MIXTURE OF BASIS FOR INTERPRETABLE CONTINUAL LEARNING WITH DISTRIBUTION SHIFTS

Anonymous authors
Paper under double-blind review

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 (i.e., task) shifts abruptly and the timing of these shifts are not known. 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 develop novel problem domains for regression tasks, evaluate MoB and other continual learning algorithms on these, and show that MoB attains better prediction error in nearly every case while using fewer models than other multiple-model approaches. We analyze latent task representations learned by MoB alongside the tasks themselves, using both qualitative and quantitative measures, to show that the learned latent task representations can be interpretably linked to the structure of the task space.

1 INTRODUCTION

Continual learning in environments with shifting data distributions is a challenging problem with several real-world applications. For example, weather and financial market data are known to have *regime shifts*. In this paper, we consider settings in which the data distribution (or *task*) shifts abruptly but the timings of these shifts are not known. In such domains, segmenting historical data into tasks (i.e., labeling the data) is often a difficult problem in itself, so often much of the available data is 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 because only a few of all possible tasks have been observed up until that point and because annotating data is costly. In the *online* setting the algorithm does not observe task boundaries and may encounter new tasks that were not present in the offline dataset.

Moreover, we focus on regression tasks in this paper. Since most benchmarks in continual learning are either image-based classification tasks or do not focus on temporal distribution shifts (Lomonaco & Maltoni, 2017; Koh et al., 2021), we construct and present new regression domains—both synthetic and based on real-data—that we believe could be useful to the research community in the study of regression tasks in the presence of temporal distribution shifts.

Our main contributions in this paper are as follows:

1. **Semi-supervised task-agnostic and interpretable continual learning algorithm.** MoB is based on a mixture of *basis models*, which is inspired by the idea of sets of basis vectors/functions, and is robust to distribution shifts. The core idea is to learn a small set of basis models and to construct a dynamic, task-dependent mixture of these models to predict for the current task.
2. **Out-of-distribution (OoD) detection method.** We propose a new methodology to detect observations that are OoD with respect to the set of existing basis models and to instantiate new models. Our methodology uses a combination of model error and model uncertainty to account

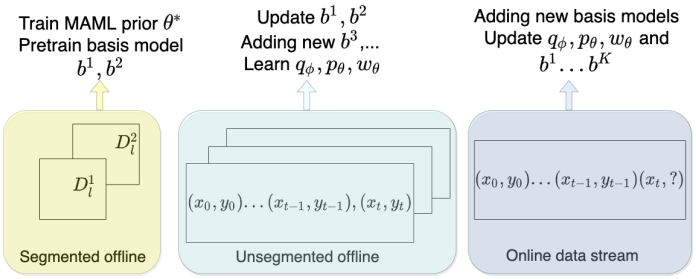

Figure 1: Offline and Online: overview of training and updates. The segmented dataset contains samples for S tasks $\mathcal{D}_l := \{\mathcal{D}_l^1, \cdots, \mathcal{D}_l^S\}$ (here $S = 2$). The unsegmented dataset contains trajectories $\tau = (x_{0:T}, y_{0:T})$.

for both covariate and concept shifts. This enables dynamic reuse of previously learned basis models across multiple tasks while simultaneously expanding the basis set *as needed* to adapt to tasks outside of the convex hull of the set of existing tasks.

3. **Synthetic regression problem domain with task similarity quantification.** We create a synthetic problem domain with distribution shifts, along with rigorous quantification of task similarity by computing the f-divergences between the tasks' data distributions. This allows systematic analysis of learned task representations.

4. **Real-world crude oil price benchmark.** Oil prices are well-known to vary in volatility over time. We construct a real-world benchmark by using over thirty years of West Texas Intermediate (WTI) crude oil daily prices to create regimes that are based on the volatility of the asset prices by fitting Markov switching models suited for heteroskedastic data ((Kim et al., 1998)).

In our experiments on a synthetic regression domain, MuJoCo environments (HalfCheetah), and real-world asset price forecasting, we show that our approach, **M**ixture **o**f **B**asis **(MoB)**, achieves better mean-squared error (MSE) than comparable methods (MOLe (Nagabandi et al., 2019), continuous adaptation, meta-learning based k-shot adaptation) in most cases. Compared to other *multiple model* methods such as MOLe, our approach uses significantly fewer basis models indicating that our mixture-based approach allows for better reuse of previously learned models. Moreover, MoB learns *interpretable* task representations that can be especially useful in a task-agnostic setting to gain insight and confidence in the model's inference and prediction. We analyze the latent task representations learned by MoB and find that similar tasks tend to cluster together in the latent space and that latent representations shift more at task boundaries when the two tasks are more dissimilar.

2 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* $P_{\mathcal{T}_t}(Y_t|X_t)$ can shift with time as $\mathcal{T}_t$ changes according to *task transition distribution* $P(\mathcal{T}_{t+1}|\mathcal{T}_t)$ and the *task process* $\mathcal{T}_t$ is Markovian.

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.1). This is similar to most recent *task-agnostic* continual learning approaches (Caccia et al., 2020; He et al., 2019; He & Sick, 2021; Jerfel et al., 2018; Nagabandi et al., 2019) that assume that task segmented data is available for offline training. *The main difference is that we also leverage unsegmented data*, and this is often much more available than segmented data in real-world problem settings for which labeling data is a costly and difficult endeavor.

We tackle two problems in this setting: (a) **Offline Pre-training:** How do we effectively train the model using the offline dataset?, and (b) **Online Adaptation:** After task shifts, how do we (1) reuse the model on tasks within the convex hull of tasks in the offline dataset and (2) quickly adapt to tasks outside of this convex hull?

