2026 IEEE/SICE International Symposium on System Integration (SII)
January 11-14, 2026. Cancún, México

Sample-Efficient Robot Learning for Supervised Effect Prediction Tasks

Mehmet Arda Eren[1] Jan Babič[3] Erhan Oztop[1,2]

Abstract— **In self-supervised robot learning, data is acquired through active interaction with the environment, which is costly. Therefore, sample-efficient exploration is vital. To this end, intrinsic motivation (IM) methods such as learning progress (LP) have been adopted in robot learning, with variable success across tasks; whereas in machine learning, active learning (AL) is considered the go-to solution, especially for classification tasks. However, there is no systematic method for fusing both approaches for continuous regression tasks encountered in robot learning. To this end, we propose MUSEL (Model Uncertainty for Sample-Efficient Learning), a novel AL framework tailored for regression tasks in robotics, such as action-effect prediction. MUSEL introduces a novel model uncertainty metric that combines total predictive uncertainty, learning progress, and input diversity to guide experience gathering. We choose Stochastic Variational Deep Kernel Learning (SVDKL) as the base learning model and validate our approach by showing its efficacy in effect prediction tasks where a manipulator robot interacts with objects on a confined tabletop. Experiments comparing MUSEL with strong baselines show that MUSEL improves learning accuracy and sample efficiency. Overall, this study offers MUSEL as an effective online learning model applicable to any robot self-learning task where experience gathering is costly.**

I. INTRODUCTION

Learning action-effect relations by interacting with the environment, a form of self-supervised learning, can be used to equip robots with reasoning and planning skills [1], [18]. This often involves executing random actions and recording outcomes, which are then learned in a supervised fashion, effectively constructing a simplified world model [8]. However, relying on random action selection for learning about the world is highly sample-inefficient, and learning problem becomes more challenging, when learning in continuous action-state spaces is required [5], [21].

In this study we propose a novel active learning algorithm, Model Uncertainty for Sample-Efficient Learning (MUSEL), and show its application on the self-supervised learning of effect prediction tasks in continuous action spaces. MUSEL uses a Stochastic Variational Deep Kernel Learning (SVDKL) [24] to jointly estimate data-and-model uncertainty and track learning progress (LP) [16]. Combined with max-min distance between input samples [27], these signals approximate model uncertainty and work together to improve sample efficiency. To evaluate MUSEL, we test it in a simulated environment where a 7-DOF robot interacts with rigid body objects. Experiments are conducted in two scenarios involving one or two sphere to assess the sample efficiency of the model. To show the synergistic contribution of the components of MUSEL, ablation experiments are also conducted. The main contributions of this work are as follows:

- Development of an active learning method for action-effect prediction in continuous action and state settings.
- Extraction of model uncertainty from total uncertainty using learning progress and input diversity.
- Use of model uncertainty-based sample selection in robot action effect prediction learning.
- Demonstration of the applicability of our method to nontrivial robot self-supervised learning tasks.

II. RELATED WORK

A. Active Learning and Its Application in Robotics

Active learning (AL) [4] is characterized by deliberate sampling data to enhance model performance while reducing costs. AL algorithms mainly address supervised tasks, including both classification [6], [11] and regression [26], [27], but unsupervised variants also exist [11]. Yet, another categorization is based on the nature of the sampling space: *pool-based methods* use a finite set, while *population-based methods* operate over an infinite set [20]. We propose MUSEL as a population-based AL framework for continuous input-output type learning tasks that can be deployed for online robot learning.

Within robotics, AL has been applied to tasks such as environmental mapping, nonparametric shape estimation, control, perception and prediction, and distributed learning [12], [13], [21]. However, these methods often treat exploration and efficiency as separate concerns. Our approach addresses this through model-uncertainty-based selection, ensuring efficiency while encouraging exploration. Intrinsic motivation (IM) is also used to improve sampling efficiency ([14], [16]). A typical IM signal is learning progress (LP), which measures improvement in performance over time and has been shown to aid sample efficiency in robot learning [3], [17]. In MUSEL, we also make use of LP, but in a novel way to estimate data uncertainty.

B. Uncertainty Quantification

Uncertainty refers to the expected error or confidence level of a model's predictions, which is a central metric in AL for sample selection [28], [29]. Accordingly, learning models in AL are typically required to output uncertainties along with predictions [7], which applies to both classification [15],

[1] Mehmet Arda Eren and Erhan Oztop are affiliated with Ozyegin University, Istanbul, Turkey. (Corresponding Author: Mehmet Arda Eren)

[2] Erhan Oztop is also with Symbiotic Intelligent Systems Research Center, Institute for Open and Transdisciplinary Research Initiatives, Osaka University, Japan. (erhan.oztop@otri.osaka-u.ac.jp)

[3] Jan Babič is with Laboratory for Neuromechanics and Biorobotics, Department of Automatics, Biocybernetics and Robotics, Jožef Stefan Institute, Ljubljana, Slovenia. (jan.babic@ijs.si)

