VAE-AUGMENTED IMITATION LEARNING WITH LLM-BASED GOAL GENERATION FOR SCALABLE ROBOT MANIPULATION

DHYEY SHAH [DHYEY28@SEAS], MAANASA RAJESHWER [MAANASA@SEAS],
PRAKRITI PRASAD [PRASADPR@SEAS], SAMHITHA VEDIRE [SVEDIRE@SEAS]

ABSTRACT. This project explores the use of latent trajectory representations to enhance Behavioral Cloning (BC) for robotic pick-and-place tasks using only proprioceptive and object-centric inputs. We train a Variational Autoencoder (VAE) as a temporal encoder on low-dimensional robot state data to capture motion intent, task phase, and context. Four configurations are evaluated: full observation (proprio + image), proprio only, VAE-enabled BC, and BC with precomputed `latent_vae`. Results show that `latent_vae` improves policy performance over raw proprioception alone and could provide promising improvements in ensuring task continuity in times of exteroceptive sensor failures. Additionally, we integrate a Large Language Model (LLM) to generate spatial goals from natural language, enabling comparison of policy generalization in both single-object and multi-object manipulation trials. Our findings demonstrate that VAE-based behavior summarization, combined with LLM-driven task specification, enables efficient, scalable policy learning without heavy reliance on exteroceptive sensors for structured manipulation.

1. INTRODUCTION

Pick-and-place is a foundational skill in robotic manipulation, forming the basis of tasks in industrial automation, assistive robotics, and home environments. Despite its simplicity in theory, executing reliable and adaptive pick-and-place actions in real-world settings remain challenging due to partial observability, sensor noise, and the need for temporal coordination. Traditional approaches often rely heavily on high-resolution visual input, which can be computationally expensive and fragile in the face of occlusions or sensor failures. This project addresses the problem of learning robust pick-and-place behaviors using only low-dimensional proprioceptive and object-centric inputs to reduce dependence on image-based observations. We further explore how compact latent representations of motion, learned through a Variational Autoencoder (VAE), can summarize trajectory intent and improve policy performance. To enable intuitive task specification, we integrate a Large Language Model (LLM) that translates natural language instructions into 3D spatial goals, allowing the robot to generalize across single- and multi-object placements in a structured environment. This work contributes toward scalable, sensor-efficient, and instruction-driven robotic manipulation.

1.1. Contributions.

- *Proprioceptive-Only Behavioral Cloning with VAE Enhancements:* We demonstrate that VAE-generated latent representations improve pick-and-place success rates compared to raw proprioceptive data, narrowing the performance gap with image-based policies.
- *LLM-Driven Task Specification Pipeline:* We implement an end-to-end pipeline using a GPT-4o-based interface that translates natural language instructions into constraint-compliant 3-D object placement goals, supporting task generalization and enabling intuitive human-robot interaction.

2. BACKGROUND

2.1. **Problem Definition.** We tackle a multi-object pick-and-place task in a constrained Robosuite tabletop setting with four known objects: milk, bread, cereal, and a can. Given natural language instructions (e.g., "place the cereal next to the milk"), GPT-4o generates diverse 3D target positions that respect workspace bounds and object spacing. A pretrained policy then sequentially places the objects using end-effector control. Several challenges make this non-trivial like:

1. Workspace constraints – Object placements must remain within defined bounds.
2. Collision avoidance – Sufficient spacing is needed to avoid object interference.
3. Goal diversity – The policy must handle a variety of spatial instructions.
4. Precision and generalization – Accurate placement is required across layouts.
5. Language grounding – The robot must interpret and act on spatial commands.

This framework integrates LLM-based symbolic planning with learned low-level control, enabling scalable, zero-shot generalization without retraining for new instructions or layouts.

2.2. **Simulation and Learning Framework.** Pick-and-place tasks require precise, generalizable control across varied objects and layouts. We use Robosuite(v1.4.1) for simulation, MimicGen for generating large-scale demonstrations from limited examples, and Robomimic(v0.3) for training and evaluation. Combined with GPT-4o, this pipeline enables scalable, robust policy learning with minimal manual data collection or task-specific engineering.

