Bayesian Optimization for Sample-Efficient Policy Improvement in Robotic Manipulation

Adrian Röfer[1], Iman Nematollahi[1], Tim Welschehold[1], Wolfram Burgard[2], Abhinav Valada[1]

***Abstract*— Sample efficient learning of manipulation skills poses a major challenge in robotics. While recent approaches demonstrate impressive advances in the type of task that can be addressed and the sensing modalities that can be incorporated, they still require large amounts of training data. Especially with regard to learning actions on robots in the real world, this poses a major problem due to the high costs associated with both demonstrations and real-world robot interactions. To address this challenge, we introduce BOpt-GMM, a hybrid approach that combines imitation learning with own experience collection. We first learn a skill model as a dynamical system encoded in a Gaussian Mixture Model from a few demonstrations. We then improve this model with Bayesian optimization building on a small number of autonomous skill executions in a sparse reward setting. We demonstrate the sample efficiency of our approach on multiple complex manipulation skills in both simulations and real-world experiments. Furthermore, we make the code and pre-trained models publicly available at http://bopt-gmm.cs.uni-freiburg.de.**

I. Introduction

Efficient methods for learning new manipulation motions in a fast and reliable manner is still an open area of research in robotics. Behavioral Cloning (BC) has become the state-of-the-art technique to address this problem [1] and shows impressive results, both in the variety of skills that are trainable, as well as the types of input modalities [2]–[6], be it robotic proprioception, camera data, or language. Although they are more efficient than pure reinforcement learning (RL), these approaches still require many demonstrations (100+) to achieve high success rates. Far more data efficient are the approaches that fit a parameterized model of the robotic skill from data. Dynamical systems fall into this category and have been shown to be able to generate physically plausible motions that provide a high level of reactivity and robustness against perturbations in the environment [7]–[10] unlike their neural network based counterparts. These dynamical systems can be trained from a handful of demonstrations, in some cases, using certain constraints [8], even using a single one. While their sample efficiency is a great strength, it is by no means a guarantee for the model's quality. Thus, it remains an open challenge to update these models given, ideally sparse, environmental feedback.

In our previous work [11], we addressed this challenge. Given an initial dynamical system model, in our case a Gaussian Mixture Model (GMM), we trained a Soft-Actor-Critic agent, proposing updates to the dynamical system at a fixed step interval based on sensor data. We call this fusion of SAC and GMM *SAC-GMM*. We demonstrated that our approach boosts the dynamical systems' performance to $80+\%$ after around 500 episodes of autonomous exploration. While this is relatively efficient in the domain of RL, there is still the need to further reduce the samples required for model improvement.

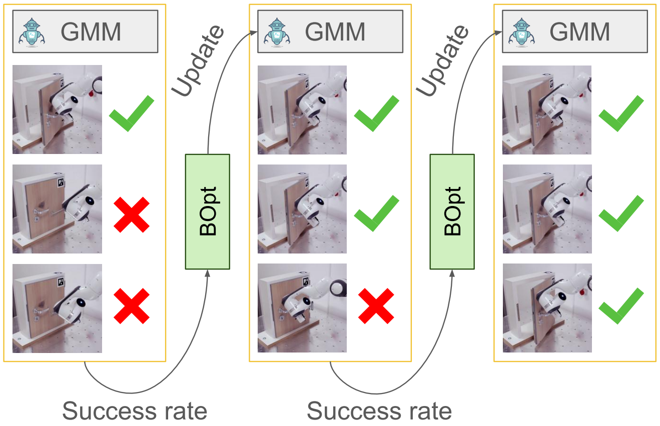


Fig. 1: We propose a simple but effective interpretation of a reinforcement learning problem as black-box optimization of a policy. The policy, encoded as a GMM, is evaluated to measure its accuracy. From this new measurement, the optimizer can regress a new improved update.

If we view the problem of learning a policy not as learning a step-wise action given an observation, but rather as optimizing the value of a very costly black-box function, a perspective examined in [12], we can leverage methods from the domain of black-box optimization to further improve sample efficiency. In this work, we propose BOpt-GMM as a fusion of sample efficient Bayesian Optimization and a GMM base policy model, as schematically represented in Fig. 1. Bayesian Optimization (BOpt) is often used for hyperparameter search in machine learning, where the evaluation of a possible set of parameters is very expensive and thus highly geared towards drawing as few samples as possible. Previous approaches [13]–[15] have used BOpt in an RL setting. Different from our approach, these works rely on predefined motion primitives with a low number of additional parameters over which they optimize, which is of very similar complexity to hyperparameter search. In our case, we leverage BOpt to search the high-dimensional space of a multivariate GMM. The question arises as to how updates in this trajectory representation can be carried out efficiently and in a physically sound manner. Addressing these points, in this paper we make the following contributions:

- We frame a sparse RL-setting as black-box optimization of a GMM policy model.

[1] Department of Computer Science, University of Freiburg, Germany.
[2] Department of Eng., University of Technology Nuremberg, Germany.
This work was funded by the BrainLinks-BrainTools center of the University of Freiburg.

- We propose two effective, low-dimensional update methods for GMM encoded policies, which reduce the parameter space independent of the optimization scheme. We demonstrate their applicability to our optimization approach as well as a reinforcement learning baseline.
- We evaluate our proposed approach thoroughly in simulation as well as the real world and demonstrate a significant improvement in sample efficiency.
- We make the code and pre-trained models publicly available at http://bopt-gmm.cs.uni-freiburg.de.

II. Related Work

Learning from human demonstrations also called *imitation learning*, is an approach that has been exploited for more than two decades [16]. Its overarching goal is to learn to reproduce the actions demonstrated by a human, either externally or through teleoperation of the robot [17]–[19]. It has been deployed successfully in both robotics and autonomous driving [20], using this approach robots have been enabled to learn many household skills [21]–[24] and, lately, this technique has even been used to train large vision and language conditioned transformer models [25] to enable long-horizon manipulations in LLM-based agents [26]. However, training deep neural network policies from demonstrations requires hundreds or thousands of demonstrations, even when the approach is geared towards efficiency. As we are looking to deploy robots in novel environments and on novel tasks, this need for data becomes a limiting factor.

