod_{t=0}^{T} p(\mathbf{s}_{t+1}|\mathbf{s}_t,\ \mathbf{a}_t, \mathbf{e}_k)p(\mathbf{a}_t|\mathbf{s}_t, \mathbf{e}_k) \tag{1}$$

Recent work has shown neural network ensembles capture uncertainty and avoid overfitting in supervised [12] and reinforcement learning problems [2]. Given their straightforward implementation and reliable training protocols, we model $p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_k)$ as an ensemble of probabilistic networks incorporating variational inference as an alternative to a fully Bayesian neural network (see Sec. A.1).

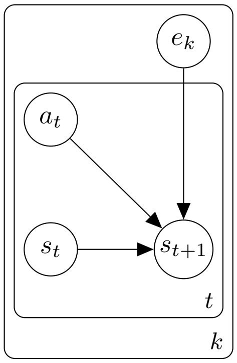


Figure 1: Graphical model of multiple environment transition dynamics probability model.

As the agent acts, new transitions $\mathcal{D}^*$ are collected and added to the dataset $\mathcal{D} = \mathcal{D} \cup \mathcal{D}^*$. For each transition in a new environment, we update the posterior over $\mathbf{e}_k$ via Bayes' rule,

$$p(e_k|\mathcal{D}) \quad = \quad \frac{p(\mathcal{D}|e_k)p(e_k)}{p(\mathcal{D})} = \frac{p(\mathcal{D}|e_k)p(e_k)}{\int p(\mathcal{D}|e_k)p(e_k)de_k} \tag{2}$$

In general, the marginal data likelihood $p(\mathcal{D})$ involves an intractable integral, but the posterior can be approximated by stochastic variational methods (**SVI**). Utilizing **SVI** in Eq. 1 we arrive at the posterior dynamics model given the observed data.

To avoid retaining a large buffer of past experience $\mathcal{D}$, we utilize Online Bayesian Inference: We posit the posterior inference problem in sequential fashion for datasets $\mathcal{D}_t$ arriving successively, where the posterior at time $t$ becomes the prior for time $t + 1$:

$$p(e_k|\mathcal{D}_{t+1}) = \frac{p(\mathcal{D}_{t+1}|e_k)p(e_k|\mathcal{D}_t)}{p(\mathcal{D}_{t+1})} \tag{3}$$

Unfortunately, in this setting we still have to keep $\mathcal{D}_t$ around to represent the posterior. However, using approximate inference techniques explained in the next section we can represent that partial posterior in a parametric form, allowing us to *forget* data from the past when learning about the current environment.

### 2.1 Bayesian learning with variational inference

Formally, transitions are sampled iid from the intractable distribution $p(\mathbf{e}_k|\mathcal{D})$ that we approximate with $q_\phi(\mathbf{e}_k)$ parameterized by a multivariate normal where $\phi = \{\mu_q, \Sigma_q\}$. The learning objective is to maximize the marginal likelihood of observed transitions with respect to $\theta$ and $\phi$. We can maximize the evidence lower bound (**ELBO**) to this,

$$\begin{aligned} \log p(\mathcal{D}) &= \sum_{t=0}^{T} \log p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \\ &\geq \mathbb{E}_{\mathbf{e}_k \sim q_\phi(\mathbf{e}_k)} \left[ \sum_{t=0}^{T} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_k) \right] - \mathrm{KL}\big(q_\phi(\mathbf{e}_k)||p(\mathbf{e}_k)\big). \end{aligned} \tag{4}$$

For simplicity, we choose the prior $p(\mathbf{e}_k)$ and variational distribution $q_\phi(\mathbf{e}_k)$ to be Gaussian with diagonal covariance. During online learning, the prior is updated from new data $\mathcal{D}_t$ and we optimize a subtly different objective holding network parameters $\theta$ fixed:

$$\begin{gathered} p(\mathbf{e}_k|\mathcal{D}_{t+1}) \geq q_{\phi^*}(\mathbf{e}_k) \\ \text{s.t. } \phi^* = \mathrm{argmax}_\phi \mathbb{E}_{\mathbf{e}_k \sim q_\phi(\mathbf{e}_k)} \left[\log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_k)\right] - \mathrm{KL}\big(q_\phi(\mathbf{e}_k|\mathcal{D}_{t+1})||p(\mathbf{e}_k|\mathcal{D}_t)\big). \end{gathered} \tag{5}$$