3. Related Work

Recent work on robust imitation of diverse behaviors ([27]) shows how VAEs can capture low-dimensional policies across varied tasks, but still relies on high-dimensional sensory inputs. By contrast, our BC + VAE pipeline discards raw images altogether, compressing only proprioceptive state sequences into a latent codebook and conditioning on compact GPT-4o goal embeddings to avoid camera-driven domain shifts. Vision-based multi-task manipulation methods ([28]) unify tasks under a single network but suffer from inflexible visual pipelines; our approach achieves similar multi-task flexibility from joint encoders. Recent conditional VAE motion models for multitask imitation ([29]) demonstrate the power of latent dynamics, which we extend by injecting language-derived goals for sequential pick-and-place. Finally, behavioral cloning with VAE predictors in Atari games (e.g., [7, 9]) highlights how learned priors stabilize off-policy learning, and our trajectory-phase latent codes similarly reduce covariate drift and boost manipulation success rates.

4. Approach

4.1. **Language-to-Goal Planning with GPT-4.** To support flexible goal specification, we use a Large Language Model (LLM), specifically OpenAI's GPT-4, to parse natural language instructions and convert them into concrete 3D pick-and-place targets for the robot. This setup enables intuitive, human-readable commands such as "place the cereal next to the milk" or "set the can far from the bread," without requiring users to define exact coordinates.

We implement a prompt-driven querying system that takes in a goal description and returns a list of relevant objects along with randomized, constraint-compliant 3D positions. These constraints—such as safe placement margins, minimum inter-object distances, and valid object heights—are enforced in the backend logic. The system also ensures that the output follows a strict JSON schema specifying active objects and their corresponding target coordinates.

Given a user instruction, the LLM returns a list of relevant objects along with randomized, constraint-compliant 3D positions. This output is parsed and passed to a trained policy (described in the next section), which attempts to move each object to its target location. This approach decouples symbolic goal planning from control execution and supports generalization to novel instructions and object configurations.

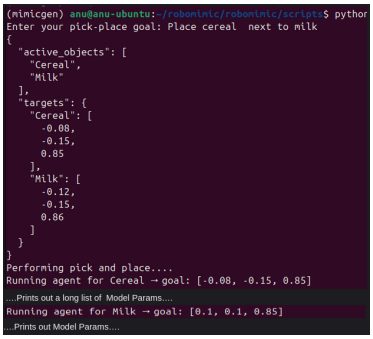

FIGURE 1. 3D Goal Generation with an LLM

4.2. **Behavioral Cloning.** Behavioral Cloning is a simple form of imitation learning, where a set of expert demonstrations is collected as state-action pairs:

$$(s_i, a_i)\,, \quad s_i \in \mathcal{S},\ a_i \in \mathcal{A}. \tag{1}$$

Here, each state s_i is a bundle of multiple modalities (e.g., an end-effector pose, joint velocities, LLM outputs), and a_i is the corresponding expert action (e.g. an end-effector velocity). In our pipeline, these N demonstrations are generated in Robosuite by tele-operating on pick-and-place tasks, then augmented via MimicGen to vary object poses and viewpoints. Each demonstration is tagged with a 3-D goal produced by GPT-4o, ensuring that $\mathcal{D}$ covers the full space of language-specified targets. We implement BC with a 2-layer MLP (1024 units/layer) trained on state-action pairs (s_i, a_i) from 1000 expert demonstrations. The state s_i combines:

$$s_i = [\underbrace{p_t, q_t, g_t, \theta_t, \dot{\theta}_t, o_t}_{\text{proprioception (25D)}}; \underbrace{z_{\text{vae}}}_{\text{latent (14D)}}\ ] \tag{2}$$

$$\mathcal{D} = \{(s_i, a_i)\}_{i=1}^{N} \tag{3}$$

4.2.1. *Objective Function.* Using supervised learning, a policy π_θ is fitted to replicate the expert's actions by minimizing the loss. The policy π_θ minimizes a composite loss:

$$\mathcal{L}(\theta) = \underbrace{\lambda_2 \|a - \hat{a}\|_2^2}_{\text{Precision}} + \underbrace{\lambda_1 \|a - \hat{a}\|_1}_{\text{Robustness}} + \underbrace{\lambda_c (1 - \cos(a, \hat{a}))}_{\text{Direction}} \tag{4}$$

- $\lambda_2 = 1.0$: L2 loss dominates for precise magnitude matching
- $\lambda_1 = 0.0$: L1 term available for outlier rejection
- $\lambda_c = 0.0$: Cosine loss available for directional alignment

Our default configuration emphasizes exact trajectory replication through squared error minimization, with optional terms for robust or directional learning.