Alternatively, motions can also be learned from fewer demonstrations by encoding them in low-dimensional models. Established examples are *dynamic motion primitives* (DMP) [9] and Gaussian Mixture Models (GMM) [8]. DMPs encode motions as a collection of attractors and repulsors, while GMMs encode a trajectory more directly as the statistical correlation of system state and its first-order derivative. Both can be used to learn from few human demonstrations as has been demonstrated in [8], [10], [27]–[29], with both methodologies being applied successfully in longer complex tasks [15], [30]. However, GMMs are the more common choice for low-dimensional imitation learning, due to their ability to encode more varied trajectories than DMPs. While it is still a rarity, in our previous work [11] we demonstrated how high-dimensional sensor information can be used in combination with GMMs to enable fast and efficient learning of reactive policies from very few demonstrations. However, the number of exploration episodes needed to perfect the policy is still quite high.

Using Bayesian Optimization methods for policy improvement is not a new approach. BOpt has been used successfully in finding threshold parameters for walking gaits [31], [32] as well as efficient impact-compensating arm movements for balancing robots [33]. In [34] the authors demonstrate sample-efficient learning of a robotic policy in simple scenarios by learning function of the task dynamics as a Gaussian Process (GP). A similar technique is employed by [35], who additionally use a linear projection to reduce the number of policy parameters to. More flexible is the approach presented in [36] which uses an auto-encoding scheme to form a low-dimensional parameter space, trained on simulated trajectories. BOpt has also been used to decide sampling a simulation or a real world execution [37], reducing the number of expensive real world examples needed to improve a policy. Different to our proposed method all of these methods require a dense reward signal. Using binary signals in combination with BOpt has been studied as well, though less throughly. In [38], the authors formulate an approach to maximizing a stochastic binary reward by using a GP to model the parameters of a binomial distribution. While they do demonstrate that this model can be used to exploit such a function efficiently, we use a different approach to integrating binary feedback, as this model only lends itself to univariate scenarios.

Most closely related to the approach that we propose are [13]–[15]. Englert *et al.* [13] propose using BOpt to identify a low-dimensional task mapping for a constrained optimization problem which they initialize from a single demonstration. They continue to sample possible mappings and finally use their motion problem with the uncovered mapping to perform control to solve their tasks. With their approach, they can optimize policies for manipulating articulated objects. They reward their agent with the negative forces exerted during the interactions. While their data efficiency is impressive, the space of their mapping is extremely low-dimensional with no more than three dimensions. Johannsmeier *et al.* [14] model a manipulation skill as a chain of primitive controllers with learnable parameters for desired contact force and superimposed oscillations. They study the suitability of different optimizers for three example tasks and find BOpt to be sufficient, though it is outperformed by an evolutionary strategy. The low success of BOpt might be explainable with their 30-dimensional parameter space and rather large parameter range. In their setups, they derive a cost from the execution duration of a robotic skill. Wu *et al.* [15] follow up the efforts of [14], [39] by again employing BOpt to learn parameters for motion primitives. Their key change is encoding the demonstration trajectories as a GMM and incorporating the probability of the current trajectory under this distribution in the optimizer's objective function, enforcing a similarity of generated trajectories to the original demonstrations. This similarity function is the main reward signal for the optimizer, with only minor influence given to task success. Otto, Celik, *et al.* [40], [41] phrase a policy improvement problem on DMPs similar to ours, but use a deep function approximator which leads to a significant data requirement.

Our approach differs from the discussed works in three main points: 1) We do not assume the existence of predefined control primitives or motion models but learn these fully as reactive systems from demonstration data. 2) We optimize directly over our policy model which is a much larger parameter space. 3) We use a simple binary reward signal to guide our optimization and do not rely on measures that require prior knowledge of the task, such as trajectory length, or interaction forces. To the best of our knowledge, we are the first to use BOpt to optimize a GMM-encoded policy.

III. Problem Formulation

In this work, we consider a sparse reinforcement learning setting, in which a policy π_θ processing observations s_t, produces an action a_t, and receives a reward $r_t \in \mathbb{R}$ in return. The objective of the policy is to accumulate the maximum possible reward R for an episode. We assume the reward to be sparse, only given out at the end of an episode for either success or failure, i. e. $R \in \{0, 1\}$. We assume the policy π_θ to be parameterized under a space $\Theta \subseteq \mathbb{R}^m$ and assume the existence of an update function $\oplus$ which can be used to derive an updated policy $\pi_{\theta,i} = \pi_\theta \oplus \Delta\theta_i$. We strive to find the optimal update $\Delta\theta^*$ which will yield the optimal policy $\pi^* = \pi_\theta \oplus \Delta\theta^*$. Performance is assessed by a non-deterministic evaluation function $h_\theta(\Delta\theta, j) \to \mathbb{R}$ which executes the policy yielded by the update for j episodes and averages the rewards obtained by the executions. The overall objective is

$$\Delta\theta^* = \arg\max_{\Delta\theta} h_\theta(\Delta\theta, j), \tag{1}$$

where j is constant. Each evaluation of h_θ yields a data point $(\Delta\theta_i, R_i)$, which is collected in a dataset $\mathcal{D} = \{(\Delta\theta_1, R_1), \dots, (\Delta\theta_I, R_I)\}$. This dataset can be used in the search for $\Delta\theta^*$.

IV. BOpt-GMM Framework

To address the formulated problem, we introduce our approach BOpt-GMM. It consists of a gradient-free Bayesian optimizer which generates updates $\Delta\theta$ for the policy π_θ which is a dynamical system encoded as a Gaussian Mixture Model. In the following, we describe all three of these components in detail.

A. GMM

In this work, we examine the challenge of improving an initial robotic motion policy π_θ trained from a set of demonstrated trajectories Ξ_D. Our approach assumes motion to be driven by a dynamical system of the form $f_\theta(s) = \dot{s}$, encoded in a GMMs as parameterization of π_θ. Here $\dot{s}$ denotes the first-order derivative of the observable state $\mathbf{s}$. We follow the Dynamical System definition of [8], [27]. Given an observable system state $\mathbf{s} \in \mathbb{R}^m$, in our case the position of the robot's endeffector relative to a frame of reference, a GMM models the dynamics of this system as K components, weighted by $\omega \in [0, 1]^K$ with $\|\omega\| = 1$, each of which consists of a $2m$-dimensional mean μ^k, and a corresponding covariance matrix Σ^k, with

$$\omega \in [0, 1]^K, \quad \mu^k = \begin{bmatrix} \mu_s^k \\ \mu_{\dot{s}}^k \end{bmatrix}, \quad \Sigma^k = \begin{bmatrix} \Sigma_s^k & \Sigma_{s\dot{s}}^k \\ \Sigma_{\dot{s}s}^k & \Sigma_{\dot{s}}^k \end{bmatrix}. \tag{2}$$

At inference time, we use Gaussian Mixture Regression (GMR) to infer $\dot{s}_t$ from s_t as

$$\dot{\mathbf{s}}_t = f_\theta(s_t) = \sum_{k=1}^{K} h^k(s_t)(\mathbf{A}^k s_t + b^k) \tag{3}$$

with

$$\mathbf{A}^k = \Sigma_{s\dot{s}}^k (\Sigma_s^k)^{-1}, \quad b^k = \mu_{\dot{s}}^k - \mathbf{A}^k \mu_s^k \tag{4}$$

and $h^k(s_t)$ being the normalized probability of $p\,(k \mid s_t)$. For a more detailed understanding of the inference procedure, please refer to [8].

