Mixture of Basis for Interpretable Continual Learning with Distribution Shifts

Mengda Xu [*]
JPMorgan AI Research
`mengda.xu@jpmorgan.com`

Sumitra Ganesh[*]
JPMorgan AI Research
`sumitra.ganesh@jpmorgan.com`

Pranay Pasula
JPMorgan AI Research
`Pranay.Pasula@jpmorgan.com`

Abstract

Continual learning in environments with shifting data distributions is a challenging problem with several real-world applications. In this paper we consider settings in which the data distribution (task) shifts abruptly and the timing of these shifts are not known to the observer. Furthermore, we consider a *semi-supervised task-agnostic* setting in which the learning algorithm has access to both task-segmented and unsegmented data for offline training. We propose a novel approach called Mixture of Basis models (MoB) for addressing this problem setting. The core idea is to learn a small set of basis models and to construct a dynamic, task-dependent mixture of the models to predict for the current task. We also propose a new methodology to detect observations that are out-of-distribution with respect to the existing basis models and to instantiate new models as needed. We test our approach in multiple domains and show that it attains better prediction error than existing methods in most cases while using fewer models. Moreover, we analyze the latent task representations learned by MoB and show that similar tasks tend to cluster in the latent space and that the latent representation shifts at task boundaries when tasks are dissimilar.

1 Introduction

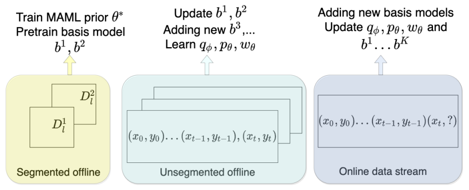
Continual learning in environments with shifting data distributions is a challenging problem with several real-world applications e.g. weather and financial market data are known to have *regime shifts*. In this paper, we consider settings where the data distribution (or *task*) shifts abruptly but the timing of these shifts are not known to the observer. In such domains, segmenting historical data into tasks is often a difficult problem in itself and much of the data is likely to be unsegmented. Motivated by this practical challenge, we consider a *semi-supervised task-agnostic* setting in which the learning algorithm has access to both task-segmented and unsegmented data for offline training. Typically segmented data will be available only for a small set of tasks. In the online setting the algorithm does not observe task boundaries and encounters new tasks that were not present in the offline dataset.

[*]Contributed equally

Contributions: Our key contribution in this paper is a new approach to the aforementioned problem that is based on a mixture of *basis models* (inspired by basis functions/vectors). The core idea is to learn a small set of basis models and to construct a dynamic, task-depdendent mixture of these models to predict for the current task. We also propose a new methodology to detect observations that are out-of-distribution with respect to the existing basis models and to instantiate new models. Our methodology uses a combination of model error and uncertainty to account for both covariate and concept shifts. Thus our approach allows for dynamic reuse of previously learned basis models across multiple tasks while simultaneously expanding the basis set *as needed* to adapt to new tasks.

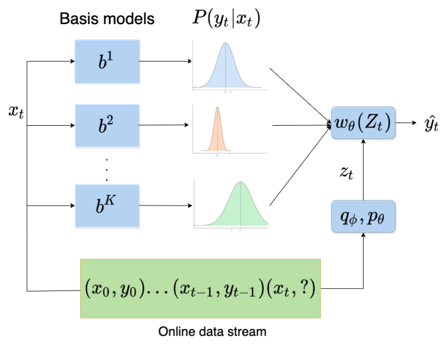
In our experiments on a synthetic regression domain and MuJoCo environments (HalfCheetah), we show that our approach, **M**ixture **of** **B**asis **(MoB)**, achieves better mean-squared error (MSE) than comparable methods in most cases. Moreover, MoB also allows for greater interpretability. We analyze the latent task representations learned by MoB and find that similar tasks tend to cluster together in the latent space and that the latent representation shifts at task boundaries when tasks are dissimilar.

2 Mixture of Basis (MoB)



Train MAML prior θ^*
Pretrain basis model
b^1, b^2
Update b^1, b^2
Adding new b^3,...
Learn $q_\phi, p_\theta, w_\theta$
Adding new basis models
Update $q_\phi, p_\theta, w_\theta$ and
$b^1 \dots b^K$
D_l^2
D_l^1
$(x_0, y_0)\dots(x_{t-1}, y_{t-1}), (x_t, y_t)$
$(x_0, y_0)\dots(x_{t-1}, y_{t-1})(x_t, ?)$
Segmented offline
Unsegmented offline
Online data stream

(a) Offline and Online: The segmented dataset contains samples for S tasks $\mathcal{D}_l := \{\mathcal{D}_l^1, \cdots, \mathcal{D}_l^S\}$ (here $S = 2$). The unsegmented dataset contains trajectories $\tau = (x_{0:T}, y_{0:T})$.



(b) MoB pipeline

Figure 1: Overview of MoB training

Problem Statement: Our goal is to learn a model that accurately predicts target variables Y_t from inputs X_t in a non-stationary environment in which the data distribution $P_{\mathcal{T}_t}(X_t, Y_t)$ can shift with time, depending on the current task $\mathcal{T}_t$. The task process $\mathcal{T}_t$ is Markovian (i.e., the current task $\mathcal{T}_t$ is conditionally independent of previous observations $X_{t'<t}$ and $Y_{t'<t}$ given the previous task $\mathcal{T}_{t-1}$).