4.2.2. *Multimodal State Encoding.* In our updated training setup, the raw state s_t consists of multiple low-dimensional proprioceptive and object-centric features, along with high-dimensional visual inputs. The low-dimensional vector includes the end-effector position (3-D), end-effector orientation quaternion (4-D), gripper joint positions (2-D), robot joint positions (7-D), robot joint velocities (7-D), and an object state descriptor (variable dimension depending on task, denoted here as d_{obj}). These are concatenated into a single flat feature vector:

$$s_t = \begin{bmatrix} \text{End\_effector position (3D)} \\ \text{End\_effector orientation (4D)} \\ \text{Gripper joints (2D)} \\ \text{Arm joints (7D)} \\ \text{Arm velocities (7D)} \\ \text{Object state } (d_{obj}) \end{bmatrix} \in \mathbb{R}^{25+d_{obj}} \tag{5}$$

- For vision-based runs: $s_t \leftarrow [s_t^{\text{low}}; \text{rgb}_t]$
- For VAE runs: $s_t \leftarrow [s_t^{\text{low}}; z_{\text{vae}}]$ (14D latent)

This multimodal encoding allows the policy to leverage both structured proprioceptive signals and unstructured visual inputs for learning.

4.2.3. *Policy Network.* Once our multimodal state s_t is constructed, it is passed to a deterministic policy network π_θ that outputs a continuous action vector:

$$\hat{a}_t = \pi_\theta(s_t).$$

In our implementation, π_θ is realized via Robomimic's `ActorNetwork` class, which supports flexible modality fusion and consists of modality-specific encoders followed by a fully connected MLP policy head.

For this project, the core policy uses a 2-layer MLP architecture:

$$\begin{aligned} h_1 &= \text{ReLU}(W_1 s_t + b_1) \\ h_2 &= \text{ReLU}(W_2 h_1 + b_2) \\ \hat{a}_t &= W_3 h_2 + b_3 \end{aligned} \tag{6}$$

where:

- $W_1, W_2 \in \mathbb{R}^{1024\times1024}$ are hidden layer weights,
- $W_3 \in \mathbb{R}^{\text{action\_dim}\times1024}$ maps to the final action space,
- The network is optimized using the Adam optimizer with learning rate $\eta = 10^{-4}$.

The policy is deterministic; no noise is injected during action generation. This allows for fast inference and stable performance at test time. The fully differentiable MLP structure ensures compatibility with gradient-based optimization and automatic differentiation.

4.2.4. *Loss Computation and Backpropagation.* Given the predicted action $\hat{a}_t = \pi_\theta(s_t)$ and the expert action a_t, we compute the training loss as a weighted sum of three components: mean squared error (L_2), absolute error (L_1), and cosine similarity alignment. The augmented loss for each time step is given by:

$$\begin{aligned} \mathcal{L}(\theta) = \lambda_2 \left\|\pi_\theta(s_i) - a_i\right\|_2^2 \\ + \lambda_1 \left\|\pi_\theta(s_i) - a_i\right\|_1 \\ + \lambda_c \left(1 - \cos\left(\pi_\theta(s_i), a_i\right)\right), \end{aligned} \tag{7}$$

This formulation balances overall prediction accuracy (L_2), robustness to outliers (L_1), and directional correctness (cosine term). Over a batch of size B ($B = 16$ in our runs). The final loss is the average loss over a batch of state-action pairs, and gradient descent updates the policy parameters by stepping in the direction that minimizes this average loss.

During backpropagation, gradients flow through all loss components and MLP layers, updating network parameters efficiently via PyTorch's autograd engine. This multi-term loss ensures smoother convergence and improved stability, especially in scenarios with noisy or inconsistent demonstrations. At test time, after E epochs of training, we freeze π_θ and evaluate on Robosuite trials. For each timestep, the model outputs an end-effector action, which is applied until the robot reaches its target goal, as specified by the LLM or predefined waypoint.

