BAKU: An Efficient Transformer for Multi-Task Policy Learning

Siddhant Haldar* **Zhuoran Peng** **Lerrel Pinto**

New York University

Abstract: Training generalist agents capable of solving diverse tasks is challenging, often requiring large datasets of expert demonstrations. This is particularly problematic in robotics, where each data point requires physical execution of actions in the real world. Thus, there is a pressing need for architectures that can effectively leverage the available training data. In this work, we present BAKU, a simple transformer architecture that enables efficient learning of multi-task robot policies. BAKU builds upon recent advancements in offline imitation learning and meticulously combines observation trunks, action chunking, multi-sensory observations, and action heads to substantially improve upon prior work. Our experiments on 129 simulated tasks across LIBERO, Meta-World suite, and the Deepmind Control suite exhibit an overall 18% absolute improvement over RT-1 and MT-ACT, with a 36% improvement on the harder LIBERO benchmark. On 30 real-world manipulation tasks, given an average of just 17 demonstrations per task, BAKU achieves a 91% success rate. Videos of the robot are best viewed at baku-robot.github.io.

1 Introduction

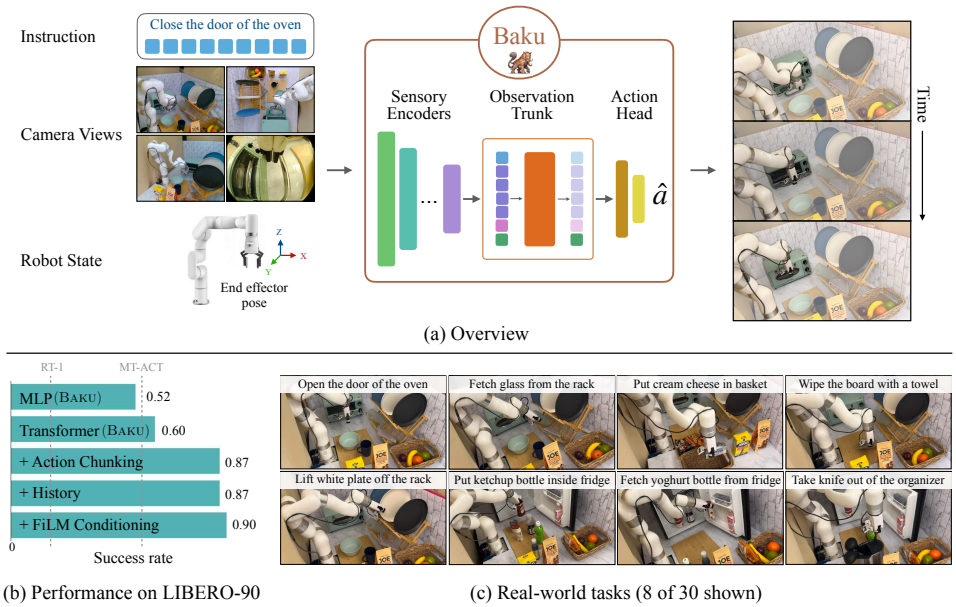
Learning generalist policies that can solve multiple tasks is a long standing problem in decision making and robotics. While significant advances have been made in computer vision [1, 2] and natural language processing [3, 4, 5], algorithms that can effectively do so for physical agents are far behind. A key reason for this is the scale of available data. While large-scale datasets in vision and language can readily be amassed from the Internet, robotics presents a unique challenge. Given its interactive nature, data acquisition requires physical engagement with the world, making robot data considerably more laborious to obtain in terms of both time and financial costs.

A prominent approach for training multi-task policies is to bite the bullet and collect large amounts of data, often by contracting teleoperators [6, 7, 8]. However, policies trained on such data are quite inefficient, often achieving performance far below independently trained single-task policies [9, 10, 11]. The current best answer to solve this problem is unfortunately to collect even more demonstration data from experts.

In this work, we present BAKU, a simple architecture for multi-task policy learning that provides highly efficient training, particularly in data-scarce problems such as robotics. BAKU builds upon recent work in multitask learning [12, 7] and has three key features. First, a transformer encoder that fuses information from multiple modalities like vision and language while incorporating temporal context. Second, a FiLM-conditioned [13] visual encoder helps the model learn task-specific representations by adapting the encoder to the task. Third, an action prediction head that is separated from the observational encoding trunk, enabling BAKU to be easily retrofitted with state-of-the-art action generation models [14, 15, 16, 17]. The novelty of BAKU hence lies in carefully combining these ideas to produce a new transformer architecture particularly suited for multitask decision making.

*Correspondence to: siddhanthaldar@nyu.edu

8th Conference on Robot Learning (CoRL 2024), Munich, Germany.



(a) Overview

(b) Performance on LIBERO-90

(c) Real-world tasks (8 of 30 shown)

Figure 1: **(a)** We present BAKU, a simple transformer architecture learning multi-task policies across a diverse range of tasks. BAKU encodes inputs from different modalities using modality-specific encoders. The encoded representations are merged in an observation trunk before predicting actions through an action head. **(b)** We develop a unified policy for 90 tasks in the LIBERO-90 benchmark, discussing design choices that impact multi-task performance. **(c)** On our xArm robot, BAKU can learn a single multi-task policy for 30 tasks with an average of 17 demonstrations collected per task.

To demonstrate the effectiveness of BAKU, we run extensive experiments on 129 simulated tasks across LIBERO [18], Meta-World [19], and DeepMind Control [20], and 30 robotic manipulation tasks on an xArm robot (see Fig. 1). Our main findings are summarized below:

1. BAKU exhibits an overall 18% absolute performance improvement over prior state-of-the-art multi-task learning algorithms on 129 tasks across 3 simulated environment suites (Section 3.1). BAKU sets a state-of-the-art performance on LIBERO with 90% average success rate, a 36% absolute improvement over prior work (Table 1).
2. On real-world tasks, with an average of 17 demonstrations per task, BAKU achieves an average success rate of 91% across 30 diverse tasks in a multi-task kitchen environment, with randomized object initialization. This outperforms prior state-of-the-art algorithms by 35% (Section 3.2).
3. Through an ablation analysis, we study the importance of each component in BAKU (Section 3.4), particularly the role of action chunking [21] and a multimodal action head in boosting performance, especially in our real-world experiments.

All of our datasets, and training and evaluation code will be made publicly available. Videos of our trained policies can be seen here: baku-robot.github.io.

2 BAKU

The design of multi-task learning algorithms involves numerous decisions regarding model architecture and component selection. This often results in complex architectures where the importance of individual components is sometimes unclear. In this work, we perform a systematic and thorough ablation study across the various multi-task learning architectures proposed by prior works [7, 12, 22] and introduce BAKU, a simple architecture for multi-task policy learning. To facilitate our analysis, we divide the overall model architecture into three main components: sensory encoders, an observation trunk, and an action head. Sensory encoders process raw sensor inputs from different modalities

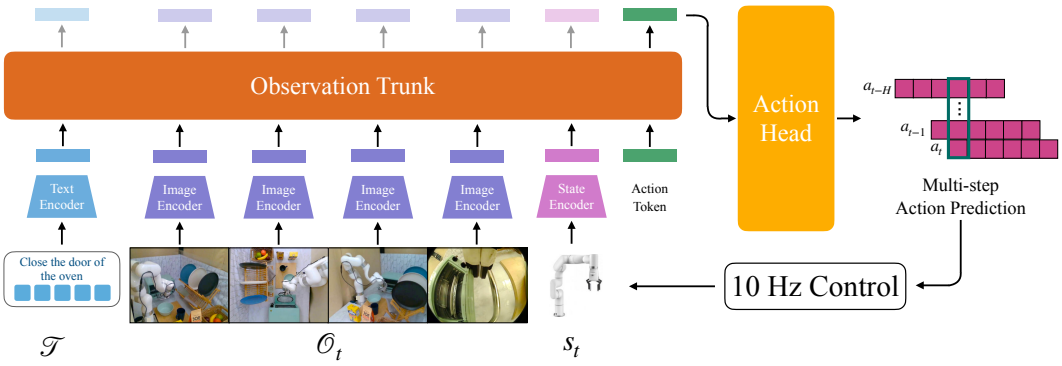


Figure 2: Overview of BAKU, broken down into modality-specific sensory encoders, an observation trunk, and an action head predicting a chunk of actions. BAKU takes as input observations from multiple camera views $\mathcal{O}_t$, robot proprioceptive state s_t and a task instruction $\mathcal{T}$ and enables performing closed-loop control at 10Hz in our real world experiments on the xArm.

into useful feature representations. The observation trunk combines the encoded information from the different modalities. Finally, the action head utilizes the combined information to predict actions. Below, we describe these three components in detail, with additional algorithmic details provided in Appendix B.