3 INTERPRETABLE CONTINUAL LEARNING WITH DISTRIBUTION SHIFTS

Our approach, **Mixture of Basis** models (**MoB**), can dynamically combine the predictions from a set of what we call *basis models* to predict the target for the active task given inputs generated by this task. **Basis Model Definition:** A *basis model* is a model b that maps a given X to some Y. With a slight abuse of notation, we will also use $b(Y|X)$ to denote the probability of Y given X under the basis model b; point estimates for Y are constructed by taking the expectation of this distribution. 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*

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

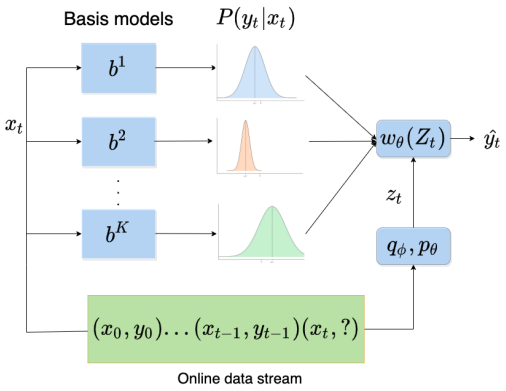

Figure 2: Overview of MoB pipeline.

where $\mathbf{w}$ maps latent task representations to the standard simplex (i.e., $\sum_{i=1}^{K} w^i(Z_t) = 1$ and $w^i(Z_t) \geq 0$ for $i = 1, \ldots, K$). Note that the mixing network $\mathbf{w}$ takes only the task representation as input, whereas the basis models are task-agnostic.

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.

3.1 LEARNING AND INFERENCE WITH A FIXED BASIS SET

Let's first consider that we have a fixed number of basis models and would like to infer the latent task representations z and make predictions y given a stream of observations $\tau = (x_{0:T}, y_{0:T})$ [1].

We use a sequential VAE construction ((Chung et al., 2015; Krishnan et al., 2015)) and learn the parameters θ and ϕ of the model P and the inference model q, respectively, by maximizing the ELBO $\mathcal{L}(\tau; \theta, \phi)$ w.r.t. θ and ϕ (see Appendix A.4.1 for details).

$$\mathcal{L}(\tau; \theta, \phi) = E_{z_t \sim q_\phi}\left[\sum_{t=0}^{T} \log P_\theta(y_t|x_t, z_t)\right] - \sum_{t=1}^{T} E_{z_t, z_{t-1} \sim q_\phi}\left[D_{KL}(q_\phi(z_t|z_{t-1}, x_t, y_t) \,||\, P_\theta(z_t|z_{t-1})\right]$$

The first term measures reconstruction error while the second term acts as a regularizer encouraging q_ϕ to match the prior P_θ over the task process Z_t unless q_ϕ improves reconstruction error enough to deviate from P_θ. In order to compute the gradients w.r.t. the inference model parameters we use the re-parameterization trick in Kingma & Welling (2013) and estimate the ELBO as:

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

where $z_t = \mu_\phi(z_{t-1}, x_t, y_t) + \sigma_\phi(z_{t-1}, x_t, y_t) \odot \epsilon$ and $\epsilon \sim \mathcal{N}(0, \mathbf{I})$. Note that the model P_θ 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 $b_\theta^{(i)}(Y|X)$, and (iii) the prior task model $p_\theta(Z_t|Z_{t-1})$.

Pre-trained basis models: In theory basis models could be learned using procedure above, but we found that in practice this often led to basis models being too similar to one another. To overcome this issue, we instead leverage the already available segmented task dataset $\mathcal{D}_l$ to pre-train a basis model for each task that can later be fine-tuned to online data.

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 (Lakshminarayanan

[1] **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

et al., 2017) 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 i.e. $b_j^{(i)}(Y|X=x) = \mathcal{N}(\mu_{ij}(x), \sigma_{ij}(x))$. As in Lakshminarayanan et al. (2017), we weigh each model in the ensemble equally to construct the predictive distribution for the i^{th} basis as mixture of Gaussians

$$b^{(i)}(Y_t|X_t) = \frac{1}{M}\sum_{j=1}^{M} b_j^{(i)}(Y_t|X_t) \tag{3}$$

and point estimate

$$\hat{y}_t = \frac{1}{M}\sum_{i=1}^{K} w^i(z_t)\sum_{j=1}^{M}\mu_{ij}(x_t) \tag{4}$$

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 prior using MAML (Finn et al., 2017). 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 $\mathcal{D}_l^i$ for that task.

3.2 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 (He & Sick, 2021) and model uncertainty (Farquhar & Gal, 2018) have been used to detect OoD samples or distribution shifts. In case of *covariate* shift, where inputs X are OoD with respect to the offline dataset, well-calibrated models would have high uncertainty. But in case of a *concept* shift, the model could be confident in its predictions (it has seen similar inputs X before) while still 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.

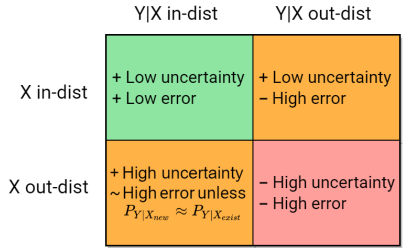

Figure 3: Model uncertainty and error for different scenarios of X and $Y|X$ being in or out of distribution.

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

$$S_{ODDS}(x, y) := \frac{P(y|D=O, x)P(D=O|x)}{P(y|D=I, x)P(D=I|x)} \tag{5}$$

with a default 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).

Computing the likelihood $\mathbf{P(y|D, x)}$: 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 $P(y|D = O, x)$, 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; the justification for this approximation is that if the data is OoD with respect to the current basis set, we need to fit a new model to evaluate the likelihood.