We consider a *semi-supervised task-agnostic* setting in which the learning algorithm has access to both task-segmented $\mathcal{D}_l$ and unsegmented data $\mathcal{D}_u$ for offline training (see Fig. 1a). The problems are: (a) How do we effectively train the model using the offline dataset?, and (b) How do we update the model in the online setting to adapt to task shifts and new tasks that were not in the offline dataset? [2]

Basis Model Definition: A *basis model* is a model that predicts Y for a given X. We will use $b(Y|X)$ to denote the probability of Y given X under the basis model. Given a set of K basis models $\{b^{(i)}\}_{i=1:K}$ and a latent task representation $Z_t \in \mathbb{R}^d$ for the current task $\mathcal{T}_t$, we express the conditional distribution of Y_t given X_t and Z_t as the mixture (see Fig. 1b)

$$P(Y_t|X_t, Z_t) = \sum_{i=1}^{K} w^i(Z_t)\, b^{(i)}(Y_t|X_t) \tag{1}$$

where $\mathbf{w}(Z_t)$ is contained in the standard $(K-1)-$simplex (i.e., $\sum_{i=1}^{K} w^i(Z_t) = 1$ and $w^i(Z_t) \geq 0$ for $i = 1, 2, \dots, K$). Note that the mixing network $\mathbf{w}()$ takes only the task representation as input, whereas the basis models are task-agnostic. The benefits of using basis models are: (a)Basis models can capture the commonality across tasks such that we can reuseing the existing basis models to

[2]Note that this is similar to current task-agnostic continual learning approaches [1, 7, 8, 9, 14] that assume that task segmented data is available for offline training

predict new tasks (b)Latent representation Z_t drives how to combine the basis models and give us interpretability. We will first describe MoB's learning and inference procedure for a fixed number of basis models and then tackle the problem of when and how to add new basis models.

2.1 Learning and Inference with a Fixed Basis Set

Let's assume we are given a stream of observations $\tau := (x_{0:T}, y_{0:T})$ [3]. We use a sequential VAE construction ([2, 11]) and learn the parameters θ and ϕ of the model P and the inference model q, respectively, by maximizing the ELBO $\mathcal{L}(\tau; P, q)$ w.r.t. θ and ϕ. In order to compute the gradients w.r.t. the inference model parameters we use the re-parameterization trick in [10] and estimate the ELBO as

$$\hat{\mathcal{L}}(\tau; \theta, \phi) = \sum_{t=0}^{T} \log P_\theta(y_t|x_t, z_t) + \sum_{t=1}^{T} \log P_\theta(z_t|z_{t-1}) - \sum_{t=1}^{T} \log q_\phi(z_t|z_{t-1}, x_t, y_t)) \tag{2}$$

where $z_t = \mu_\phi(z_{t-1}, x_t, y_t) + \sigma_\phi(z_{t-1}, x_t, y_t) \odot \epsilon$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. Details of the derivation are included in the Appendix A.1.1. Note that the model P here uses the mixture of basis formulation in Eq. 1 and has three components: (i) the mixture network $\mathbf{w}_\theta(Z)$, (ii) the basis models $\{b_\theta^{(i)}(Y|X)\}_{i=1:K}$, and (iii) the prior task model $p_\theta(Z_t|Z_{t-1})$.

Pre-trained basis models: In theory the basis models could be learned using the above procedure. However, we found in practice that this often led to the basis models being too similar to one another. To overcome this issue, we instead use the segmented task dataset $\mathcal{D}_l$ to pre-train a basis model for each task and then allow for adaptation.

Uncertainty estimation using ensembles: We would like to estimate the uncertainty in each basis model in order to assess when to instantiate new models. We use deep ensembles [12] for uncertainty estimation by training an ensemble of M networks $\{b_j^{(i)}\}_{j=1:M}$ for each basis model $b^{(i)}$. Each model $b_j^{(i)}$ in the ensemble outputs the mean and variance of an isotropic Gaussian.

Fast basis instantiation using MAML: During offline and online training, we might have only few data points available for instantiation of a basis model. To enable fast instantiation, we use a meta-learned (MAML) [4] prior. Since we need ensemble basis models, we train an ensemble MAML prior $\{\theta_j^*\}_{j=1}^M$ using the segmented task dataset and b_j^i is adapted to task $\mathcal{T}_i$ from θ_j^* using the segmented data for that task.

2.2 Adding New Basis Models

For our approach to be able to adapt to new tasks in the offline or online setting, it needs to detect when the observations are out-of-distribution (OoD) with respect to the current set of models. Prediction errors [8] and model uncertainty [3] have been used to detect OoD samples. In case of *covariate* shift, well-calibrated models would have high uncertainty. But in case of a *concept* shift the model could be confident in its predictions while making errors because $P(Y|X)$ had changed. To reliably handle both types of shifts, we propose a novel **O**ut-of-**D**istribution **D**etection **S**core (ODDS) that combines model uncertainty and error.

To decide whether a data point (x, y) is OoD or not, we define a binary random variable D that can take values $\{I, O\}$ where I and O denote in and out of distribution, respectively. The decision on whether (x, y) is OoD can then be based on the score $S_{ODDS} := \frac{P(y|D=O,x)P(D=O|x)}{P(y|D=I,x)P(D=I|x)}$ with a default threshold of 1. Details on how the score is computed are included in Appendix A.1.2.

MoB creates an OoD buffer Q for each trajectory. If S_{ODDS} is below the threshold, 1, for a sample (x_t, y_t), it is added into the buffer. Similar to [16], once the buffer size exceeds some threshold L, MoB will create a new basis model by adapting the MAML prior to the samples in Q.

[3]**Notation**: We will use capital letters (e.g. Z_t) to denote random variables and small letters (e.g. z_t) to denote realizations/observations of the random variable

3 Experiments

Our goal is to study the following three aspects of our proposed approach: (a) **Performance**: how does our approach (MoB) compare to others in terms of average prediction error (MSE) over an online stream of tasks? (b) **Scaling**: how many basis models does MoB instantiate? (c) **Interpretability**: does the learned latent representation of the task(s) capture objectively measurable task similarity?

We evaluate our method on a synthetic regression domain and on building environment models for HalfCheetah (MuJoCo [16]). [4] In both domains we define multiple tasks (as described in the Appendix A.2). A subset of the tasks are included in the segmented and unsegmented offline dataset, and these task partitions are described in Appendix A.2, Table 1. After training on the offline dataset, all approaches are evaluated on an online data stream generated by a Markovian task process with a fixed, uniform probability per time step of switching to a different task. In particular, the online stream contains tasks that were not present in the offline data set in order to test the ability of the approaches to adapt to new tasks. For each domain and task partition, we report results on 10 different seeds.

We compare our method with three baselines - all of which use a meta-learned (MAML) [4] prior trained using offline data: (a) **MOLe:** MOLe [14] is an online algorithm that instantiates multiple *task* models from a meta-learned prior. For fair comparison to MoB, we add an offline training phase to MOLe by running it on the offline data and carrying over any generated models, (b) **MAML k-shot:** This approach uses a single prediction model that is adapted from MAML prior θ^* at each time step, using latest k data points, and (c) **MAML continuous:** This approach uses a single prediction model that is is continuously updated using the most recent observations; model is initialized at the start of the online phase using a meta-learned prior θ^*.