2.1 Sensory Encoders

In the real-world, robots encounter diverse data modalities, including vision, depth feedback, proprioceptive feedback, and task instructions in various forms such as text, goal images, or task videos. In BAKU, we focus on vision, robot proprioception, and text or goal image based task instructions. For vision, we use a ResNet-18 [23] visual encoder to process images of the scene, enhanced with a FiLM [13] layer to integrate task-specific information. Robot proprioception data is processed through a two-layer multilayer perception (MLP) encoder. For text, we use a 6-layer version of MiniLM [24] provided in Sentence Transformers [25]. We project the representations obtained from all modalities to the same dimensionality through additional MLP layers, to facilitate combining the encoded information. We have included a description of FiLM conditioning in Appendix B.1.

2.2 Observation Trunk

The encoded inputs from all sensory modalities are combined in the observation trunk. We explore two variants of the trunk network:

Multilayer Perceptron (MLP) The encoded inputs are concatenated into a single feature vector and passed through a multilayer perceptron. When using a history of observations, the inputs corresponding to all time steps are concatenated.

Transformer Each encoded input is treated as an observation token and passed through a transformer decoder network [26]. A learnable action token is appended to the list of observation tokens and used to predict the action. When using a historical observation, a separate action token is added for each time step to enable predicting actions for all time steps. A causal mask is applied to the transformer to ensure that actions are predicted solely based on past observations.

Both variants output action feature vectors (corresponding to the action tokens for a transformer), which are then passed through an action head to predict actions.

2.3 Action Head

The final component of our architecture is the action head, an action prediction module that takes as input the action feature vectors obtained from the observation trunk and predicts the corresponding actions. An independent action prediction module enables us to unify several state-of-the-art action generation models within the same framework. We experiment with five action head variants: vanilla MLP, Gaussian Mixture Model (GMM) [27], Behavior Transformer (BeT) [15], Vector-Quantized Behavior Transformer (VQ-BeT) [16], and diffusion policy [28, 17, 29]. Considering the temporal correlation in robot movements, we follow prior work [21, 12] and include action chunking with exponential temporal averaging to produce smoother behaviors and counteract the covariate shift often seen in low-data imitation learning scenarios. In contrast to previous works [21, 12] that decode actions for each time step separately, we predict the action chunk as a single concatenated vector. We find that this simplification improves performance (see Table 1). More details about each action head variant has been provided in Appendix B.2 along with details about the exponential temporal smoothing technique in Appendix B.3.

2.4 Putting it all together

Our proposed architecture is depicted in Figure 2. Through extensive experimentation (see Section 3), our final architecture includes a modified FiLM-conditioned ResNet-18 vision encoder (provided with the LIBERO benchmark [18]), an MLP encoder for robot proprioception, and a pre-trained text encoder for task instructions. For environments with multiple camera views, we use a common visual encoder across all views. We provide only the current observation as an input and the observation trunk uses a causal transformer decoder architecture [30]. The base version of our model uses an MLP action head with action chunking and temporal smoothing to produce smoother motions. We also experiment with multimodal variants of action heads and have provided the results in Section 3.4.

The parameter counts are approximately 2.1M for the sensory encoders, 6.5M for the observation trunk, and 1.4M for the action head, bringing the total model size to approximately 10M parameters.

3 Experiments

Our experiments are designed to answer the following questions: (a) How well does BAKU work for multi-task learning? (b) How does BAKU perform on real-world tasks? (c) How does BAKU perform on long-horizon tasks? (d) What design decisions affect multi-task policy learning? Additional results and analysis have been provided in Appendix F.

Simulation Tasks: We experiment with 90 manipulation tasks from the LIBERO-90 benchmark [18], 30 manipulation tasks from Meta-World suite [19], and 9 locomotion tasks from DeepMind Control Suite (DMC) [20]. Figure 3 depicts the simulated environments. For LIBERO-90, we use 50 demonstrations per task provided with the benchmark and use images from third-person and gripper camera views, as well as robot proprioception as input. For Meta-World, we obtain 35 demonstrations per task from an expert policy trained with demonstration-guided reinforcement learning [31, 32], using only the third-person view as input. We use images of size 128×128 for LIBERO-90 and 84×84 for Meta-World. For DMC, we train state-based locomotion policies using 500 demonstrations per task obtained from experts trained with DrQ-v2 [33]. All evaluations are conducted using 10 policy rollouts per task. More details have been included in Appendix C.

Robot Tasks: Our real-world experiments are performed on a Ufactory xArm 7 robot with an xArm Gripper in a multi-task kitchen environment. The policies are trained on RGB images of size 128×128 obtained from four different camera views, including an egocentric camera attached to the robot gripper. The action space comprises the robot end effector pose and the gripper state. We collect a total of 520 demonstrations across 30 tasks, averaging 17 demonstrations per task. The demonstrations were collected using a VR-based teleoperation system [34] at a 30Hz frequency. The learned policies are deployed at 10Hz. More details can be found in Appendix D.

Table 1: Performance of multi-task policies learned using BAKU on 3 simulated benchmarks - LIBERO-90, Meta-World, and DM Control - and a real xArm robot. We observe that BAKU significantly outperforms prior work on both simulated and real world tasks.

Method	LIBERO-90 (90 tasks)	Meta-World (30 tasks)	DMC (9 tasks)	Real Robot (20 tasks)
RT-1	0.16	0.65	0.66	0.37
MTACT	0.54	0.13	0.59	0.56
BAKU **(Ours)**	**0.9**	**0.79**	**0.7**	0.86
BAKU w/ VQ-BeT **(Ours)**	**0.9**	0.78	**0.7**	**0.91**

Strong Baselines: We compare BAKU with two state-of-the-art approaches, MT-ACT and RT-1. We have provided a detailed description of both in Appendix E.

3.1 How well does BAKU work for multi-task learning?

We evaluate the multi-task performance of BAKU on 90 tasks from the LIBERO-90 benchmark, 30 tasks from Meta-World, and 9 tasks from DMC. Table 1 compares the performance of BAKU with our baselines, RT-1 [7] and MT-ACT [12]. BAKU outperforms the strongest baseline by 36% and 14% on LIBERO-90 and Meta-World respectively, demonstrating more effective multi-task learning on complex manipulation tasks. On the simpler DMC locomotion tasks, BAKU outperforms the strongest baseline by 4%. Overall, these results suggest that BAKU more effectively leverages relationships between tasks to achieve superior multi-task learning performance compared to prior methods.

3.2 How does BAKU perform on real-world tasks?

We evaluate BAKU on 30 manipulation tasks in our real-world kitchen environment, comparing it with MT-ACT and RT-1. During evaluations, the xArm was always initialized at the same pose and the objects being manipulated were placed in a fixed set of positions for all methods. We conducted 5 evaluation runs per task, totaling 150 trials per method. Table 1 includes our results. We observe that BAKU achieves an 86% success rate across all tasks, outperforming the strongest baseline by 30%. Replacing the MLP action head with a multimodal VQ-BeT [16] head further improves the success rate to 91%, outperforming the strongest baseline by 35%. The real-world rollouts have been included in Appendix D. Appendix F.1 provides the task-wise performance for each method. Overall, these results indicate BAKU's promise for deploying multi-task policies on real-world robotic systems.

3.3 How does BAKU perform on long-horizon tasks?

We also evaluate BAKU on long-horizon tasks in the simulated LIBERO-10 benchmark and our real-world multi-task kitchen environment. Table 2 provides the results on 10 tasks in LIBERO-10 and 5 long-horizon tasks in the real kitchen environment, each composed of two shorter tasks chained sequentially. We use 50 demonstrations per task for LIBERO-10 and an average of 19 demonstrations per task for the real robot. We observe that BAKU significantly outperforms our strongest baseline, MT-ACT, on these long horizon tasks, achieving on average 19% higher success rate. This highlights BAKU's ability to learn policies that can effectively plan and execute sequences of actions over extended time horizons. Real world rollouts of these long-horizon tasks have been included in Appendix D with the task-wise performance and demonstration details in Appendix F.1.

3.4 What design decisions affect multi-task policy learning?

As described in Section 2, our multi-task policy architecture consists of three main components: sensory encoders, an observation trunk, and an action head. In this section, we analyze the design choices within each component and their effect on overall multi-task performance. We consider BAKU with an MLP action head (described in Section 2.4) as our base model. For ablations, we vary only a single property at a time while keeping all other aspects identical. This experimental setup allows us to clearly isolate the impact of individual design decisions. We examine different observation trunks,

Table 2: Performance of multi-task policies learned using BAKU on long-horizon tasks in the LIBERO-10 simulated benchmark and a real xArm robot. We observe that BAKU significantly outperforms prior work on both simulated and real world tasks.

Method	LIBERO-10 (10 tasks)	Real Robot (5 tasks)
MT-ACT	0.68	0.64
BAKU **(Ours)**	**0.86**	**0.84**

Table 3: Study of design decisions for BAKU that affects multi-task performance.

