

Bridging the Resource Gap: Deploying Advanced Imitation Learning Models onto Affordable Embedded Platforms

Haizhou Ge^{1*}, Ruixiang Wang^{2*}, Zhu-ang Xu^{3*}, Hongrui Zhu⁴, Ruichen Deng⁶,
Yuhang Dong⁴, Zeyu Pang⁵, Guyue Zhou¹, Junyu Zhang⁵, Lu Shi^{1†}

Abstract— Advanced imitation learning with structures like the transformer is increasingly demonstrating its advantages in robotics. However, deploying these large-scale models on embedded platforms remains a major challenge. In this paper, we propose a pipeline that facilitates the migration of advanced imitation learning algorithms to edge devices. The process is achieved via an efficient model compression method and a practical asynchronous parallel method Temporal Ensemble with Dropped Actions (TEDA) that enhances the smoothness of operations. To show the efficiency of the proposed pipeline, large-scale imitation learning models are trained on a server and deployed on an edge device to complete various manipulation tasks.

I. INTRODUCTION

The recent advancements in transformer architectures [1]–[3] and large-scale models [4], [5] have significantly propelled the field of embodied intelligence, enabling agents to interact with their environments in increasingly sophisticated ways. Among these advancements, imitation learning has emerged as a promising approach for addressing complex manipulation tasks [6]–[8]. By allowing agents to learn from expert demonstrations, imitation learning offers an efficient pathway to acquiring intricate behaviors without requiring extensive trial and error. The increasing reliance on imitation learning highlights its potential to overcome the challenges faced by traditional reinforcement learning methods, especially in environments that demand real-time decision-making and fine motor control [9]. This has led to its growing adoption in robotic manipulation [10], where rapid adaptability and high precision are critical.

However, the deployment of these advanced algorithms often requires expensive computing platforms with rich resources [11], [12], which poses significant challenges when attempting to implement them in resource-limited environments. Examples of such environments include autonomous drones [13], mobile robots [14], and other low-power, embedded systems where computational resources are constrained, and energy efficiency is paramount. The reliance

on powerful hardware limits the accessibility and practicality of these large-scale models in many real-world applications.

Existing solutions to mitigate these challenges, such as cloud-based inference, have their limitations. While offloading computation to remote servers can reduce the local processing burden, it introduces dependencies on network connectivity. These dependencies can lead to latency issues, potential privacy concerns, and reliability problems in unstable network conditions [15]. The drawbacks highlight the need for alternative approaches that can operate effectively within the constraints of resource-limited platforms, leading to the growing interest in Edge AI, which refers to the deployment of artificial intelligence algorithms and models directly on edge devices. However, their limited computational power, storage, and memory make it challenging to deploy sophisticated algorithms or models directly onto edge devices.

In this paper, we propose a pipeline (as shown in Fig. 1) to implement advanced imitation learning algorithms and models on low-cost embedded platforms to address the aforementioned problem. To bridge the cost gap and achieve a smoother transfer of the learning model from high-performance platforms to edge devices, we provide techniques for efficient model compression through input shape unification as well as parameter quantization. In the meantime, a practical asynchronous parallel method Temporal Ensemble with Dropped Actions (TEDA) is introduced to achieve high efficiency and avoid discontinuity during deployment. To validate the approach, we developed a complete system and conducted a series of robotic manipulation tasks involving both single-arm and dual-arm operations.

II. RELATED WORK

A. Embodied Intelligence System

Embodied intelligence systems aim to develop cognitive abilities and further motor skills through interactions with the environment [16]. Both the models for single modality such as visual [17] as well as language representations [18] and multimodal models have been introduced for robotic implementations, improving task decomposition, inference, and control [19]–[21], enabling more advanced robotic behavior. However, deploying these models for complex tasks poses significant challenges, including model deployment, accurate sensor data acquisition, and precise calibration, all of which drive up system costs. To address these challenges,

*These authors contributed equally to this work. †Corresponding author: shilu@air.tsinghua.edu.cn, ¹Tsinghua University, China, ²Harbin Institute of Technology, China, ³Jiangnan University, China, ⁴Zhejiang University, China, ⁵Tongji University, China, ⁶University of Pennsylvania, USA

We gratefully acknowledge the support of Discover Robotics and Horizon Robotics. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies.