Performance and Scaling: MoB achieves the lowest mean squared-error (MSE) on the online stream of tasks, in most task partitions and across domains. On both the regression task and HalfCheetah ice-slope, MoB achieves significantly better performance than all baselines while instantiating significantly fewer models than MOLe. For example, in the HalfCheetah ice-slope domain, MoB instantiates $13, 15$ and 2 basis models (on average) in the three task partitions, in comparison to the $26, 53$ and 9 task models instantiated by MOLe. In the regression domain, MoB instantiates 3 models vs. MOLe's average of 12 (details in Appendix A.5, Table 2). This demonstrates MoB's ability to reuse previously learned basis models in order to achieve comparable or superior performance.

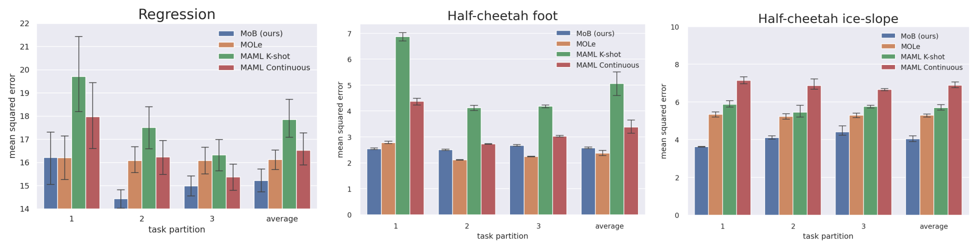


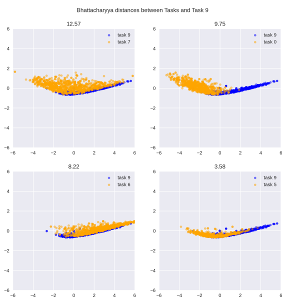
Figure 2: MSEs and 95% confidence intervals over 10 seeds for each algorithm on each task partition.

Interpretability: We analyzed the principal components of the Z_t inferred by MoB to study whether similar tasks cluster together in latent space. In the regression domain, we can quantify the task similarity by the Bhattacharya distance between the distribution of Y under different tasks. As we can see in Fig. 3a, latent task representations are well separated if tasks are sufficiently different and overlap more as the similarity increases. We also confirmed that the trajectories of the Z-components with the highest variance typically show shifts that are aligned with the task boundaries if the tasks are dissimilar (see Fig. 3c; additional plots can be found in Appendix A.5).

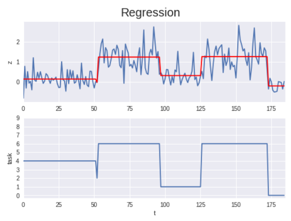
4 Related Work

Recent works in continual learning [5, 1, 6], have leveraged advances in meta-learning (e.g. MAML [4]) to enable fast adaptation to the current task. However, these approaches don't allow for modular-

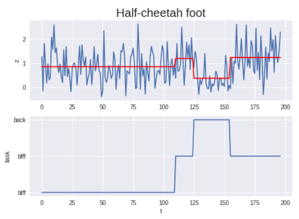
[4]Note that though the MuJoCo environment (HalfCheetah) is typically used in an reinforcement learning setting, here we are only interested in a regression task constructed using this domain (i.e., the task of predicting the next state, given the current state and action.)



(a) **Regression.** First two principal components of the 32-dimensional latent space Z. Bhattacharyya distance between tasks is above each subplot.



(b) **Regression.** Activations of a randomly selected z component (z_{15}) over a randomly selected trajectory. Mean values of task segments are shown in red. The activations of all other 31 z components is shown in Figure 7.



(c) **HalfCheetah Foot.** z component with maximum variance (mean values in red). When the task switches from *btff* to *bfff*, the activation doesn't change much, likely because *ff* is present in both. However, it shifts significantly when the task changes to *back*.

Figure 3: Analysis of learned latent representations.

ization and interpretability. A handful of works have proposed ways to combine model components but in ways that are different from our approach. [7] combines model components in parameter space whereas [9] proposes an approach where the MAML prior itself is modeled as a mixture distribution. MOLe [14], similarly to our approach, creates multiple prediction models by leveraging MAML and is hence used as a baseline for comparison.

5 Conclusions

We proposed a new approach (MoB) based on a mixture of basis models that is able to learn robustly in the presence of distribution shifts. Our experiments showed that MoB outperformed comparable methods, such as MOLe, in the majority of cases while instantiating significantly fewer models. Moreover, MoB learned interpretable latent task representations that captured information about task similarity and distribution shifts.

References

[1] Massimo Caccia, Pau Rodriguez, Oleksiy Ostapenko, Fabrice Normandin, Min Lin, Lucas Caccia, Issam Laradji, Irina Rish, Alexandre Lacoste, David Vazquez, and Laurent Charlin. Online fast adaptation and knowledge accumulation: a new approach to continual learning. March 2020.

[2] Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. A recurrent latent variable model for sequential data. In C Cortes, N D Lawrence, D D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 2980–2988. Curran Associates, Inc., 2015.

[3] Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. May 2018.

[4] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017.

[5] Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online Meta-Learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1920–1930. PMLR, 2019.

[6] Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-MAML: Look-ahead meta learning for continual learning. July 2020.

[7] Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A Rusu, Yee Whye Teh, and Razvan Pascanu. Task agnostic continual learning via meta learning. June 2019.

[8] Yujiang He and Bernhard Sick. CLeaR: An adaptive continual learning framework for regression tasks. January 2021.

[9] Ghassen Jerfel, Erin Grant, Thomas L Griffiths, and Katherine Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. December 2018.

[10] Diederik P Kingma and Max Welling. Auto-Encoding variational bayes. December 2013.

[11] Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. November 2015.

[12] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles, 2017.

[13] Davide Maltoni and Vincenzo Lomonaco. Continuous learning in Single-Incremental-Task scenarios. June 2018.

[14] Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via Meta-Learning: Continual adaptation for Model-Based RL. December 2018.