B. GMM Parameterization

Although GMMs are low-dimensional models compared to common neural network architectures, they do still hold too many parameters to be exposed directly in $\Delta\theta$. In addition, properties such as positive-definiteness of the components' covariances need to be preserved during the update integration $\pi_\theta \oplus \Delta\theta$. Thus, we are concerned with finding a small space $\Delta\Theta$, which preserves the necessary properties of the GMM. We propose to perform norm-preserving updates to the weights. The generated updates $\Delta\theta_{\omega,i}$ at step i are added to ω and normalized, while a small minimum activation ϵ is enforced for numeric stability

$$\omega_i^k = \frac{\max(\omega^k + \Delta\theta_{\omega^k,i}, \epsilon)}{\sum_j^K \max(\omega^j + \Delta\theta_{\omega^j,i}, \epsilon)}. \tag{5}$$

Updates for the means μ are integrated additively without any further post-processing.

The number of parameters in the covariances Σ^k is $(2 \cdot |S|)^2$ and thereby quadratic in the number of degrees of freedom $|S|$. Even by exploiting the symmetry of covariances and the fact that only Σ_s^k and $\Sigma_{s\dot{s}}^k$ but not $\Sigma_{\dot{s}}^k$ are relevant for inference, the number of parameters to estimate still is quadratic in $|S|$. Therefore, we are interested in a view of Σ^k which enables us to formulate a much lower-dimensional update space. We propose two schemes for updating the covariance which is linear in $|S|$. Our first scheme is based on eigenvalue decomposition as a natural lower-dimensional parameterization. Given the eigenvectors $\mathbf{Q}$ and diagonal matrix of eigenvalues $\mathbf{\Lambda}$, we form the updated covariance as

$$\Sigma_i^k = \mathbf{Q}\mathbf{\Lambda}\,\mathrm{diag}(\mathbf{\Delta}\theta_{\Sigma^k,i})\mathbf{Q}^{-1}, \tag{6}$$

with $\mathbf{\Delta}\theta_{\Sigma^k,i} \in [1 - \sigma, 1 + \sigma]$.

In our second update scheme, we have the optimizer produce updates $\mathbf{\Delta}\theta_{\Sigma^k,i} \in [-\sigma, \sigma]$ per component which encodes Euler rotations, which we integrate as

$$\Sigma_{x,i}^k = R_{XYZ}(\mathbf{\Delta}\theta_{\Sigma^k,i})\Sigma_x^k. \tag{7}$$

The general intuition behind these update rules stems from the interpretation of the covariance matrix in 3-dimensional space as an ellipsoid. In the first case, we assume that the direction of the correlation of the initial model is reasonable. By changing the eigenvalues of the decomposition, we restrict the optimization to a scaling of the ellipsoid axes. In the second case, the assumption is the opposite: we preserve the scaling and instead allow for a rotation of the axes. In both cases, the number of parameters is reduced to $|S|$. Note that these are only meaningful interpretations in task space and not in latent spaces such as joint space.

We also consider a rank-1 vector-base update for Σ_i^k based on a singular value decomposition and rank reduction. We find this update method to be less stable than the ones presented above, but we include a brief description of it for the interested reader in Sec. VI.

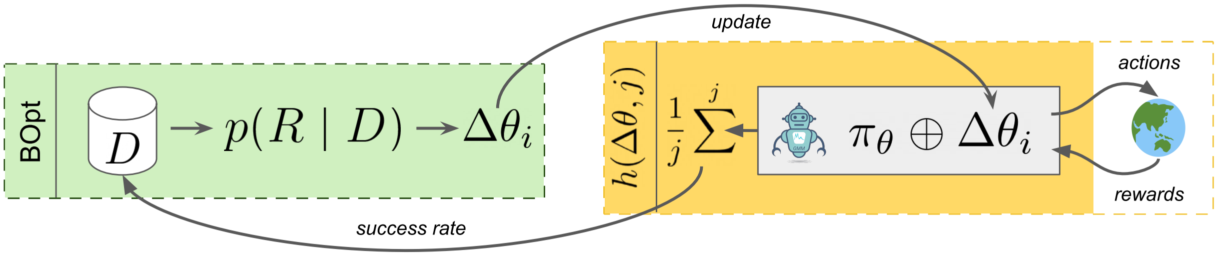

update
BOpt
D
$\rightarrow p(R \mid D) \rightarrow \Delta\theta_i$
$h(\Delta\theta, j)$
$\frac{1}{j}\sum^{j}$
$\pi_\theta \oplus \Delta\theta_i$
actions
rewards
success rate

Fig. 2: Our approach consists of two parts: A Bayesian optimizer estimating the value $p\left(\Delta\theta \mid D\right)$ and proposing potential new updates $\Delta\theta_i$. The second part is the evaluation function $h(\Delta\theta_i, j)$ which plays the update $\Delta\theta$ for j steps and averages the returns. The results are used to inform the optimizer.

C. Bayesian Optimization

Bayesian Optimization (BOpt) is the state-of-the-art technique for hyperparameter tuning in automated machine learning tasks. In this domain, its problem is formulated as finding a vector of hyperparameters $\mathbf{x} \in A$ which maximizes model performance as $\max_{x \in A} f(x)$, where A is typically a bounded hypercube in $\mathbb{R}^n$. Unlike model-free techniques such as grid search and random search, BOpt algorithms build a probabilistic surrogate function $p\left(y \mid \mathbf{x}, D\right)$, where D is a dataset of evaluated hyperparameter samples, as defined in Sec. III and $\mathbf{x}$ is a newly generated hyperparameter sample. The function's probabilistic nature allows the optimizer to explore the parameter space according to the expected value as estimated by the surrogate [42], [43]. A common surrogate implementation is to use Gaussian Processes (GP), however, these lend themselves mostly to lower-dimensional parameter spaces and are costly to evaluate on large datasets [42]. Random forests (RF) are a common alternative to using GPs in BOpt [42], [43]. By using multiple random trees as regressors and averaging their output, RFs are able to provide both an expectation and uncertainty estimate of a given $\mathbf{x}$, while being much more time and space-efficient than GPs. As our parameter space is quite large with up to 70 parameters, and our data becomes plentiful over time, we use RFs in this work.