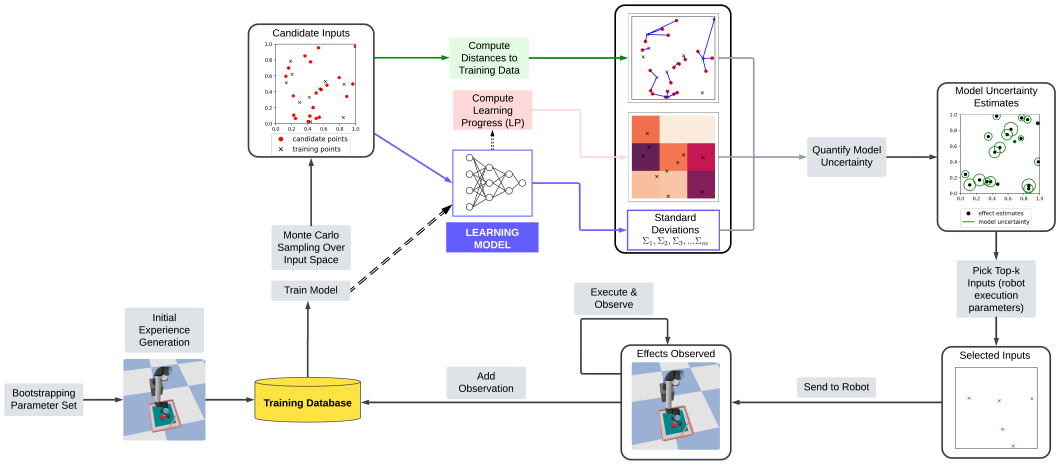

Fig. 1. The proposed model, MUSEL is illustrated. See text for details.

[19] and regression [7], [22] tasks. In regression, Gaussian Processes (GPs) are widely used for this purpose, modeling the predictive output as a Normal distribution, with the mean representing the prediction and the standard deviation representing the uncertainty [23]. However, its scalability is limited due to increased computational demands with larger datasets [22].

Stochastic Variational Gaussian Processes (SVGPs) [9] address this scalability issue by introducing virtual points, known as inducing points m, distributed over the input space and variationally optimized through stochastic (mini-batch) training via the Evidence Lower Bound (ELBO) [2]. Jointly optimizing $m \ll n$ inducing points, where n is the number of samples, reduces inference complexity from $\mathcal{O}(n^3)$ to $\mathcal{O}(nm^2)$ for training and to $\mathcal{O}(m^2)$ per-point for prediction, enabling SVGPs to scale to large or streaming datasets [9].

While standard GPs may struggle to capture complex high-dimensional patterns, Deep Kernel Learning (DKL) [25] uses a neural network to learn feature representations before applying the GP kernel, thus combining expressive deep features with the non-parametric flexibility of GPs. Stochastic Variational Deep Kernel Learning (SVDKL) [24] further builds on this by applying an SVGP to the learned features, with joint optimization of both the neural network and the GP via a mini-batch ELBO. This setup supports scalable training on large datasets while maintaining well-calibrated uncertainty estimates. In MUSEL, we employ SVDKL as the core learning engine, as explained in Methods.

III. Methods

A. Problem Formalization

In a typical action-effect prediction task, a robot is tasked to learn the *effects* of its *actions* through interaction with the surrounding environment. For this, the robot collects experience tuples of the form *(state, action)* $\rightarrow$ *effect*, where *state* describes the configuration of both the robot and the environment. While we present our method in the context of a specific example (see Section IV-A), the approach is generalizable to any effect prediction setting. For prediction, we represent the input, i.e., state-action pair, as $x = (s,a)$. Thus input set is defined as

$$\mathbb{X} = \left\{ \{x_t = (s_t, a_t)\}_{t=1}^{T},\ s_t \in S \subset \mathbb{R}^{\gamma}, a_t \in A \subset \mathbb{R}^{\zeta} \right\},$$

Here, t denotes the current step out of T total steps, and a_t is the action executed at step t. The pre-action state s_t (referred to as the *state* in our work) and a_t define the state space S and action space A. After each action, the robot observes a post-action state $s_{t'} = f(s_t, a_t)$, where $(s_t, a_t) \in \mathbb{X}$ and f is the environment transition function. As interaction continues, the dataset grows to T experiences. The target space $\mathbb{Y}$, associated with $\mathbb{X}$, consists of effect vectors y given by the difference between pre- and post-action states.

$$\mathbb{Y} = \{y_t \mid y_t = |s_{t'} - s_t|,\ y_t \in \mathbb{R}^{\gamma}\}_{t=1}^{T},$$

Having the growing sets of $\mathbb{X}$ and $\mathbb{Y}$ at a given instant of interaction, one can learn the mapping $\mathbb{X} \rightarrow \mathbb{Y}$ to capture the experienced action-effect relation so far.