### 2.2 Control with Auxiliary Variable Models

Given a learned dynamics model, agents can plan into the future by recursively predicting future states $\mathbf{s}_{t+1}, ..., \mathbf{s}_{t+h}$ induced by proposed action sequences $\mathbf{a}_t, \mathbf{a}_{t+1}, ..., \mathbf{a}_{t+h}$ such that $\mathbf{s}_{t+1} \sim p(\cdot|\mathbf{s}_t, \mathbf{a}_t)$. If actions are conditioned on the previous state to describe a policy $\pi(\mathbf{a}_t|\mathbf{s}_t)$, then planning becomes learning a policy $\pi$ to maximize expected reward over the predicted state-action sequence. A limitation of this approach is that modeling errors are compounded at each time step, resulting in sub-optimal policies when the learning procedure overfits to the imperfect dynamics model. Alternatively, we use *model predictive control (MPC)* to find the action trajectory $\mathbf{a}_{t:t+h}$ that optimizes $\sum_t^{t+h} \mathbb{E}_{\mathbf{s}_t, \mathbf{a}_t \sim p(\mathbf{s_t}, \mathbf{a}_t)}[r(\mathbf{s}_t, \mathbf{a}_t)]$ at run-time [1]. At each time step, the MPC controller plans into the future, finding a good trajectory over the planning horizon $h$ but applying only the first action from the plan, and re-plans again at the next step. Because of this, MPC is better able to tolerate model bias and unexpected conditions.

Algorithm 1 describes a learning procedure that uses the cross-entropy method (CEM) as the optimizer for an MPC controller [4]. On each iteration, CEM samples 500 proposed action sequences $\mathbf{a}_{t:t+h}$ from $h$ independent multivariate normal distributions $\mathcal{N}(\mathbf{a}_t|\mu_t, \Sigma_t)$, one for each time step in the planning horizon (line 8), and calculates the expected reward for each sequence. The top 10% performing of these are used to update the proposal distribution mean and covariance. However, evaluating the expected reward exactly is intractable, so we use a particle-based approach called trajectory sampling (TS) from [2] to propagate the approximate next state distributions.

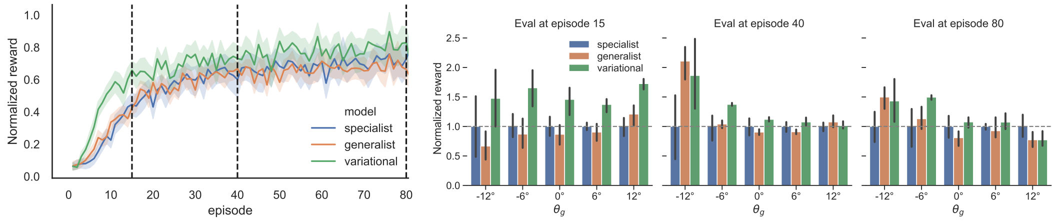
On a new environment, we skip training line 4 to keep the dynamics model $f_\theta$ fixed. Our task then is to iterate between acting at step $t$ and inferring $p(\mathbf{e}_k|\mathcal{D}_t)$ in order to align the expected dynamics with the current system the agent is acting in, even as it may be changing.

## 3 Experiments

To demonstrate learning and performance across environments, we show preliminary results in the HalfCheetah Mujoco simulator on proof-of-concept tasks that vary the direction of gravity $\theta_g$ where $0°$ is vertical and positive/negative values tilt forwards/backwards (like walking up or down a hill.) Otherwise environments were set up as in [2]. We split these environments into training and test environments, $\theta_g^{\text{train}} \in \{-12°, -6°, 0°, 6°, 12°\}$ and $\theta_g^{\text{test}} \in \{-15°, -9°, -3°, 3°, 9°, 15°\}$, such that four test environments are interpolated and the two end environments are extrapolated relative to the training distribution. (See Appendix C for results using only 2 training environments). To test continual learning, we test models trained in static environments on a dynamic environment where $\theta_g$ changes from $-6°$ to $6°$ halfway through the episode. For each environment, we collect an episode rollout via MPC given the current dynamics model. (The first episode is generated via random actions.) Then the ensemble is trained incrementally for 10 iterations on data seen so far, and the process is repeated. At test time, neural network parameters are fixed but the variational