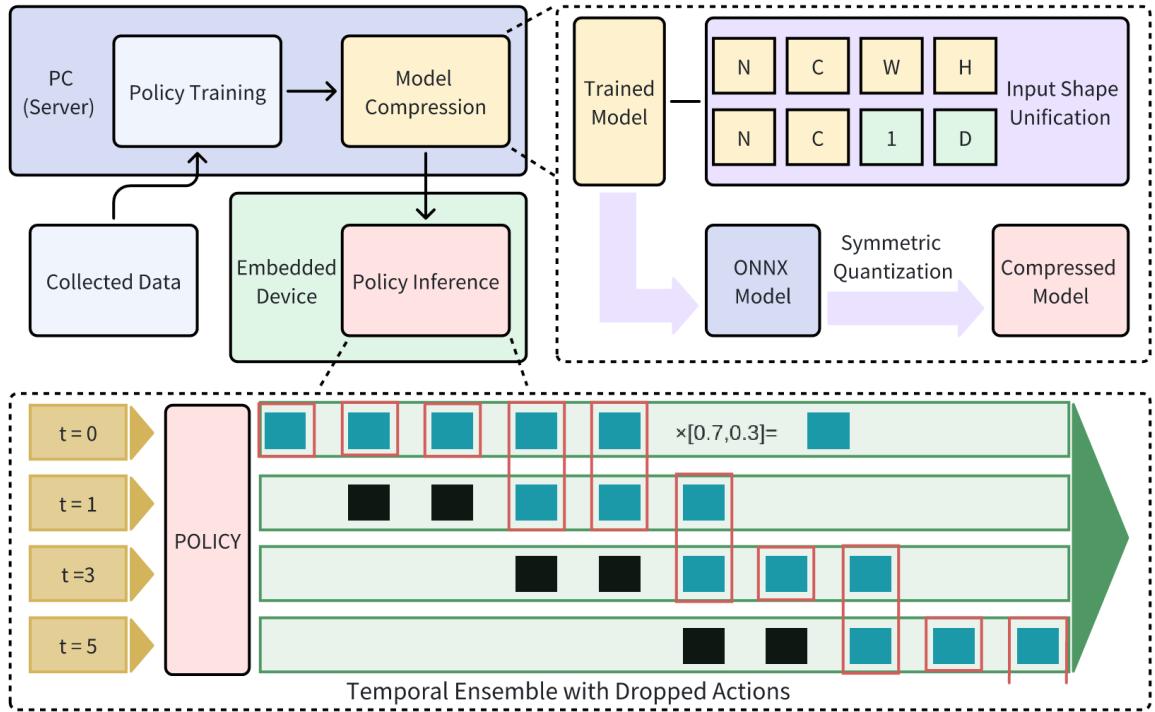


Fig. 1: Pipeline for deploying Imitation Learning algorithms on embedded devices. The policy trained with collected data is deployed into the embedded device through two key components: model compression and Temporal Ensemble with Dropped Actions (TEDA) with a trunking size of 5 for policy inference ($k = 5$). In the TEDA section, the blue block represents the normal action, while the black block represents the dropped action and the red box represents Temporal Ensemble when applying actions.

cost-effective solutions like ALOHA [22] have been developed, enabling end-to-end imitation learning directly from real demonstrations gathered using a custom teleoperation interface. These solutions highlight the ongoing efforts to balance the sophistication of AI models with the practical constraints of deployment in real-world environments.

