Object-Factored Models with Partially Observable State

Isaiah Brand*, Michael Noseworthy*, Sebastian Castro, and Nicholas Roy
CSAIL, MIT
Cambridge, MA, USA
{ibrand,mnosew,scastro,nickroy}@csail.mit.edu

Abstract

In a typical robot manipulation setting, the physical laws that govern object dynamics never change, but the set of objects does. To complicate matters, objects may have intrinsic properties that are not directly observable (e.g., center of mass or friction coefficients). In this work, we introduce a latent-variable model of object-factored dynamics. This model represents uncertainty about the dynamics using *deep ensembles* while capturing uncertainty about each object's intrinsic properties using object-specific latent variables. We show that this model allows a robot to rapidly generalize to new objects by using information theoretic active learning. Additionally, we highlight the benefits of the deep ensemble for robust performance in downstream tasks.

1 Introduction

Predictive models are indispensable in the roboticist's toolkit. The ability to predict the outcome of an action allows a robot to plan robustly in a task-agnostic manner [6, 19]. For such models to be useful, they need to accurately capture the nuances of each object the robot interacts with. This is challenging due to the vast diversity of objects present in the world. Previous work has shown that factoring the model to decouple object state from the global dynamics leads to effective generalization when these properties are fully observable [4, 10, 19]. However, the assumption that object state is fully-observed does not generally hold; many objects have intrinsic properties that influence how they behave and that are not visually observable. For example, an object's *center of mass* will influence how it can be stacked, and a ball's *rolling resistance* will influence how far it rolls. When interacting with novel objects, we generally do not have labels for these properties (even at training time, these labels can be laborious to collect).

The key question we are interested in is: *how do we learn predictive models that support quick adaptation to objects with unobserved properties?* To do so, we propose to use an object-factored latent variable model that explicitly captures uncertainty about the object's unobserved properties *and* the global dynamics. The object-specific latent variables support information theoretic data acquisition in the learned latent space, allowing for rapid adaptation. The forward model, or dynamics function, is represented using a deep ensemble, allowing the model to capture uncertainty about complex dynamics. Both types of uncertainty can be incorporated when performing downstream tasks, leading to more robust uncertainty-aware planning.

We evaluate the proposed model in two domains: stacking blocks with unobserved centers of mass and throwing balls with unobserved rolling resistance. We demonstrate the utility of representing uncertainity at both the object and global levels. At the object level, the robot can use the learned latent space to quickly adapt to novel objects. At the dynamics level, the deep ensemble leads to more robust task performance when compared to a model that does not use ensembles.

*Equal contribution.

Bayesian Deep Learning workshop, NeurIPS 2021.

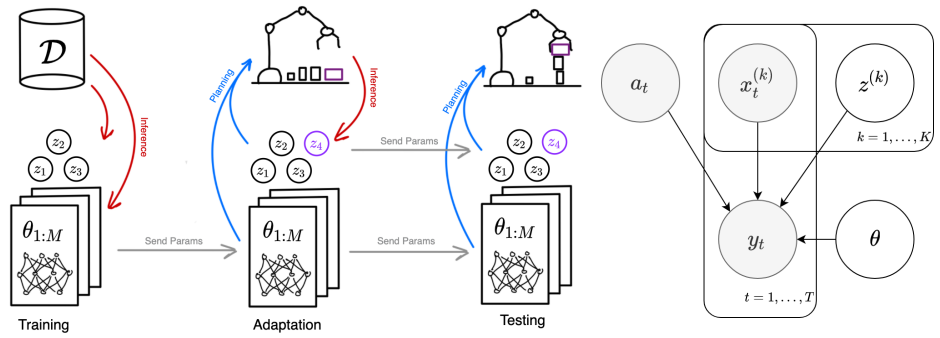

Figure 1: **Left.** Following a *training phase*, the robot is given new objects (purple) in the *adaptation phase* before being evaluated in the *testing phase*. **Right.** The outcome, y_t, depends on the state, $\mathcal{X}_t = \{x_t^{(k)}\}$, action, a_t, dynamics, θ, and each object's unobservable static properties, $\mathcal{Z} = \{z^{(k)}\}$.