Computing the prior $\mathbf{P(D|x)}$: To approximate the prior $P(D|x)$, we create a normalized uncertainty score for each basis and convert it into a probability-like measure. To create a normalized uncertainty score for the i-th basis, we normalize the total variance of the ensemble by the variance of the

components

$$score_i(x) = \frac{M^{-1} \sum_{j=1}^{M} (\sigma_{ij}^2(x) + \mu_{ij}^2(x)) - \mu_i^2(x)}{M^{-1} \sum_{j=1}^{M} \sigma_{ij}^2(x),} \tag{6}$$

where $\mu_i(x) := M^{-1} \sum_{j=1}^{M} \mu_{ij}$. Intuitively, this score normalizes the total uncertainty by the aleatoric or *irreducible* uncertainty σ_{ij}^2; so as the *epistemic* uncertainty reduces and $\mu_{ij} \to \mu_i, \ \forall j$ (the models in the ensemble overlap), the $score_i \to 1$. We convert the normalized score into a probability by exponential scaling, $P(D = I|x) = e^{(1-score(x))/\eta}$ where η is a temperature parameter that modulates the sensitivity to the score (less sensitive as η increases). To be conservative, we use the minimum score from all basis models for the S_{ODDS} calculation.

MoB maintains a buffer of OoD samples and instantiates a new basis when the buffer size exceeds a threshold. The procedure to add a new basis is summarized in Algorithm **??** and is used by the offline and online algorithms summarized in Algorithm 1 and 2.

Algorithm 1: Mixture of Basis (MoB) - Offline

Input: segmented data $\mathcal{D}_l = \{D_l^1, \cdots, D_l^S\}$; unsegmented data $\mathcal{D}_u$
Return: $\{b_\theta^{(i)}\}$, q_ϕ, p_θ, $\mathbf{w}_\theta$, θ^*
Train meta-learned prior $\{\theta_j^*\}_{j=1:M}$ using $\mathcal{D}_l$
for *i = 1:S* **do**
    Adapt $b_j^{(i)}$ to D_l^i from θ_j^* for $j = 1, \cdots, M$
end
while *Not converged* **do**
    Sample a minibatch of B trajectories from D_u
    for *each* $\tau = \{x_{0:T}, y_{0:T}\}$ *in minibatch* **do**
        Init z_0
        for *t=1:T* **do**
            Sample $\epsilon_t \sim \mathcal{N}(0, \mathbf{I})$
            $z_t = \mu_\phi(z_{t-1}, x_t, y_t) + \sigma_\phi(z_{t-1}, x_t, y_t) \odot \epsilon_t$
            Add new basis using ODDS(x_t, y_t)
        end
        Compute $\hat{\mathcal{L}}(\tau; \theta, \phi)$ using Eq. (2)
    end
    Compute minibatch loss $\hat{\mathcal{L}}_{\theta,\phi} = -\sum_{i=1}^{B} \hat{\mathcal{L}}(\tau^{(i)}; \theta, \phi)$
    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 2: Mixture of Basis (MoB) - Online

Input: online data stream τ
Init z_0
for *each time step t* **do**
    Sample $\epsilon_t, \tilde{\epsilon}_t \sim \mathcal{N}(0, \mathbf{I})$
    Compute $z_t = \mu_\theta(z_{t-1}) + \sigma_\theta(z_{t-1}) \odot \epsilon_t$
    Compute $\hat{y}_t$ using Eq.(4)
    Update $z_t = \mu_\phi(z_t, x_t, y_t) + \sigma_\phi(z_t, x_t, y_t) \odot \tilde{\epsilon}_t$
    Compute $\hat{\mathcal{L}}(\tau_{1:t}; \theta, \phi)$ 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
    Add new basis using $ODDS(x_t, y_t)$
end

4 EXPERIMENTS

Our goal is to evaluate the following three aspects of our proposed approach, MoB: 1) **Performance.** How does MoB compare in terms of average prediction error (MSE) over an online stream of tasks over a diverse array of problem domains? 2) **Scaling.** How many basis models does MoB instantiate?

How does this compare to other modular continual learning approaches? 3) **Interpretability.** Do latent representations of tasks $\{Z\}$ reflect the similarity of tasks? For example, do latent representations for similar tasks cluster together? Are task shifts reflected in shifts within the latent space?

4.1 ENVIRONMENTS

We evaluate MoB and baselines on four regression domains: one synthetic, two regression problems in modified MuJoCo environments [2] Qureshi et al. (2019), and one real-world forecasting environment. In all domains we define multiple tasks as described below. A subset of the tasks are included in the segmented and unsegmented offline dataset (task partitions are described in the table 2). 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.

Regression: We use randomly initialized neural networks to create 10 different regression tasks. The input observation X is sampled uniformly at random in $[-1, 1]$ and $Y \sim \mathcal{N}(\mu_i(x); \sigma_i(x))$ for the i-th task, 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. KL divergence, Bhattacharyya distance) between the conditional distributions $P(Y|X)$ under different tasks to quantify task similarity. See Fig. 10 in A.1 for dissimilarity across tasks.

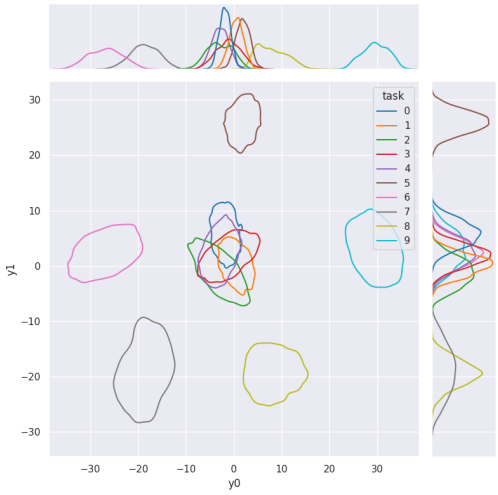

Figure 4: Marginal and joint probability density function estimates of variable y for each task in the synthetic Regression domain. For clarity only one level set per task is shown.