New samples are evaluated on the basis of an acquisition function that rates potential new samples $\mathbf{x}$ according to the surrogate function. While there are many acquisitions functions, *expected improvement* [44] is criterion that is used most commonly [42]. In the automated ML literature, it is often pointed out that a drawback of BOpt is its sequential nature which bars it from parallelization. This does not concern us, as we are interested in improving a policy on a single robot, and thus have no opportunity to parallelize.

We connect BOpt to our problem by setting $A = \Delta\Omega$ and $f = h$, as defined in Sec. III. We include the averaging evaluation function h as a measurement function that measures the accuracy of a proposed sample for j episodes. While existing optimizers such as SMAC [45] provide functionality for optimizing stochastic functions, they do assume these to depend on a seed they provide. Since we cannot affect the state of the external world, we model h as a deterministic function and reduce the variance by selecting a sufficiently large j. We represent the process of surrogate update, sampling, and sample evaluation schematically in Fig. 2.

V. Experimental Evaluation

We evaluate our proposed approach BOpt-GMM in 3 simulated scenarios and their matching real-world counterparts, which are shown in Fig. 3. We use the simulated scenarios not only to contrast our approach with other baselines but also to evaluate the impact of our proposed covariance update schemes. In our real-world evaluation, we study if the well-performing optimizer configurations we have identified in simulation can also be applied to train and optimize policies directly on real robotic systems.

A. Experiment Setup

In all evaluation scenarios, we collect 10 demonstrations by teleoperation of the robot and fit a GMM to the data using Expectation Maximization (EM). We also explored the usage of the SEDS framework [8] but did not find the resulting GMM models to improve performance while introducing additional complexity in model fitting. Using these models as a starting point, we compare BOpt-GMM to two baselines:

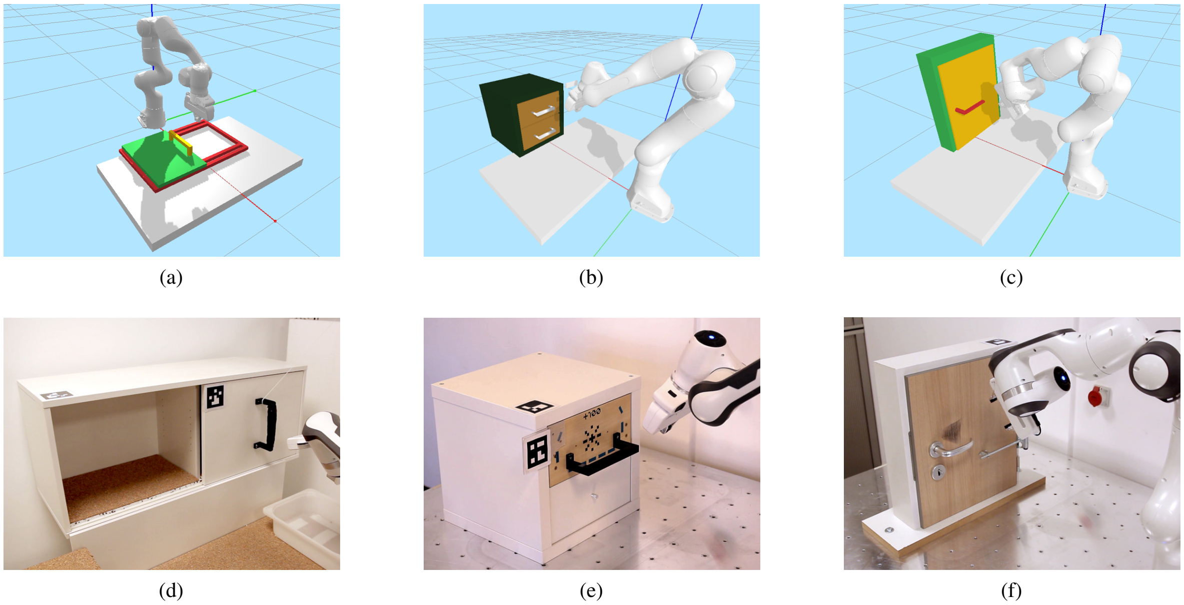
(1) In simulation, we introduce a naive *Online GMM* (OG) approach in which we add all successful trajectories to a growing dataset Ξ_O. We then refit π_θ to the full dataset $\Xi = \Xi_D \cup \Xi_O$ yielding our updated GMM $\pi_{\theta,i}$, where Ξ_D is the set of original demonstration trajectories.

(2) Our SAC-GMM approach [11] which learns an additional policy π_Δ which generates GMM-updates $\Delta\theta_t$ every n environment steps to dynamically update the GMM. In addition to the position of the end effector, SAC-GMM receives the wrench experienced by the robot at its wrist, as we found it to learn too slowly when using solely proprioceptive observations.

In the simulation, we also compare to a Behavior Cloning (BC) policy similar to [2] trained on the same initial demonstrations Ξ_D. As this baseline shows very limited performance, we train a variant *BC 100* on an extended demonstration set Ξ_D^{100}. We do not compare against plain SAC as we already determined in our previous work [11] that, due to the sparse reward setting, it does not learn any successful policy on the time horizon of interest to us. Since we are interested in both performance and training efficiency, we track two metrics: 1) the overall policy success rate; 2) the number of episodes taken to achieve 80% success rate.