**Algorithm 1** Learning and control with Model Predictive Control

```
Initialize data $\mathcal{D}$ with random policy
for Episode m = 1 to M do
    Sample an environment indexed by $k$
    If learning, train a dynamics model $f_\theta$ with $\mathcal{D}$ using Eq. 4
    Initialize starting state $\mathbf{s}_0$ and episode history $\mathcal{D}_k = \varnothing$
    for Time t = 0 to TaskHorizon do                                              ▷ MPC loop
        for Iteration i = 0 to MaxIter do                                         ▷ CEM loop
            Sample actions $\mathbf{a}_{t:t+h} \sim \mathrm{CEM}(\cdot)$
            Sample latent $\mathbf{e}_k \sim q_\phi(\mathbf{e}_k)$ for each particle state particle $s_p$
            Propagate next state predictions $s_{t+1,p} \leftarrow s_{t+1,p}$ using $f_\theta$ and TS-$\infty$   ▷ See [2]
            Evaluate expected reward $\sum_{\tau=t}^{t+h} {}^{1}\!/_{P} \sum_{p=1}^{P} r(\mathbf{s}_{\tau,p}, \mathbf{a}_\tau)$
            Update $\mathrm{CEM}(\cdot)$ distribution
        end for
        Execute first action $\mathbf{a}_t$ determined by final $\mathrm{CEM}(\cdot)$ distribution
        Record outcome $\mathcal{D}_t \leftarrow \{(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}\}$
        Record outcome $\mathcal{D}_k \leftarrow \mathcal{D}_k \cup \mathcal{D}_t$
        Update approximate posterior $q_\phi(\mathbf{e}_k|\mathcal{D}_t)$ using Eq. 5
    end for
    Update data $\mathcal{D} \leftarrow \mathcal{D} \cup \mathcal{D}_k$
end for
```



(a) Normalized episode reward averaged across training environments.

(b) Normalized episode reward per training environment.

Figure 2: Comparing normalized episode reward on training environments. **(a)** For each environment $k$, calculate the best mean reward (averaged over 5 different seeds) across all models $R_k$. The minimum reward $B_k$ is the mean reward from 50 rollouts of a random policy. Then, the observed rewards at episode $m$ are rescaled by the maximum and minimum mean reward, i.e. $\hat{r} = {}^{r-B_k}\!/_{R_k-B_k}$. The shaded regions and error bars are 95% CI from 500 bootstraps. **(b)** Rewards are normalized to the specialist mean (=1) and random policy (=0) per environment. The dashed vertical lines indicate checkpoints 15, 40, and 80 where the models are also evaluated on novel test environments (see Figure 3, lower panel).

parameters are updated online after every time step (using only the last transition) with 60 iterations and 5× larger learning rate. All experiments are repeated with 5 random seeds.

We compare our method against two baselines, a specialist and generalist agent. A specialist is an ensemble dynamics model (based on [2]) trained separately for each environment (no auxiliary variables or inference) and executed via MPC, achieving impressive performance in only dozens of episodes. This single-environment agent provides a strong performance baseline, but as we will see, can be beat because of positive transfer during training. The generalist is functionally the same as the specialist, but is trained on all data collected from the environments in $\theta_g^{\text{train}}$ but also *without auxiliary latent variables*. Thus the generalist demonstrates the ability of the model-based ensemble to generalize without our approach. Our latent model uses variational inference to jointly learn both a dynamics model and a latent representation $e_k$ of environment parameters, and so is expected to compare favourably against the generalist.

Fig. 2a shows the training performance as a function of number of episodes seen per environment. Overall, the variational approach learns faster than both baselines, doing at least as well as an agent specialized for each environment. However the performance difference closes with more training data (see Fig. 2b for a breakdown), as expert baselines can learn to model the

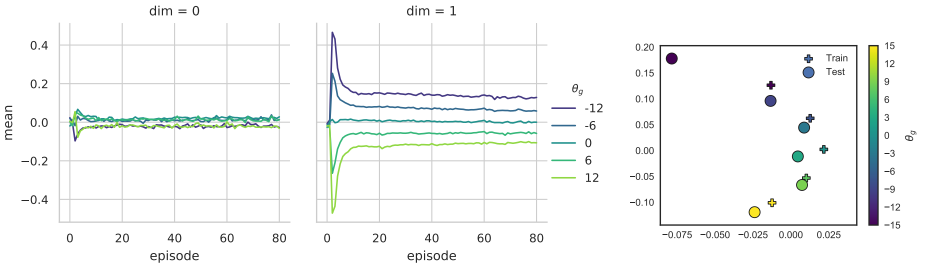


Figure 4: **Learned latent variable. Left and Center:** Means of the two dimensions of the latent variable during training. Dimension two is in order of $\theta_g$. **Right:** Mean of 2D inferred latent variables from training and test environments.