B. Proposed Method: MUSEL

In this section, we explain our proposed framework, MUSEL, which aims to minimize the number of observation queries while learning an input-output mapping over continuous spaces without sacrificing prediction accuracy. At the core of MUSEL is the model uncertainty estimation method introduced in Section III-D, which is embedded in a loop comprising *execution sample selection, robot execution, model learning, and input sampling*, as illustrated in Figure 1 and detailed in Algorithm 1.

The MUSEL processing pipeline begins by generating m_{init} i.i.d. random state-action parameters to initialize the observation set $\mathbb{X}_O$. The robot executes these inputs and records the resulting effects as $\mathbb{Y}_O$ (Algorithm 1, lines 1-2). Subsequently, the model chosen for learning input-output relations with uncertainty estimates (i.e., SVDKL) is initialized. Each iteration starts with learning from the observed input-output samples $(\mathbb{X}_O, \mathbb{Y}_O)$. Then a candidate set $\mathbb{X}_C$ is generated by approximating the true set, sampling m_{cand} i.i.d. random points from the input space (Algorithm 1, lines 5–6). This step makes MUSEL a population-based active learning approach. Model uncertainty values U_{model} are then

Algorithm 1 Active Learning with MUSEL

Require: environment *Env*, n_{iter}, m_{init}, m_{cand}, k
Ensure: model $\mathbf{M}$, dataset $\mathbb{X}_O$, dataset $\mathbb{Y}_O$
1: $\mathbb{X}_O \leftarrow$ CREATESET(m_{init}) {Initialize the training dataset}
2: $\mathbb{Y}_O \leftarrow$ **ROBOT** : EXECUTEANDOBSERVE($\mathbb{X}_O$)
3: $\mathbf{M} \leftarrow \mathbf{M}$: INITIALIZE
4: **for** $i \leftarrow 1$ **to** n_{iter} **do**
5: $\quad \mathbf{M} \leftarrow \mathbf{M}$: TRAIN($\mathbb{X}_O, \mathbb{Y}_O$)
6: $\quad \mathbb{X}_C \leftarrow$ SAMPLEINPUTSPACE(m_{cand}) {(Re)create candidate inputs}
7: $\quad U_{\text{model}} \leftarrow$ ESTIMATEMODELUNCERTAINTY($\mathbb{X}_C, \mathbf{M}$)
8: $\quad X_C^* \leftarrow$ SELECTTOPK($\mathbb{X}_C, U_{\text{model}}, k$)
9: $\quad Y_C^* \leftarrow$ **ROBOT** : EXECUTEANDOBSERVE(X_C^*)
10: $\quad \mathbb{X}_O \leftarrow \mathbb{X}_O \cup X_C^*$
11: $\quad \mathbb{Y}_O \leftarrow \mathbb{Y}_O \cup Y_C^*$
12: **end for**
13: **return** $\mathbf{M}, \mathbb{X}_O, \mathbb{Y}_O$

estimated over $\mathbb{X}_C$, as described in Section III-D. Based on these estimates, candidate inputs are ranked, and the top k are selected as X_C^*. The corresponding state-action pairs are executed by the robot, their outcomes Y_C^* are collected, and both X_C^* and Y_C^* are added to the training dataset (Algorithm 1, lines 9–11). At the end of each iteration, if n_{iter} iterations have been completed, the system returns the trained model along with the collected samples. Otherwise, the next iteration begins with training on the updated dataset.

C. SVDKL Model Implementation

As reviewed in Section II-B, Stochastic Variational Deep Kernel Learning (SVDKL), uses a trainable neural network to map inputs into a rich feature space and an SVGP with a small set of inducing points to perform regression. Let $h_\phi : \mathbb{R}^d \to \mathbb{R}^\kappa$ be the neural network parameterized by ϕ. This network acts as a feature extractor by mapping each input $x_t = (s_t, a_t)$ to a κ-dimensional feature $z_t = h_\phi(x_t)$. An SVGP model is then placed on top of these features [24], containing M inducing inputs $Z = \{z_j\}_{j=1}^M$ (where $M \ll N$, and N is the number of samples) and corresponding latent outputs $u = f(Z)$. Since the true posterior $p(u \mid \{z_i, y_i\}_{i=1}^N)$ is intractable, we make a variational approximation by defining $q(u) = \mathcal{N}(u; m, S)$ [9], where $m \in \mathbb{R}^M$ and $S \in \mathbb{R}^{M \times M}$ represent the mean and covariance of $q(u)$. Thus, training and inference are conducted using information summarized by these M inducing points, rather than directly on the full dataset. We maximize the mini-batch ELBO to ensure that $q(u)$ explains the observed data while remaining close to the prior $p(u)$ [9]:

$$\mathscr{L}(\phi, m, S) = \sum_{i \in \mathscr{B}} \mathbb{E}_{q(u)}\big[\log p(y_i \mid f(z_i))\big] - \mathrm{KL}\big[q(u) \,\|\, p(u)\big], \quad (1)$$

where $\mathscr{B}$ denotes a random subset of training indices. Monte Carlo sampling through the GP head and stochastic gradient descent on (ϕ, m, S) enable end-to-end learning of both feature and GP parameters at scale [24]. This SVDKL backbone yields calibrated predictive means and variances for any input x, which are utilized for uncertainty-driven sample selection in the following section. Details of the model are provided in Section IV-A.