Category	Variant	LIBERO-90	Meta-World	DMC
Observation Trunk	MLP	0.81	0.78	0.68
	Transformer	**0.90**	**0.79**	**0.70**
Model Size	4.4M	0.85	0.78	0.68
	10M	**0.9**	0.79	**0.70**
	31M	0.87	**0.81**	**0.70**
	114M	0.19	**0.81**	0.68
Action Head	MLP	**0.90**	**0.79**	**0.70**
	GMM	0.84	0.65	0.67
	BeT	0.89	0.78	0.60
	VQ-BeT	**0.90**	0.78	**0.70**
	Diffusion	0.89	0.45	0.61
Action Chunking	✗	0.76	0.78	**0.74**
	✓	**0.90**	**0.79**	0.70
Observation History	✗	**0.90**	0.79	**0.70**
	With last-step loss	0.54	0.08	0.37
	With multi-step loss	**0.90**	**0.82**	0.68
Goal Modality	Text	0.90	0.79	N/A
	Goal Image	0.88	**0.81**	N/A
	Intermediate Image	**0.91**	0.80	N/A
FiLM	✗	0.87	**0.79**	N/A
	✓	**0.90**	**0.79**	N/A

model sizes, action heads, goal modalities, and the use of action chunking, observation history, and task conditioning through FiLM [13]. The results of our ablation study are provided in Table 3 with more analysis in Appendix F. The results provide insights into which components and properties are most important for effective multi-task learning with BAKU.

Effect of Observation Trunk: We experiment with two trunk types: an MLP and a transformer architecture. In Table 3, we observe a slight performance dip when using an MLP trunk on Meta-World and DMC. For LIBERO-90, our most complex simulated benchmark, an MLP trunk resulted in a 9% lower success rate than a transformer trunk. This highlights the efficacy of transformers for modeling complex relationships between observations from multiple sensing modalities and actions.

Effect of Model Size: We study the effect of model size on multi-task performance by evaluating configurations with 4.4M, 10M, 31M, and 114M parameters. For each variant, we vary the size of the observation trunk and the action head while keeping the sensory encoders constant. The results in Table 3 show that the 4.4M, 10M, and 31M parameter models achieve similar performance across benchmarks. Surprisingly, the largest 114M parameter model severely underperforms on the harder LIBERO-90 benchmark. We suspect this poor performance may have been due to overfitting on the training data with a larger capacity. Based on these results, we use the 10M parameter model for BAKU since it is the smallest model with the best performance on 2 of the 3 simulated benchmarks.

Effect of Action Head: We compare the performance of BAKU retrofitted with five different action heads: MLP, GMM [27], BeT [15], VQ-BeT [16], and diffusion [17]. Having an independent action head enables us to extend these state-of-the-art action generation models to multi-task settings. Table 3 shows that on simulated benchmarks, a simple MLP head performs just as well or better than other multimodal action heads. Among the multimodal variants, VQ-BeT achieves the best performance. As a result, we also evaluate BAKU with a VQ-BeT action head in our real-world setup (Table 1). On the real robot, having a multimodal action head proves advantageous with the VQ-BeT head achieving a success rate of 91%, a 5% improvement over an MLP action head. Hence, our experiments demonstrate that while multimodal heads may provide benefits for real-world deployment, a simple MLP head can perform well on simulated data with limited behavioral diversity.

Effect of Action Chunking: We study the effect of action chunking on multi-task policy performance. For the image-based LIBERO-90 and Meta-World benchmarks, we predict a chunk of 10 future actions. For the locomotion tasks in DMC, we predicted 3 future actions. Based on the results in Table 3, we observe the largest difference on LIBERO-90, where removing action chunking and instead predicting a single action led to a 14% drop in performance. In contrast, there is no perceptible difference in performance on Meta-World with and without chunking. For the locomotion domains in DMC, we see a 4% performance increase when removing chunking. Hence, action chunking benefits manipulation tasks while mildly hindering locomotion tasks from our experiments.

Effect of Observation History: We study the effect of using an observation history on multi-task performance. As shown in Table 3, naively using an observation history where the action prediction loss is only computed for the last time step significantly degrades performance. However, since BAKU uses a transformer observation encoder, it allows predicting actions for all observations in the history and computing the prediction loss over all time steps. Empirically, we found this multi-step prediction loss provides richer supervision and improves the single-step loss performance by an average of 47% across all benchmarks. However, incorporating an observation history with multi-step action prediction did not noticeably improve overall policy performance compared to using no history. Therefore, our final architecture only uses the most recent observation as an input.

Effect of Goal Modality: We experiment with 3 different goal modalities: text instruction, goal image, and intermediate goal image. The text instructions are directly obtained from the task data. The goal image is obtained by randomly sampling a demonstration from the training dataset and taking the last frame. For an intermediate goal image, we consider this randomly sampled task demonstration, and for every time step, treat the frame k steps in the future as the goal image [35]. Table 3 contains the results on LIBERO-90 and Meta-World, as goal images do not apply to the state-based DMC tasks. We set k to 50 steps for LIBERO-90 and 30 steps for Meta-World. Since LIBERO-90 has two camera views, we use the third-person view to obtain goal images. We observe that all three goal modalities show a similar performance with slight variations. Overall, our approach supports different goal representations with only minor variations in performance.

Effect of FiLM Conditioning: We examine the impact of using a FiLM-conditioned vision encoder for language-guided multi-task policies. As shown in Table 3, on the image-based LIBERO-90 and Meta-World benchmarks, a FiLM-conditioned vision encoder performs equally well or better than an unconditional encoder. FiLM conditioning allows modulating the vision encoder's parameters conditioned on the language input. This provides an effective way to fuse visual and linguistic information for solving tasks. Therefore, BAKU employs a FiLM-conditioned vision encoder for our image-based experiments.

4 Related Work

Imitation Learning (IL) IL [36] refers to the setting where agents learn from demonstrations without access to environment rewards. IL can be broadly categorized into Behavior Cloning (BC) [37, 38] and Inverse Reinforcement Learning (IRL) [39, 40]. BC solely learns from offline

demonstrations but suffers on out-of-distributions samples [41] whereas IRL focuses on learning a robust reward function through online interactions but suffers from sample inefficiency [31, 32]. In this work, we focus on using BC to learn multi-task policies. In recent years, there have been significant advances in single-task behavior cloning with the development of multimodal action generation models using GMMs [27, 14], EBMs [42], BeT [15, 43, 16], and diffusion [28, 17, 29, 44]. There has also been notable progress in solving long-horizon tasks through imitation learning with some works relying solely on robot data [45, 46, 47, 21, 48, 49] while others attempt to bootstrap learning from human data [35]. Further, these advances in policy learning combined with significant strides in self-supervised representation learning [50, 51, 52] have enabled deploying these policies in messy and unpredictable environments such as our homes [8] as well as zero-shot deployment in-the-wild [53, 54]. However, despite the large body of work advancing single-task robotic policy learning, there still exists a gap between single-task and multi-task performance for policy learning [9, 10, 11].

Multi-task Learning Robotics has a long history of multi-task learning. There is a significant body of work focusing on learning policies for robotic grasping with the aim of generalizing to new tasks [55, 56, 57, 58, 59], robotic language understanding [60, 61, 62, 63], and framing multi-task learning as a goal reaching problem [64, 65, 66]. Additionally, several works have collected varied multi-task robotics datasets [45, 18, 12, 67, 68]. Recently, there has been an increased use of transformer-based architectures for multi-task robot learning, spanning across robot navigation [53, 54], locomotion [69, 70, 71, 72], and manipulation [7, 73, 43, 12]. While most of these works use text conditioning for task specification, some go beyond text to use goal images [43, 74] and videos [75, 76] as well. Another emerging trend is co-training these robot policies with tasks such as visual question answering and image captioning [77, 73], to develop more generalizable policies. Overall, multi-task learning has been widely applied in robotics and, more recently, using high-capacity transformer models to learn robot control policies has become common practice in the field. Despite their effectiveness, the architectures for these policies often become complicated, with the necessary components sometimes being unclear. Our proposed model, BAKU, combines key ideas from prior work into a single architecture to produce a model that is both simple and outperforms state-of-the-art methods in multi-task policy learning.

5 Conclusion and Limitations

In this work, we presented BAKU, a simple transformer architecture that demonstrates improved multi-task policy learning performance on a variety of simulated and real-world domains compared to prior state-of-the-art methods. We recognize a few limitations in this work: (a) In our real-world experiments, while BAKU achieved good performance on most tasks, it struggled on some precise manipulation tasks, such as *opening an oven door* or *placing a tea bottle in the fridge*. This suggests that data sharing across tasks of varying difficulty may hinder performance on more precise skills. Developing techniques to learn a single policy for different task complexity levels could help address this. (b) Currently, we focus on performing a single skill at a time. Developing algorithms capable of chaining multiple such skills can enable effective long-horizon robot manipulation. (c) In this work, we primarily studied the policy architecture and did not analyze the generalization benefits of multi-task learning as the number of tasks increases. A study of the emergence of such generalization with greater task diversity would be another interesting direction. Overall, we hope that BAKU serves as an important step towards developing multi-task policies capable of performing precise robotic manipulation.