HalfCheetah Foot: We create different tasks in the HalfCheetah environment by clipping the action space of one or two actuators to one-third of the original. To generate the data, we rollout a policy trained in the standard HalfCheetah environment (using SAC (Haarnoja et al., 2018)) under the different task settings.

HalfCheetah Ice-Slope: We create another set of 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.

West Texas Intermediate (WTI) Crude Oil Prices: Oil prices are well-known to vary in volatility over time. We leverage over thirty years of publicly-available data ((wti)) to create a new real-world continual learning benchmark. We construct regimes based on the volatility of the asset prices by fitting Markov switching models that perform maximum likelihood estimation and are well-suited for heteroskedastic data (Kim et al. (1998)). We illustrate our process of constructing regimes from raw data in Figure 5.

4.2 EXPERIMENT RESULTS

We compare our method with three baselines—all of which use meta-learned priors trained using offline data:

[2]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)

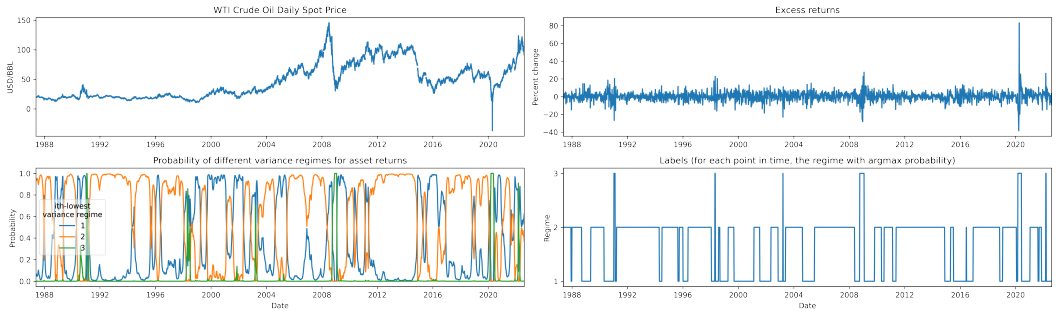

Figure 5: *(Top left)* WTI daily spot price. *(Top right)* Excess returns, a proxy of volatility and variance. *(Bottom right)* Fitted model regime probabilities over time. *(Bottom right)* Task labels for each point in time, computed via argmax probabilites.

- **MOLe:** MOLe (Nagabandi et al., 2019) is an online algorithm that instantiates multiple *task* models from a meta-learned prior. To enable fair comparisons between MOLe and other algorithms, we add an offline training phase to MOLe by running MOLe on each trajectory in the offline data and carry over all generated models to the online phase.
- **MAML k-shot:** This approach uses a single prediction model that is adapted from MAML prior θ^* using latest k data points.
- **MAML continuous:** This approach uses a single prediction model that is initialized using a meta-learned prior θ^* (learned using offline data). At each time step, the current model is updated using the most recent observations.

While MAML k-shot and continuous optimize for adaptation to the most recent task, MOLe also tries to optimize recall of previous tasks by maintaining multiple task models. These baselines provide a useful contrast against our approach which also tries to achieve adaptation to new tasks by reusing previously learned basis models. **MOLe** Nagabandi et al. (2019) is a particularly strong baseline for MoB because it operates on non-stationary data stream, uses multiple models and the well-established *expectation-maximization* algorithm, guaranteeing local *maximum a posteriori* estimates of model parameters.

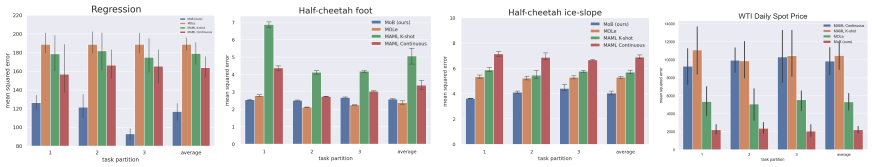

Figure 6: MSEs and 95% confidence intervals over 10 seeds for each algorithm on each task partition. *Note: the order for the bottom right subfigure is reversed.*

Performance. MoB attained the lowest MSE on the online stream of tasks in most task partitions across all domains other than HalfCheetah Foot, in which it attained MSE comparable to the lowest attained over all algorithms tested.In HalfCheetah Foot MoB outperformed MOLe only in the first task partition, which contains segmented tasks with only one actuator clipped; the other two partitions have segmented tasks with two actuators clipped. Intuitively the first task partition provides more 'primitive' segmented tasks in the offline phase.

Scaling. On Regression, HalfCheetah Ice-Slope, and WTI Daily Spot Price, MoB instantiated significantly fewer models than MOLe (Table 1), the only baseline that also instantiated new models. On HalfCheetah Foot, MoB instatiated fewer models for two of the three task partitions as well as the average over task partitions, but the difference was less pronounced than for all other domains.

Overall, MoB instantiated far fewer models than MOLe over three random partitions and ten random seeds per partition over four diverse domains. On HalfCheetah Foot, MoB still instantiated fewer models than MOLe over for two of the three partitions. Combined with the lower errors that MoB usually attained over MOLe and the other two baselines, we believe that MoB generally reused learned basis models more efficiently than MOLe did, leading to comparable and usually superior performance over all four baselines.

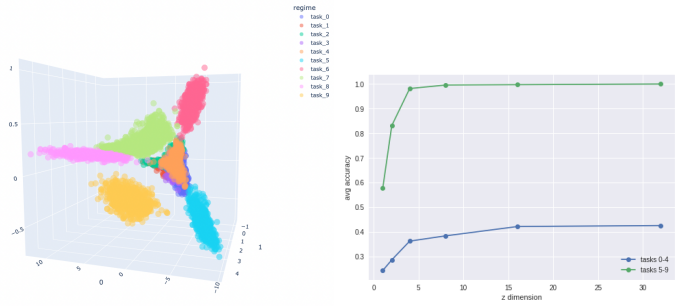

Figure 7: *(Left)* $\{z\}$ projected onto first 3 principal components. z for different inner tasks overlap, and z for different outer tasks are separable. *(Right)* In fact we validate this by computing accuracies of linear regression *one versus all* classifers on inner tasks 0-4 and outer tasks 5-9 as **dim** $z = 1, 2, 4, 8, 16, 32$.