2 Learning Phases for Object Manipulation

We consider a robot operating in a domain with K objects from the same class. Each object's state can be divided into observed and unobserved properties. The observed state, $\mathcal{X}_t = \{x_t^{(k)}\}_{k=1}^K$, can be time-varying, whereas the unobservable state, $\mathcal{Z} = \{z^{(k)}\}_{k=1}^K$, is assumed to be static. This covers a wide range of object properties including friction coefficients, mass, and coefficient of restitution. Our objective is to learn a model that predicts some aspect of the environment, y_t, which we will refer to as the *outcome*, that can be useful for downstream tasks. Concretely, the goal is to estimate $p(y_t|\mathcal{X}_t, \mathcal{Z}, a_t)$, where $a_t \in \mathcal{A}$ is the action space of the robot.

We assume the robot's operation is divided into three distinct phases: *training*, *adaptation*, and *testing* (see Figure 1). In the *training* phase, the set of objects is fixed, and the robot has some finite amount of time to act in the environment without any task descriptions. In the *adaptation* phase, the robot is given a new object and a short amount of time to experiment and adapt. Finally, there is a *testing* phase, in which a robot is supplied with an external reward function, and interacts with the objects in order to maximize that reward.

3 Object-Factored Latent Variable Model

We represent the predictive model using the probabilistic graphical model shown in Figure 1. The outcome, y_t, is generated via a stochastic process governed by global parameters, θ, unobserved object parameters, $\mathcal{Z}$, the current state, $\mathcal{X}_t$, and action, a_t. Due to the complexity of object dynamics, we represent θ using an ensemble [3] of domain-dependent neural networks (see Appendix B for architecture details for each domain). We also include a prior over object-specific latent variables, $p(\mathcal{Z})$.In the *training phase*, the primary goal is to learn about the global parameters, θ, that will generalize to the downstream *adaptation* and *testing phases*. Then, in the *adaptation phase*, we only need to infer the unobserved properties of novel objects, $\mathcal{Z}$.

3.1 Inference in Factored-Object Models

In the training and adaptation phases, the robot will have a collection of transitions: $\mathcal{D} = \{(\mathcal{X}_t, a_t, y_t)_i\}_{i=1}^N$. With this dataset, we would like to estimate a posterior distribution over the unobserved parameters: $p(\mathcal{Z}, \theta \mid \mathcal{D})$ during the training phase and $p(\mathcal{Z} \mid \mathcal{D})$ during the adaptation phase. A posterior that accurately represents *epistemic uncertainty* will allow us to gather data more efficiently and plan more robustly with models resulting from limited data. In the training phase, we assume the robot has access to a dataset, while in the adaptation phase, it must gather its own.

In general, computing the true posterior is intractable, so we take a *Variational Inference* (VI) approach, and compute an approximate posterior distribution, $q(\mathcal{Z}, \theta) = q_z(\mathcal{Z})q_\theta(\theta)$.We represent each $q(z^{(k)})$ as a diagonal Gaussian distribution. Although VI in theory can apply to both $\mathcal{Z}$ and θ, recent work has proposed representing uncertainty over neural network parameters implicitly using

an ensemble of M networks: $\theta = \theta_1, \ldots, \theta_M$ [16]. A longer discussion about these approximation can be found in Appendix A.1.

We propose an inference algorithm for generative models that uses ensembles to represent the uncertainty of the dynamics parameters while using VI for inference of the latent variables.[2] The algorithm alternates between optimizing $\mathcal{Z}$ with respect to the ELBO and updating each model in the ensemble to maximize the likelihood of the data. The pseudo-code can be found in Appendix A.3.

Optimizing $\mathcal{Z}$: To optimize the parameters of $q_z(\mathcal{Z})$, we maximize the ELBO with respect to the variational distribution parameters, while fixing the ensemble parameters. We derive a batch-loss which we minimize via gradient descent on the parameters of q_z (full derivation in Appendix A.2),

$$\mathcal{L}_z(B) = \frac{N}{|B|} \sum_{t \in B} \mathbb{E}_q[-\log p(y_t \mid \mathcal{X}_t, a_t, \mathcal{Z}, \theta)] + \frac{1}{|B|} \mathrm{D}_{\mathrm{KL}}(q_z(z) \parallel p(z)). \tag{1}$$

Optimizing θ: Unlike for $\mathcal{Z}$, we do not explicitly place a prior on θ. Instead, we can view the prior as being implicitly defined by the weight-initialization and stochasticity in SGD. Each ensemble model receives batches in a different order to further encourage ensemble diversity.

$$\mathcal{L}_{\theta_i}(B) = \frac{N}{|B|} \sum_{i \in B} \mathbb{E}_q[-\log p(y_t \mid \mathcal{X}_t, a_t, \mathcal{Z}, \theta_i)]. \tag{2}$$

During the adaptation phase, we only fit $q_z(\mathcal{Z})$ and freeze θ. Because $\mathcal{Z}$ is low dimensional, this can be done using VI or by particle filtering methods as described in Appendix A.4.