4.3. **Variational Autoencoder (VAE).** While Behavioral Cloning (BC) performs well with both proprioceptive and visual inputs, high-resolution image data can be costly, noisy, and computationally demanding in real-world robotics. To reduce this dependency while preserving task-relevant structure, we use a Variational Autoencoder (VAE) to learn compact latent representations from proprioceptive and object-centric inputs alone.

4.3.1. *VAE Architecture.* In our implementation, the Variational Autoencoder (VAE) is trained on sequences of proprioceptive and object-centric features, rather than RGB images. The goal is to learn compact latent representations that summarize task dynamics, such as motion intent, phase of execution, or object interaction structure.

The VAE consists of:

- Encoder: $[300, 400]$ MLP $\rightarrow \mathbb{R}^{14}$ latent vector
- Decoder: $[400, 300]$ MLP (used only during training, discarded afterward)
- KL divergence weight (β): 1.0

This structure allows the encoder to summarize trajectories into semantically meaningful latent embeddings without relying on vision. During deployment, we retain only the encoder to generate '`latent_vae`' vectors for downstream BC policy input.

4.3.2. *Objective Function.* The VAE optimizes a weighted evidence lower bound (ELBO):

$$\mathcal{L}_{\text{VAE}} = \underbrace{\mathbb{E}_{z\sim q_\phi}[\log p_\psi(x|z)]}_{\text{Reconstruction term}} - \beta \underbrace{D_{\text{KL}}(q_\phi(z|x)\|p(z))}_{\text{Regularization term}} \tag{8}$$

Here, x is a sequence of proprioceptive and task-relevant features, such as the pose of the end effector, joint angles, velocities, and state of the object. The encoder $q_\phi(z|x)$ maps the input to a latent distribution over $z \in \mathbb{R}^{14}$, while the decoder $p_\psi(x|z)$ reconstructs the original input.

4.3.3. *Dataset and Training Procedure.* In our implementation, we train the VAE using images collected from Robosuite rollouts. These frames include the robot's gripper interacting with common kitchen objects such as milk cartons, cereal boxes, and cans. The VAE learns to abstract features like object shape, location, and relative orientation into a latent vector z_{vae} of fixed dimension, which we later use as a substitute for the full image.

5. Experimental Results

5.1. **Vanilla Behavior Cloning (Proprioception + RGB images.** Leverages both proprioceptive and visual inputs to train a policy with rich state information for improved manipulation.

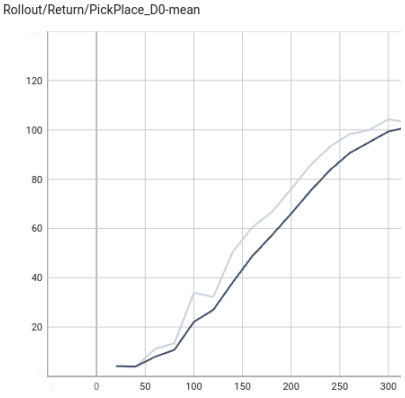

FIGURE 2. Average rollout reward during evaluation episodes.

5.2. **Run_1: Behavior Cloning with Proprioception Only (Reduced Modalities).** Trains the policy using only robot state without any visual feedback, representing a sensor-limited baseline.

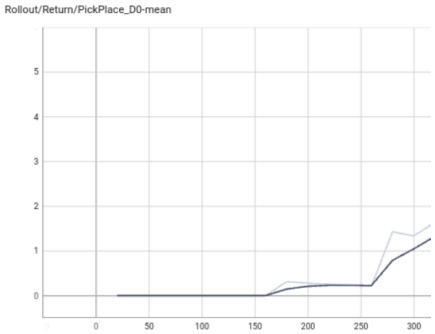

FIGURE 3. Average rollout reward during evaluation episodes.

5.3. **Run_2: Behavior Cloning with VAE Enabled Training on Reduced Modalities.** Introduces a VAE during training to encode temporal structure in proprioceptive data, enhancing policy learning.