B. Experiments in Simulation

We first evaluate our approach in the simulated scenarios depicted in Fig. 3. All of our scenarios (a-c) are manipulations



(a) (b) (c)

(d) (e) (f)

Fig. 3: (a) Simulated sliding of a horizontal hatch. The location and orientation of the hatch's frame are varied between episodes. (b) Simulated Drawer Opening. The location of the cabinet is varied in the XY plane. (c) Simulated opening of a door. The location of the door is varied per episode. The handle must be pressed to move the door. Real opening of a (d) sliding door, (e) drawer, (f) door.

of articulated objects and each poses a different challenge. The first scenario (a) requires the robot to open a sliding hatch. The robot starts above the hatch, has to loop behind the handle and push open the hatch. This task does not require greater precision but a looping steady motion. The second scenario (b) requires the robot to open a drawer. Therefore the robot must successfully hook the handle and move in the opening direction. This task requires greater precision for the hooking of the handle but once this has been achieved it is rather forgiving. Setting (c) is the most difficult of our scenarios. Opening the door requires precise and measured motions to successfully press and hook the handle to open the door. In all scenarios, we fit the GMMs to the relative location of the end-effector to the object. We vary the location of the object, requiring the agents to make the policy robust against variance in scenarios.

We collect 10 demonstrations in each scenario and fit a GMM for each setting. As the number k of Gaussians used has an impact on the performance of the model while also scaling the number of parameters to optimize, we perform evaluations with $k \in \{3, 5, 7\}$, which yield at most $30, 50$ and 70 parameters respectively. Throughout our experiments we use $j = 8$ evaluation episodes. We give both SAC-GMM and the Bayesian optimizer update ranges of

$$\begin{aligned} \Delta\theta_\omega &\in [-0.1, 0.1] \\ \Delta\theta_\mu &\in [-0.05, 0.05] \\ \Delta\theta_\Sigma &\in [-0.1, 0.1]. \end{aligned} \tag{8}$$

We use the Bayesian Optimization implementation from SMAC3 [45], a collection of mature gradient-free optimizers for black-box optimization. Specifically, we use the hyperparameter optimizer with logarithmic expected improvement as an acquisition function. In addition, we set SAC-GMM's learning rate as $2 \cdot 10^{-3}$. The baseline performances of these models are presented in Fig. 4.

The success rates achieved by the approaches are reported in Fig. 4. We find that both BOpt-GMM and SAC-GMM improve over the baseline performance of the GMM, independent of the specific k. The simple Online-GMM baseline is not reliable. While it is able to increase its success rate over the starting GMM in some cases, in others it deteriorates performance dramatically. Behavioral Cloning cannot be initialized successfully from the 10 episodes used to fit the GMMs. With an additional 90 demonstrations, it does start to achieve noticeable performance, however, this is not the sample efficiency we aim to achieve.

While we can see from Fig. 4 that our approach and SAC-GMM work well with any k, we choose $k = 5$ for a detailed analysis, as we have found that setting to be a good tradeoff between model performance and optimization/learning speed. For a detailed analysis, we present Tab. I where we compare the different GMM updates in SAC-GMM and BOpt-GMM. We find that BOpt-GMM achieves 80% success rate 40% (87 episodes earlier) faster than SAC-GMM. In Fig. 5, we present a qualitative comparison of the success rates of the two approaches over the training duration. On the other hand, we also we observe that SAC-GMM achieves overall higher performance than BOpt-GMM. With respect to our newly proposed covariance update strategies, we find them to achieve a much higher success rate on average in our 500 episode timeframe than updating only the means μ as done in previous work [11]. The combination of covariance update and means update, however, does not exceed this success rate, likely due to the larger number of parameters. A minor trend we seem to identify in our data is that the R_{XYZ} update performs better for SAC-GMM than the eig update, while this is reversed for

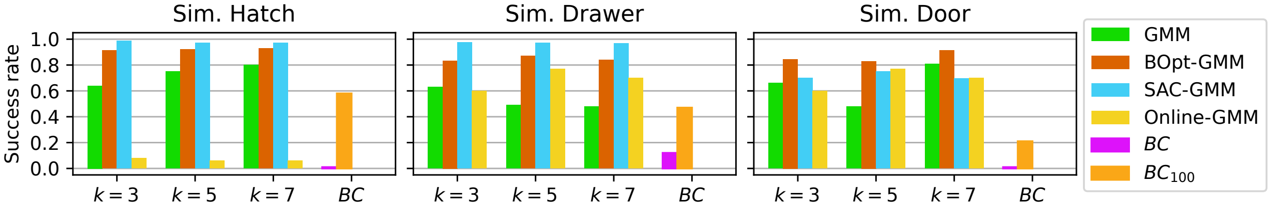


Fig. 4: Comparison of the mean performances of GMM, SAC-GMM, BOpt-GMM, and Online-GMM baseline in our three simulated scenarios in Fig. 3). k indicates the number of GMM components. We run each method for 500 episodes. We find BOpt-GMM and SAC-GMM to improve significantly over the initial GMM, while Online-GMM does not do so reliably, or even deteriorates performance. Additionally, we introduce BC trained on the same demonstrations as the GMM, and BC_{100} trained on a full 100 demonstrations. The latter achieves recognizable but not comparable performance.

TABLE I: Detailed analysis of the effect of GMM updates in BOpt-GMM and SAC-GMM. The table shows the maximal success rate of models optimized only over their means (μ), their covariances using our update approaches *eig* and R_{XYZ}, as well as their combinations. We optimize a GMM ($k = 5$) with these update strategies using both BOpt and SAC. We report the mean final performances of our models, as well as the mean number of episodes needed to achieve 80% success rate. The *Mean* column reports the average of these metrics across the scenarios, while the *Mean* row reports the averages across the update modalities. From the mean along both axes, we draw the overall conclusion that BOpt-GMM performs much faster, while SAC-GMM achieves slightly higher overall performance. From the bottom row we can also conclude that the *eig* and R_{XYZ} updates work better on their own, than when paired with μ and also work better than μ alone. We note only two exceptions to this observation in the *Door* scenario for BOpt-GMM.