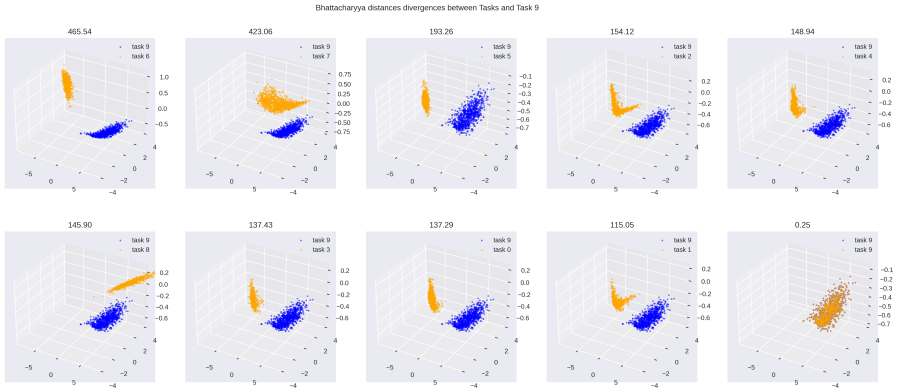

Figure 8: **Regression.** First 3 principal components of the 32-dimensional latent space Z. Bhattacharyya distance between tasks is above each subplot; plots ordered in terms of decreasing distance (increasing task similarity) to reference task 9 shown in blue.

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. 8, 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. 9; additional plots can be found in the Appendix).

Table 1: Average number of basis (task) models instantiated over 10 seeds per partition.

Domain	Partition	MoB	MOLe
Regression	1	**5.6**	12.6
	2	**5.1**	9.9
	3	**5.6**	8.5
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
WTI Daily Spot Price	1	**2.7**	6.4
	2	**3.1**	7.1
	3	**3.1**	8.1

5 Related Work

Early approaches: One of the earliest works that looked at learning combinations of models to solve multiple tasks was Jacobs et al. (1991). All tasks identities and boundaries are known, and all test tasks were included as training tasks. The approach uses a predefined number n of *expert* networks, a

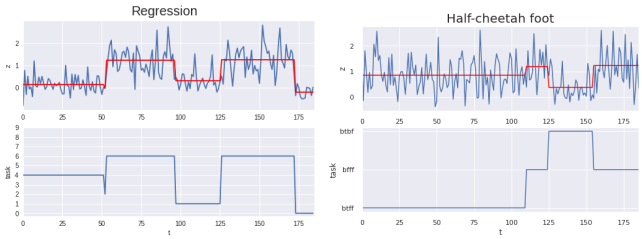

Figure 9: *(Left)* **Regression.** Activations of a randomly selected z component over randomly selected trajectory. Mean values of task segments shown in red. Activations of all other 31 z components are shown in A.4. *(Right)* **HalfCheetah Foot.** When task switches from *btff* to *bfff*, activation doesn't change much, likely because *ff* present in both. But it shifts significantly when task changes to *back*.

gating network, and a *selector* network. Aljundi et al. (2017) is an approach that builds on this classic algorithm using autoencoders and reconstruction error to measure task similarity.

Avoidance of catastrophic forgetting: The avoidance of catastrophic forgetting, or the event that a model's parameters are overwritten to its detriment due to an update rule on its current task, has been a known problem for decades McCloskey & Cohen (1989). *Regularization and rehearsal-based* approaches to continual learning (Maltoni & Lomonaco, 2018) typically use a single network and employ strategies Kirkpatrick et al. (2017) to prevent catastrophic forgetting of previously learned tasks. *Architectural* approaches such as Rusu et al. (2016) instantiate a new networks for each task but scale poorly with the number of tasks and are not applicable in task-agnostic settings.

Meta-learning for fast adaptation: Recent works in continual learning (Finn et al., 2019; Caccia et al., 2020; Gupta et al., 2020), leverage advances in meta-learning (e.g. MAML (Finn et al., 2017)) to enable fast adaptation to the current task, instead of attempting to strictly remember previous tasks with no adaptation. However, these approaches lack modularization and interpretability. A handful of works have proposed ways to combine model components that are different from our approach. He et al. (2019) combines model components in parameter space whereas Jerfel et al. (2018) proposes an approach where the MAML prior itself is modeled as a mixture distribution. MOLe (Nagabandi et al., 2019), similar to our approach, creates multiple prediction models by leveraging MAML and is hence used as a baseline for comparison. Our approach of separating the task-specific and task-agnostic model components is similar to the "What and How" framework proposed in He et al. (2019), but the latter uses a meta-learned model as an instantiation for task inference models whereas MoB uses a meta-learning model as an instantiation for new task-specific models (i.e., basis models).

Out of Distribution (OoD) Detection: Several methods to detect OoD samples have been proposed in the context of classification tasks - Hendrycks & Gimpel (2016) take the maximum of softmax probability as the confidence score to detect OoD samples while Lee et al. (2018) define the score based on the Mahalanobis distance; Hendrycks et al. (2018) and Roy et al. (2021) assume the models have access to a large data set of known outliers. He & Sick (2021) focus on a regression setting and use prediction errors with set thresholds to detect *novel* samples. In contrast, we cast the detection problem as a binary decision and handle combinations of shifts in $P(X)$ and $P(Y|X)$ by jointly considering the model uncertainty and likelihood.

