MEAT: Mixture of Experts in Action Transformer for Robotic Arm Control

Naeem Ul Islam[1] Hung Phan Quoc Mai[1,2] Ying-Jen Chen[1]
[1]Yuan Ze University, Taiwan
[2]National Economics University, Vietnam

***Abstract*—Learning robot control policies from demonstrations can be framed as supervised regression mapping observations to actions, but challenges such as multimodal action distributions, temporal consistency, and high-precision demands complicate this task. Recent advances leverage architectures including transformers, diffusion models, and VAEs to improve policy learning, with transformer-based methods particularly effective at integrating language and visual inputs. Precision-critical tasks remain difficult due to compounding errors in imitation learning; action chunking policies address this by predicting multiple future steps jointly to reduce temporal noise. Mixture of Experts (MoE) techniques have further enhanced Transformers by enabling scalable specialization, as demonstrated in language models like DeepSeek, Kimi, or Mixtral. This work introduces MEAT (Mixture of Experts in Action Transformer Policy), where we integrate MoE into the Action Chunking Transformer (ACT) Policy, showing upto 20% accuracy gains in simulation and real-world experiments. These results suggest that MoE-enhanced transformers hold promise for scalable and high-performance robot control.**

***Index Terms*—Imitation Learning, Human Teaching Robot, Robot Control Policy**

I. Introduction

At its core, learning robotic manipulation policies from demonstrations can be framed as a supervised regression problem, where the goal is to map observations to actions. However, predicting robot actions presents unique challenges that set it apart from typical supervised learning tasks. These include handling multimodal action distributions, maintaining temporal consistency across sequences, and meeting the demands of high-precision control.

Recent research has shown promise in building autonomous robot control models by learning from human demonstrations, often leveraging advanced architectures like Transformers [1], Diffusion models [2], or VAEs [3] for policy learning. Despite learning extensively from human behavior, these models still face significant limitations.

Transformer-based approaches are particularly notable for their ability to integrate natural language instructions with rich and flexible vector representations, which has led to their widespread adoption in recent robot control models such as the Action Chunking Transformer (ACT) [1], Octo [4], and π_0 [5]. These models leverage the transformer architecture's capacity for modeling long-range dependencies and multi-modal inputs to map observations and instructions to action sequences effectively. Nonetheless, tasks that require high precision and rely heavily on visual feedback remain especially challenging for imitation learning, even when trained on high-quality demonstrations. Small prediction errors in actions can accumulate over time, causing significant deviations in state and exacerbating the well-known compounding error problem associated with imitation learning [1]. To mitigate such issues, the ACT policy predicts joint positions for the next k timesteps jointly as an *action chunk*, rather than predicting one step at a time. This reduces the effective task horizon and helps the model handle temporal confounders such as pauses or noise in the demonstration trajectories more robustly.

However, despite these advances, existing transformer-based policies remain relatively simple in their architectural design, leaving considerable room for further improvement. Originally designed for natural language processing, transformers have since seen numerous architectural enhancements, one of the most prominent being the Mixture of Experts (MoE) [6] technique. MoE introduces sparse expert routing to improve scalability and expressiveness, and has shown strong effectiveness in cutting-edge models such as DeepSeek [7] and Mixtral [6]. These successes underscore the potential of MoE and related innovations to further enhance transformer-based policies for robot control and imitation learning tasks.

In this study, we introduce MEAT: Mixture of Expert in Action Transformer policy. We explore model-level enhancements to the ACT by incorporating MoE into its architecture and investigating the effectiveness of auxiliary loss functions during training. Our experiments demonstrate that augmenting ACT with MoE improves task performance and robustness, offering a promising direction for scaling up to larger, multimodal robot control models. We validate our proposed approach through extensive simulations on benchmark datasets.

In general, our main contributions can be summarized as follows:

- We propose an enhanced version of the ACT model by integrating a MoE mechanism into the transformer layers, enabling better utilization of model capacity and improving generalization.
- We investigate the impact of auxiliary loss functions on training stability and performance of the ACT model, providing insights into effective training strategies for action-chunking policies.
- We empirically demonstrate that MEAT achieves 5-15% higher accuracy compared to the original ACT across multiple simulated experiments.

979-8-3315-9205-9/25/$31.00 ©2025 IEEE
DOI 10.1109/CogMI67134.2025.00033

- We evaluate and deploy our MEAT model on both the simulation dataset proposed in the original ALOHA paper as well as the widely-used RoboMimic [8] dataset, and also on a real-world robot, showcasing its effectiveness and adaptability across different benchmarks.