B. Edge AI

Edge AI refers to the deployment of AI models and enables them to run directly on edge devices, close to the data source [23]. Edge AI significantly reduces latency, decreases bandwidth usage, and enhances reliability by minimizing dependence on cloud connectivity. It has been applied across various domains, including autonomous driving [24], smart surveillance [25], and other IoT devices [26]. However, optimizing AI models for low-power devices and effectively utilizing limited computational resources remain primary challenges [27]. Recent advancements in model compression techniques, such as pruning and quantization, have made it possible to deploy sophisticated AI algorithms on edge devices [28], [29], which provides the way for achieving high inference accuracy in resources-limited devices.

III. METHOD

In this section, we introduce the comprehensive pipeline (as shown in Fig. 1) to deploy advanced imitation learn-

ing algorithms into affordable embedded platforms. Firstly, policies are trained using the collected data for each task. Then, after unifying the input shapes, the model undergoes a compression phase via Symmetric Quantization (SQ). Following compression, the optimized model is deployed onto the embedded device, where it performs inference efficiently by utilizing TEDA. This technique not only improves the model's responsiveness but also ensures that it can make real-time decisions in dynamic environments. The subsequent subsections provide a detailed explanation of each stage in the pipeline, highlighting the methods and techniques employed to achieve robust performance on embedded platforms.

A. Model Compression

Due to the significant disparity in storage capacity and computational power between server-grade hardware and edge devices, directly deploying advanced models trained on servers to edge devices is impractical. To overcome this limitation, model compression is necessary. Before the compression process, we first apply an Input Shape Unification (ISU) technique. The inputs for the intelligent embodiments usually enjoy multi-modalities with different data structures, which leads to extra effort for storage and management. To deal with the problem, we express all the modalities

into the typical image tensor form of NCHW, where N represents the batch size, C represents the number of color channels, H and W describe the height and width of the image. For example, a joint position input with the shape of 1D can be viewed as an image tensor with one color channel, i.e., its shape can be represented as 111D. By using this consistent formation, redundant data transmission ¹ and extra computational overhead can be reduced.

Then, we compressed the model by converting its parameters from 32-bit floating-point (float) to 16-bit integer (int16) using Symmetric Quantization (SQ). This approach reduces the model's memory footprint and computational requirements, enabling more efficient inference on resource-constrained embedded processors. By employing symmetric quantization, which uses a uniform scaling factor across all values, we maintain computational simplicity while significantly enhancing the model's deployability on low-power hardware without sacrificing too much accuracy.

B. Temporal Ensemble with Dropped Actions (TEDA)

Action chunking, which groups individual actions together and executes them as one unit, effectively reduces cumulative error in IL but can introduce motion jitter. Temporal Ensemble (TE), which performs a weighted average over predicted actions at the same timestep, can smooth out this jitter, yet it requires inference before each action. Those are two key techniques used in recent advanced imitation learning algorithms [22], [30]. However, on embedded platforms, where the inference frequency is close to and even lower than the action execution frequency, inference before each action can cause discontinuous motion. This discontinuity not only increases the completion time but also tends to cause jitter, which reduces the success rate of the task.

To solve this problem, we propose a simple yet novel approach TEDA, which parallelizes the execution of action sequences and policy prediction. As shown in Fig. 1 and Alg. 1, this approach performs one prediction at $t = 0$ (t_0) to obtain k predicted actions $\hat{a}_{0:k}^{t_0}$, then executes the first action $\hat{a}_0^{t_0}$ and performs another prediction at t_1 . During the prediction process, the subsequent actions $\hat{a}_{1:N}^{t_0}$ ($N \leq k$) are executed in parallel. Then at t_{N+1} , the prediction is finished and we get the newly predicted actions $\hat{a}_{1:k+1}^{t_1}$. Since the subsequent actions $\hat{a}_{1:N}^{t_1}$ lag behind the current time step t_{N+1} , they will not be executed, which we called dropped actions. The prediction and execution of actions will continue into the next cycle.