[15] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. MCP: learning composable hierarchical control with multiplicative compositional policies. *CoRR*, abs/1905.09808, 2019.

[16] Ahmed Hussain Qureshi, Jacob J. Johnson, Yuzhe Qin, Byron Boots, and Michael C. Yip. Composing ensembles of policies with deep reinforcement learning. *CoRR*, abs/1905.10681, 2019.

[17] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. June 2016.

A Appendix

A.1 Algorithms

A.1.1 Derivation of ELBO

We assume a generative model with the following structure: the latent task process Z_t is Markov

$$P(Z_t|X_{<t}, Y_{<t}, Z_{t'<t}) = P(Z_t|Z_{t-1}) \tag{3}$$

and the target variable Y_t is conditionally independent of past observations $X_{<t}$ and $Y_{<t}$ given the current input observation X_t and latent task Z_t i.e.

$$P(Y_t|X_{\leq t}, Y_{<t}, Z_{\leq t}) = P(Y_t|X_t, Z_t) \tag{4}$$

Derivation of Variational Lower Bound: Let's assume we are given a stream of observations $\tau := (x_{0:T}, y_{0:T})$. [5] We would like to approximate the (in general) intractable posterior distribution $P(Z_{\leq t}|X_{\leq t}, Y_{\leq t})$ using a distribution $q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})$ from a class of tractable distributions $\mathcal{Q}$. We can find the best approximating q by minimizing the KL divergence between the approximating distribution and the true posterior distribution

$$\min_{q \in Q} D_{KL}(q||P)$$

where

$$\begin{aligned} D_{KL}(q||P) &= E_{Z_{\leq t}\sim q}\left[\log \frac{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}{P(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}\right] \\ &= E_{Z_{\leq t}\sim q}\left[\log \frac{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}{P(Z_{\leq t}, X_{\leq t}, Y_{\leq t})} \cdot P(X_{\leq t}, Y_{\leq t})\right] \\ &= E_{Z_{\leq t}\sim q}\left[\log \frac{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}{P(Z_{\leq t}, X_{\leq t}, Y_{\leq t})}\right] + \log P(X_{\leq t}, Y_{\leq t}) \end{aligned} \tag{5}$$

[5]**Notation**: We will use capital letters (e.g. Z_t to denote random variables and small letters z_t to denote realizations/observations of the random variable

Rearranging terms,

$$\log P(X_{\leq t}, Y_{\leq t}) = D_{KL}(q||P) + E_{Z_{\leq t}\sim q}\left[\log \frac{P(Z_{\leq t}, X_{\leq t}, Y_{\leq t})}{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}\right] \tag{6}$$

$$\geq E_{Z_{\leq t}\sim q}\left[\log \frac{P(Z_{\leq t}, X_{\leq t}, Y_{\leq t})}{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}\right] \tag{7}$$

since $D_{KL}(q||P) \geq 0$. That is, the log-likelihood of the observed trajectory τ is lower bounded by the evidence lower bound (ELBO) or variational lower bound $\mathcal{L}(\tau; P, q)$ defined as follows

$$\log P(\tau) \geq \underbrace{E_{Z_{\leq T}\sim q}\left[\log \frac{P(Z_{\leq T}, x_{0:T}, y_{0:T})}{q(Z_{\leq T}|x_{0:T}, y_{0:T})}\right]}_{\mathcal{L}(\tau;P,q)} \tag{8}$$

We further assume a factored form of q (as in [11])

$$q(Z_0)\prod_{t=1}^{T} q(Z_t|Z_{t-1}, X_t, Y_t)$$

Plugging into Eq (8), we get

$$\mathcal{L}(\tau; P, q) = E_{Z_{\leq T}\sim q}\left[\log \frac{P(Z_{\leq T}, x_{\leq T}, y_{\leq T})}{q(Z_0)\prod_{t'<t=1}^{T} q(Z_{t'<t}|Z_{t'<t-1}, x_{t'<t}, y_{t'<t}))}\right] \tag{9}$$

Using the generative model assumptions, the numerator can be factorized as

$$P(Z_{\leq T}, X_{\leq T}, Y_{\leq T}) = \prod_t P(Z_t, X_t, Y_t|Z_{<t}, X_{<t}, Y_{<t}) \tag{10}$$

$$= \prod_{t'<t} P(Y_{t'<t}|Z_{t'<t}, X_{t'<t})P(X_{t'<t})P(Z_{t'<t}|Z_{t'<t-1}) \tag{11}$$

Plugging in, the ELBO $\mathcal{L}(\tau; P, q)$ is given by

$$E_{Z_{\leq T}\sim q}\left[\log \frac{P(Z_0)P(y_0|x_0, Z_0)P(x_0)\prod_{t=1}^{T} P(y_{t'<t}|Z_t, x_t)P(x_t)P(Z_t|Z_{t-1})}{q(Z_0)\prod_{t=1}^{t} q(Z_t|Z_{t-1}, x_t, y_t)}\right]$$

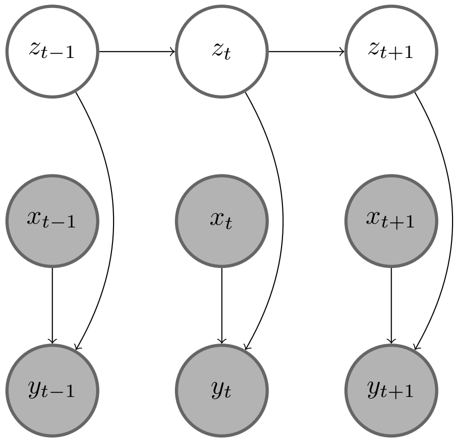


Figure 4: Generative model.

The $P(x_t)$ terms can be treated as a constant in the ELBO which then simplifies to

$$\begin{aligned}\mathcal{L}(\tau; P, q) = E_{Z_t \sim q}&\left[\sum_{t=0}^{T} \log P(y_t|x_t, Z_t)\right] \\ &- \sum_{t=1}^{T} E_{Z_{t-1} \sim q}\left[D_{KL}(q(Z_t|Z_{t-1}, x_t, y_t)||P(Z_t|Z_{t-1})\right] \\ &- D_{KL}(q(Z_0)||P(Z_0))\end{aligned} \tag{12}$$

The form of the ELBO is similar to that in the original VAE in that the first term measures the average reconstruction error using the model P and the latent variables sampled from q, while the second and third terms act as a regularization constraining KL-divergence between the approximating posterior distribution q and the prior.