system arbitrarily well given enough data. We explore 5 episodes of online learning on an environment with changing dynamics, showing that the variational approach outperforms the generalist (see Fig. 3; Top). On the static (Bottom) test environment, the variational approach is superior in the low-data regime, demonstrating positive transfer without sacrificing performance overall. Yet its ability to extrapolate to the most extreme test angles does not exceed that of the generalist significantly. Either the ensemble-based generalist is a stronger baseline than anticipated for this task or further algorithmic improvements remain to be explored.

We visualize the unsupervised embeddings of dynamics for a qualitative assessment of the learned systems in Fig. 4. We observe that the order of the hidden parameter $\theta_g$ on training and test environments is captured accurately, while the $\mathbf{e}_k$ of the steepest novel test environments are inferred to lie outside those of the training environments.

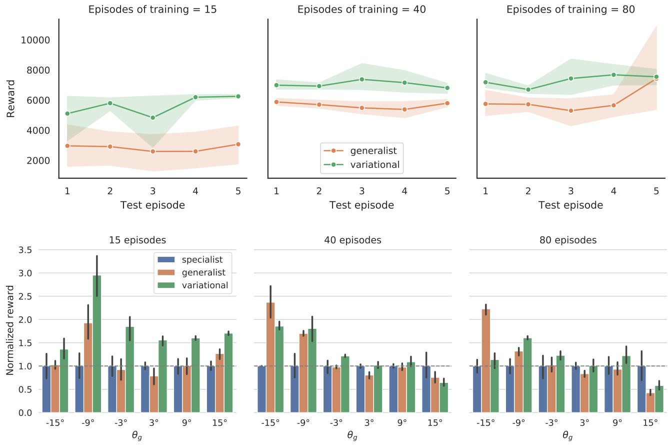


Figure 3: **Top:** Model performance on dynamically changing environment. $\theta_g$ changes from $-6°$ to $6°$ after 500 time steps in each episode. **Bottom:** Final normalized scores obtained on the novel environments after a few episodes of continuous learning.

## 4 Discussion

A long-standing aim in reinforcement learning is the search for robust learning algorithms that learn efficiently with less data and produce adaptable agents that generalize to many situations. We have shown that (ensemble) dynamics models with auxiliary latent variables can be used to learn quickly across environments with varying physical parameters, and also that they allow robust control on novel environments. This suggests that the latent variable approximately captures relevant degrees of freedom of the true dynamics of the environment and utilizes an explicit belief about the state of its environment beneficially. Our method suggests an alternate approach to popular implicit approaches (e.g. meta-learning such as MAML) to learning adaptive models across environment dynamics, and can readily incorporate other information available to a robotic agent, e.g. other perceptions or informative priors. Online inference relaxes constraints on environment stationarity without resorting to, e.g., sample-inefficient recurrent neural network policies. This combination of modeling elements affords our agents strong performance in terms of data efficiency and generalization on high-D continuous control task. Our results suggest future work on harder and more varied environments to further test the performance benefit of auxiliary latent variables.

## References

[1] E. F. Camacho and C. Bordons. *Model Predictive Control.* Springer Science & Business Media, 2013.

[2] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. *ArXiv preprint*, 2018.

[3] Ignasi Clavera, Anusha Nagabandi, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to Adapt: Meta-Learning for Model-Based Control. *ArXiv preprint*, 2018.

[4] Pieter-Tjerk De Boer, Dirk P Kroese, Shie Mannor, and Reuven Y Rubinstein. A tutorial on the cross-entropy method. *Annals of operations research*, 134(1):19–67, 2005.

[5] Marc P Deisenroth and Carl E. Rasmussen. PILCO: A Model-Based and Data-Efficient Approach to Policy Search. *Proceedings of the International Conference on Machine Learning*, 2011.

[6] Marc Peter Deisenroth. A Survey on Policy Search for Robotics. *Foundations and Trends in Robotics*, 2011.

[7] Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *IJCAI: proceedings of the conference*, volume 2016, page 1432. NIH Public Access, 2016.

[8] Yarin Gal, Rowan Thomas Mcallister, and Carl Edward Rasmussen. Improving PILCO with Bayesian Neural Network Dynamics Models. In *Data-Efficient Machine Learning Workshop, ICML*, 2016.