Essentially, a dropped action is an action that has not been predicted at a specific time step. The number of dropped actions in each chunk can be calculated as $\lceil f_a/f_p \rceil$. Similar to the Temporal Ensemble in ACT [22], overlapping action chunks can lead to multiple predicted actions at the same time step. Following the previous example, at time t_{N+1} , we will have both the predicted action $\hat{a}_{N+1}^{t_0}$ at time t_0 and $\hat{a}_{N+1}^{t_1}$

¹The inconsistent formation can lead to operators that are unsupported by the edge AI acceleration chip and can only run on the CPU. This would result in repeated data transfers between the AI chip and the CPU, ultimately causing a significant decrease in inference speed.

Algorithm 1 Inference with TEDA

```

1: Given: trained policy  $\pi_\theta$ , chunk size  $k$ , episode length  $T_a$ , weight  $w$ , prediction frequency  $f_p$ , execution frequency  $f_a$ .
2: Initialize the number of dropped actions  $D = \lceil f_a/f_p \rceil$ .
3: Initialize maximum prediction steps  $T_p = 2 + \lceil (T_a - 1)/D \rceil$ .
4: Initialize a container  $B[0 : T_p, 0 : C]$ , where  $B[t_p]$  stores actions predicted at step  $t_p$ , and  $C = 1 + (T_p - 2) \times D + k$ .
5: Initialize actions  $\hat{a}_{0:k}^0$  predicted at  $t_p = 0$ .
6: for timestep  $t = 1, 2, \dots, T_a$  do
7:   if  $t \% D == 1$  then
8:     if  $t == 1$  then
9:       Add  $\hat{a}_{0:k}^0$  to  $B[0, 0 : k]$  respectively.
10:    else
11:      Add  $\hat{a}_{t-N:t-N+k}^{t_p}$  to  $B[t_p, t-1 : t-1+k]$  respectively.
12:    end if
13:    Start predicting  $\hat{a}_{t:t+k}^{t_p}$  with  $\pi_\theta(\hat{a}_{t:t+k}^{t_p} | o_t)$ .
14:     $t_p = t_p + 1$ .
15:  end if
16:  Obtain current step actions  $A_t = B[t_x : t_y, t]$ , where  $t_x$  and  $t_y$  are the indices of the first and last non-zero actions at step  $t$  in  $B$ , respectively.
17:  Apply  $a_t = \sum_i w_i A_t[i] / \sum_i w_i$ .
18: end for

```

at t_1 . Then we can average these actions to reduce the chance of jitter and increase the smoothness. TEDA does not affect the implementation of the original model and its training procedure but only adapts the inference process, which can be applied to any policy that uses action chunking, such as ACT, Diffusion Policy [30], etc.

IV. EXPERIMENT

In this section, we designed a comprehensive set of experiments to evaluate the performance of the proposed architecture in different manipulation tasks. For each platform and task, we collect data using an operable teleoperation system (as shown in Fig. 2) and then train the policy model on the server (PC) with the collected measurements. After that, the policy is transferred to an embedded platform (RDK X5)¹ with the proposed pipeline. Details regarding the system specifications and resources can be found in Tab. I.

A. Manipulation Platforms and Tasks

We designed three distinct hardware systems (as illustrated in Fig. 2) for performing various manipulation tasks: a system with a single arm mounted with a two-finger gripper for tasks that require simple and precise grasping actions, a system with dual arms mounted with two-finger grippers for more complex tasks that require coordinated movement between the two arms, and a system with a single arm mounted with a three-finger gripper for tasks that require fine motor control. The robotic arms, both leaders (used in data collection) and followers, utilized in our setup were 6-DoF AIRBOT² Play robotic arms.