A.1.2 Computation of the ODDS Score

To formalize our decision-making around whether a data point (x, y) is OOD or not, we define a binary random variable D that can take values $\{I, O\}$ where I and O denote in and out of distribution, respectively. Note that

$$P(D|x, y) = \frac{P(y|D, x)P(D|x)}{P(y|x)} \tag{13}$$

The decision on whether (x, y) is OOD can then be based on the score $S_{ODDS} := \frac{P(y|D=O,x)P(D=O|x)}{P(y|D=I,x)P(D=I|x)}$ with a threshold of 1. Note that, just as with model uncertainty, $P(D|x)$ only depends on the input observation x. On the other hand, the likelihood $P(y|D, x)$ is evaluated after the ground truth y is observed (similar to the model error).

We determine the in-distribution likelihood as $P(y|D = I, x) = \max_i\ b^i(y|x)$, by considering the basis model with the highest likelihood for the sample. To approximate the out-of-distribution likelihood $P(y|D = O, x)$, similarly to [14], we create a new basis b^{new} by adapting from the MAML prior θ^* using recently observed data, and use the likelihood $b_{new}(y|x)$ under this model. Assuming the data is OOD, fitting a new model is the best approximation.

To approximate the prior $P(D|x)$, we create a normalized uncertainty score for each basis and convert it into a probability-like measure. As in [12], we treat the ensemble for the basis b^i as uniformly weighted mixture of Gaussians. We further normalize the total variance of the mixture by the variance of the components

$$score_i(X) = \frac{M^{-1}\sum_{j=1}^{M}(\sigma^2_{b^i_j}(x) + \mu^2_{b^i_j}(x)) - (b^i(y|x))^2}{M^{-1}\sum_{j=1}^{M}\sigma_{b^2_{i_j}}(x)} \tag{14}$$

In order to determine the prior $P(D|x)$, we need to convert the model uncertainty $score$ into a probability-like measure. We use $\exp((1 - score(x))/\tau)$ to approximate $P(D = I|x)$, where τ is a temperature parameter that modulates the sensitivity to the score (less sensitive as τ increases). To be conservative, we use the minimum score from all basis models for the S_{ODDS} calculation.

MoB creates a OOD buffer Q for each trajectory. If ODDS is below threshold 1, we add the (x_t, y_t) into the buffer. Similar to [16], once the buffer size exceeds threshold L, Mob will create a new basis model by adapting the MAML prior to the buffer.

A.2 Domain Definition

Regression: We use randomly initialized neural networks to create different regression tasks. The input observation X is sampled uniformly at random in $[-1, 1]$ and $Y \sim \mathcal{N}(\mu_i(x); \sigma_i(x))$ where μ_i and σ_i are randomly initialized networks. The advantage of creating regression tasks in this manner is that (a) the regression tasks are not overly simplistic, and (b) we can construct informative and well-understood dissimilarity measures (e.g. Kullback-Leibler divergence, Bhattacharyya distance) between the conditional distributions $P(Y|X)$ under different tasks to quantify task similarity.

HalfCheetah foot: We create different tasks in the HalfCheetah environment by clipping the action space of one or two actuators into one-third of the original. To generate the data under for different

Algorithm 1: Mixture of Basis (MoB) - Instantiating new basis

```
Input: data $x_t, y_t$; meta-learned prior $\{\theta_m^*\}_{m=1}^M$; OOD buffer Q
Compute the $P(D = I|x_t, y_t)$ and $P(D = O|x_t, y_t)$ using Eq. (13)
if $\frac{P(D=I|x_t,y_t)}{P(D=O|x_t,y_t)} < 1$ then
    Add $(x_t, y_t)$ into Q
end
if $|Q| > L$ then
    Initialize new basis model $b^{new}$ with ensemble size of $M$.
    Adapt $b_j^{new}$ from $\theta_j^*$ to Q, $j = 1, .., M$
    Add $b^{new}$ into mixture model.
    Clear the Q
end
```

Algorithm 2: Mixture of Basis (MoB) - Offline

```
Input: Dataset $D_l$ with $K$ segmented task data and $_U$ without any label; $\{\theta_m^*\}_{m=1}^M$ from
  meta-learning; OOD buffer threshold T ; Temperature $\tau$
Initialize: $K \times M$ basis functions $\{b_{\theta_j^i}\}$; Inference network $q_\phi$; Prior network $p_\theta$
Return: $N \times M$ basis functions $\{b_{\theta_j^i}\}$; Inference network $q_\phi$; Prior network $p_\theta$
Start:
Sample from $D_l$ to train meta learning prior $\{M_{\theta i}\}$.
for i = 1:K do
    for j=1:M do
        Adapt $b_j^i$ to $D_{li}$ from $\theta_j^*$
    end
end
while Not converged do
    Sample a minibatch of $B$ trajectories $\eta = \{x_{0:T}, y_{0:T}\}$, each of length $T + 1$, from $D_U$
    for each $\eta$ in minibatch do
        Compute $z_t = \mu_\phi(z_{t-1}, x_t, y_t) + \sigma_\phi(z_{t-1}, x_t, y_t) \odot \epsilon_t, \quad t = 1, .., T$
        Compute $\hat{\mathcal{L}}(\eta; \theta, \phi)$ by plugging into Eq (2)
        OODS$(x_t, y_t), \quad t = 1, .., T$
    end
    Compute minibatch loss $\hat{\mathcal{L}}_{\theta,\phi} = -\sum_{i=1}^B \hat{\mathcal{L}}(\tau^{(i)}; \theta, \phi)$ (ELBO is to be maximized)
    Do gradient update: $\theta \leftarrow \theta - \nabla_\theta \hat{\mathcal{L}}_{\theta,\phi}; \quad \phi \leftarrow \phi - \nabla_\phi \hat{\mathcal{L}}_{\theta,\phi}$
end
```