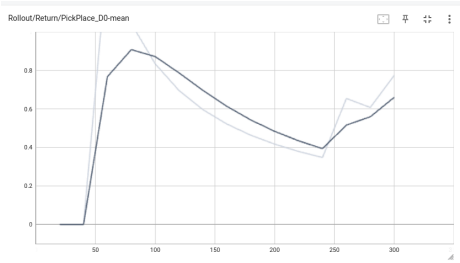

FIGURE 4. Average rollout reward during evaluation episodes.

5.4. **Run_3: Behavior Cloning with VAE Latent on Reduced Modalities.** Uses precomputed VAE latents as compact state embeddings to drive policy learning without an online encoder.

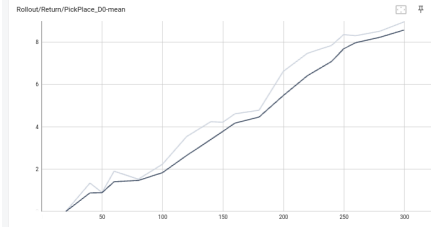

FIGURE 5. Average rollout reward during evaluation episodes.

5.5. **Working Pipeline Demo with Model 1.**

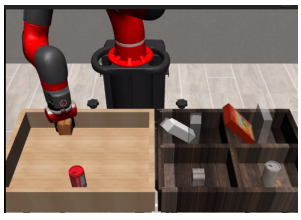

FIGURE 6. Picks up bread.

FIGURE 7. Places bread (even after it was dropped).

Drop Correction: The drop detection and correction from Figure 7 can be seen in the video: `https://youtu.be/JKXO8kdKLlg`

Model Config (BC with)	Loss	Max No. of Picks	Avg No. of Picks	Outcome	Train Time (hrs)
Proprioception only	0.021	13	1.0	Baseline	2.2
Proprioception only + VAE	0.03	4	0.6	Similar to Baseline	2.3
Proprioception only + latent VAE	0.0197	26	8.5	Slightly improved	5.5

TABLE 1. Evaluation Results for Behavior Cloning Models

We first evaluate a standard Behavioral Cloning (BC) policy over an extended training horizon, confirming its suitability for our structured pick-and-place task. Next, we apply our custom observation modalities to eliminate image input and test performance under proprioception-only conditions (Run_1 and Run_2). To evaluate the effect of learned temporal structure, we further compare this baseline against a BC model augmented with `latent_vae` embeddings (Run_3). All custom models were trained for 300 epochs on a reduced dataset due to hardware and time constraints– allowing us to experiment with multiple configurations. From the return curves, we observe that the vanilla BC (Run_1) steadily improves, crossing a final return of 2. Proprioception-only training (Run_2) shows reduced performance, with lower reward and less consistent task completion, discussed in the next section. However, the `latent_vae`-enhanced model (Run_3) recovers and exceeds the baseline, reaching a final return above 9. Supporting this the Table 1 shows the experimental results for our overall pipeline execution by showing the maximum and average rollout of successful picks (number of times an object was picked and place at the correct spot, we show a combined result for single and multiple sequence tasks in the table), evaluation metrics show that Run_3 achieves the highest average number of successful picks (8.5) and peak performance (26 picks), with the lowest loss (0.0197) among all runs, albeit with longer training time (5.5 hrs). These results reinforce that structured low-dimensional representations can meaningfully compensate for the absence of visual input in learned robotic policies.

5.6. **Demonstration Video.** A video showcasing the final execution of the trained policy in the kitchen environment with command input is available at: `https://youtu.be/CqmEF2GN3Ao`.

6. DISCUSSION

In our experiments, Run_1 served as a reliable baseline, demonstrating that Behavioral Cloning can perform well in structured manipulation tasks without image data. Run_2, which enabled the VAE during training but did not use its output, performed slightly worse—likely due to representational constraints introduced by the VAE during encoding. In contrast, Run_3, which incorporated precomputed `latent_vae` embeddings into the BC input, achieved the best performance. This suggests that the learned latent features effectively summarize trajectory structure and phase information, improving the policy's ability to handle multi-stage tasks. This aligns with prior work demonstrating the utility of latent behavior priors in temporally extended tasks [7, 9]. The rollout statistics confirm this: Run_3 achieved the highest number of successful picks and the lowest loss even in our combined single and multiple pick-place task sequence run by our final pipeline. Our language-to-goal system, powered by GPT-4o, enabled structured and consistent evaluation of model performance across both single-object and compound pick-and-place sequences. Performance are shown in Table 1.