References

[1] J. Betker, G. Goh, L. Jing, T. Brooks, J. Wang, L. Li, L. Ouyang, J. Zhuang, J. Lee, Y. Guo, et al. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8, 2023.

[2] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. L. Denton, K. Ghasemipour, R. Gontijo Lopes, B. Karagol Ayan, T. Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35: 36479–36494, 2022.

[3] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[4] M. Reid, N. Savinov, D. Teplyashin, D. Lepikhin, T. Lillicrap, J.-b. Alayrac, R. Soricut, A. Lazaridou, O. Firat, J. Schrittwieser, et al. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*, 2024.

[5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[6] A. Mandlekar, Y. Zhu, A. Garg, J. Booher, M. Spero, A. Tung, J. Gao, J. Emmons, A. Gupta, E. Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, pages 879–893. PMLR, 2018.

[7] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[8] N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.

[9] T. Yu, S. Kumar, A. Gupta, S. Levine, K. Hausman, and C. Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33:5824–5836, 2020.

[10] E. Parisotto, J. L. Ba, and R. Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[11] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[12] H. Bharadhwaj, J. Vakil, M. Sharma, A. Gupta, S. Tulsiani, and V. Kumar. Roboagent: Generalization and efficiency in robot manipulation via semantic augmentations and action chunking. *arXiv preprint arXiv:2309.01918*, 2023.

[13] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[14] 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.

[15] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[16] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[17] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023.

[18] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[19] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[20] Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. d. L. Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, et al. Deepmind control suite. *arXiv preprint arXiv:1801.00690*, 2018.

[21] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[22] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.

[23] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[24] W. Wang, F. Wei, L. Dong, H. Bao, N. Yang, and M. Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[25] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. URL https://arxiv.org/abs/1908.10084.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[27] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. In *Conference on robot learning*, pages 1113–1132. PMLR, 2020.

[28] T. Pearce, T. Rashid, A. Kanervisto, D. Bignell, M. Sun, R. Georgescu, S. V. Macua, S. Z. Tan, I. Momennejad, K. Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.

[29] M. Reuss, M. Li, X. Jia, and R. Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

[30] A. Karpathy. mingpt: A minimal pytorch re-implementation of the openai gpt. https://github.com/karpathy/minGPT, 2021.

[31] S. Haldar, V. Mathur, D. Yarats, and L. Pinto. Watch and match: Supercharging imitation with regularized optimal transport. In *Conference on Robot Learning*, pages 32–43. PMLR, 2023.

[32] S. Haldar, J. Pari, A. Rai, and L. Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023.

[33] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Mastering visual continuous control: Improved data-augmented reinforcement learning. *arXiv preprint arXiv:2107.09645*, 2021.

[34] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto. Open teach: A versatile teleoperation system for robotic manipulation. *arXiv preprint arXiv:2403.07870*, 2024.

[35] C. Wang, L. Fan, J. Sun, R. Zhang, L. Fei-Fei, D. Xu, Y. Zhu, and A. Anandkumar. Mimicplay: Long-horizon imitation learning by watching human play. In *7th Annual Conference on Robot Learning*, 2023.

[36] A. Hussein, M. M. Gaber, E. Elyan, and C. Jayne. Imitation learning: A survey of learning methods. *ACM Computing Surveys (CSUR)*, 50(2):1–35, 2017.

[37] D. Pomerleau. An autonomous land vehicle in a neural network. *Advances in Neural Information Processing Systems*, 1, 1998.

[38] F. Torabi, G. Warnell, and P. Stone. Recent advances in imitation learning from observation. *arXiv preprint arXiv:1905.13566*, 2019.

[39] A. Y. Ng, S. J. Russell, et al. Algorithms for inverse reinforcement learning. In *Icml*, volume 1, page 2, 2000.

[40] P. Abbeel and A. Y. Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004.

[41] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011.

[42] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[43] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[44] L. Chen, S. Bahl, and D. Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *Conference on Robot Learning*, pages 2012–2029. PMLR, 2023.

[45] 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.

[46] M. Heo, Y. Lee, D. Lee, and J. J. Lim. Furniturebench: Reproducible real-world benchmark for long-horizon complex manipulation. *arXiv preprint arXiv:2305.12821*, 2023.

[47] Y. Chen, C. Wang, L. Fei-Fei, and C. K. Liu. Sequential dexterity: Chaining dexterous policies for long-horizon manipulation. *arXiv preprint arXiv:2309.00987*, 2023.

[48] T. Lin, Y. Zhang, Q. Li, H. Qi, B. Yi, S. Levine, and J. Malik. Learning visuotactile skills with two multifingered hands. *arXiv:2404.16823*, 2024.

[49] K. Sridhar, S. Dutta, D. Jayaraman, J. Weimer, and I. Lee. Memory-consistent neural networks for imitation learning. *arXiv preprint arXiv:2310.06171*, 2023.

[50] X. Chen, S. Xie, and K. He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9640–9649, 2021.

[51] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[52] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[53] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*, 2023.

[54] A. Sridhar, D. Shah, C. Glossop, and S. Levine. Nomad: Goal masked diffusion policies for navigation and exploration. *arXiv preprint arXiv:2310.07896*, 2023.

[55] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[56] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016.

[57] A. Gupta, A. Murali, D. P. Gandhi, and L. Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018.

[58] U. Viereck, A. Pas, K. Saenko, and R. Platt. Learning a visuomotor controller for real world robotic grasping using simulated depth images. In *Conference on robot learning*, pages 291–300. PMLR, 2017.

[59] H.-S. Fang, C. Wang, H. Fang, M. Gou, J. Liu, H. Yan, W. Liu, Y. Xie, and C. Lu. Anygrasp: Robust and efficient grasp perception in spatial and temporal domains. *IEEE Transactions on Robotics*, 2023.

[60] H. Mei, M. Bansal, and M. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

[61] C. Lynch and P. Sermanet. Language conditioned imitation learning over unstructured data. *arXiv preprint arXiv:2005.07648*, 2020.

[62] S. Stepputtis, J. Campbell, M. Phielipp, S. Lee, C. Baral, and H. Ben Amor. Language-conditioned imitation learning for robot manipulation tasks. *Advances in Neural Information Processing Systems*, 33:13139–13150, 2020.

[63] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[64] A. Raffin, A. Hill, R. Traoré, T. Lesort, N. Díaz-Rodríguez, and D. Filliat. Decoupling feature extraction from policy learning: assessing benefits of state representation learning in goal based robotics. *arXiv preprint arXiv:1901.08651*, 2019.

[65] T. Jurgenson, O. Avner, E. Groshev, and A. Tamar. Sub-goal trees a framework for goal-based reinforcement learning. In *International conference on machine learning*, pages 5020–5030. PMLR, 2020.

[66] D.-A. Huang, Y.-W. Chao, C. Paxton, X. Deng, L. Fei-Fei, J. C. Niebles, A. Garg, and D. Fox. Motion reasoning for goal-based imitation learning. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4878–4884. IEEE, 2020.

[67] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[68] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, et al. Droid: A large-scale in-the-wild robot manipulation dataset. *arXiv preprint arXiv:2403.12945*, 2024.

[69] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.

[70] A. Gupta, L. Fan, S. Ganguli, and L. Fei-Fei. Metamorph: Learning universal controllers with transformers. *arXiv preprint arXiv:2203.11931*, 2022.

[71] I. Radosavovic, T. Xiao, B. Zhang, T. Darrell, J. Malik, and K. Sreenath. Learning humanoid locomotion with transformers. *arXiv e-prints*, pages arXiv–2303, 2023.

[72] I. Radosavovic, B. Zhang, B. Shi, J. Rajasegaran, S. Kamat, T. Darrell, K. Sreenath, and J. Malik. Humanoid locomotion as next token prediction. *arXiv preprint arXiv:2402.19469*, 2024.

[73] B. Zitkovich, T. Yu, S. Xu, P. Xu, T. Xiao, F. Xia, J. Wu, P. Wohlhart, S. Welker, A. Wahid, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. In *Conference on Robot Learning*, pages 2165–2183. PMLR, 2023.

[74] S. Haldar and L. Pinto. Polytask: Learning unified policies through behavior distillation. *arXiv preprint arXiv:2310.08573*, 2023.

[75] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*, pages 991–1002. PMLR, 2022.

[76] V. Jain, M. Attarian, N. J. Joshi, A. Wahid, D. Driess, Q. Vuong, P. R. Sanketi, P. Sermanet, S. Welker, C. Chan, I. Gilitschenski, Y. Bisk, and D. Dwibedi. Vid2robot: End-to-end video-conditioned policy learning with cross-attention transformers, 2024.

[77] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, et al. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[78] M. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova. Tokenlearner: Adaptive space-time tokenization for videos. *Advances in neural information processing systems*, 34: 12786–12797, 2021.