3.2 Experimental Design

When presented with a new object, a robot will need to adapt its model in a self-supervised manner. Because data collection on a real robot platform is expensive, it is important to be efficient. Random exploration is unlikely to lead to transitions that elicit the effects of the unobservable object properties, so we employ an information-theoretic active learning strategy. Specifically, during the adaptation phase, we choose actions that maximize the information gain between the label and the object-specific latent variable: $I(\mathcal{Z}; y_t | a_t, \mathcal{X}_t, \mathcal{D})$. We compute the information gain in label space as proposed in Houlsby et al. [12]. Note that we marginalize over the remaining uncertainity in the ensemble when selecting actions that inform us about the latents. The resulting acquisition function is,

$$\underset{a_t}{\operatorname{argmax}} \;\; \mathrm{H}(y_t \mid \mathcal{X}_t, a_t, \mathcal{D}) - \mathbb{E}_{\mathcal{Z} \sim q}\left[\mathrm{H}(y_t \mid \mathcal{X}_t, a_t, \mathcal{Z}, \mathcal{D})\right]. \tag{3}$$

4 Evaluation on Downstream Tasks

To evaluate the effectiveness of our approach, we analyze and ablate components of our model in two domains: *stacking* and *throwing*. In each domain, the agent first fits the model parameters using a fixed dataset in the training phase, and is then given a novel object to adapt to (see Appendix A.3 for an outline of the adaptation phase). In the stacking domain, each block's center of mass is unobserved. In the testing phase, the robot is tasked with maximizing the novel block's overhang above a base block: placing it as close to the edge as possible without falling over. In the throwing domain, the ball's rolling resistance and radius are unknown and the robot is tasked with throwing the ball such that it comes to rest at a desired distance. To optimize task performance, we calculate the expected reward under the posterior distribution of our model. More details about the domains and planning algorithms can be found in Appendix B.

Rapid Adaptation To measure how quickly different methods can adapt to novel objects, we plot task performance after each data point was collected with a given acquisition method (see Figure 2). In the stacking domain (left), information-theoretic data acquisition consistently outperforms the random strategy, successfully adapting after only 5 placements. This is because figuring out the center of mass accurately requires several consecutive and carefully planned placements.On the other hand, in the throwing domain, both random and active achieve similar good performance. We believe this is due to the smoother relationship between the actions, latents, and outcome, meaning most throws are going to be similarly informative about the latents. In both domains, we also compare to a baseline where we adapt a deep ensemble without latent variables at test time (see Appendix C for a description). The baseline fails to adapt quickly in each domain.

[2] In Appendix A.2, we show this is related to a complete VI approach if we ignore the prior on the dynamics.

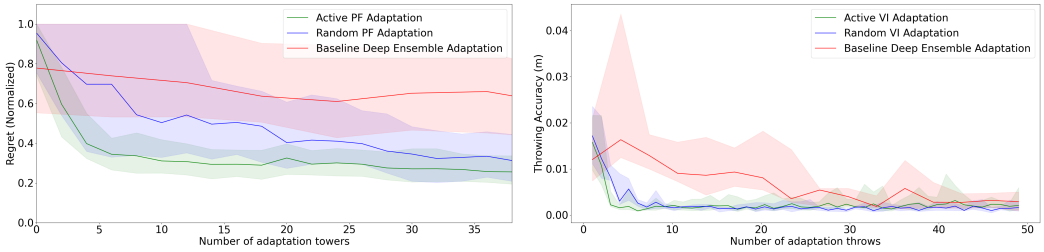

Figure 2: Task performance on each domain comparing adaptation strategies. Shaded areas are performance quartiles across 20 adaptation/testing runs. **Left.** Regret for the tower domain. **Right.** RMSE for the throwing domain.