Even though our results were not significantly higher than our declared baseline, we observed that they keep becoming more significant with increasing epochs and for a small study, valide our approach of training the VAE enabled code without image observations aimed to avoid visual overfitting and reduce computational overhead to mimic resource and sensing constraint environments but have hard automation sequences which need to ensure safety and task completion even in times of sensor failures. Our LLM-driven query module enables symbolic-to-control translation through structured language prompts, offering an efficient mechanism to evaluate policy generalization in varied task setups. All our correct results are also attached on our Github Repo.

6.1. **Limitations and Future Work.** The current pipeline requires a two-stage process: VAE training followed by BC, adding overhead and checkpoint complexity. The VAE is sensitive to latent dimensionality and KL regularization, requiring careful hyperparameter tuning. Dataset preparation across Robosuite, Robomimic, and MimicGen demanded custom wrappers to align observation formats, posing severe integration challenges. Our hardware and time constraints limited joint training with visual inputs. Future work should include training a vision-based VAE using a single image-enabled run to extract compressed `latent_vae` on our custom tuned models, allowing image-free execution with improved spatial awareness ([7]). While our primary experiments ran for 300 epochs (result plots), extended training for a couple of our initial runs (2000 epochs) confirmed the same performance trends with incremental improvements. We also experimented with custom policy variants, including Coherent Soft Imitation Learning and Diffusion Policy, and plan to explore advanced offline RL techniques (e.g., IQL, TD3-BC) and online fine-tuning under our latent-augmented setting.

REFERENCES

[1] P. Prasad, D. Shah, M. Rajeshwer, and S. Vedire, "Project Repository: Reward-Driven Motion Primitives for Pick-and-Place," GitHub, 2024. [Online]. Available: https://github.com/prasadpr09/Reinforcement-learning--PickPlace

[2] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman, "Relay Policy Learning: Solving Long-Horizon Tasks via Imitation and Reinforcement Learning," in *Conf. Robot Learning (CoRL)*, PMLR, 2020, pp. 1025–1037.

[3] A. Hussein, M. Gaber, E. Elyan, and C. Jayne, "Imitation Learning: A Survey of Learning Methods," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–35, 2017, doi: 10.1145/3054912.

[4] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín, "What matters in learning from offline human demonstrations for robot manipulation," *arXiv preprint arXiv:2108.03298*, 2021.

[5] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox,
"MimicGen: A Data Generation System for Scalable Robot Learning using Human Demonstrations,"
arXiv preprint arXiv:2310.17596, 2023. [Online]. Available: https://arxiv.org/abs/2310.17596

[6] C. C. Beltran-Hernandez, N. Erbetti, and M. Hamaya, "SliceIt!—A Dual Simulator Framework for Learning Robot Food Slicing," *arXiv preprint arXiv:2404.02569*, 2024.

[7] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet, "Learning latent plans from play," in *Proc. Conf. Robot Learning (CoRL)*, vol. 100, pp. 1113–1132, 2020.

[8] D. Guo, "Kitchen Robot Case Studies: Learning Manipulation Tasks from Human Video Demonstrations," M.S. thesis, Carnegie Mellon University, Pittsburgh, PA, USA, Dec. 2023. [CMU-RI-TR-23-87].

[9] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi, "Dream to control: Learning behaviors by latent imagination," *arXiv preprint arXiv:1912.01603*, 2019.

[10] E. Xing, A. Gupta, S. Powers, and V. Dean, "KitchenShift: Evaluating Zero-Shot Generalization of Imitation-Based Policy Learning Under Domain Shifts," in *NeurIPS 2021 Workshop on Distribution Shifts*, 2021. [Online]. Available: https://openreview.net/forum?id=DdglKo8hBq0

[11] F. Torabi, G. Warnell, and P. Stone, "Behavioral Cloning from Observation," in *Proc. of the 27th Int. Joint Conf. on Artificial Intelligence (IJCAI)*, 2018, pp. 4950–4957, doi: 10.24963/ijcai.2018/687.