Tasks for evaluation are illustrated in Fig. 3, 4, and 5, where we break down the tasks into more specific subtasks. These tasks encompass a series of operations, including

¹<https://developer.d-robotics.cc/rdkx5>

²<https://airbots.online/>

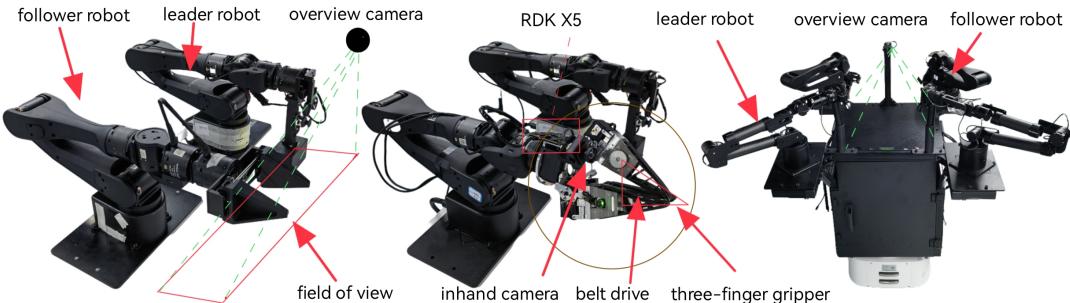


Fig. 2: Overview of teleoperation systems for data collection. *Left*: Single-arm with a two-finger gripper. *Middle*: Single-arm with a three-finger gripper. *Right*: Dual-arm with grippers.

TABLE I: Comparison of the hardware specifications between the server and the embedded device.

Platform	CPU	GPU/BPU	Memory
PC	Intel Core i9, 8-core/16-thread, @5.5GHz	NVIDIA RTX 4060, 8GB GDDR6	16GB DDR5, 5600MHz
X5	8-core Arm Cortex-A55, @1.5GHz	Bayers-architecture BPU @1.0GHz, 0.5GB, 10 TOPS	2GB/4GB LPDDR4

controlling the movement of robotic arms, managing the grasping actions of grippers, and manipulating the rotational motion in some of the cases.

B. Data Collection and Policy Training

We utilized a teleoperation system for data collection, where operator-controlled leader arms guided follower arms to gather precise and effective data. The dataset comprises joint positions of both leader and follower robots, along with RGB images (each at 480×640 resolution). Considering the complexity of the tasks, we recorded 120, 1000, and 400 timesteps for tasks 1 ~ 3, respectively, resulting in 300, 50, and 50 demonstration episodes. The sampling frequency was set at 25 Hz for all tasks. To improve spatial generalization, objects for manipulation were randomly positioned within a $5 \text{ cm} \times 5 \text{ cm}$ area during each data collection process.

The imitation learning model for evaluation in our experiments is the ACT model [10], which includes both the transformer and CNN structure. All models are trained with the same hyperparameters and the chunk size is 25.

C. Experiment Results

To evaluate the importance of different factors of our proposed structure, we first did the ablation tests on the Input Shape Unification (ISU) component. It is important to note that Symmetric Quantization (SQ) is essential, as the model cannot run on the X5 without it due to hardware limitations. Thus we are showing the effect of ISU in the ablation test. Then we did the overall evaluation among all the scenarios to complete various manipulation tasks.

1) *With/Without ISU*: In this study, we evaluate the impact of ISU on performance across three settings: baseline inference on the PC, SQ on the X5, and SQ combined with ISU on the X5. The accuracy and prediction time of the compressed models are presented in Table II. The results indicate that the model compressed solely with SQ requires approximately 47 times longer to perform inference on the X5 compared

to the uncompressed model running on a PC. However, by incorporating ISU, the inference time on the X5 is reduced by a factor of 6, without any loss in model accuracy. This demonstrates the effectiveness of ISU in optimizing inference time while maintaining model performance. Consequently, the model with SQ and ISU will be used for all subsequent experiments on the X5 platform.

TABLE II: Comparison results for with/without ISU. The prediction accuracy of the model is calculated as e_0/e_1 , where e_0 and e_1 are the MSE values between the predicted results and the ground truth for the model before and after compression, respectively among 25 trails of data.