6 Conclusions

We propose a new approach, Mixture of Basis (MoB), that can learn robustly in the presence of distribution shifts. Our approach learns a set of basis models and constructs a dynamic, task-dependent mixture of the existing basis models to predict for the current task. We also introduce a new methodology (ODDS) to instantiate new basis models which takes into account both the uncertainty of the existing models before the ground truth is observed, as well as the likelihood of the ground truth after it is observed. We show in our experiments that MoB is able to perform better than comparable methods like MOLe in most cases, while instantiating significantly fewer models. Moreover, our analysis of the latent task representations learned by MoB indicate that it learns *interpretable* task representations - we show that similar tasks cluster together in the latent space and that the latent representation shifts at task boundaries when the tasks are dissimilar. Thus, our proposed approach learns meaningful task representations while achieving good performance and scaling.

REFERENCES

Rahaf Aljundi, Punarjay Chakravarty, and Tinne Tuytelaars. Expert gate: Lifelong learning with a network of experts. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3366–3375, 2017.

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. URL http://arxiv.org/abs/2003.05856.

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 (eds.), *Advances in Neural Information Processing Systems 28*, pp. 2980–2988. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5653-a-recurrent-latent-variable-model-for-sequential-data.pdf.

Sebastian Farquhar and Yarin Gal. Towards robust evaluations of continual learning. May 2018. URL http://arxiv.org/abs/1805.09733.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. *CoRR*, abs/1703.03400, 2017. URL http://arxiv.org/abs/1703.03400.

Chelsea Finn, Aravind Rajeswaran, Sham Kakade, and Sergey Levine. Online Meta-Learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 1920–1930. PMLR, 2019. URL http://proceedings.mlr.press/v97/finn19a.html.

Gunshi Gupta, Karmesh Yadav, and Liam Paull. La-MAML: Look-ahead meta learning for continual learning. July 2020. URL http://arxiv.org/abs/2007.13904.

Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications. *CoRR*, abs/1812.05905, 2018. URL http://arxiv.org/abs/1812.05905.

Xu He, Jakub Sygnowski, Alexandre Galashov, Andrei A Rusu, Yee Whye Teh, and Razvan Pascanu. Task agnostic continual learning via meta learning. June 2019. URL http://arxiv.org/abs/1906.05201.

Yujiang He and Bernhard Sick. CLeaR: An adaptive continual learning framework for regression tasks. January 2021. URL http://arxiv.org/abs/2101.00926.

Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *CoRR*, abs/1610.02136, 2016. URL http://arxiv.org/abs/1610.02136.

Dan Hendrycks, Mantas Mazeika, and Thomas G. Dietterich. Deep anomaly detection with outlier exposure. *CoRR*, abs/1812.04606, 2018. URL http://arxiv.org/abs/1812.04606.

Robert A Jacobs, Michael I Jordan, Steven J Nowlan, and Geoffrey E Hinton. Adaptive mixtures of local experts. *Neural computation*, 3(1):79–87, 1991.

Ghassen Jerfel, Erin Grant, Thomas L Griffiths, and Katherine Heller. Reconciling meta-learning and continual learning with online mixtures of tasks. December 2018. URL http://arxiv.org/abs/1812.06080.

Chang-Jin Kim, Charles R Nelson, and Richard Startz. Testing for mean reversion in heteroskedastic data based on gibbs-sampling-augmented randomization. *Journal of Empirical finance*, 5(2): 131–154, 1998.

Diederik P Kingma and Max Welling. Auto-Encoding variational bayes. December 2013. URL http://arxiv.org/abs/1312.6114v10.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114 (13):3521–3526, 2017.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International Conference on Machine Learning*, pp. 5637–5664. PMLR, 2021.

Rahul G Krishnan, Uri Shalit, and David Sontag. Deep kalman filters. November 2015. URL http://arxiv.org/abs/1511.05121.

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

Kimin Lee, Kibok Lee, Honglak Lee, and Jinwoo Shin. A simple unified framework for detecting out-of-distribution samples and adversarial attacks, 2018.

Vincenzo Lomonaco and Davide Maltoni. Core50: a new dataset and benchmark for continuous object recognition. In *Conference on Robot Learning*, pp. 17–26. PMLR, 2017.

Davide Maltoni and Vincenzo Lomonaco. Continuous learning in Single-Incremental-Task scenarios. June 2018. URL http://arxiv.org/abs/1806.08568.

Michael McCloskey and Neal J Cohen. Catastrophic interference in connectionist networks: The sequential learning problem. In *Psychology of learning and motivation*, volume 24, pp. 109–165. Elsevier, 1989.

Anusha Nagabandi, Chelsea Finn, and Sergey Levine. Deep online learning via meta-learning: Continual adaptation for model-based rl, 2019. URL http://arxiv.org/abs/1812.07671.

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. URL http://arxiv.org/abs/1905.10681.

Abhijit Guha Roy, Jie Ren, Shekoofeh Azizi, Aaron Loh, Vivek Natarajan, Basil Mustafa, Nick Pawlowski, Jan Freyberg, Yuan Liu, Zachary Beaver, Nam Vo, Peggy Bui, Samantha Winter, Patricia MacWilliams, Gregory S. Corrado, Umesh Telang, Yun Liu, A. Taylan Cemgil, Alan Karthikesalingam, Balaji Lakshminarayanan, and Jim Winkens. Does your dermatology classifier know what it doesn't know? detecting the long-tail of unseen conditions. *CoRR*, abs/2104.03829, 2021. URL https://arxiv.org/abs/2104.03829.

Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. June 2016. URL http://arxiv.org/abs/1606.04671.