		μ		R_{XYZ}		eig		$\mu + R_{XYZ}$		$\mu + eig$		Task mean	
		Success rate	# eps. > 80%	Success rate	# eps. > 80%	Success rate	# eps. > 80%	Success rate	# eps. > 80%	Success rate	# eps. > 80%	Success rate	# eps. > 80%
Hatch	SAC	90.9%	125	100.0%	75	100%	100	97.5%	150	98.3%	125	**97.3%**	115.0
	BOpt	88.8%	28	98.6%	14	99.2%	14	88.6%	28	85.6%	28	92.2%	**22.0**
Drawer	SAC	95.0%	150	99.0%	50	99.2%	50	97.5%	125	95.8%	225	**97.3%**	120.0
	BOpt	82%	252	87.6%	49	94.3%	35	84.6%	133	88.6%	98	87.4%	**113.0**
Door	SAC	79.2%	–	87.0%	250	83.3%	400	69.2%	–	58.3%	–	75.4%	430.0
	BOpt	83%	301	81.6%	168	85.0%	203	90.3%	161	77%	–	**83.4%**	**266.0**
Mean	SAC	**88.4%**	258.3	**95.3%**	125.0	**94.2%**	183.3	**88.1%**	258.3	**84.1%**	283.3	**90.0%**	221.6
	BOpt	84.6%	**193.6**	89.3%	**77**	92.8%	**84.0**	87.8%	**107.3**	83.7%	**208.7**	87.6%	**134.1**

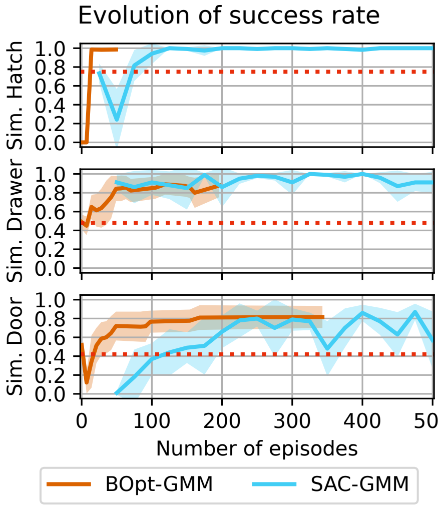


Fig. 5: To illustrate the significance of the difference in sampling efficiency, we overlay the evolution of model performances of the three basic updates. The dashed red line shows the performance of the base GMM. Note: BOpt-GMM is only evaluated when a new incumbent is generated, while SAC-GMM is evaluated at regular intervals. Hence the different graph lengths.

BOpt-GMM. We conclude from our simulated experiments that SAC-GMM yields a higher policy success rate in the long run, but BOpt-GMM achieves *good* performance much sooner. Further, we will deploy the R_{XYZ} and eig updates in our real-world scenarios, as they promise to be the most effective.

Finally, we would like to note a peculiarity in working with Bayesian optimization. The optimizer we use uses the surrogate model to generate a so-called *incumbent* configuration which is the assumed highest performing set of parameters. The incumbent does not change after every data sample, as a new sample does not need to reveal a new optimal set of parameters. While this does not improve sample efficiency in improving the policy, it does mean that we only have to re-evaluate the performance of our policy whenever a new incumbent is generated. As can be seen in Fig. 5, in some scenarios, such as *Hatch*, the best performing incumbent is found early, while in others this can take longer. These discrete moments for policy evaluation distinguish using BOpt significantly from other learning methods. It is outside of the scope of this work, but we believe there to be a potential for terminating the learning process early based on a trade-off of the remaining uncertainty in the surrogate model and the remaining possible improvement.

C. Real World Experiments

Our real-world scenarios (d, e, f in Fig. 3) mimic the scenarios we explored in the simulation. We again have a sliding door, a drawer, and a door for the robot to manipulate. We monitor the completion of the tasks using AR markers by measuring the displacement or rotation of the marker on the moving part in the frame of a static marker. The sliding door is registered as open when moved $0.4\,\mathrm{m}$ to the left,

TABLE II: Results of the real-world experiments. We report the maximal success rate of models and the number of episodes required to achieve 80% performance. In this experiment, we find BOpt-GMM to outperform SAC-GMM. However, the real experiments ended after 100 episodes, which explains the comparatively low maximum performance of SAC-GMM.

		R_{XYZ}		eig		Mean	
		Success rate	# eps. > 80%	Success rate	# eps. > 80%	Success rate	# eps. > 80%
Hatch	SAC	63.3%	–	53.3%	–	58.3%	100.0
	BOpt	76.6%	–	85.5%	14	81.1%	57.0
Drawer	SAC	86.6%	25	93.3%	50	90.0%	37.5
	BOpt	86.6%	14	93.3%	14	90.0%	14.0
Door	SAC	93.3%	50	71.1%	–	82.2%	75.0
	BOpt	93.3%	14	95.0%	7	94.2%	10.5
Mean	SAC	81.1%	58.3	72.6%	83.3	76.9%	70.8
	BOpt	**85.5%**	**42.7**	**91.3%**	**11.7**	**88.4%**	**27.2**

the drawer is considered open at $0.2\,\mathrm{m}$, and the door is at an angle of $25°$. The scenarios are randomized by sampling different starting locations of the robot's end-effector.

In the real-world scenarios, we reduce the scale of the experiments. Where we verified our results in simulation on 16 different seeds and ran each method for 500 episodes, here we reduced the numbers to 3 seeds and 100 episodes. We use the insights we have gained from our simulations and only study the R_{XYZ} and eig updates. We collect 10 demonstrations per scenario by teleoperating the robot using a gamepad and fitting an initial GMM with $k = 5$ components to these. We set $\Delta\theta_\omega$, $\Delta\theta_\Sigma$, and SAC's learning rate as before. We find our observations from the simulation experiments to be confirmed in our real-world experiments. Both SAC-GMM and BOpt-GMM can use our update strategies to successfully improve the baseline policy. Once again, BOpt-GMM passes the 80% success rate threshold sooner than SAC-GMM. The minor trend we observed in our simulated experiments is born out more starkly in our real experiments: we observe that the R_{XYZ} update performs better in combination with SAC-GMM, while the eig update performs better with BOpt-GMM. Different from our simulation experiments, BOpt-GMM also stays ahead of SAC-GMM in overall success rate. This is likely due to the much shorter runtime of the experiments in the real-world setting.

VI. Conclusion

In this work, we proposed employing gradient-free Bayesian Optimization (BOpt) in a sparse reinforcement learning setting with the aim of achieving greater sample efficiency. We enabled the application of BOpt to this space by encoding our underlying policy as GMM and letting BOpt find suitable updates to this policy. We coin this combination BOpt-GMM. To keep the updates low-dimensional but still enable the optimizer to access the entire GMM, we proposed two low-dimensional methods for updating the GMM's covariance. We compared our approach to three other baselines in simulation and were able to successfully deploy the parameters we identified in three real-world scenarios. In both simulation and real world, we found that our approach is significantly more efficient, achieving a success rate of 80%, 40% faster than our baselines. We find that our covariance updates are more effective for our approach and our SAC-GMM baseline.