Ensemble Robustness The proposed model represents uncertainity at two levels: at the object level through $\mathcal{Z}$ and at the global level through θ. We show that the ensemble is important for having robust performance in noisy domains by ablating the ensemble and using a single network for θ in the stacking domain (see Figure 3). Without the ensemble, the model does not consistently perform well, often receiving a regret of 1 (indicating the tower fell over). On the other hand, the ensemble leads to consistently good performance, showing the utility of the uncertainty representation for planning in downstream tasks.

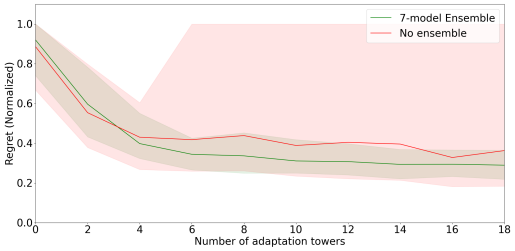

Figure 3: Task performance in the stacking domain when using a single neural network as opposed to a full ensemble.

5 Related Works

Several previous works have introduced Bayesian models for rapid adaptation. For example, Doshi-Velez and Konidaris [8] and Sæmundsson et al. [22] both use latent variables to parameterize a task and Gaussian Processes as the global dynamics. Killian et al. [14] have a similarly structured model but use a BNN as the dynamics model. Closely related to our model, Perez et al. [20] and Belkhale et al. [2] use both deep ensembles and low-dimensional latent variables within their models. We extend this framework to show that learned latent space can be used for active learning and analyze the importance of the ensemble.

Other work has focused on learning object dynamics with unobservable intrinsic object properties. In some works, the authors assume the global dynamics are known a priori which can inform the values of the object-specific properties [1, 24, 25]. In our work, we desire to jointly infer the dynamics and object properties. Related work which does not assume the dynamics are known often rely on either a fixed dataset [17, 23, 26, 27] at adaption time or a task definition in order to infer the latent properties [7, 8, 21, 22]. Instead, we propose to use active learning with respect to the model output which can be beneficial even when the task is not yet known. Our approach can yield zero-shot performance on a new task, because all the experimentation was performed in a task-agnostic adaptation phase.

6 Conclusion

In this work, we presented a model that represents uncertainty about the world at two levels: at the object-level using object-specific latent variables, and at the global level using deep ensembles. Our experiments show that information gain with respect to the learned latent space leads to rapid adaptation and that deep ensembles are critical for robust task performance. A key limitation that we plan to address in future work is the use of a fixed dataset during the training phase and a priori knowledge of the number of object-specific latent dimensions.

Acknowledgments and Disclosure of Funding

The authors would like to thank Caris Moses, Leslie Pack Kaelbling, and Tomás Lozano-Pérez for invaluable discussions relating to the ideas presented. This research was generously sponsored by the Honda Research Institute.

References

[1] P. W. Battaglia, J. B. Hamrick, and J. B. Tenenbaum. Simulation as an engine of physical scene understanding. *Proceedings of the National Academy of Sciences (PNAS)*, 110(45), 2013.

[2] Suneel Belkhale, Rachel Li, Gregory Kahn, Rowan McAllister, Roberto Calandra, and Sergey Levine. Model-Based Meta-Reinforcement Learning for Flight with Suspended Payloads. *IEEE Robotics and Automation Letters (RAL)*, 6(2), 2021.

[3] W. H. Beluch, T. Genewein, A. Nurnberger, and J. M. Kohler. The Power of Ensembles for Active Learning in Image Classification. In *2018 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[4] M. B. Chang, T. Ullman, A. Torralba, and J. B. Tenenbaum. A compositional object-based approach to learning physical dynamics. In *International Conference on Learning Representations (ICLR)*, 2017.

[5] N. Chopin. A sequential particle filter method for static models. *Biometrika*, 89(3):539–552, August 2002. ISSN 0006-3444, 1464-3510. doi: 10.1093/biomet/89.3.539. URL `https://academic.oup.com/biomet/article-lookup/doi/10.1093/biomet/89.3.539`.

[6] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPs)*, 2018.

[7] Misha Denil, Pulkit Agrawal, Tejas D. Kulkarni, Tom Erez, Peter Battaglia, and Nando de Freitas. Learning to Perform Physics Experiments via Deep Reinforcement Learning. In *Proceedings of The International Conference on Learning Representations*, 2017.

[8] Finale Doshi-Velez and George Konidaris. Hidden parameter markov decision processes: A semiparametric regression approach for discovering latent task parametrizations. In *Proceedings of the International Joint Conferences on Artificial Intelligence (IJCAI)*, volume 2016, 2016.