[12] ISRI-AIST, "RoboManipBaselines," GitHub repository, 2024. [Online]. Available: https://github.com/isri-aist/RoboManipBaselines

[13] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for Deep Data-Driven Reinforcement Learning," *arXiv preprint arXiv:2004.07219*, 2020.

[14] J. Liang *et al.*, "Code as Policies: Language Model Programs for Embodied Control," *arXiv preprint arXiv:2209.07753*, 2022.

[15] K. Lee, Y. Zhang, L. Snyder, X. Meng, S. Abbeel, and C. Finn, "Contextual reinforcement learning with finite sample guarantees," *arXiv preprint arXiv:2005.04611*, 2020.

[16] L. P. Cinelli *et al.*, "Variational Autoencoder," in *Variational Methods for Machine Learning with Applications to Deep Networks*, Springer, 2021, pp. 111–149, doi: 10.1007/978-3-030-70679-1_5.

[17] L. Tai, G. Paolo, and M. Liu,
"A Survey of Deep Network Solutions for Learning Control in Robotics,"
arXiv preprint arXiv:1612.07139, 2016. [Online]. Available: https://arxiv.org/abs/1612.07139

[18] N. Dilokthanakul, P. A. M. Mediano, M. Garnelo, M. C. H. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep Unsupervised Clustering with Gaussian Mixture Variational Autoencoders," arXiv preprint arXiv:1611.02648, 2017.

[19] P. Florence *et al.*, "Implicit Behavioral Cloning," in *Proc. of the 4th Conf. on Robot Learning (CoRL)*, 2021, pp. 158–168.

[20] R. de Lazcano, A. Kallinteris, J. J. Tai, S. R. Lee, and J. Terry, "Gymnasium Robotics," 2024. [Online]. Available: http://github.com/Farama-Foundation/Gymnasium-Robotics

[21] S. Osa, A. Puiutta, and E. K. R. Ko,
"An Algorithmic Perspective on Imitation Learning,"
Foundations and Trends in Robotics, vol. 7, no. 2–3, pp. 1–179, 2018. [Online]. Available: https://arxiv.org/abs/1809.02929

[22] V. Pallagani *et al.*, "On the Prospects of Incorporating Large Language Models (LLMs) in Automated Planning and Scheduling (APS)," in *Proc. Int. Conf. Automated Planning and Scheduling (ICAPS)*, vol. 34, no. 1, pp. 432–444, 2024, doi: 10.1609/icaps.v34i1.31503.

[23] W. Huang, C. Wang, Y. Li, R. Zhang, and L. Fei-Fei, "ReKep: Spatio-Temporal Reasoning of Relational Keypoint Constraints for Robotic Manipulation," *arXiv preprint arXiv:2409.01652*, 2024.

[24] X. Gao *et al.*, "VRKitchen: An Interactive 3D Virtual Environment for Task-Oriented Learning," 2019. [Online]. Available: https://xfgao.github.io/paper/Arxiv2019_VRKitchen.pdf

[25] Y. Zhu, J. Wong, A. Mandlekar, and R. Martín-Martín, "robosuite: A Modular Simulation Framework and Benchmark for Robot Learning," *arXiv preprint arXiv:2009.12293*, 2020.

[26] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," in *Proc. Int. Conf. Learning Representations (ICLR)*, 2014. [Online]. Available: https://arxiv.org/abs/1312.6114

[27] Z. Wang, J. Merel, S. Reed, G. Wayne, N. de Freitas, and N. Heess, "Robust Imitation of Diverse Behaviors," *arXiv preprint arXiv:1707.02747*, 2017. [Online]. Available: https://arxiv.org/abs/1707.02747

[28] R. Rahmatizadeh, P. Abolghasemi, L. Bölöni, and S. Levine, "Vision-Based Multi-Task Manipulation for Inexpensive Robots Using End-To-End Learning from Demonstration," *arXiv preprint arXiv:1707.02920*, 2017. [Online]. Available: https://arxiv.org/abs/1707.02920

[29] B. Xu, M. U. Din, and I. Hussain, "Conditional Variational Auto Encoder Based Dynamic Motion for Multi-task Imitation Learning," *Scientific Reports*, vol. 15, no. 1, p. 9196, 2025. doi: 10.1038/s41598-025-93888-4