Platform	Method	Accuracy	Prediction time
PC	—	1	0.013
X5	SQ	0.997	0.614
X5	SQ + ISU	0.997	0.103

TABLE III: Comparison of inference frequency. t_1 : observation time, t_2 : prediction time, t_3 : communication time, t_4 : execution time. For the baseline, the total inference time is the sum of all four components ($t_1 t_2 t_3 t_4$). In contrast, for ours (policy with TEDA), the total inference time is only the sum of t_3 and t_4 since the prediction and execution are parallel.

Platform	Time per step (s)				Total Inference Time
	t_1	t_2	t_3	t_4	
PC (Baseline)	0.013	0.012	0.001	0.040	0.066
X5 (Baseline)	0.017	0.103	0.001	0.040	0.161
X5 (Ours)	0.017	0.103	0.001	0.040	0.041

2) *Overall Task Results*: As introduced above, we designed three tasks to evaluate our pipeline. For each task, we compare the results across three scenarios: inference on a PC (PC (Baseline)), direct inference on the embedded



Fig. 3: *Task 1: Single-Arm Cup Stacking*: This task involves using a single robotic arm with a two-finger gripper to place a blue cup into a pink cup. First, the robotic arm approaches the blue cup and picks it up (Subtask 1: Grasp). Then, the arm adjusts the position of the blue cup to place it into the pink cup. Finally, the arm opens the gripper and returns to the initial position (Subtask 2: Place).



Fig. 4: *Task 2: Dual-Arm Cup Stacking*: This task involves using a two-arm system to stack cups into a bowl. Both robotic arms simultaneously approach these two cups, with each arm executing specific subtasks. First, the right arm grasps the pink cup (Subtask 1) while the left arm grasps the blue cup (Subtask 3). Once the cups are secured, both arms move the cups over the bowl. The right arm then places the pink cup into the bowl, opens its gripper, and returns to the initial position (Subtask 2). Following this, the left arm places the blue cup inside the pink cup, opens its gripper, and also returns to its initial position (Subtask 4).



Fig. 5: *Task 3: Whiteboard Eraser Rotation*: In this task, a single robotic arm mounted with a three-finger gripper is used to manipulate a whiteboard eraser. The arm first picks up the whiteboard eraser (Subtask 1: Grasp the board eraser), then rotates it using the conveyor belt of the three fingers to change its orientation (Subtask 2: Rotate the board eraser). Finally, the arm shifts the eraser to a specific location, activates the conveyor belt to place the eraser down, and then opens the gripper to complete the task (Subtask 3: Place the eraser).

device (**X5 (Baseline)**), and deployment on the embedded device using our proposed pipeline (**X5 (Ours)**). Each test was repeated 25 times to avoid bias.

As shown in Table IV and Table III, the policies evaluated on the PC consistently achieved the highest success rates across all tasks and subtasks. In contrast, the success rate of the baseline strategy on the X5 embedded device is significantly lower. A primary cause of task failure, as observed, is the jitteriness in motion, attributed to the low inference frequency during the robotic arm's movement. Additionally, the low inference frequency hinders the system's ability to make timely adjustments to changes in the scene, which further reduces the success rate. After incorporating the proposed pipeline, we can achieve a similar inference performance as PC does in terms of the success rate and completion time, especially for the Dual-Arm Cup Stacking task and the Whiteboard Eraser Rotation task. The results showcase the efficiency of our approach in more complex, multi-step tasks.

V. LIMITATIONS AND DISCUSSIONS

This paper presents a method for deploying high-performance algorithms on resource-constrained edge devices. By employing Input Shape Unification (ISU), we significantly improved the inference frequency of quantized models on the X5 embedded platform. Additionally, we introduced a practical asynchronous parallel method, Temporal Ensemble with Dropped Actions (TEDA), which ensures continuity during action execution. Experimental results validate the effectiveness of our approach, with success rates approaching those observed on PC platforms.