A Appendix

A.1 Environments

Regression: We use randomly initialized neural networks to create 10 different regression tasks. The input observation X is sampled uniformly at random in $[-1, 1]$ and $Y \sim \mathcal{N}(\mu_i(x); \sigma_i(x))$ for the i-th task, 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. Bhattacharyya distance) between the conditional distributions $P(Y|X)$ under different tasks to quantify task similarity. See Fig. 10 for dissimilarity across tasks.

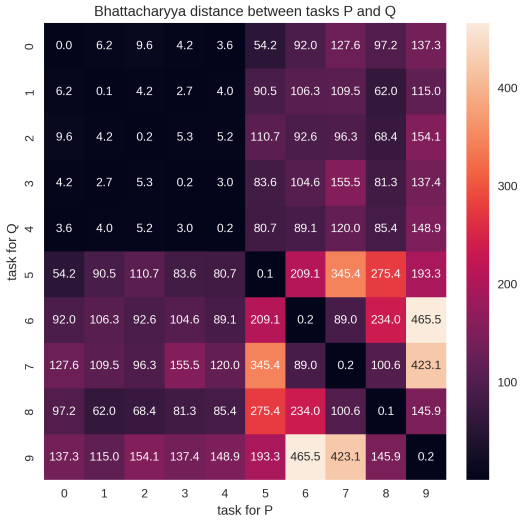

Figure 10: The pairwise Bhattacharyya distances between tasks, which connect intuitive qualitative differences between tasks with rigorous quantitative measures.

HalfCheetah Foot: We create different tasks in the HalfCheetah environment by clipping the action space of one or two actuators to one-third of the original. To generate the data, we rollout a policy trained in the standard HalfCheetah environment (using SAC (Haarnoja et al., 2018)) under the different task 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 another set of 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 $\mathcal{D}_u$. 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. [3]

A.2 Task partition

The task partitions in the experiment are listed in table 2.

A.3 Implementation details

For the MAML prior, we train an ensemble of size 4. Similar to Lakshminarayanan et al. (2017), we use all available task data but with random minibatches 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 $\eta = 10$. To optimize our model, we use Adam with learning rate 1e-4. During the online training, we alternatively update between the (i) (basis models) and (ii),(iii) (mixture network $w_\theta(Z)$,the prior task model $p_\theta(Z_t|Z_{t-1})$) to stabilize the learning. We trained on a 4-GPU machine.

[3]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 2: Task partition for experiments. *Note:* the tasks listed under segmented are present in *both* 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
WTI Daily Spot Price	1	2	3
	2	3	1
	3	1	2

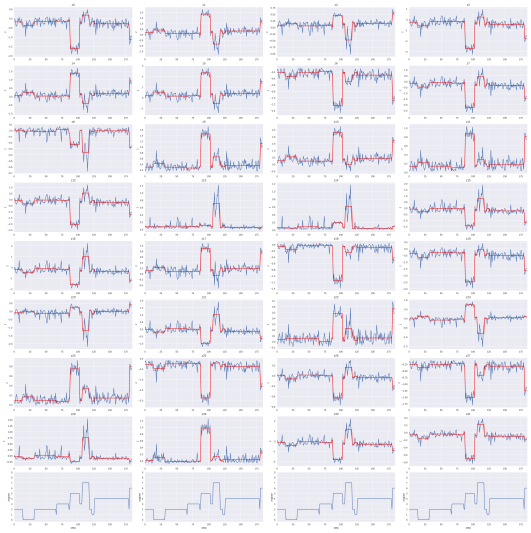

Figure 11: **Regression Latent Space:** Following the Offline training phase on the Regression domain, these are activations of all 32 Z components as the active task varies according to the task plots at the bottom. The plots in each column are aligned so the activations at any time t vertically line up across subplots with other activations and the active task at time t.

A.4 Latent space

For all four domains, we plots the latent space as the active task varies along the time and the task pairwise plot over PCA embedding of latent space (See Figures 8, 12, and 13).

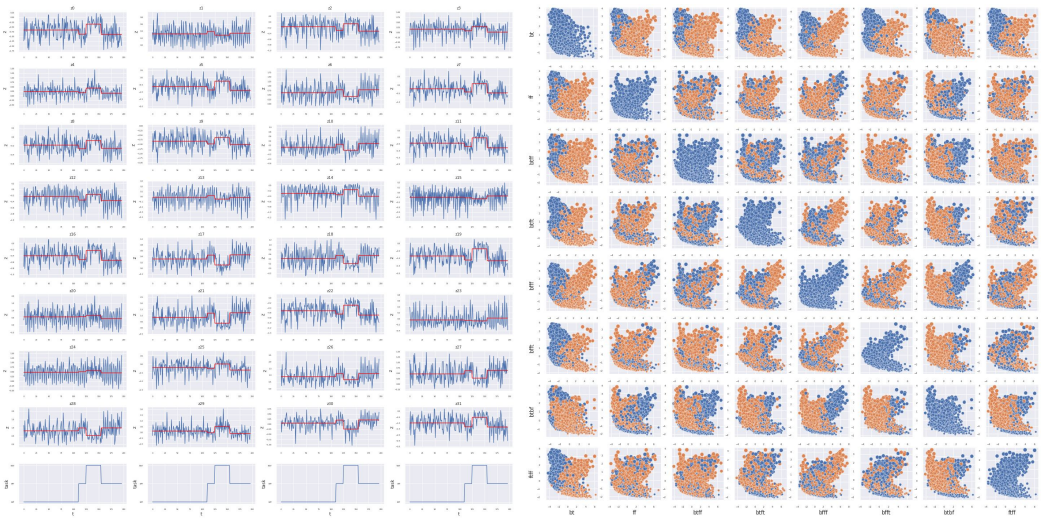

Figure 12: **Half-Cheetah Foot:** Left: Following the Offline training phase on the HalfCheetah foot, these are activations of all 32 Z components as the active task varies according to the task plots at the bottom. Right: 8x8 task pairwise plot over PCA embedding of latent space Z in HalfCheetah foot. 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*.