D. Estimating Model Uncertainty

The total uncertainty associated with a prediction is a compound of data and model uncertainties [7]. We take the *std* output of SVDKL for input x (i.e., a state-action pair), denoted $\sigma(x)$, as the total uncertainty $U(x)$. During the data collection process, only model uncertainty can be reduced by acquiring additional samples; therefore, our objective is to estimate this uncertainty to effectively guide sample selection. Following the approach in [10], we assume statistical independence between model uncertainty U_{model} and data uncertainty U_{data}, and thus write:

$$U(x) = \sigma(x) = U_{model}(x) U_{data}(x). \quad (2)$$

Since $\sigma(x)$ is predicted by SVDKL, if $U_{data}(x)$ can be estimated, then $U_{model}(x)$ can also be inferred. Regions where dense training data coincide with high total uncertainty, reflecting persistent error, indicate raised U_{data}. To exploit this, we define $\ell(x)$ as the minimum distance (MD) to the observed inputs, given by $\ell(x) = \min\{||x - x_k||\}$ where $x_k \in \mathbb{X}_O$. The maximum of this value is often used in active learning algorithms for sample selection to encourage diversity [27].

Likewise, as LP measures the decrease in prediction error obtained from the model, a low LP indicates saturated learning and suggests a high U_{data} relative to U_{model}. We assume that the model can learn the task, with its error metric decreasing over time during training to a desired level, but never perfectly, so the error remains nonzero. Since the input space is continuous, computing the LP for every individual input is infeasible. To address this, we make a locality assumption, and assign a single LP value to all points within a neighborhood. We use square neighborhoods (local regions) induced by partitioning the input space into equally spaced η-dimensional boxes.

We define r_i as the set of the training samples inside each region; at each step t. With SVDKL model, average RMSE errors for each r_i are computed with $E^t(r_i) = \frac{1}{|r_i|} \sum_{x' \in r_i} e(x')$, where $e(x')$ indicates the prediction error on the executed input x' from r_i. The last p values are kept and used to make a linear fit against $t = 1, 2, .., p$ so that $E^t(r_i) \approx \beta_i t + c_i$. With the following definition, LP is mapped to $[10^{-4}, 1]$, so that it reflects the (normalized) error reduction rate when the error is decreasing; otherwise, it is set to a small positive value to ensure mathematical consistency in Equation 4. When there are no samples inside a region for computing LP, it is set to 1 to encourage diversity. Thus we have:

$$\underset{x \in r_i}{LP(x)} = LP(r_i) = \begin{cases} 1, & \text{if } r_i = \emptyset \\ -\frac{2}{\pi} arctan(\beta_i), & \text{if } \beta_i \le 0 \\ 10^{-4}, & \text{otherwise} \end{cases} \quad (3)$$

Based on the inverse relationship between these effects, we propose to estimate data uncertainty with $U_{data}(x) = (\ell(x) LP(x))^{-1}$. By substituting it in U_{data} in Equation 2, the model uncertainty becomes

$$U_{model}(x) = \sigma(x) \ell(x) LP(x). \quad (4)$$

IV. EXPERIMENTS AND RESULTS

The environment for the learning experiments is designed as a confined table within the workspace of the robot. Objects

can bounce off the boundary walls according to the simulated physics. The table area is a square with sides measuring 40 cm. A diagonal wall spanning two adjacent edges is included to break spatial symmetry (see Figure 2).

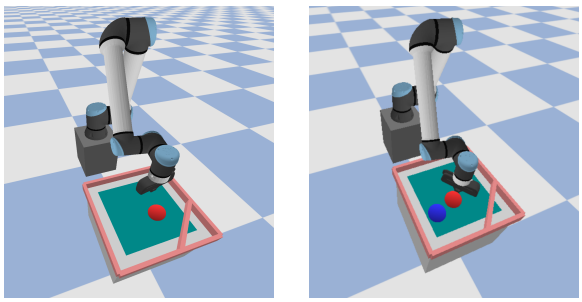

Fig. 2. (a) One-sphere setting: The target sphere (red) is placed at the location selected by the robot prior to the push action. The cyan region indicates allowed positions for the sphere. The diagonal wall is included to increase the task difficulty. (b) Two-sphere setting: This configuration is identical to (a) except for a second sphere (blue), which moves according to physical dynamics but is set to a fixed location before action execution.