[9] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *International Conference on Machine Learning (ICML)*. PMLR, 2016.

[10] J. B. Hamrick, K. R. Allen, V. Bapst, T. Zhu, K. R. McKee, J. B. Tenenbaum, and P. W. Battaglia. Relational inductive bias for physical construction in humans and machines. In *the Annual Meeting of the Cognitive Science Society (CogSci)*, 2018.

[11] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 14(5), 2013.

[12] N. Houlsby, F. Huszar, Z. Ghahramani, and M. Lengyel. Bayesian Active Learning for Classification and Preference Learning, 2011.

[13] Neil Houlsby. *Efficient Bayesian active learning and matrix modelling*. PhD thesis, University of Cambridge, 2014.

[14] Taylor W Killian, Samuel Daulton, George Konidaris, and Finale Doshi-Velez. Robust and Efficient Transfer Learning with Hidden Parameter Markov Decision Processes. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPs)*, 2017.

[15] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *International Conference on Learning Representations (ICLR)*, 2014.

[16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NeurIPs)*, 2017.

[17] Peter Y Lu, Samuel Kim, and Marin Soljačić. Extracting interpretable physical parameters from spatiotemporal systems using unsupervised learning. *Physical Review X*, 10(3):031056, 2020.

[18] Kevin P Murphy. *Machine learning: a probabilistic perspective*. MIT press, 2012.

[19] Michael Noseworthy, Caris Moses, Isaiah Brand, Sebastian Castro, Leslie Pack Kaelbling, Tomás Lozano-Pérez, and Nicholas Roy. Active learning of abstract plan feasibility. In *Proceedings of Robotics Science and Systems (RSS)*, 2021.

[20] Christian F. Perez, Felipe Petroski Such, and Theofanis Karaletsos. Generalized Hidden Parameter MDPs Transferable Model-based RL in a Handful of Trials. In *Proceedings of The Thirty-Fourth AAAI Conference on Artificial Intelligence (AAAI)*, 2020.

[21] Kate Rakelly, Aurick Zhou, Deirdre Quillen, Chelsea Finn, and Sergey Levine. Efficient Off-Policy Meta-Reinforcement Learning via Probabilistic Context Variables. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

[22] Steindór Sæmundsson, Katja Hofmann, and Marc Peter Deisenroth. Meta reinforcement learning with latent variable gaussian processes. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2018.

[23] Rishi Veerapaneni, John D Co-Reyes, Michael Chang, Michael Janner, Chelsea Finn, Jiajun Wu, Joshua Tenenbaum, and Sergey Levine. Entity Abstraction in Visual Model-Based Reinforcement Learning. In *Conference on Robot Learning (CoRL)*, 2019.

[24] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[25] Jiajun Wu, Joseph Lim, Hongyi Zhang, Joshua Tenenbaum, and William Freeman. Physics 101: Learning Physical Object Properties from Unlabeled Videos. In *Procedings of the British Machine Vision Conference 2016*, 2016.

[26] Zhenjia Xu, Jiajun Wu, Andy Zeng, Joshua B. Tenenbaum, and Shuran Song. DensePhysNet: Learning Dense Physical Object Representations via Multi-step Dynamic Interactions. In *Proceedings of Robotics: Science and Systems (RSS)*, 2019.

[27] David Zheng, Vinson Luo, Jiajun Wu, and Joshua B. Tenenbaum. Unsupervised Learning of Latent Physical Properties Using Perception-Prediction Networks. In *Proceedings of Uncertainty in Artificial Intelligence (UAI)*, 2018.

A Modeling and Inference Details

A.1 Posterior Approximations

Ensemble Various approaches have been explored for representing uncertainty in neural networks (e.g., Monte Carlo Dropout [9] or Bayesian Neural Networks with mean-field weight posteriors). In this work, we choose to use ensembles of neural networks [16] as they afford expressive approximate posteriors while being simple to implement and train. Although ensembles are computationally expensive, in robotic domains where the rate of data collection is limited by the physical robot, we believe it is desirable to trade computational efficiency for sample efficiency.

Prior $p(\mathcal{Z})$ The original VAE [15] showed that samples from a Gaussian prior can encourage networks to learn a *smooth* latent space. We expect that in our object-factored model, learning a smooth representation will aid in generalizing to new objects.