Algorithm 3: Mixture of Basis (MoB) - Online

```
Input: $\{\theta_m^*\}_{m=1}^M$ from meta-learning; OOD buffer threshold T; Temperature $\tau$ ; $N \times M$ basis
  functions $\{b_{\theta_j^i}\}$; Inference network $q_\phi$; Prior network $p_\theta$; Input trajectory $\eta$
Start:
Sample $z_0 \sim \text{Unif}(\mathbb{R}^N)$
for each time step t do
    Compute $z_t = \mu_\theta(z_{t-1}, x_t) + \sigma_\theta(z_{t-1}, x_t)$
    Compute $\hat{y}_t$ using Eq (1)
    Compute $z_t = \mu_\phi(z_t, x_t, y_t) + \sigma_\phi(z_t, x_t, y_t)$
    Compute $\hat{\mathcal{L}}(\eta_{1:t}; \theta, \phi)$ by plugging into Eq (2)
    Do gradient update: $\theta \leftarrow \theta - \nabla_\theta \hat{\mathcal{L}}_{\theta,\phi}; \quad \phi \leftarrow \phi - \nabla_\phi \hat{\mathcal{L}}_{\theta,\phi}$s
    $OODS(x_t, y_t)$
end
```

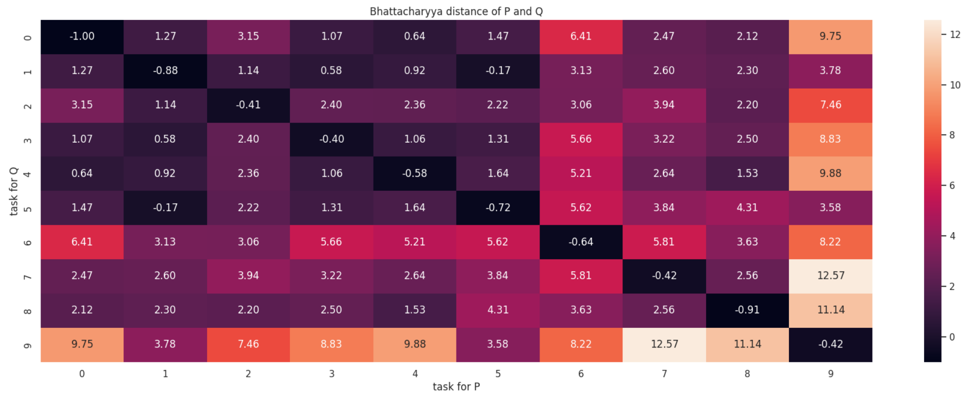
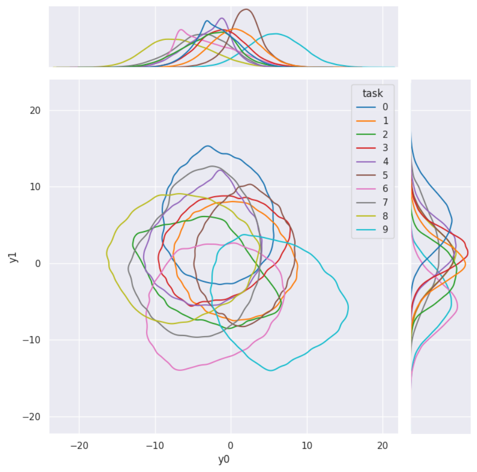
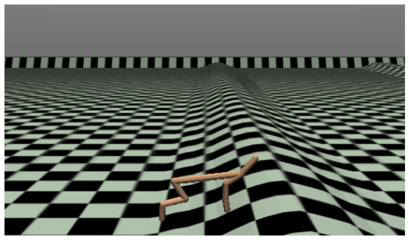


Figure 5: **Regression Domain:** Bhattacharyya distance between tasks.



(a) **Regression.** Marginal and joint probability density function estimates of variable y for each task in the synthetic Regression domain. For clarity only one level set per task is shown.



(b) **HalfCheetah ice-slope**, shown climbing a slope

Figure 6: Experiment domains.

tasks, we rollout a policy trained in the standard HalfCheetah environment (using SAC) under the different clipping settings. Specifically, we clip the front foot(ff), front thigh(ft), back foot(bf), and back thigh(bt) to create 8 tasks - bt, ff, ftff, bfft, btff, btft, bfff and btbf. We randomly pick two tasks to be included in the segmented offline data $\mathcal{D}_l$ and four tasks (including the ones in $\mathcal{D}_l$) to be included in the unsegmented offline data set $\mathcal{D}_u$. All tasks are included in the online data stream.

HalfCheetah ice-slope: We create different tasks using the HalfCheetah environment by changing the slopes and friction of the terrain. We create 4 different tasks for the offline training: 1) flat terrain with low friction / ice, 2) medium slope - medium slope with normal friction, 3) mixed (easy, medium, and hard) slopes with normal friction, and 4) medium slope with ice terrain. The data generation process is similar to the HalfCheetah foot. We randomly pick two tasks for the segmented offline dataset $\mathcal{D}_l$ and an additional two tasks for unsegmented offline dataset $\mathcal{D}_u$ (see Table 1 for details). During the online phase, we always test our method in a low friction terrain with different slopes (mixed ice) which was never seen during offline training. [6]

[6]The construction of tasks here is limited by the technical difficulty of not being able to switch the friction easily for a Mujoco environment within a single rollout.

Table 1: Task partition for experiments: the tasks listed under segmented are present both in the segmented and unsegmented offline datasets.

Domain	**Partition**	**Segmented**	**Unsegmented**
Regression	1	0, 5	1, 2, 6
	2	4, 7	5, 6, 9
	3	2, 7	0, 3, 9
HalfCheetah Foot	1	bt, ff	btff, bfft
	2	bfff, btft	btbf, ftff
	3	btbf, ftff	bfff, btft
HalfCheetah Ice-Slope	1	medium slope, flat ice	medium ice slope, mixed slope
	2	mixed slopes, flat ice	medium ice slope, medium slope
	3	medium ice slope, medium slope	mixed slopes, flat ice

A.3 Implementation details

For MAML prior, we train an ensemble of size 4. Similar to [12], we use all available task data but with random mini-batch during the training. For each basis model, it will adapt to one specific task from MAML prior. Each basis model will also have an ensemble of size 4. Each NN in the ensemble will adapt from one of the MAML prior ensemble. We use one layer LSTM followed by a two layers projection head to predict the next Y. Each layer has 128 nodes. We use batch size of 32 to train both MAML and basis models. For Mob training, θ and ϕ are both parameterized by three layers fully-connected network with 128 hidden units. They output the Z_t in the 32-dimensional latent space Z. To create new task, we use OOD buffer size T=20 and temperature $\tau = 10$. To optimize our model, we use Adam with learning rate 1e-4. We trained on a 4-GPU machine.