[9] Masahiko Haruno, Daniel M Wolpert, and Mitsuo Kawato. Mosaic model for sensorimotor learning and control. *Neural computation*, 13(10):2201–2220, 2001.

[10] Karol Hausman, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin Riedmiller. Learning an embedding space for transferable robot skills. 2018.

[11] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and efficient transfer learning with hidden parameter markov decision processes. In *Advances in Neural Information Processing Systems*, pages 6250–6261, 2017.

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles. *NIPS*, 2017.

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 2015.

[14] Anusha Nagabandi, Gregory Kahn, Ronald S Fearing, and Sergey Levine. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7559–7566. IEEE, 2018.

[15] Jiayu Yao, Taylor Killian, George Konidaris, and Finale Doshi-Velez. Direct policy transfer via hidden parameter markov decision processes. *LLARLA Workshop, FAIM 2018*, 2018.

# A Learning environment dynamics

We define the transition model of an environment by $T_\eta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ parameterized by $\eta$, which includes physical constants like gravity, friction, and dampening, or properties of an agent like actuator gain or noise [7, 11]. In order to perform model-based control for an agent acting in such an environment, one requires knowledge of the transition dynamics, which are composed of the dynamic mechanisms and the constants $\eta$.

When the quantities $\eta$ and the underlying mechanisms that govern environment dynamics are unknown, one can resort to learning a model of these dynamics $f_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t)$ given data observed from the environment $\mathcal{D} = \{(\mathbf{s}_t, \mathbf{a}_t), \mathbf{s}_{t+1}\}_{t=1}^{N}$.

Because environment dynamics can be stochastic, one can use a generative model of transition dynamics. Since these are continuous quantities, they can be modeled with a Gaussian likelihood parameterized by mean $\mu_\theta$ and covariance $\Sigma_\theta$ by a neural network $f_\theta$ with parameters $\theta$:

$$\begin{aligned} p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) &= p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t; \theta) \\ &= \mathcal{N}\left(\mu_\theta(\mathbf{s}_t, \mathbf{a}_t), \Sigma_\theta(\mathbf{s}_t, \mathbf{a}_t)\right) \end{aligned} \tag{6}$$

Here as elsewhere, instead of $\mathbf{s}_{t+1}$, a neural network predicts the change in the states $\Delta_s = \mathbf{s}_{t+1} - \mathbf{s}_t$ given the state and action $p(\Delta_s|\mathbf{s}_t, \mathbf{a}_t) = f_\theta$.

## A.1 Ensembles of networks

In order to be robust to model mis-specification and handle the small data setting, one can model uncertainty about parameters $\theta$ and marginalize to obtain

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = \int p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \theta) p(\theta) d\theta \,. \tag{7}$$

However, Bayesian neural networks can be computationally expensive to train. A practical way to approximate drawing samples from the posterior $p(\theta|\mathcal{D})$ is through ensembles of predictors each trained on different bootstraps or shuffles of the training data [12]:

$$p(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) = \frac{1}{|E|} \sum_{\theta \in E} p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t) \,. \tag{8}$$

In this work, each member of the ensemble is trained on distinct shuffles of the same data, and are reshuffled at every epoch as suggested in [12]. In contrast, Chua et al. [2] use bootstrap samples.

## A.2 Learning curves

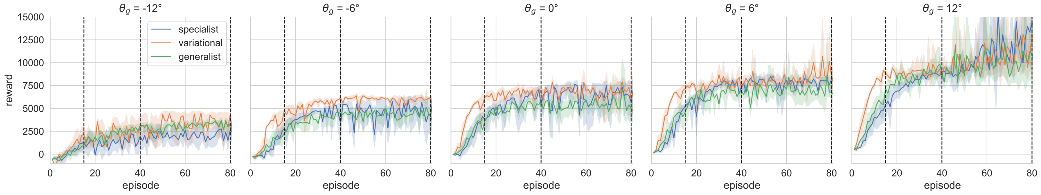


Figure 5: Episode reward per environment as a function of episodes seen on that environment. This implies that the generalist and variational models see 5X as much data as the specialist in total. The dashed vertical lines indicate checkpoints 15, 40, and 80 where the models are evaluated on novel test environments.