For future work, we see an opportunity for combining BOpt-GMM and SAC-GMM. BOpt-GMM would yield the first drastic improvement, while SAC-GMM would be tasked with developing the local reactivity needed to achieve a final couple of percentage points of success. As a minor improvement, we believe there is an opportunity for exploiting the variance in the surrogate function to determine the number of episodes to evaluate a sample. As a larger improvement, we are also considering if there is an early stopping criterion that could be used to reliably end the optimization process when it stagnates. While this does not improve the overall policy success rate, it would potentially further improve sample efficiency.

Appendix - Low-rank Covariance Updates

As additional GMM size $|S|$ update, we examine a rank-1 update scheme. The scheme is based on constructing a low-rank approximation of Σ^k, performing an update under this approximation and reintegrating the update into the full-rank matrix. We do so by extracting $\mathbf{U}, \mathbf{s}, \mathbf{V}$ from Σ^k as

$$\Sigma^k - \mathrm{diag}(\Sigma^k) = \mathbf{U}\,\mathrm{diag}(\mathbf{s})\mathbf{V}^T, \tag{9}$$

using singular value decomposition. We use this decomposition to formulate a rank-1 update by only considering the first column $\mathbf{U}_1, \mathbf{V}_1$ of $\mathbf{U}, \mathbf{V}$ and the first component $\mathbf{s}_1$ of $\mathbf{s}$. Given the i-th update vector $\Delta\theta_{\Sigma^k,i} \in [1-\sigma, 1+\sigma]$, we compute the updated matrix Σ^k_i as

$$\Sigma^k_i = \mathrm{diag}(\Sigma^k) + \mathbf{U}_1\,\mathrm{diag}(\mathbf{s}_1)(\mathbf{V}_1 \cdot \Delta\theta_{\Sigma^k,i})^T. \tag{10}$$

We investigate this update type in our simulated scenarios with both BOpt-GMM and SAC-GMM. We find that this update type is much less reliable at achieving strong performance with either method and varying number of GMM components, see Tab. III. Upon inspection, we find that the rank-1 update performs well ($\geq 90\%$) in some settings, but fails in others ($\leq 30\%$). We suspect this is due to the required decomposition being executed on $\Sigma^k - \mathrm{diag}(\Sigma^k)$, which can become numerically unstable as Σ^k becomes more and more diagonal. However, we need this decomposition to maintain the invertibility of Σ^k_i as required by the GMR inference, see Eq. (4) for details. In addition, this method allows us to maintain the positive determinant of Σ^k, which is required by the function $h^k(s_t)$ also required during inference.

TABLE III: We display the average final success rates achieved by our ablated rank-1 update method. The reported success rates are averages over all simulated scenarios and all numbers of GMM components k. While we see in Fig. 4 that BOpt-GMM achieves an average success rate of over 80% independent of the update type and k, this is not the case for the low-rank update. Similarly SAC-GMM underperforms with this update. We show the aggregated performance of the other update types in the right-most column.

	Rank-1	$\mu+$Rank-1	Mean Rank-1	Mean o. Updates
SAC	73.9%	72.0%	73%	88.7%
BOpt	74.1%	72.1%	73.1%	87.7%

References

[1] T. Osa, J. Pajarinen, G. Neumann, J. Bagnell, P. Abbeel, and J. Peters, “An algorithmic perspective on imitation learning,” *Foundations and Trends in Robotics*, vol. 7, pp. 1–179, 11 2018.

[2] E. Chisari, T. Welschehold, J. Boedecker, W. Burgard, and A. Valada, “Correct me if i am wrong: Interactive learning for robotic manipulation,” *IEEE Robotics and Automation Letters*, 2022.

[3] J. O. von Hartz, E. Chisari, T. Welschehold *et al.*, “The treachery of images: Bayesian scene keypoints for deep policy learning in robotic manipulation,” *IEEE Robotics and Automation Letters*, 2023.

[4] D. Honerkamp, M. Büchner, F. Despinoy, T. Welschehold, and A. Valada, “Language-grounded dynamic scene graphs for interactive object search with mobile manipulation,” *arXiv preprint arXiv:2403.08605*, 2024.

[5] F. Schmalstieg, D. Honerkamp, T. Welschehold, and A. Valada, “Learning hierarchical interactive multi-object search for mobile manipulation,” *IEEE Robotics and Automation Letters*, 2023.

[6] D. Honerkamp, T. Welschehold, and A. Valada, “N^2m^2: Learning navigation for arbitrary mobile manipulation motions in unseen and dynamic environments,” *IEEE Transactions on Robotics*, 2023.

[7] S. Schaal, S. Kotosaka, and D. Sternad, “Nonlinear dynamical systems as movement primitives,” *Int. Journal of Humanoid Robotics*, 2000.

[8] S. M. Khansari-Zadeh and A. Billard, “Learning stable nonlinear dynamical systems with gaussian mixture models,” *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[9] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, “Dynamical movement primitives: learning attractor models for motor behaviors,” *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[10] S. Manschitz, M. Gienger, J. Kober, and J. Peters, “Mixture of attractors: A novel movement primitive representation for learning motor skills from demonstrations,” *IEEE Rob. and Aut. Letters*, 2018.

[11] I. Nematollahi, E. Rosete-Beas, A. Röfer, T. Welschehold, A. Valada, and W. Burgard, “Robot skill adaptation via soft actor-critic gaussian mixture models,” in *Int. Conf. on Rob. and Aut.*, 2022.

[12] F. Stulp and O. Sigaud, “Policy improvement: Between black-box optimization and episodic reinforcement learning,” in *Journées Francophones Planification*, 2013.

[13] P. Englert and M. Toussaint, “Learning manipulation skills from a single demonstration,” *The International Journal of Robotics Research*, vol. 37, no. 1, pp. 137–154, 2018.

[14] L. Johannsmeier, M. Gerchow, and S. Haddadin, “A framework for robot manipulation: Skill formalism, meta learning and adaptive control,” in *International Conference on Robotics and Automation*, 2019.

[15] Z. Wu, W. Lian, C. Wang, M. Li, S. Schaal, and M. Tomizuka, “Prim-lafd: A framework to learn and adapt primitive-based skills from demonstrations for insertion tasks,” *IFAC-PapersOnLine*, 2023.

[16] M. Bain and C. Sammut, “A framework for behavioural cloning.” in *Machine Intelligence 15*, 1995, pp. 103–129.