However, when the model prediction frequency is much lower than the action execution frequency, the number of dropped actions in TEDA will be large, which requires the model to have a very large chunk size. However, the maximum value of chunk size is limited by the episode length of the specific task, and a large chunk size may also potentially reduce the success rate of the task [10], so TEDA

TABLE IV: Success rate (%) comparison for the tasks with three scenarios.

Platform	Task 1		Task 2				Task 3		
	Grasp	Place	Subtask1	Subtask2	Subtask3	Subtask4	Subtask1	Subtask2	Subtask3
PC (Baseline)	92	84	88	80	84	72	76	72	56
X5 (Baseline)	68	56	64	52	60	48	60	56	44
X5 (Ours)	88	76	80	72	80	64	72	68	52

may no longer be applicable in this case.

REFERENCES

- [1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, “Attention is all you need,” *NeurIPS*, vol. 30, 2017. [1](#)
- [2] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. S. Ryoo, G. Salazar, P. R. Sanketi, K. Sayed, J. Singh, S. A. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. H. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitzkovich, “Rt-1: Robotics transformer for real-world control at scale,” *ArXiv*, vol. abs/2212.06817, 2022. [1](#)
- [3] H. Kim, Y. Ohmura, and Y. Kuniyoshi, “Transformer-based deep imitation learning for dual-arm robot manipulation,” *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 8965–8972, 2021. [1](#)
- [4] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. M. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan, “Do As I Can, Not As I Say: Grounding Language in Robotic Affordances,” 2022. [Online]. Available: <https://www.semanticscholar.org/paper/cb5e3f085caefdf1f3d5e08637ab55d39e61234fc> [1](#)
- [5] C. Chi, S. Feng, Y. Du, Z. Xu, E. A. Cousineau, B. Burchfiel, and S. Song, “Diffusion Policy: Visuomotor Policy Learning via Action Diffusion,” *ArXiv*, vol. abs/2303.04137, p. null, 2023. [Online]. Available: <https://www.semanticscholar.org/paper/bdb3bd30a49ea4c5b20b43dbd8f0eb59e9d80e2> [1](#)
- [6] K. Rakelly, A. Zhou, C. Finn, S. Levine, and D. Quillen, “Efficient off-policy meta-reinforcement learning via probabilistic context variables,” in *ICML*, 2019. [1](#)
- [7] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, “A Survey of Imitation Learning: Algorithms, Recent Developments, and Challenges,” Sept. 2023, arXiv:2309.02473 [cs, stat]. [Online]. Available: <http://arxiv.org/abs/2309.02473> [1](#)
- [8] I. Radakovovic, X. Wang, L. Pinto, and J. Malik, “State-only imitation learning for dexterous manipulation,” in *IROS*, 2021. [1](#)
- [9] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath, “Deep reinforcement learning: A brief survey,” *IEEE Signal Processing Magazine*, vol. 34, no. 6, pp. 26–38, 2017. [1](#)
- [10] T. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning fine-grained bimanual manipulation with low-cost hardware,” 2023. [1](#), [4](#), [5](#)
- [11] L. Ke, J. Wang, T. Bhattacharjee, B. Boots, and S. Srinivasa, “Grasping with chopsticks: Combating covariate shift in model-free imitation learning for fine manipulation,” in *International Conference on Robotics and Automation (ICRA)*, 2021. [1](#)
- [12] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, “One-shot imitation learning,” *ArXiv*, vol. abs/1703.07326, 2017. [1](#)
- [13] A. T. Azar, A. Koubaa, N. Ali Mohamed, H. A. Ibrahim, Z. F. Ibrahim, M. Kazim, A. Ammar, B. Benjdira, A. M. Khamis, I. A. Hameed, et al., “Drone deep reinforcement learning: A review,” *Electronics*, vol. 10, no. 9, p. 999, 2021. [1](#)