Posterior $q_z(\mathcal{Z})$ The variational distributions for the object-specific latent variables are represented using diagonal Gaussian distributions. The variational parameters are, $\{\mu^{(k)}, (\sigma^2)^{(k)}\}_{k=1}^K$,

$$q_z(\mathcal{Z}) = \prod_{k=1}^{K} q_{z^{(k)}}(z^{(k)}) = \prod_{k=1}^{K} \mathcal{N}(\mu^{(k)}, (\sigma^2)^{(k)}) \tag{4}$$

A.2 ELBO Derivation

Although the derivation of the evidence lower bound for the proposed model follows the standard derivation and is not novel [18], we include it here for completeness. It is also helpful to highlight the difference between our proposed algorithm (with ensembles) and a full VI approach.

Objective

First, we will derive the ELBO specifically for our model. Here, x represents both the observed state and action. The overall goal is to maximize the marginal likelihood of the data (x, y) in the presence of latent variables (θ, z):

$$\begin{aligned}
\log p(y \mid x) &= \log \int_{z,\theta} p(y, z, \theta \mid x) = \log \int_{z,\theta} p(y, z, \theta \mid x) \frac{q(z,\theta)}{q(z,\theta)} \\
&= \log \mathbb{E}_{z,\theta \sim q}\left[\frac{p(y, z, \theta \mid x)}{q(z,\theta)}\right] \geq \mathbb{E}_q\left[\log \frac{p(y, z, \theta \mid x)}{q(z,\theta)}\right] \\
&= \mathbb{E}_q\left[\log \frac{\prod_{i=1}^N p(y_i, z, \theta \mid x_i)}{q(z,\theta)}\right] = \mathbb{E}_q\left[\log \frac{\prod_{i=1}^N p(y_i, z, \theta \mid x_i)}{q_z(z) q_\theta(\theta)}\right] \\
&= \mathbb{E}_q\left[\log \frac{\prod_{i=1}^N p(y_i, \mid \theta, x_i, z) p(\theta) p(z)}{q_z(z) q_\theta(\theta)}\right] \\
&= \sum_{i=1}^{N} \mathbb{E}_q\left[\log p(y_i, \mid \theta, x_i, z)\right] - \mathrm{D}_{\mathrm{KL}}(q_z(z) \parallel p(z)) - \mathrm{D}_{\mathrm{KL}}(q_\theta(\theta) \parallel p(\theta))
\end{aligned}$$

Model Update

In updating the model parameters, the goal is to maximize the EBLO. A minibatch loss can be derived as in [11] by ignoring terms that do not depend on θ and accounting for the batch size:

$$\mathcal{L}_\theta(B) = \frac{N}{|\mathcal{B}|} \sum_{i \in \mathcal{B}} \mathbb{E}_q[-\log p(y_i \mid \theta, x_i, z)] + \frac{1}{|\mathcal{B}|} \mathrm{D}_{\mathrm{KL}}(q_\theta(\theta) \parallel p(\theta)). \tag{5}$$

Because we represent our model using deep ensembles, we do not have an analytical prior and drop the KL-divergence term. Instead, each model is initialized independently and sees the data in a different order. Practically, we compute this quantity by sampling a minibatch from our dataset, sampling $z \sim q_z$, sampling $\theta \sim q_\theta$ and running a forward pass of the network to compute the log-likelihood of the labels. We can reduce the variance of this term by drawing multiple samples from q_z and q_θ.

Latent Update

The object specific latent variables follow the same derivation which results in a batch loss. This time, we do have an analytical prior for z,

$$\mathcal{L}_z(B) = \frac{N}{|B|} \sum_{i \in B} \mathbb{E}_q[-\log p(y_i \mid \theta, x_i, z)] + \frac{1}{|B|} \mathrm{D}_{\mathrm{KL}}(q_z(z) \parallel p(z)). \tag{6}$$

A.3 Algorithm Pseudocode

Algorithm 1 Batch Update

Input: Batches $\{B_i\}_{i=1}^{M}$, θ, $\mathcal{Z}$

for i in $1, \ldots, M$ **do**
  Calculate $\mathcal{L}_{\theta_i}(B_i)$
  $\theta_i \leftarrow$ Update(θ_i, $\mathcal{L}_{\theta_i}(B_i)$)
end for
Calculate $\mathcal{L}_{\mathcal{Z}}(B_0)$
$\mathcal{Z} \leftarrow$ Update($\mathcal{Z}$, $\mathcal{L}_{\mathcal{Z}}(B_0)$)