A. Experiments Overview

The robot interacts with a "target" sphere of radius 3 cm placed on the table in both experimental settings (see Figure 2). The positions of the sphere along the axes parallel to the table are used as the state $s = s_{\mathrm{x}}, s_{\mathrm{y}} \in \mathbb{R}^2$ ($\gamma = 2$) (see Section III-A). The state space is restricted to a 30 cm $\times$ 30 cm square, smaller than the table to prevent wall collisions. The robot is equipped with a push action, $a = push(\alpha)$, where $\alpha \in [-\pi/3, \pi/3] \subset \mathbb{R}$ ($\zeta = 1$). To execute this action, the robot end effector first moves to p_{start} and then strikes the object by moving to p_{end}, with an end-effector translation of $r = 5$ cm as follows:

$$\begin{aligned} p_{start} &= (pos_{\mathrm{x}} - \mathrm{r}\cos\alpha,\ pos_{\mathrm{y}} - \mathrm{r}\sin\alpha) \\ p_{end} &= (pos_{\mathrm{x}} + \mathrm{r}\cos\alpha,\ pos_{\mathrm{y}} + \mathrm{r}\sin\alpha). \end{aligned} \tag{5}$$

At each time step t, the robot records the object's initial position s_t, executes push action a_t, and collects the final position once the object comes to rest. The simulation is queried for the position of ball giving a noise-free reading. Then the difference between the ball's final and initial positions is used as the effect of the action $y_t = \{\delta_{\mathrm{x}}, \delta_{\mathrm{y}}\}$. Then with the input $x_t = \{\sin\alpha, \cos\alpha, pos_{\mathrm{x}}, pos_{\mathrm{y}}\}$, this pair $\{x_t, y_t\}$, is added to the observation set (Algorithm 1, lines 10–11). As standard, α is encoded with $(\sin\alpha, \cos\alpha)$ to avoid input discontinuities over the range over its range. In our experiments, sample selection includes both state and action (Algorithm 1, line 7). In the second experiment, an additional sphere is introduced at a fixed initial position but is allowed to move freely after collisions, making the task dynamics richer. The robot is not allowed to directly manipulate the second sphere (see Figure 2b). Its position is excluded from the input and output spaces but constrains the initial placement of the first sphere to avoid collisions.

To instantiate MUSEL, the following meta-parameters are used: the number of AL iterations, $n_{iter} = 3000$; the initial training dataset size $m_{init} = 1$; and, for candidate inputs, $m_{cand} = 500$ and $k = 1$ representing the numbers of generated and selected inputs, respectively. LP regions are defined as a $10 \times 7 \times 7$ grid in the $\alpha \times pos_{\mathrm{x}} \times pos_{\mathrm{y}}$ input space. In SVDKL, the feature extractor is a three-layer MLP with dimensions $4 \times 32 \times 64 \times 32$, using ReLU activations after every layer except the last; the SVGP part utilizes $M = 20$ inducing points and an RBF kernel. We implement a prioritized training scheme, selecting the $m_{\mathrm{train}} = 2000$ least-used samples for the current training epoch to ensure uniform utilization of the collected data. A learning rate of 5×10^{-3} is chosen empirically and used throughout our experiments.

All experiments evaluate each method in ten independent runs with different random seeds. Evaluation is conducted on test sets uniformly sampled over $25 \times 20 \times 20$ and $20 \times 25 \times 25$ grids for the one-sphere and two-sphere settings, respectively, with grid dimensions given by $\alpha \times pos_{\mathrm{x}} \times pos_{\mathrm{y}}$.

1) LP Values: To analyze the relationship between LP and U_{data}, the evolution of LP values in representative regions of the one-sphere task is monitored using a sampling strategy based solely on LP (see Figure 3). The results show that the LP values tend to decrease over time, as expected (see Section III-D). LP in boundary regions (blue curves in (a) and (d); orange in (c)) remains elevated longer than in central regions, indicating higher U_{model}. Furthermore, the observed general trend is that LP approaches zero by 500 iterations, matching the model convergence (Figure 4). These findings align with the assumption that lower LP values indicate a greater likelihood of irreducible error. It is worth noting, however, the high variability between consecutive values suggests that LP should be supplemented with other metrics for more accurate data uncertainty estimation, as discussed in Section III-D.

B. One-Sphere Interaction Experiments

To assess the efficacy of model uncertainty-based sampling in robot self-learning, two comparative tests were conducted. In each test, the sampling strategy was replaced with the one under evaluation (line 7 of Algorithm 1). Both evaluations included the full MUSEL framework and a random sampling baseline, in which the prioritized top-k selection was replaced by drawing $k = 1$ i.i.d. sample from the input space. The first evaluation compares the performance of our model uncertainty estimation method against each of its individual components, using each as the sole sampling approach; some of which are state-of-the-art for finite input spaces, e.g., GSx [27]. The second evaluation involves ablation experiments to quantify the impact of individually removing each component from MUSEL.