A Background

Imitation Learning: The goal of imitation learning is to learn a behavior policy π^b given access to either the expert policy π^e or trajectories derived from the expert policy $\mathcal{T}^e$. While there are a multitude of settings with differing levels of access to the expert [38], this work operates in the setting where the agent only has access to observation-based trajectories, i.e. $\mathcal{T}^e \equiv \{(o_t, a_t)_{t=0}^T\}_{n=0}^N$. Here N and T denote the number of trajectory rollouts and episode timesteps respectively. We choose this specific setting since obtaining observations and actions from expert or near-expert demonstrators is feasible in real-world settings [21, 34] and falls in line with recent work in this area [21, 43, 16, 17].

Multi-task Behavior Cloning: Behavior Cloning (BC) corresponds to solving the maximum likelihood problem shown in Eq. 1. Here $\mathcal{T}^e$ refers to expert demonstrations. When parameterized by a normal distribution with fixed variance, the objective can be framed as a regression problem where, given observations o^e, π^{BC} needs to output a^e.

$$\mathcal{L}^{BC} = \mathbb{E}_{(o^e,a^e)\sim\mathcal{T}^e} \|a^e - \pi^{BC}(o^e)\|^2 \tag{1}$$

After training, it enables π^{BC} to mimic the actions corresponding to the observations seen in the demonstrations. In multi-task settings, we use the same formulation for BC but condition the action prediction on a goal variable g^e. Thus, the loss function for multi-task BC becomes the following.

$$\mathcal{L}^{BC} = \mathbb{E}_{(o^e,a^e,g^e)\sim\mathcal{T}^e} \|a^e - \pi^{BC}(o^e|g^e)\|^2 \tag{2}$$

In this work, we represent goals as either a text description of the task [7, 12] or a goal image [43, 35].

B Algorithmic Details

B.1 FiLM Conditioning

Feature-wise Linear Modulation (FiLM) [13] is a technique used for conditioning neural networks that allows the network to modulate its behavior based on an external conditioning signal, such as text instructions or observations. In the context of text conditioning for policy learning, the text instructions are first encoded into a conditioning vector. This conditioning vector is then used to modulate the activations of the neural network through FiLM layers. FiLM applies a feature-wise affine transformation (scaling and shifting) to the activations of the network, conditioned on the text embedding. In other word, assuming $\mathbf{x}$ is a FiLM layer's input, $\mathbf{z}$ is a conditioning input, and γ and β are $\mathbf{z}$-dependent scaling and shifting vectors,

$$FiLM(\mathbf{x}) = \gamma(\mathbf{z}) \odot \mathbf{x} + \beta(\mathbf{z}) \tag{3}$$

This allows the network to adapt its computation and output based on the given text instructions, enabling tasks like instruction following or conditioning the policy on language descriptions.

B.2 Action Heads

Having a separate action prediction module allows BAKU to leverage state-of-the-art techniques for action generation. In this work, we evaluate five different action head variants. Below we briefly describe each variant. For more details on these methods, please refer to the original publications.

Multilayer Perceptron (MLP) This is a simple neural network comprising multiple dense layers. We use a two-layer MLP for our experiments.

Gaussian Mixture Model (GMM) [27] A Gaussian mixture model (GMM) action head models the policy as a mixture of Gaussians, enabling multi-modal action sampling for continuous control problems. The GMM parameters are part of the learned policy network. For our experiments, we employ a two-layer GMM action head with five action modes and a Softplus activation function.

Behavior Transformer (BeT) [15] The Behavior Transformer (BeT) models continuous action prediction as a two-part problem. Actions in the training data are first clustered into k bins using k-means clustering. A discrete action head classifies the cluster an action belongs to, while an offset action head predicts an offset value added to the corresponding cluster center. The discrete head uses a focal loss, while the offset head uses L2 loss. For our experiments, we use BeT with 64 action clusters.

Vector-Quantized Behavior Transformer (VQ-BeT) [16] The Vector-Quantized Behavior Transformer (VQ-BeT) extends BeT by replacing k-means clustering with residual VQVAE-based tokenization, significantly improving performance over BeT. For our experiments, we employ VQ-BeT with two residual VQ layers of codebook size and latent dimension 16 and 256, respectively.

Diffusion [28, 17, 29] A diffusion action head models action prediction as a diffusion process that generates actions over time by iteratively denoising samples from a Gaussian distribution. While highly effective for multi-modal distributions, the iterative denoising during inference slows deployment speed. In this work, we use a transformer-based diffusion head introduced by prior work [28, 17]. We use a two-layer diffusion head for our experiments.

B.3 Temporal smoothing over action chunking

A naïve implementation of action chunking, where a new environment observation in incorporated every k steps can be suboptimal and can result in jerky robot motion. To improve the smoothness in robot motion, we incorporate an exponential temporal ensembling technique, following prior work [21, 12]. Instead of querying the policy every k steps, we query it at every timestep. This results in an overlap in predicted action chunks and at any given timestep, there will be more than one predicted actions. Instead of using only the current action prediction, we use a temporal ensemble to combine all the past predictions. This temporal ensemble performs a weighted average over these predictions with an exponential weighing scheme $w_i = exp(-m * i)$, where w_0 is the weight for the oldest action. The speed for incorporating a new observation is governed by m, where a smaller m means faster incorporation. It must be noted that this ensembling incurs no additional training cost, only extra inference-time computation. In our experiments, similar to prior work [21, 12], we find both action chunking and temporal ensembling to be important for producing precise and smooth motion.

B.4 Hyperparameters

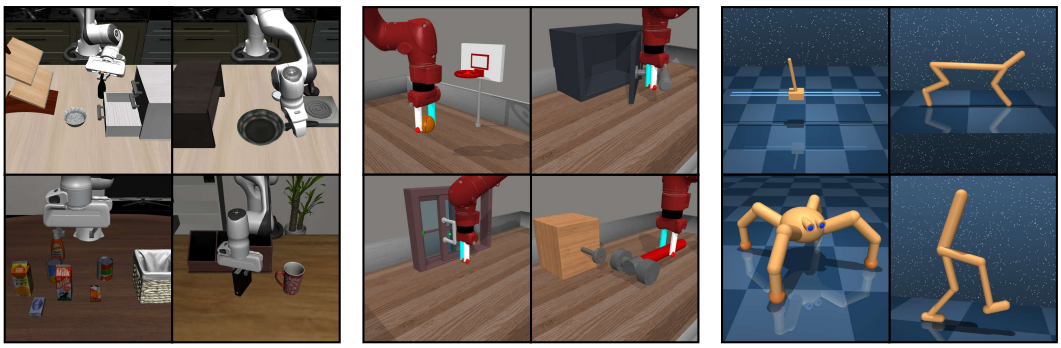
The complete list of hyperparameters is provided in Table 4. For RT-1 [7], we use our implementation with an RT-1 action head that discretizes the continuous action into discrete bins uniformly. For MT-ACT [12], we use the open-source implementation with the default hyperparameters. We vary the action chunk length for MT-ACT for different benchmarks, the values for which have been provided in Table 4.

Training time Below we provide details about the time required to train BAKU on a single NVIDIA RTX A4000 GPU.

1. **LIBERO:** Training for $600k$ steps with a batch size of 64 and 2 camera views and robot proprioception as input requires around 10.5 hours.
2. **Meta-World:** Training for $600k$ steps with a batch size of 64 and 1 camera view as input requires around 8 hours.
3. **DM Control:** Training for $2M$ steps with a batch size of 128 and robot state as input requires around 26 hours.
4. **xArm Robot:** Training for $200k$ steps with a batch size of 64 and 4 camera views and robot proprioception as input requires around 6 hours.

Table 4: List of hyperparameters.

Method	Parameter	Value
Common	Learning rate	$1e^{-4}$
	Image size	128×128 (LIBERO-90, xArm)
		84×84 (Meta-World)
	Mini-batch size	64 (LIBERO-90, Meta-World, xArm)
		128 (DM Control)
	Optimizer	Adam
	Number of training steps	600000 (LIBERO-90, Meta-World)
		2000000 (DM Control)
		200000 (xArm)
	Number of demonstrations	50 (LIBERO-90)
		35 (Meta-World)
		500 (DM Control)
		15 (xArm)
	Transformer architecture	minGPT [30] (with 8 layers and 4 heads)
	Action chunk length	10 (LIBERO-90, Meta-World)
		3 (DMC)
		20 (xArm)
BAKU	Observation trunk	Transformer
	Action head	MLP (base)
		GMM, BeT, VQ-BeT, Diffusion (variants)
	Hidden dim	256
	Observation history	False
	Action chunking	True
	Intermediate goal steps (k)	50 (LIBERO-90)
		30 (Meta-World)
RT-1	Observation trunk	Transformer
	Action head	MLP (base)
	Hidden dim	512
	Observation history	True
	History length	6
	Action chunking	False
MT-ACT	Observation history	False
	Action chunking	True



(a) LIBERO-90 (b) Metaworld (C) DM Control

Figure 3: BAKU is evaluated on 3 simulated benchmarks - LIBERO, Meta-World, and DM Control.