# B Implementation

The ELBO for a single environment $\mathcal{D}_k$ with $T$ timepoints is given as follows:

$$\log p_\theta(\mathcal{D}_k|e_k) \quad \geq \mathbb{E}_{\mathbf{e}_k \sim q_\phi(\mathbf{e}_k)} \left[ \sum_{t=0}^{T} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_k) \right] - \mathrm{KL}\big(q_\phi(\mathbf{e}_k)||p(\mathbf{e}_k)\big) \tag{9}$$

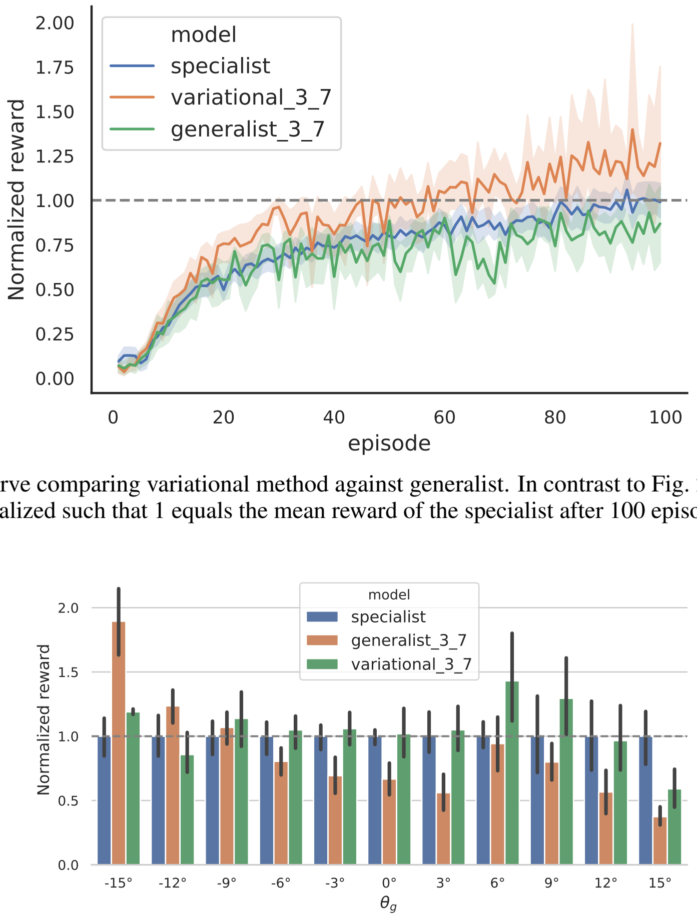
$$\geq \mathbb{E}_{\mathbf{e}_k \sim q_\phi(\mathbf{e}_k)} \left[ \sum_{t=0}^{T} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_k) \right] - \mathrm{KL}\big(q_\phi(\mathbf{e}_k)||p(\mathbf{e}_k)\big). \tag{10}$$

For clarity, we define $\log p_\theta(\mathcal{D}_k|e_k) = \sum_{t=0}^{T} \log p_\theta(\mathbf{s}_{t+1}|\mathbf{s}_t, \mathbf{a}_t, \mathbf{e}_k)$. Given ensemble $E$ of networks and Eq. 8, the complete data likelihood is an expectation over ensembles. Note that because each ensemble member is trained independently on different data, the expectation is *outside* the log. Hence,

$$\log p(\mathcal{D}_k) = \mathbb{E}_{\theta \sim E}[\log p_\theta(\mathcal{D}_k)] \geq \mathbb{E}_{\mathbf{e}_k \sim q_\phi(\mathbf{e}_k), \theta \sim E} \left[\log p_\theta(\mathcal{D}_k|\mathbf{e}_k)\right] - \mathrm{KL}\big(q_\phi(\mathbf{e}_k)||p(\mathbf{e}_k)\big)\,. \tag{11}$$

## C Generalizing from fewer environments

As in multi-task and meta-learning scenarios, the number of tasks/environments to sample is an important hyperparameter. Below, both the variational method and the generalist baseline are trained on only two environments $\theta_g = \pm 6°$ and evaluated after 100 episodes. In contrast to 5 environment training in Fig. 2a, the advantage of the variational method grows larger with more training, and remains competitive with the specialist on novel environments except for the steepest forward sloping environment (where the specialist gets very high reward.)



(a) Learning curve comparing variational method against generalist. In contrast to Fig. 2a, both panels in this figure are normalized such that 1 equals the mean reward of the specialist after 100 episodes.

(b) Normalized reward on training and test environments after 100 episodes.