A.4 Related Work

Much of the work in continual learning (CL) is focused on classification tasks, with few works like [8] that are primarily focused on regression tasks. In our paper, we consider a task-agnostic continual learning problem for regression tasks. Regularization and rehearsal based approaches to CL [13] typically use a single network and employ strategies to prevent catastrophic forgetting. Architectural approaches like PNNs [17] instantiate a new neural network (a column) for each task being solved and prevent forgetting by freezing weights. However, these approaches scale poorly with the number of tasks and are not applicable in a task-agnostic setting.

Recent works have leveraged advances in meta-learning (e.g. MAML [4]) to shift the focus towards fast adaptation to new tasks using a meta-learned prior model, instead of attempting to strictly remember previous tasks with no adaptation. These class of techniques are highly suited to task-agnostic, streaming settings [5], [1], [6] - but they do not allow for modularization of the prediction model and interpretability as the task-specific model is typically created on-the-fly through adaptation and then discarded or continuously updated.

In MoB, we try to strike a balance between adaptability and interpretability by leveraging MAML for fast instantiation of basis models. Our approach of separating the task-specific and task-agnostic model components is similar to the "what and how" framework proposed in [7], and indeed can be thought of as a specific implementation. A handful of works have proposed ways to modularize and combine model components, similar to the spirit of MoB but in very different ways. The implementations proposed in [7] combine model components in parameter space whereas [9] propose an approach where the MAML prior itself is modeled as a mixture distribution. In a different problem setting of reinforcement learning, several works have also proposed composing policies or primitive skills in a multiplicative manner [15] or through a weighted sum [16].

A.5 Experimental Results

In this section, we list the average number of basis models generated by Mob and number of tasks generated by MOLe in our experiment. Mob generates less basis(tasks) than MOLe in almost all task partitions and domains.

Then we demonstrate the latent space Z shifts with switching tasks. We show that the PCA embedding space of different tasks are well separated if tasks are sufficiently different and start to overlap as the similarity increases.

Table 2: Average number of basis (task) models instantiated over 10 different seeds per partition.

Domain	**Partition**	**MoB**	**MOLe**
		# basis models	**# task models**
Regression	1	**3.4**	12.0
	2	**3.6**	11.8
	3	**3.3**	11.6
HalfCheetah Foot	1	3.0	**2.3**
	2	**3.0**	3.9
	3	**3.0**	3.4
HalfCheetah Ice-Slope	1	**13.4**	26.5
	2	**14.8**	52.8
	3	**2.0**	9.2

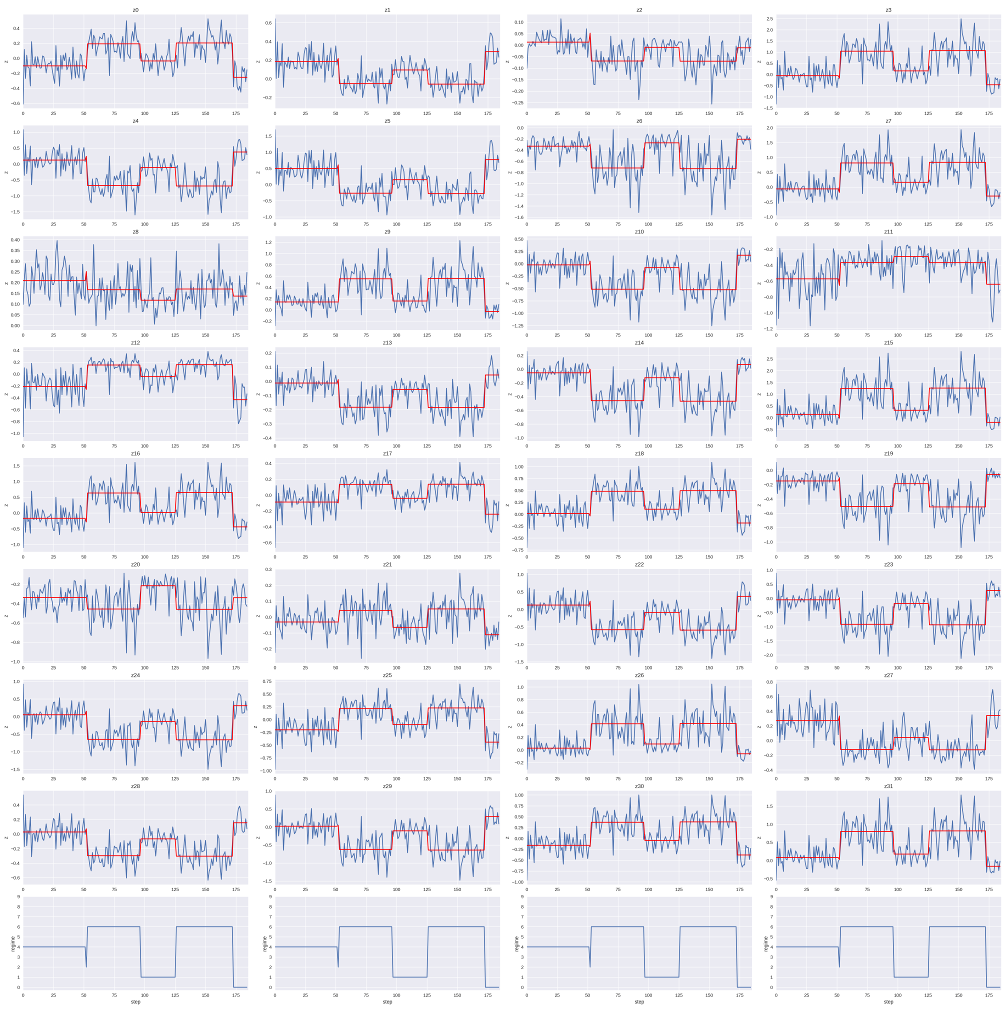


Figure 7: Following the Offline training phase on the Regression domain, these are activations of all 32 Z components as the active task varies according to the task plots at the bottom. The plots in each column are aligned so the activations at any time t vertically line up across subplots with other activations and the active task at time t.