C Simulation Tasks

We evaluate BAKU on three simulated benchmarks: LIBERO-90 [18], MetaWorld [19], and DM Control [20]. For LIBERO-90, we directly use the dataset provided, which includes demonstrations for all 90 tasks. For details on the specific LIBERO-90 tasks, please refer to the original paper [18]. For MetaWorld and DM Control, we collected demonstrations from expert agents trained with reinforcement learning (RL). We include only the tasks for which we were able to obtain expert demonstration data. Table 5 lists the 30 MetaWorld tasks and 9 DM Control tasks used in our experiments. Figure 3 shows a visualization of our simulated benchmarks.

D Robot Tasks

We evaluate BAKU on 30 tasks in our real-world multi-task kitchen environment. We provide the task description along with policy deployment rollouts with BAKU for each task in Figures 4, 5, 6, 7, and 8. The long-horizon task rollouts have been shown in Figure 9.

Robot control We deploy our learned policies at 10Hz using a high-level controller. To facilitate smooth motion on the robot, we deploy a low-level Minimum-Jerk Controller at 100Hz.

E Baselines

In this section, we provide a detailed explanation of our baselines and a comparison with BAKU to highlight their differences.

MT-ACT [12] Multi-task Action-Chunking Transformer (MT-ACT) is a state-of-the-art transformer encoder-decoder architecture for learning multi-task policies. MT-ACT extends Action-Chunking Transformer (ACT) [21] to a multi-task setting. MT-ACT takes as input observations from multiple camera views, robot proprioception, and task instructions. Each input modality passes through dedicated encoders. The encoded observations are then fused in a transformer encoder, the output of which conditions a transformer decoder to predict chunks of future actions. Each predicted action corresponds to a position embedding input to the decoder. In ACT and MT-ACT, a conditional variational autoencoder (CVAE) is used to learn a multimodal style variable which conditions the encoder to deal with multimodal action distributions. During inference, the style variable is set to zero, leading to unimodal behavior. In contrast, BAKU uses a decoder-only transformer architecture that directly predicts action features corresponding to past observations. This enables us to (1) leverage recent advances in multimodal action generation by plugging in several unimodal and multimodal heads for action prediction, and (2) incorporate a history of observations to predict actions for each time step in the history (see Section 3.4 for results on the use of observation history). Further, using a

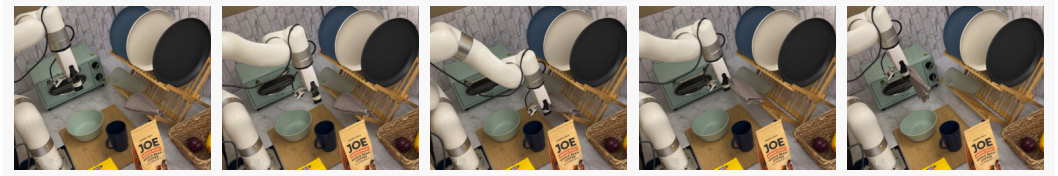
Table 5: List of tasks in Meta-World and DM Control.

Meta-World	DM Control
basketball-v2	cartpole swingup
bin-picking-v2	cheetah run
button-press-v2	hopper stand
button-press-topdown-v2	quadruped run
button-press-topdown-wall-v2	quadruped walk
button-press-wall-v2	teacher easy
coffee-button-v2	walker stand
coffee-pull-v2	walker walk
coffee-push-v2	walker run
dial-turn-v2	
disassemble-v2	
door-lock-v2	
door-open-v2	
door-unlock-v2	
drawer-close-v2	
drawer-open-v2	
faucet-close-v2	
faucet-open-v2	
hammer-v2	
handle-press-v2	
handle-press-side-v2	
handle-pull-v2	
handle-pull-side-v2	
peg-insert-side-v2	
peg-unplug-side-v2	
plate-slide-v2	
plate-slide-back-v2	
plate-slide-back-side-v2	
plate-slide-side-v2	
shelf-place-v2	
soccer-v2	
stick-push-v2	
sweep-v2	
sweep-into-v2	
window-close-v2	
window-open-v2	

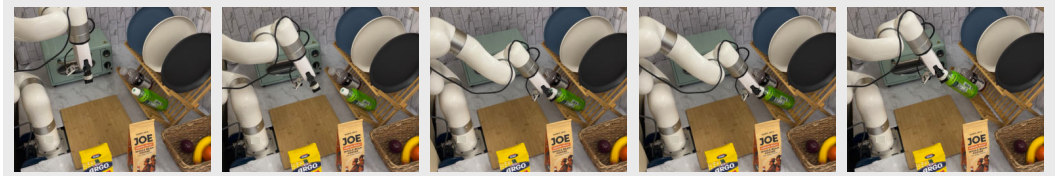
multimodal action head enables BAKU to exhibit multimodal behavior during inference, improving real-world performance (see Table 1).

RT-1 [7] RT-1 is a transformer-based multi-task policy learning architecture that models actions as discrete classes by uniformly discretizing them into bins. RT-1 uses a FiLM-conditioned vision encoder (ResNet-18 in our implementation), but instead of directly using the final 512-dimensional representation, it splits an intermediate feature map of size $k \times k \times 512$ into k^2 tokens of 512 dimensions each. These tokens are passed through a Token Learner [78] module to reduce them to 8 tokens per image. Similar to BAKU, these tokens are then passed through a decoder-only transformer architecture to predict a discrete action. In contrast, BAKU directly uses the final 512-dimensional representation from the vision encoder, without summarizing tokens via a token learner. Additionally, BAKU predicts a continuous action through an unimodal or multimodal action head. Based on

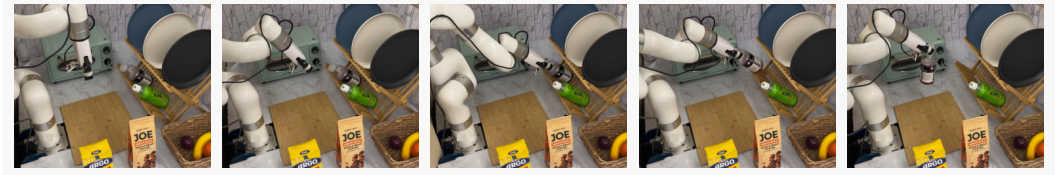
Fetch glass from rack: Fetch the glass from the lower rack.



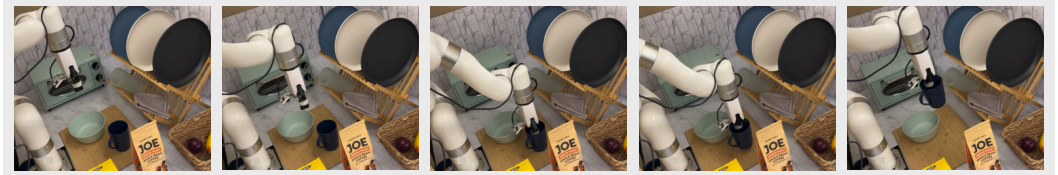
Fetch towel from rack: Fetch the towel from the lower rack.



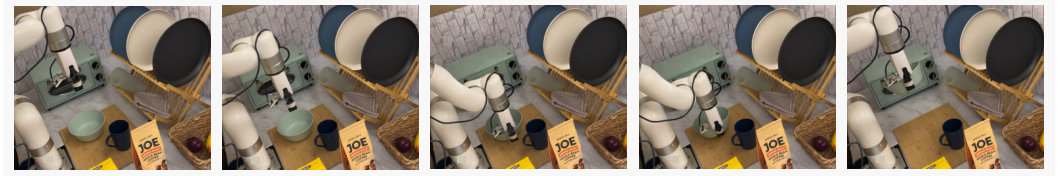
Fetch tea bottle from rack: Fetch the bottle of green tea from the lower rack.



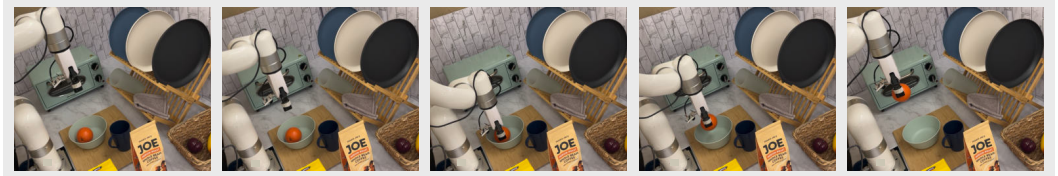
Fetch water bottle from rack: Fetch the bottle of vitamin water from the lower rack.



Pick blue mug: Pick up the blue mug from the kitchen counter.