Figure 4 summarizes the learning performance of all tested methods from both evaluation types. The RMSE for each method is plotted against the number of training iterations to illustrate sampling efficiency.

Comparisons with singleton methods. In the single-measure sampling comparisons (Figure 4, top), σ-only selection yielded the lowest performance, being surpassed even by random sampling, which highlights the downside of relying solely on model prediction variance estimation. LP sampling performed marginally better than random selection due to

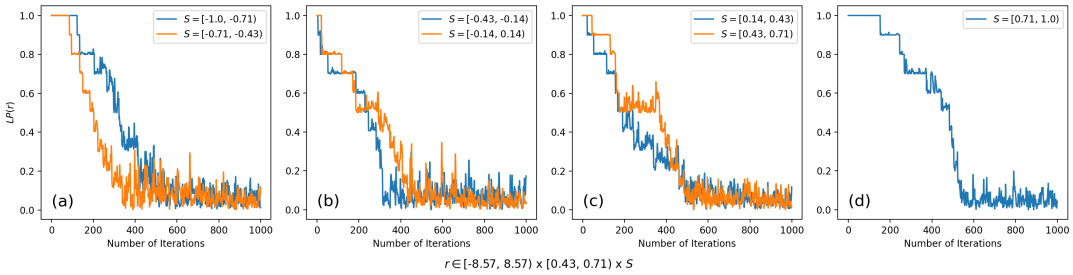

Fig. 3. Typical LP profiles of illustrative regions defined with $\alpha, pos_x, pos_y \in [-8.57, 8.57] \times [-0.43, 0.71) \times S$ are shown, where S is indicated in the legend of each LP curve. For ease of comparison some LP curves are superimposed.

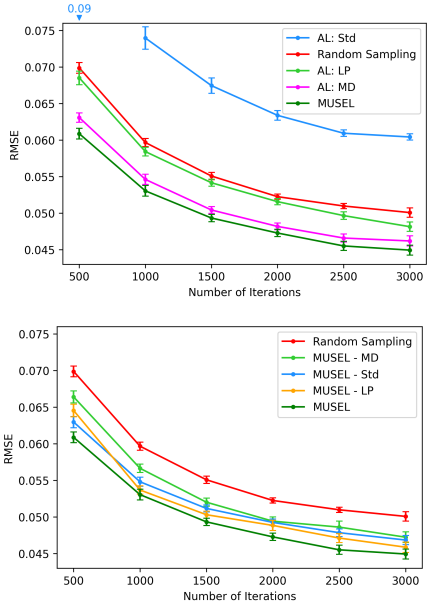

Fig. 4. Top: RMSE values obtained with single-measure-based and random sampling are contrasted with those of MUSEL as a function of sampling iterations. Error bars indicate the standard error of the mean (SEM). Bottom: Performance of MUSEL is contrasted with when individual sampling components are ablated from it.

its coarse region-level granularity: with identical LP values within each, samples are chosen randomly from the one with the highest LP. This suggests that LP should be combined with other methods for substantially better performance. MD-based selection shows performance close to MUSEL, both achieving significantly low RMSE values compared to other methods ($p < 0.05$). Nevertheless, MUSEL exhibits a statistically significant advantage at the 500th iteration.

Computational complexity of the methods. In addition to accuracy, Table I reports the time and memory requirements of the evaluated methods to rate samples, excluding sample-selection overhead. Average per-iteration computational costs of each method were evaluated on the one-sphere task using a PC equipped with an AMD Ryzen 5 7600X 6-core CPU, an NVIDIA GeForce RTX 4070 Ti GPU, 16 GB RAM, and Linux OS. Here, the σ-based method requires 2.14 ms and 3 kB of memory per iteration. The LP-based method is the fastest, taking only 0.02 ms but using 52.4 kB, while the MD-based method achieves 0.25 ms with 4.1 kB. Thus, LP attains the highest speed at the expense of $\sim 13\times$ greater memory usage compared to σ and MD-based methods. Thus, in total, MUSEL sample selection takes 2.41, ms and uses 59.5, kB of memory per iteration on the one-sphere task.

TABLE I

TIME AND MEMORY COMPLEXITIES OF THE EVALUATED METHODS.

Method	Time Complexity	Memory Complexity
σ-based	$\mathcal{O}\big(I(C^2 T_{NN} + Cm^2 + m^3)\big)$	$\mathcal{O}\big((C+kI)d\big)$
LP	$\mathcal{O}(kI^2 + IR)$	$\mathcal{O}(IR)$
MD	$\mathcal{O}(CdkI^2)$	$\mathcal{O}\big((C+kI)d\big)$

Note: $T_{NN} = \sum_{\ell=1}^{L} n_{\ell-1} n_\ell$ denotes the multiplication count for an MLP with layer sizes $(n_0, \ldots, n_L)$; C is the candidate set size, d the input dimension, k the top-k samples per step, I the number of iterations, m the number of inducing points in SVDKL, and R the number of regions in the LP grid.