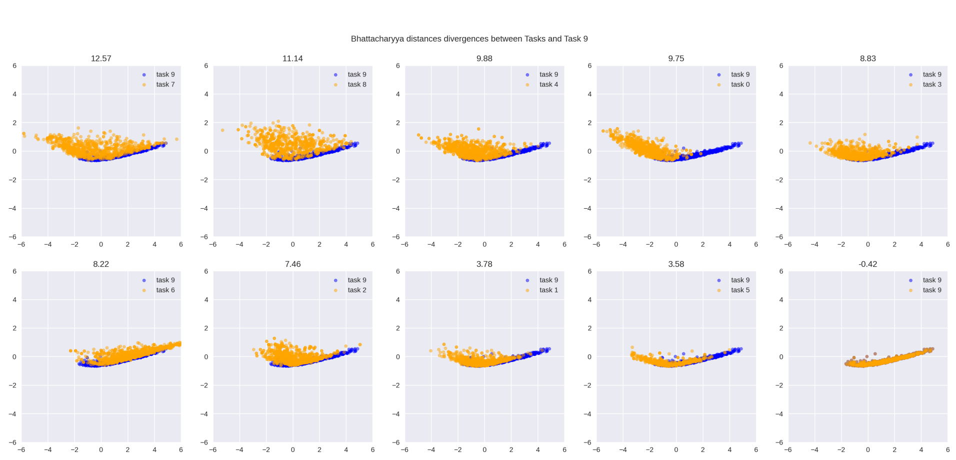
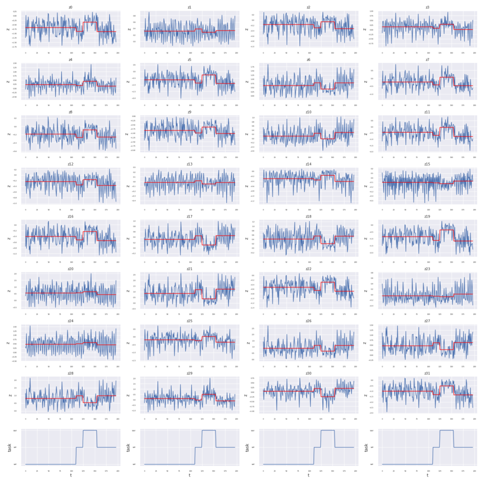
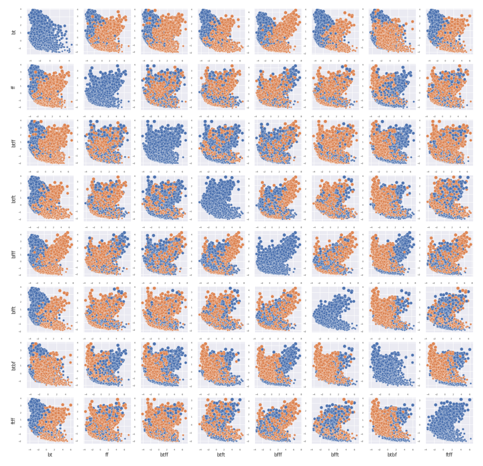


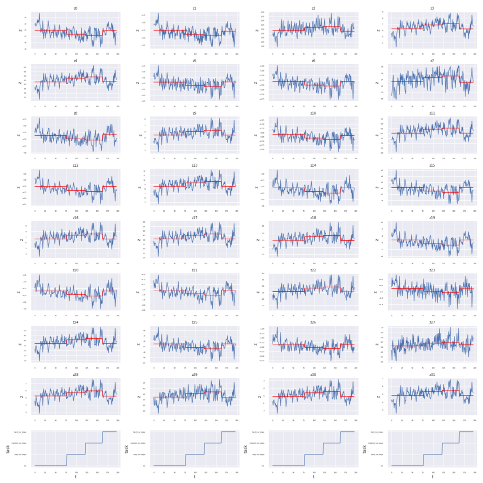
Figure 8: PCA embedding of latent space Z of regression dataset. Bhattacharyya distance between tasks is above each subplot.


(a) Following the Offline training phase on the HalfCheetah foot, these are activations of all 32 Z components as the active task varies according to the task plots at the bottom.


(b) 8x8 task pairwise plot over PCA embedding of latent space Z in HalfCheetah foot. First, task *bt* and *ff* distribution are quite different in the PCA embedding. Though they overlap on some region, it might because they still have some commonality between tasks. Tasks containing clipping *bt* is closer to task *bt*. Similarly, any task containing clipping *ff* or *ft* is closer to *ff*.

.

Figure 9: Latent space Z of HalfCheetah foot


(a) Following the Offline training phase on the HalfCheetah ice-slope, these are activations of all 32 Z components as the active task varies according to the task plots at the bottom.

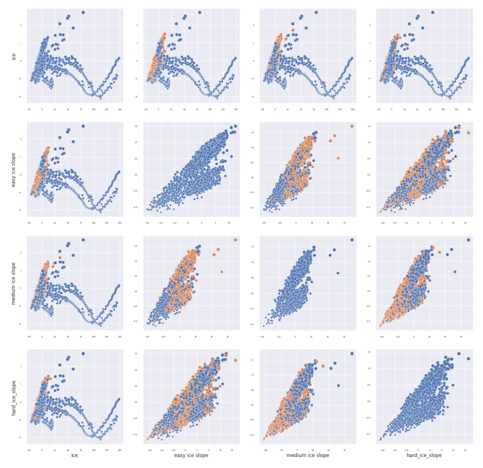

(b) 8x8 task pairwise plot over PCA embedding of latent space Z in HalfCheetah ice-slope. One interesting observation is that medium ice-slope and hard ice-slope distribution are different subsets of easy ice-slope distribution. We conjugate that tasks are similar and different level of ice-slopes can be expressed by different subset of easy ice-slope latent space.

Figure 10: Latent space Z of HalfCheetah ice-slope