Algorithm 2 Adaptation Phase

$z' \leftarrow$ InitializeNewLatent()
$\theta, \mathcal{Z} \leftarrow$ FromTrainingPhase
$\mathcal{D} \leftarrow \{\}$
for nx in $0, \ldots,$ max_acquire **do**
  $\mathcal{X}_0 \leftarrow$ ResetEnvironment()
  $a \leftarrow$ ExperimentDesign($\theta, \mathcal{Z}, z'$)
  $y \leftarrow$ Execute($\mathcal{X}_0, a$)
  $\mathcal{D} \leftarrow \mathcal{D} + (\mathcal{X}_0, a, y)$
  $z' \leftarrow$ InferLatents($\theta, \mathcal{Z}, z', \mathcal{D}$)
end for

Algorithm 1 describes a single VI batch update to both the ensemble parameters and the object-specific latent variables. To encourage ensemble diversity we use an independent data loader for each ensemble such that each receives a separate batch, B_i, for each update. To update $\mathcal{Z}$, we use the batch from the first data loader, B_0. Although, our exposition considers block-coordinate descent, optimizing q_z and q_θ alternately, we found that this conferred no advantage over stochastic gradient ascent to optimize the joint ELBO directly.

Algorithm 2 shows at a high level the flow of the adaptation phase. At every timestep, we acquire a new datapoint based on the current uncertainity estimates. After collecting the label, we update or re-fit the model.

A.4 Adaptation Phase Particle Filter

During the fitting phase, we no longer wish to infer the dynamics and instead wish to compute the marginal posterior, $p(\mathcal{Z}|\mathcal{D})$. This will allow us to quickly gain information about a new object taking into account any remaining uncertainty about the dynamics. We could proceed as in the training phase and infer this distribution using a variational inference approach, however we found that this could result in over-confident posterior distributions. Instead, we also experiment with using a *particle filter* on the latent space as $\mathcal{Z}$ is low dimensional. Our implementation of the particle filter follows the approach of Chopin [5].

B Experimental Domains

We evaluate our proposed method in two simulated environments, a block stacking domain, and an object throwing domain. These diverse domains show that the proposed framework applies to both continuous (throwing) and discrete (stacking) dynamics.

B.1 Throwing

Domain Description: In the throwing domain (shown in Figure 4 left), a simulated robot arm learns to throw a variety of balls with an obstacle present. It is assumed that the robot will interact with each object one at a time and that an unfortunate intern is present to return the thrown objects to the robot workspace such that they may be thrown again. Each ball varies in its radius and rolling resistance. However, both these properties are not observed, and the model will need to use the object-level latent space, $\mathcal{Z}$, to represent them. The outcomes, $y_t \in \mathbb{R}$, capture how far the balls comes to rest along the x-axis from the initial release position.

The action space of the robot consists of a parameterized throwing primitive. The throw action is parameterized by the release angle and angular velocity of the ball. In the testing phase, the robot's goal is throw the ball such that it comes to rest at a desired distance. We report the RMSE averaged over many different goal distances.

Implementation Details: In the throwing domain, the dynamics function is represented as a standard multilayer perceptron. The inputs are the observed state, concatenated with the object-level latent for the current input ball, and action. We used a latent space of size 2 for this domain. The entropy for Equation 3 is calculated by approximating the differential entropy of a Mixture of Gaussians.

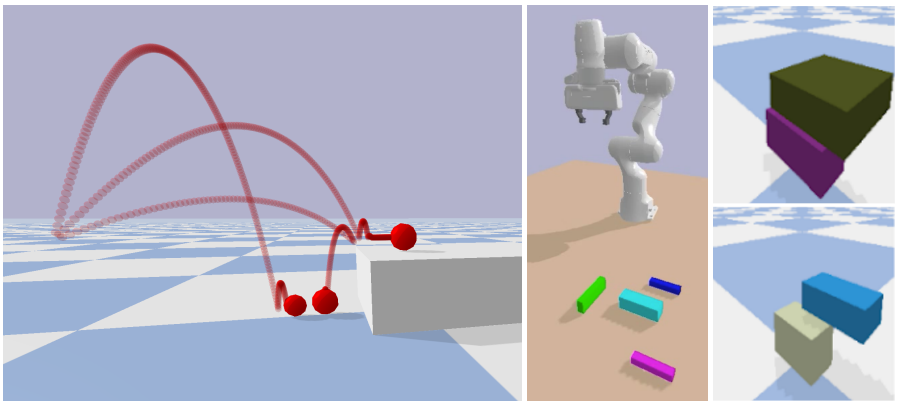

Figure 4: We evaluate our self-supervised dynamics learning framework in throwing (left) and stacking (center) domains. The towers on the right show examples of towers with small overhang (top) and large overhang (bottom).