Sampling patterns. When the sampling patterns of the methods are examined, random sampling, as expected, produces a uniform distribution. In contrast, alternative methods such as σ-based, LP-based, MD-based, and MUSEL tend to concentrate samples near the boundaries of the state space, where system behavior is more complex due to ball–boundary interactions. The extent of this boundary focus influences whether the task is fully learned. Among these, the σ-based method overemphasizes boundaries, whereas the LP-based method tends to overlook them. In contrast, MUSEL and MD-based approaches achieve more balanced sampling, with MUSEL distributing samples more evenly across the sampling space (e.g., boundary counts at iteration 1000: 189.5 for MUSEL vs. 221.0 for MD; at iteration 3000: 487.3 vs. 562.5). This balanced sampling results in an improvement in MUSEL's learning efficiency.

Ablation experiments. The ablation study revealed that MD contributed most to sample efficiency; however, even without it, the ablated model maintained a statistically significant advantage over Random Sampling ($p < 0.05$) across all learning iterations. The performance of the MD-ablated model differed significantly ($p < 0.05$) from MUSEL, except at the final iteration. Nevertheless, the MD-ablated MUSEL maintained a better learning curve than models using σ- or LP-based selection alone, indicating a synergistic effect between LP and σ. Removing LP from MUSEL caused a minor decline, significant only at the 500th iteration. This is likely due to LP's coarse neighborhood granularity.

C. Two-Sphere Interaction Experiments

In this section, we compare MUSEL with random sampling and MD based sampling in a dynamically richer environment. Figure 5 shows that MUSEL and MD-based sampling achieve significantly higher sample efficiency compared to random sampling, paralleling the findings from the one-sphere experiments (see Figure 4). An interesting observation is that the improvement over random selection is even more pronounced in the two-sphere interaction task,

with statistical significance ($p < 0.05$) at iterations 500, 1000, 2000, and 3000, underlining that complex learning tasks benefit more from guided sample selection. Although MD-based sampling–the closest competitor in the one-sphere task (see Section IV-B)–outperforms random selection and remains competitive, MUSEL consistently delivers better results, with a wider margin than in the one-sphere task, particularly during the first half of training. These findings demonstrate the improved sample efficiency of MUSEL over MD-based sampling in more complex tasks.

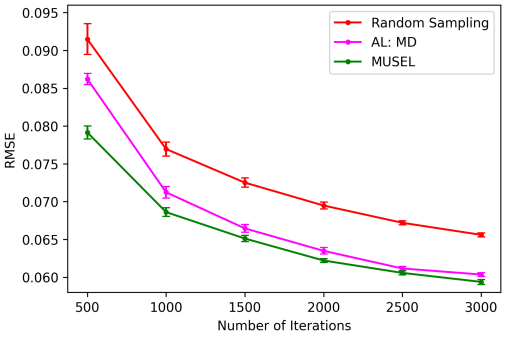

Fig. 5. Comparison of MUSEL with MD-based and random sampling strategies is presented in the two-sphere interaction task. The test settings follows the same convention used in the one-sphere task (Figure 4).

V. Conclusion

In this study, we introduced MUSEL (Model Uncertainty for Sample-Efficient Learning), a novel sample-efficient active learning framework for robot self-supervised learning. A model-uncertainty-based sample selection strategy is introduced through the novel use of learning progress (LP) and minimum-distance (MD) metrics, enabling MUSEL to adaptively choose the most informative actions and address the high costs associated with extensive data collection in robot learning. The simulation experiments showed that, when compared with the other baselines, our strategy achieves the lowest RMSE values during learning. Importantly in the more complex task, MUSEL shows early convergence, which further adds to its suitability for robot learning tasks. These results indicate that MUSEL is better suited for higher-dimensional scenarios, whereas MD may perform competitively in low dimensional problem as input diversity alone is often sufficient to cover the input space effectively.

Future work will address testing MUSEL in higher-dimensional and more complex tasks with added noise to assess its scalability and robustness. In particular, testing on physical robotic setups would demonstrate its real-world applicability in terms computational requirements as well as sensor and actuator noise tolerance.

Acknowledgment

This work was supported by JSPS KAKENHI Grants JP25K03159, JP25H01236, and JP23K24926, and EU Horizon Europe Grant 101120408.