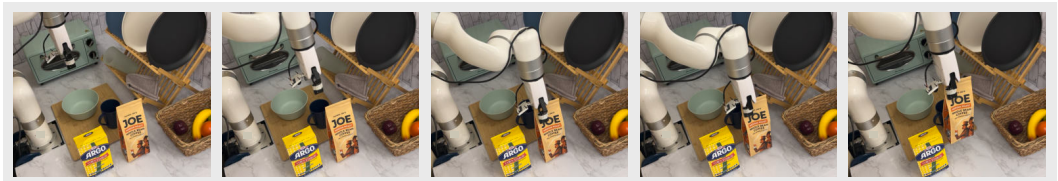
Pick light blue bowl: Pick up the light blue bowl from the kitchen counter.



Pick orange from bowl: Pick up the orange from inside the light blue bowl kept on the kitchen counter.

Figure 4: Real-world policy rollouts showing BAKU's capability in complex manipulation tasks.

our experiments (see Table 1), we observe that these design choices in BAKU lead to significant improvements in performance over RT-1.



Pick coffee bag: Pick up the bag of coffee from the kitchen counter.

Pick box of corn starch: Pick up the box of corn starch from the kitchen counter.

Lift blue plate from rack: Lift the blue plate kept on the upper rack.

Lift white plate from rack: Lift the white plate kept on the upper rack.

Lift black plate from rack: Lift the black plate kept on the upper rack.

Open oven door: Open the door of the oven.

Close oven door: Close the door of the oven.

Figure 5: Real-world policy rollouts showing BAKU's capability in complex manipulation tasks.

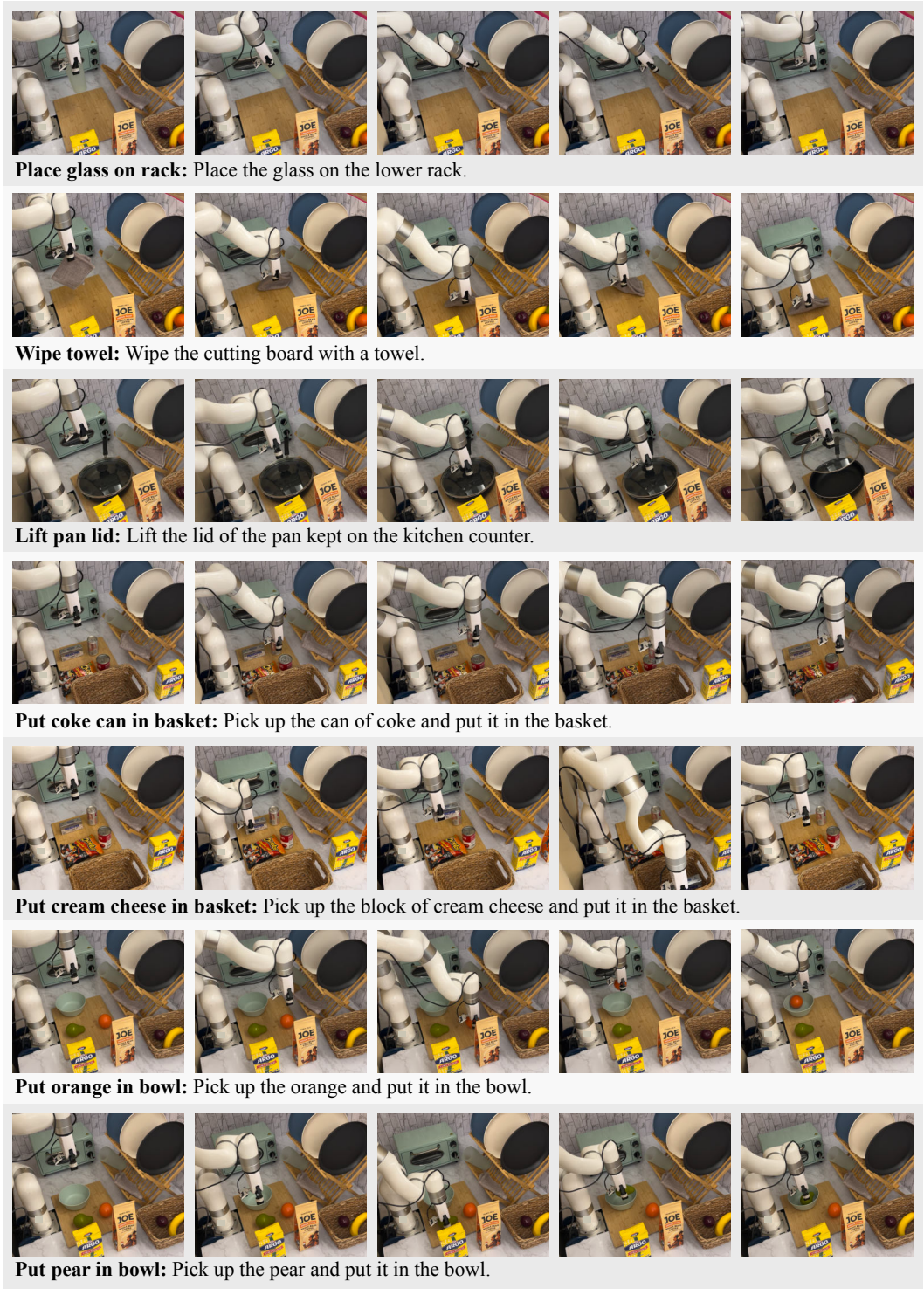


Figure 6: Real-world policy rollouts showing BAKU's capability in complex manipulation tasks.

F Additional Results and Analysis

F.1 Real-World Task-wise Results

Table 6 provides the task-wise performance for all 30 tasks in our real-world multi-task kitchen environment. We collect an average of 17 demonstrations per task, with a total of 520 demonstrations

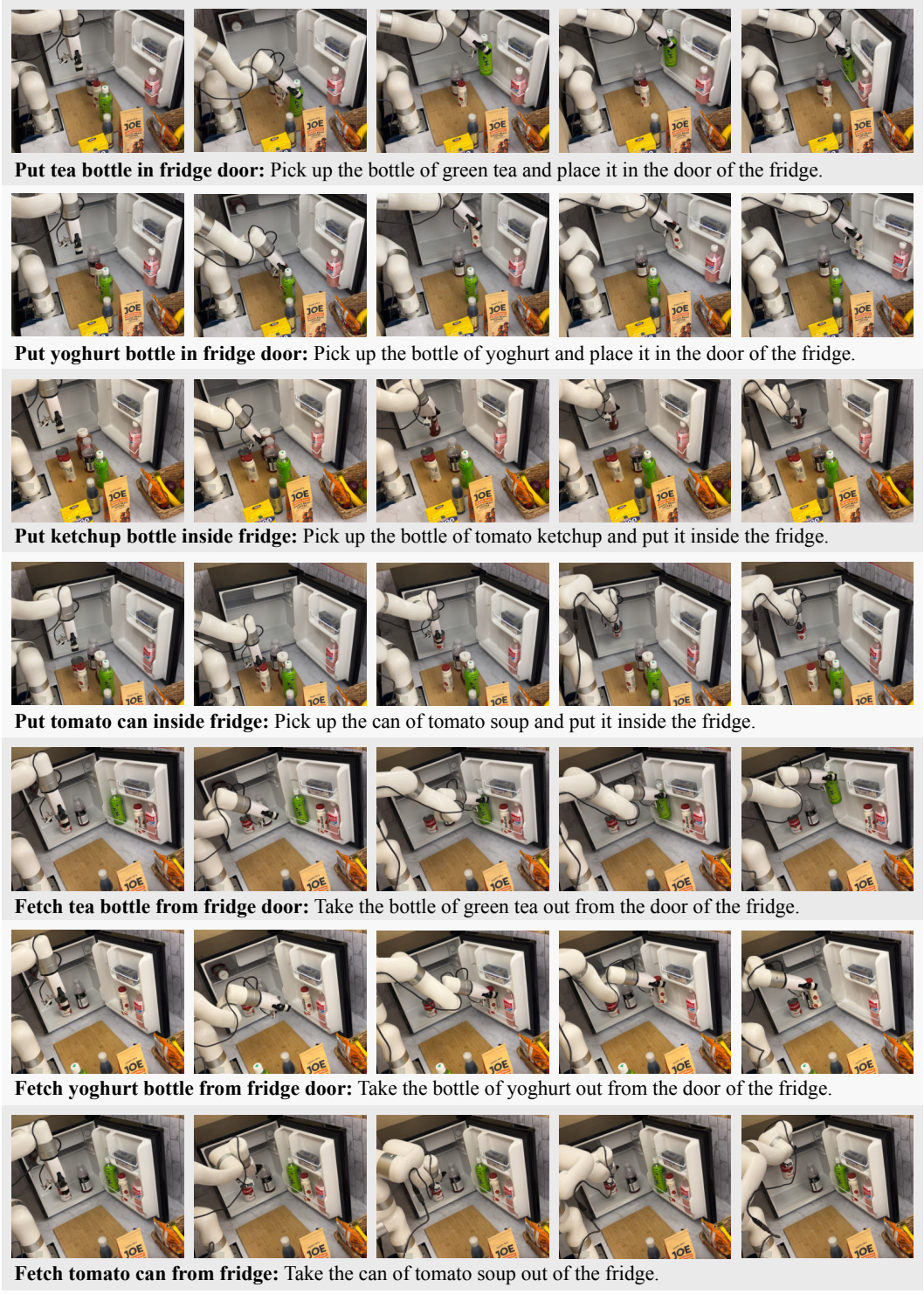
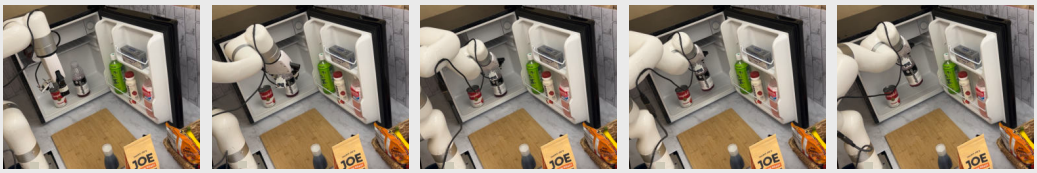
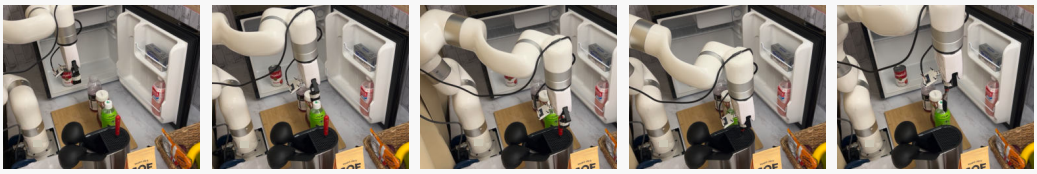