[17] A. Billard, S. Calinon, R. Dillmann, and S. Schaal, “Survey: Robot programming by demonstration,” *Springer Handbook of Robotics*, pp. 1371–1394, 2008.

[18] C. Celemin, R. Pérez-Dattari, E. Chisari, G. Franzese, L. de Souza Rosa, R. Prakash, Z. Ajanović, M. Ferraz, A. Valada, J. Kober *et al.*, “Interactive imitation learning in robotics: A survey,” *Foundations and Trends® in Robotics*, vol. 10, no. 1-2, pp. 1–197, 2022.

[19] B. Zheng, S. Verma, J. Zhou, I. W. Tsang, and F. Chen, “Imitation learning: Progress, taxonomies and challenges,” *IEEE Transactions on Neural Networks and Learning Systems*, no. 99, pp. 1–16, 2022.

[20] L. Le Mero, D. Yi, M. Dianati, and A. Mouzakitis, “A survey on imitation learning techniques for end-to-end autonomous vehicles,” *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[21] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, “One-shot visual imitation learning via meta-learning,” in *Conference on robot learning*, 2017, pp. 357–368.

[22] J. Wong, A. Tung, A. Kurenkov, A. Mandlekar, L. Fei-Fei, S. Savarese, and R. Martín-Martín, “Error-aware imitation learning from teleoperation data for mobile manipulation,” in *Conference on Robot Learning*, 2022, pp. 1367–1378.

[23] A. Mandlekar, D. Xu, R. Martín-Martín, S. Savarese, and L. Fei-Fei, “Learning to generalize across long-horizon tasks from human demonstrations,” *arXiv preprint arXiv:2003.06085*, 2020.

[24] M. Shridhar, L. Manuelli, and D. Fox, “Perceiver-actor: A multi-task transformer for robotic manipulation,” in *Conference on Robot Learning*, 2023, pp. 785–799.

[25] A. Brohan, N. Brown, J. Carbajal *et al.*, “Rt-1: Robotics transformer for real-world control at scale,” *arXiv preprint arXiv:2212.06817*, 2022.

[26] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes *et al.*, “Do as i can, not as i say: Grounding language in robotic affordances,” *arXiv preprint arXiv:2204.01691*, 2022.

[27] N. Figueroa and A. Billard, “A physically-consistent bayesian non-parametric mixture model for dynamical system learning.” in *Proc. of the Conf. on Robot Learning*, 2018, pp. 927–946.

[28] È. Pairet, P. Ardón, M. Mistry, and Y. Petillot, “Learning generalizable coupling terms for obstacle avoidance via low-dimensional geometric descriptors,” *IEEE Robotics and Automation Letters*, 2019.

[29] Z. Lu, N. Wang, and C. Yang, “A constrained dmps framework for robot skills learning and generalization from human demonstrations,” *IEEE/ASME Transactions on Mechatronics*, 2021.

[30] Y. Wang, N. Figueroa, S. Li, A. Shah, and J. Shah, “Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations,” *arXiv preprint arXiv:2206.04632*, 2022.

[31] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, “Bayesian optimization for learning gaits under uncertainty: An experimental comparison on a dynamic bipedal walker,” *Annals of Mathematics and Artificial Intelligence*, vol. 76, pp. 5–23, 2016.

[32] A. Rai, R. Antonova, S. Song, W. Martin, H. Geyer, and C. Atkeson, “Bayesian optimization using domain knowledge on the atrias biped,” in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1771–1778.

[33] S. Kuindersma, R. Grupen, and A. Barto, “Learning dynamic arm motions for postural recovery,” in *Int. Conf. on Humanoid Robots*. IEEE, 2011, pp. 7–12.

[34] K. Chatzilygeroudis, R. Rama, R. Kaushik, D. Goepp, V. Vassiliades, and J.-B. Mouret, “Black-box data-efficient policy search for robotics,” in *Int. Conf. on Intelligent Robots and Systems*. IEEE, 2017, pp. 51–58.

[35] L. P. Fröhlich, E. D. Klenske, C. G. Daniel, and M. N. Zeilinger, “Bayesian optimization for policy search in high-dimensional systems via automatic domain selection,” in *Int. Conf. on Intelligent Robots and Systems*. IEEE, 2019, pp. 757–764.

[36] R. Antonova, A. Rai, T. Li, and D. Kragic, “Bayesian optimization in variational latent spaces with dynamic compression,” in *Conference on Robot Learning*. PMLR, 2020, pp. 456–465.

[37] A. Marco, F. Berkenkamp, P. Hennig, A. P. Schoellig, A. Krause, S. Schaal, and S. Trimpe, “Virtual vs. real: Trading off simulations and physical experiments in reinforcement learning with bayesian optimization,” in *Int. Conf. on Rob. and Aut.* IEEE, 2017, pp. 1557–1563.

[38] M. Tesch, J. Schneider, and H. Choset, “Expensive function optimization with stochastic binary outcomes,” in *Int. Conf. on Machine Learning*. PMLR, 2013, pp. 1283–1291.

[39] F. Voigt, L. Johannsmeier, and S. Haddadin, “Multi-level structure vs. end-to-end-learning in high-performance tactile robotic manipulation.” in *Proc. of the Conf. on Robot Learning*, 2020, pp. 2306–2316.

[40] F. Otto, O. Celik, H. Zhou, H. Ziesche, V. A. Ngo, and G. Neumann, “Deep black-box reinforcement learning with movement primitives,” in *Conference on Robot Learning*. PMLR, 2023, pp. 1244–1265.

[41] O. Celik, D. Zhou, G. Li, P. Becker, and G. Neumann, “Specializing versatile skill libraries using local mixture of experts,” in *Conference on Robot Learning*. PMLR, 2022, pp. 1423–1433.

[42] F. Hutter, L. Kotthoff, and J. Vanschoren, *Automated machine learning: methods, systems, challenges*. Springer Nature, 2019.

[43] L. Yang and A. Shami, “On hyperparameter optimization of machine learning algorithms: Theory and practice,” *Neurocomputing*, vol. 415, pp. 295–316, 2020.

[44] D. R. Jones, M. Schonlau, and W. J. Welch, “Efficient global optimization of expensive black-box functions,” *Journal of Global optimization*, vol. 13, pp. 455–492, 1998.

[45] M. Lindauer, K. Eggensperger, M. Feurer *et al.*, “Smac3: A versatile bayesian optimization package for hyperparameter optimization,” *Journal of Machine Learning Research*, 2022.