References

[1] Ahmetoglu, A., Seker, M.Y., Piater, J., Oztop, E., Ugur, E.: Deepsym: Deep symbol generation and rule learning for planning from unsupervised robot interaction. J. Artif. Intell. Res. **75**, 709–745 (2022)

[2] Blei, D.M., Kucukelbir, A., McAuliffe, J.D.: Variational inference: A review for statisticians. Journal of the American statistical Association **112**(518), 859–877 (2017)

[3] Bugur, S., Oztop, E., Nagai, Y., Ugur, E.: Effect regulated projection of robot's action space for production and prediction of manipulation primitives through learning progress and predictability-based exploration. IEEE Trans. Cogn. Dev. Syst. **13**(2), 286–297 (2019)

[4] Cohn, D.A., Ghahramani, Z., Jordan, M.I.: Active learning with statistical models. Journal of Artif. Intell. Research **4**, 129–145 (1996)

[5] van Deursen, M.: Population-based Active Learning for Black-Box Regression. Master's thesis, Delft University of Technology, Delft, Netherlands (October 2020)

[6] Fu, Y., Li, B., Zhu, X., Zhang, C.: Active learning without knowing individual instance labels: a pairwise label homogeneity query approach. IEEE Trans. Knowl. Data Eng. **26**(4), 808–822 (2013)

[7] Gawlikowski, J., Tassi, C.R.N., Ali, M., Lee, J., Humt, M., Feng, J., Kruspe, A., Triebel, R., Jung, P., Roscher, R., et al.: A survey of uncertainty in deep neural networks. arXiv:2107.03342 (2021)

[8] Ha, D., Schmidhuber, J.: World models. arXiv preprint arXiv:1803.10122 (2018)

[9] Hensman, J., Fusi, N., Lawrence, N.D.: Gaussian processes for big data. arXiv preprint arXiv:1309.6835 (2013)

[10] Lee, H., Lee, S., Song, B.C.: Data and model uncertainty aware salient object detection. IEEE Access (2024)

[11] Liu, Z., Jiang, X., Luo, H., Fang, W., Liu, J., Wu, D.: Pool-based unsupervised active learning for regression using iterative representativeness-diversity maximization (irdm). Pattern Recognition Letters **142**, 11–19 (2021)

[12] Lluvia, I., Lazkano, E., Ansuategi, A.: Active mapping and robot exploration: A survey. Sensors **21**(7), 2445 (2021)

[13] Maeda, G., Ewerton, M., Osa, T., Busch, B., Peters, J.: Active incremental learning of robot movement primitives. In: Conference on Robot Learning. pp. 37–46. PMLR (2017)

[14] Marshall, J., Blank, D., Meeden, L.: An emergent framework for self-motivation in developmental robotics. In: Proceedings of the International Conference on Development and Learning (2004)

[15] Mena, J., Pujol, O., Vitrià, J.: A survey on uncertainty estimation in deep learning classification systems from a bayesian perspective. ACM Computing Surveys (CSUR) **54**(9), 1–35 (2021)

[16] Oudeyer, P.Y., Kaplan, F., Hafner, V.V.: Intrinsic motivation systems for autonomous mental development. IEEE transactions on evolutionary computation **11**(2), 265–286 (2007)

[17] Say, H., Oztop, E.: A model for cognitively valid lifelong learning. In: IEEE Int. Conf. on Robotics and Biomimetics (ROBIO) (2023)

[18] Seker, M.Y., Tekden, A.E., Ugur, E.: Deep effect trajectory prediction in robot manipulation. Robot. Auton. Syst. **119**, 173–184 (2019)

[19] Sensoy, M., Kaplan, L., Kandemir, M.: Evidential deep learning to quantify classification uncertainty. Advances in neural information processing systems **31** (2018)

[20] Sugiyama, M., Nakajima, S.: Pool-based active learning in approximate linear regression. Machine Learning **75**, 249–274 (2009)

[21] Taylor, A.T., Berrueta, T.A., Murphey, T.D.: Active learning in robotics: A review of control principles. Mechatronics **77**, 102576 (2021)

[22] Tripathy, R., Bilionis, I., Gonzalez, M.: Gaussian processes with built-in dimensionality reduction: Applications to high-dimensional uncertainty propagation. J. Comput. Phys. **321**, 191–223 (2016)

[23] Williams, C., Rasmussen, C.: Gaussian processes for regression. Advances in neural information processing systems **8** (1995)

[24] Wilson, A.G., Hu, Z., Salakhutdinov, R.R., Xing, E.P.: Stochastic variational deep kernel learning. Advances in neural information processing systems **29** (2016)

[25] Wilson, A.G., Hu, Z., Salakhutdinov, R., Xing, E.P.: Deep kernel learning. In: Artif. Intell. and Statist. pp. 370–378. PMLR (2016)

[26] Wu, D.: Pool-based sequential active learning for regression. IEEE Trans.Neural Netw. Learn. Syst. **30**(5), 1348–1359 (2018)

[27] Wu, D., Lin, C.T., Huang, J.: Active learning for regression using greedy sampling. Information Sciences **474**, 90–105 (2019)

[28] Yang, Y., Loog, M.: Active learning using uncertainty information. In: IEEE Int. Conf. on Pattern Recognition (ICPR). pp. 2646–2651 (2016)

[29] Yang, Y., Ma, Z., Nie, F., Chang, X., Hauptmann, A.G.: Multi-class active learning by uncertainty sampling with diversity maximization. International Journal of Computer Vision **113**, 113–127 (2015)