Figure 7: Real-world policy rollouts showing BAKU's capability in complex manipulation tasks.

across all tasks. Task-wise performance for the real-world long-horizon tasks has been included in Table 7.

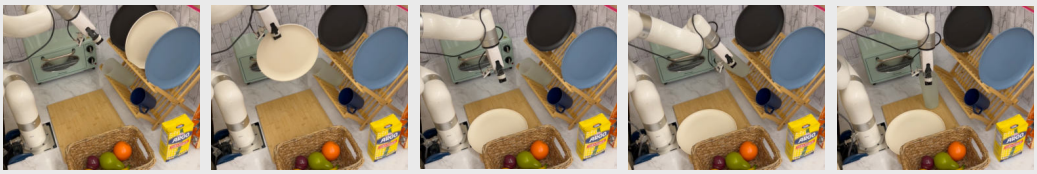


Fetch water bottle from fridge: Take the bottle of vitamin water out of the fridge.

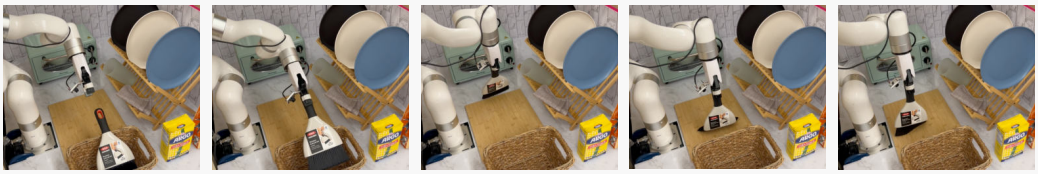


Fetch knife from organizer: Fetch the knife from the organizer placed on the kitchen counter.

Figure 8: Real-world policy rollouts showing BAKU's capability in complex manipulation tasks.



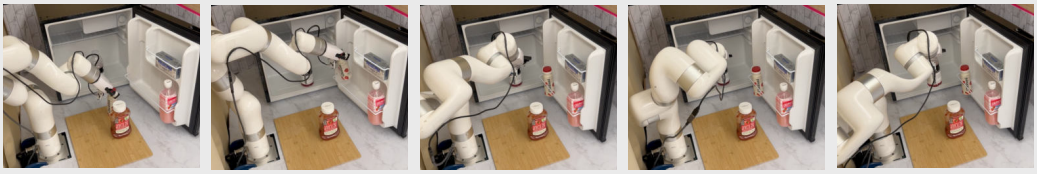
Set up table: Place the white plate and the glass on the kitchen counter.



Pick broom and sweep: Pick up the broom and sweep the cutting board.

Pick towel and wipe: Pick up the towel from the lower rack and wipe the cutting board.

Take bowl out of the oven: Take the bowl out of the oven and place it on the kitchen counter.



Put yoghurt inside and take water bottle out of fridge: Put the yoghurt in the door of the fridge and take out the bottle of water from inside the fridge.

Figure 9: Real-world policy rollouts showing BAKU's capability on long-horizon manipulation tasks.

Table 6: Real task-wise performance

Task	Number of Demonstrations	Successes (out of 5)			
		RT-1	MTACT	Baku	Baku w/ VQ-BeT
Fetch glass from rack	20	5	5	5	5
Fetch towel from rack	28	5	2	5	5
Fetch tea bottle from rack	16	0	3	5	5
Fetch water bottle from rack	16	0	0	5	5
Pick blue mug	16	5	5	5	5
Pick light blue bowl	25	5	5	5	5
Pick orange from bowl	27	0	0	3	4
Pick coffee bag	19	3	5	5	5
Pick box of corn starch	14	0	3	5	5
Lift blue plate from the rack	18	0	4	5	5
Lift white plate from the rack	18	5	5	5	5
Lift black plate from the rack	12	2	3	5	5
Open oven door	17	0	0	0	3
Close oven door	27	0	3	3	4
Place glass on rack	19	5	5	5	5
Wipe towel	17	4	5	5	5
Lift pan lid	18	1	2	4	4
Put coke can in basket	19	0	0	3	3
Put cream cheese in basket	19	0	3	5	5
Put orange into bowl	14	0	0	4	5
Put pear into bowl	17	0	0	3	5
Put tea bottle in fridge door	18	0	0	1	0
Put yoghurt bottle in fridge door	17	3	5	3	5
Put ketchup bottle inside fridge	15	5	4	5	5
Put tomato can inside fridge	11	0	0	5	4
Fetch tea bottle from fridge door	11	5	5	5	5
Fetch tomato can from fridge door	11	0	1	5	5
Fetch yoghurt bottle from fridge door	10	0	3	5	4
Fetch water bottle from fridge	11	2	3	5	5
Fetch knife from organizer	20	0	5	5	5
Mean	17	1.83	2.8	4.3	**4.53**
Mean success rate (out of 1)	–	0.37	0.56	0.86	**0.91**

F.2 Additional Analysis

In addition to the analysis in Section 3.4, we provide further comparisons here to better justify our design choices.

Separate vs. Shared Vision Encoders On the LIBERO-90 benchmark, environment observations include images from two camera views. Table 8 compares multi-task performance using either a common encoder for both views or separate view-specific encoders. While separate encoders provide a 2% boost in performance, this minor gain comes at the cost of a 15% increase in parameter count per camera view added (since the visual encoders comprise 1.5M parameters in our 10M parameter model). For our real-world experiments involving 4 camera views, this parameter increase would be even more significant. Therefore, in BAKU, we use a shared encoder for all views to keep the model compact, assisting with faster inference speeds.

Table 7: Real task-wise performance for long-horizon tasks

Task	Number of Demonstrations	Successes (out of 5)	
		MTACT	Baku
Set up table	34	3	3
Pick broom and sweep	13	4	5
Pick towel and wipe	14	2	4
Take bowl out of the oven	18	5	5
Put yoghurt inside and take water bottle out of fridge	17	2	4
Mean	19	3.2	**4.2**
Mean success rate (out of 1)	–	0.64	**0.84**

Table 8: Study of design decisions for the model architecture that affects multi-task performance.

Category	Variant	LIBERO-90	Meta-World	DMC
Separate vs. Shared Vision Encoders	Common	0.90	–	–
	Separate	**0.92**	–	–
Observation Trunk Input	Separate	**0.90**	**0.79**	**0.70**
	Concatenated	0.87	**0.79**	**0.70**

Observation trunk input In our proposed architecture (see Section 2.4), the encoded observations from different modalities are passed individually as tokens into the observation trunk along with the action token to output the action feature representation. An alternative approach is to concatenate all the encoded inputs into a single vector and pass it through the observation trunk. As shown in Table 8, for Meta-World and DMC, which each have only a single input source, there is no difference in performance, as expected. However, for LIBERO-90, which uses two camera views and the robot's proprioceptive state as inputs, there is a 3% absolute improvement in performance when using separate observation tokens as compare to a single concatenated vector.

G Broader Impacts

In this work, we present BAKU, a simple and efficient transformer architecture for multi-task policy learning. This work takes an important step toward enabling more efficient training of generalist robotic agents capable of performing diverse tasks, reducing the need for large datasets of expert demonstrations which are costly and time-consuming to collect. Further, BAKU focuses on improving data efficiency by maximally leveraging available training data, which is particularly valuable in robotics where data collection is expensive.