- [14] Z. Fu, T. Zhao, and C. Finn, “Mobile ALOHA: Learning Bimanual Mobile Manipulation with Low-Cost Whole-Body Teleoperation,” *ArXiv*, vol. abs/2401.02117, p. null, 2024. [Online]. Available: <https://www.semanticscholar.org/paper/fc3819a50705fc3cf90ab92f2a206b858fef3b19> [1](#)
- [15] T. Guo, “Cloud-based or on-device: An empirical study of mobile deep inference,” in *2018 IEEE International Conference on Cloud Engineering (IC2E)*. IEEE, 2018, pp. 184–190. [1](#)
- [16] J. Duan, S. Yu, H. L. Tan, H. Zhu, and C. Tan, “A survey of embodied ai: From simulators to research tasks,” *IEEE Transactions on Emerging Topics in Computational Intelligence*, 2022. [1](#)
- [17] A. Zeng, P. R. Florence, J. Tompson, S. Welker, J. Chien, M. Attaran, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee, “Transporter networks: Rearranging the visual world for robotic manipulation,” in *Conference on Robot Learning*, 2020. [1](#)
- [18] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *ArXiv*, vol. abs/1810.04805, 2019. [1](#)
- [19] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, “Voxposer: Composable 3d value maps for robotic manipulation with language models,” *arXiv preprint arXiv:2307.05973*, 2023. [1](#)
- [20] M. Shridhar, L. Manuelli, and D. Fox, “Cliport: What and where pathways for robotic manipulation,” in *Conference on robot learning*. PMLR, 2022, pp. 894–906. [1](#)
- [21] A. Stone, T. Xiao, Y. Lu, K. Gopalakrishnan, K.-H. Lee, Q. Vuong, P. Wohlhart, S. Kirmani, B. Zitzkovich, F. Xia, et al., “Open-world object manipulation using pre-trained vision-language models,” *arXiv preprint arXiv:2303.00905*, 2023. [1](#)
- [22] T. Zhao, V. Kumar, S. Levine, and C. Finn, “Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware,” *ArXiv*, vol. abs/2304.13705, p. null, 2023. [Online]. Available: <https://www.semanticscholar.org/paper/91eb20f923ea3b0246868902aef4e9bea572b800> [2](#), [3](#)
- [23] D. Liu, H. Kong, X. Luo, W. Liu, and R. Subramaniam, “Bringing ai to edge: From deep learnings perspective,” *Neurocomputing*, vol. 485, pp. 297–320, 2022. [2](#)
- [24] D. Katare, D. Perino, J. Nurmi, M. Warnier, M. Janssen, and A. Y. Ding, “A survey on approximate edge ai for energy efficient autonomous driving services,” *IEEE Communications Surveys & Tutorials*, 2023. [2](#)
- [25] R. Ke, Y. Zhuang, Z. Pu, and Y. Wang, “A smart, efficient, and reliable parking surveillance system with edge artificial intelligence on iot devices,” *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 8, pp. 4962–4974, 2020. [2](#)
- [26] M. Merenda, C. Porcaro, and D. Iero, “Edge machine learning for ai-enabled iot devices: A review,” *Sensors*, vol. 20, no. 9, p. 2533, 2020. [2](#)
- [27] R. Singh and S. S. Gill, “Edge ai: a survey,” *Internet of Things and Cyber-Physical Systems*, vol. 3, pp. 71–92, 2023. [2](#)
- [28] B. Reidy, M. Mohammadi, M. Elbtty, H. Smith, and Z. Ramtin, “Work in progress: real-time transformer inference on edge ai accelerators,” in *2023 IEEE 29th Real-Time and Embedded Technology and Applications Symposium (RTAS)*. IEEE, 2023, pp. 341–344. [2](#)
- [29] G. Chen, H. Meng, Y. Liang, and K. Huang, “Gpu-accelerated real-time stereo estimation with binary neural network,” *IEEE Transactions on Parallel and Distributed Systems*, vol. 31, no. 12, pp. 2896–2907, 2020. [2](#)
- [30] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” 07 2023. [3](#)