B.2 Stacking

Domain Description: In the stacking domain, a simulated Franka Emika Panda robot arm learns to build towers out of adversarially weighted blocks. Unlike in the previous domain, multiple objects will interact with each other as well as the robot. The observed state, $\mathcal{X}_t$ is the shape and color of each block. The object-level latent variables, $\mathcal{Z}$, will need to represent the mass and center of mass of each block which are not visually observable. The robot's action space consists of parameterized pick-and-place actions where the parameters include which block to pick and its final position (we add $3mm$ of Gaussian distributed placement noise to each action). The robot does not change the block's orientation.

In quasi-static domains such as this, most of the dynamics are known while the block is attached to the gripper (and aren't influenced by the hidden state). It is when the robot lets go of a block that the unoberseved properties come into play: a tower may fall if we do not know its center of mass. We add additional structure to the dynamics model to represent this observation. We assume either the world stays in the state predicted by a known deterministic quasi-static dynamics function, $f_{qs}(X_t, a_t)$, or the state results in something else which we cannot predict. The outcomes are the Bernoulli random variable y_t and describe whether the robot actually ends up in the state predicted by the deterministic dynamics or not (this is the notion of *feasibility* described in [19]):

$$p_\theta(\mathcal{X}_{t+1}|\mathcal{X}_t, a_t, \mathcal{Z}) = \begin{cases} \mathcal{X}_{t+1} = f_{qs}(X_t, a_t) & \text{with prob } p_\theta(y_t|\mathcal{X}_{t+1}, \mathcal{X}_t, \mathcal{Z}, a_t) \\ \emptyset \end{cases} \tag{7}$$

Thus the model we are interested in learning predicts whether a tower is stable or not.

Implementation Details: In the blocks domain, θ is represented as a graph neural network. We use the same model architecture as in Noseworthy et al. [19]. Many previous works have shown graph networks to be well suited to modeling the dynamics of multiple interacting objects [10]. $\mathcal{Z}$ is a 3-dimensional random variable. Equation 3 is computed using standard calculations for entropy of a Bernoulli random variable. Section 2.3.4 of Houlsby [13] describes the calculation for marginal information gain used in the adaptation phase. The training phase uses 50 blocks and 2500 total towers.

Task Details: To investigate how well the learned latent space captures center of mass, we design a downstream task that requires detailed knowledge of this property. Specifically, the task is *maximum overhang* where the robot must place the novel block on top of another block such that the distance between the base of the bottom block and the farthest edge of the top block is maximized. We also disincentive the robot from building unstable towers by giving the robot a large negative reward of $-1m$ if the tower falls. Thus, given our model which predicts stability, the expected reward, R, is,

$$R = -1 * (1 - \mathbb{E}_{z\sim q_z,\ \theta\sim q_\theta}\left[p(y_t|\mathcal{X}_t, a_t, \mathcal{Z}, \theta)\right]) + r * \mathbb{E}_{z\sim q_z,\ \theta\sim q_\theta}\left[p(y_t|\mathcal{X}_t, a_t, \mathcal{Z}, \theta)\right],$$

where r is the reward computed from the observable state (position, and dimensions) the robot would receive if the tower is stable. The robot is only tested on towers with two blocks as this most clearly elicits the effects of the latent space (the model is trained on towers with $2-5$ blocks).

C Baselines

We compare our object-factored model to a baseline without factorization, which makes no explicit distinction between object properties and global dynamics. Concretely, the baseline is a deep ensemble with the same network architecture as the dynamics ensemble in our object factored model. During the adaptation phase, active learning using ensembles is applied to generate experiments with the new object [3, 19]. However, the baseline must update the weights of the neural network to fit the new data, as it has no latent variables.

The baseline ensemble achieves lower training accuracy; even with a relatively small number of training objects (50), the ensemble struggles to store the latent properties for those objects within the network weights. During the adaptation phase the baseline is considerably slower to adapt to the new objects, as it must adjust the entire network to explain the new observations.