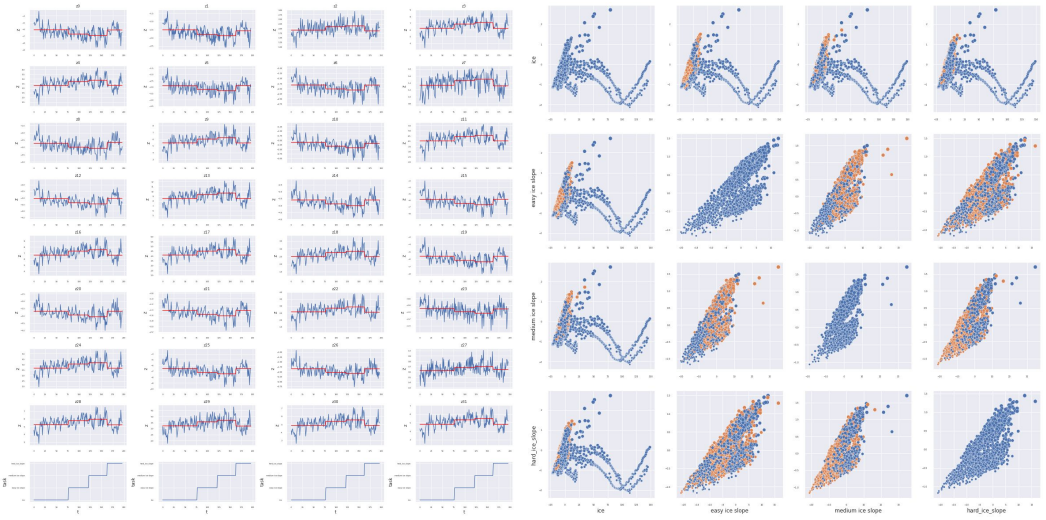

Figure 13: **Half-Cheetah Ice:** Left: Following the Offline training phase on the HalfCheetah ice-slope, these are activations of all 32 Z components as the active task varies according to the task plots at the bottom. Right: 8x8 task pairwise plot over PCA embedding of latent space Z in HalfCheetah 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.

A.4.1 DERIVATION OF ELBO

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

$$P(Z_t|X_{<t}, Y_{<t}, Z_{t'<t}) = P(Z_t|Z_{t-1}) \tag{7}$$

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.

$$P(Y_t|X_{\leq t}, Y_{<t}, Z_{\leq t}) = P(Y_t|X_t, Z_t) \tag{8}$$

Derivation of Variational Lower Bound: Let's assume we are given a stream of observations $\tau := (x_{0:T}, y_{0:T})$. [4] 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

$$\begin{aligned} D_{KL}(q||P) & \tag{9} \\ &= E_{Z_{\leq t} \sim q}\left[\log \frac{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}{P(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}\right] \\ &= E_{Z_{\leq t} \sim q}\left[\log \frac{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}{P(Z_{\leq t}, X_{\leq t}, Y_{\leq t})} \cdot P(X_{\leq t}, Y_{\leq t})\right] \\ &= E_{Z_{\leq t} \sim q}\left[\log \frac{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}{P(Z_{\leq t}, X_{\leq t}, Y_{\leq t})}\right] + \log P(X_{\leq t}, Y_{\leq t}) \tag{10} \end{aligned}$$

Rearranging terms,

$$\begin{aligned} \log P(X_{\leq t}, Y_{\leq t}) &= D_{KL}(q||P) + E_{Z_{\leq t} \sim q}\left[\log \frac{P(Z_{\leq t}, X_{\leq t}, Y_{\leq t})}{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}\right] \tag{11} \\ &\geq E_{Z_{\leq t} \sim q}\left[\log \frac{P(Z_{\leq t}, X_{\leq t}, Y_{\leq t})}{q(Z_{\leq t}|X_{\leq t}, Y_{\leq t})}\right] \tag{12} \end{aligned}$$

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

$$\log P(\tau) \geq \underbrace{E_{Z_{\leq T} \sim q}\left[\log \frac{P(Z_{\leq T}, x_{0:T}, y_{0:T})}{q(Z_{\leq T}|x_{0:T}, y_{0:T})}\right]}_{\mathcal{L}(\tau; P, q)} \tag{13}$$

We further assume a factored form of q (as in Krishnan et al. (2015))

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

Plugging into Eq (13), we get

$$\mathcal{L}(\tau; P, q) = E_{Z_{\leq T} \sim q}\left[\log \frac{P(Z_{\leq T}, x_{\leq T}, y_{\leq T})}{q(Z_0) \prod_{t'<t=1}^{T} q(Z_{t'<t}|Z_{t'<t-1}, x_{t'<t}, y_{t'<t}))}\right] \tag{14}$$

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

$$\begin{aligned} P(Z_{\leq T}, X_{\leq T}, Y_{\leq T}) &= \prod_t P(Z_t, X_t, Y_t|Z_{<t}, X_{<t}, Y_{<t}) \tag{15} \\ &= \prod_{t'<t} P(Y_{t'<t}|Z_{t'<t}, X_{t'<t}) P(X_{t'<t}) P(Z_{t'<t}|Z_{t'<t-1}) \tag{16} \end{aligned}$$

[4] **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

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

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

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

$$\begin{aligned}\mathcal{L}(\tau; P, q) = E_{Z_t \sim q}\left[\sum_{t=0}^{T} \log P(y_t|x_t, Z_t)\right] \\ - \sum_{t=1}^{T} E_{Z_{t-1} \sim q}\left[D_{KL}(q(Z_t|Z_{t-1}, x_t, y_t)||P(Z_t|Z_{t-1})\right] \\ - D_{KL}(q(Z_0)||P(Z_0))\end{aligned} \tag{17}$$

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.