II. Background

Advances in Imitation Learning. Much of the recent research has been directed towards imitation learning for robots, and many approaches have been presented. Following the success of the Diffusion [9] and Transformer [10] models, these architectures have been extensively explored and adopted as policies for robotic manipulation. [2] introduces Diffusion Policy, a conditional denoising diffusion approach for robot visuomotor control that enables stable training, handles multimodal actions, and supports high-dimensional spaces using receding horizon control, visual conditioning, and a time-series diffusion transformer. DiT-Block Policy [11] improved diffusion transformer architecture with adaLN-zero layers and optimized input encodings, achieving state-of-the-art performance on long-horizon bimanual dexterous tasks and demonstrating strong scaling on 10 hours of multimodal, language-annotated ALOHA demonstrations. In addition, the approach using transformer-based models is the most studied, which will be discussed in more detail below.

Transformer models in Robotics Transformer-based models offer the advantage of unifying diverse tasks such as vision, language commands, and speech into a single end-to-end framework using vector embeddings, enabling the creation of foundation models that can be further fine-tuned. [1] implement action chunking policies using Transformers, training them as conditional VAEs (CVAEs) to capture human data variability. Action Chunking with Transformers (ACT) significantly outperforms previous imitation learning approaches on various simulated and real-world fine manipulation tasks. [4] presents Octo, a large transformer-based policy trained on 800k trajectories from the Open X-Embodiment dataset, designed as a generalist foundation model for robotic manipulation that can be efficiently finetuned across diverse platforms, sensors, and action spaces using language or image goals. With a similar approach, π_0 [5] proposes a generalist robot policy using a novel flow matching architecture built on a pre-trained vision-language model to leverage semantic knowledge for dexterous tasks, demonstrating strong performance across diverse platforms and tasks through prompting, instruction following, and fine-tuning. Rather than using Transformer for learning manipulating policy, Q-Transformer [12] introduces a scalable offline reinforcement learning method that leverages both human and autonomous data by modeling Q-functions with a Transformer. By discretizing action dimensions into tokens, the approach enables effective sequence modeling for Q-learning and demonstrates strong performance across diverse real-world robotic manipulation tasks.

Mixture of Experts. MoE models selectively route information through a network, using a gating mechanism to activate only a subset of specialized expert modules per input. This approach was popularized by [13], which introduced conditional routing via a gating network. With the success of Transformers, MoE architectures evolved most notably with Switch Transformers [14], which incorporated sparse expert feed-forward layers to scale model capacity while keeping computation efficient. Sparsely activated expert networks within layers allowing only a subset of experts to process each input, which improves model capacity and efficiency. This approach enables scaling to larger models while maintaining manageable computational costs. This design became foundational for many Large Language Models: Mixtral [6], Deepseek-v3 [7], and GLaM [15]. MoE was integrated in Mixtral [6], at the time Mixtral 8x7B – Instruct achieving superior performance over GPT-3.5 Turbo, Claude-2.1, Gemini Pro, and Llama 2 70B on human evaluation benchmarks. In addition, the components of MoE are also studied including: Gating Function, Expert Network, Routing Strategy, Training Strategy. Jointly optimizing both routers and experts poses challenges like expert collapse, where multiple experts converge to similar functions, and router collapse, where only a few experts are consistently selected. Load balancing losses [13], [14] address this by encouraging more uniform expert usage. Various routing strategies have since been proposed, including expert choice routing [16], differential k-selection [17], frozen hashing [18], and linear assignment [19]. Instead of learning an auxiliary loss for gating expert balancing, [20] introduced Loss-Free Balancing, a gradient-free load balancing method for MoE models that avoids auxiliary loss interference by applying expert-wise biases to routing scores, dynamically adjusting them based on recent usage to maintain balanced expert load. In robotics arm manipulation policy learning the idea of using MoE is gradually being studied more, for example MoDE [21] introduces Mixture-of-Denoising Experts in Diffusion Transformer Policy that surpasses current state-of-the-art Diffusion Policies while enabling parameter-efficient scaling through sparse experts and noise-conditioned routing. MoVEInt [3] proposes a Mixture-of-Experts VAE framework that learns a shared latent space from human demonstrations using an MDN prior, enabling accurate and reactive robot actions in human-robot interactions.

Demonstration Datasets. In order to facilitate the fast-growing Imitation Learning, many datasets are introduced from real environments to simulated environments. For example, in real-world environments, Open X-embodiment [22] introduces a large, standardized dataset of 160,266 tasks across 22 robots from 21 institutions to enable training and evaluation of generalist robot policies adaptable across diverse platforms and skills. Besides, there are numerous of demonstration datasets have been published such as DROID [23] with 76k demonstration trajectories or 350h of interaction data, collected across 564 scenes and 86 tasks by 50 data collectors; and Mobile-ALOHA [24] which is a low-cost and whole-body teleoperation system for data collection. In simulation, Robomimic [8] presents an extensive evaluation of six offline learning algorithms on diverse simulated and real-world multi-stage manipulation tasks using human demon-

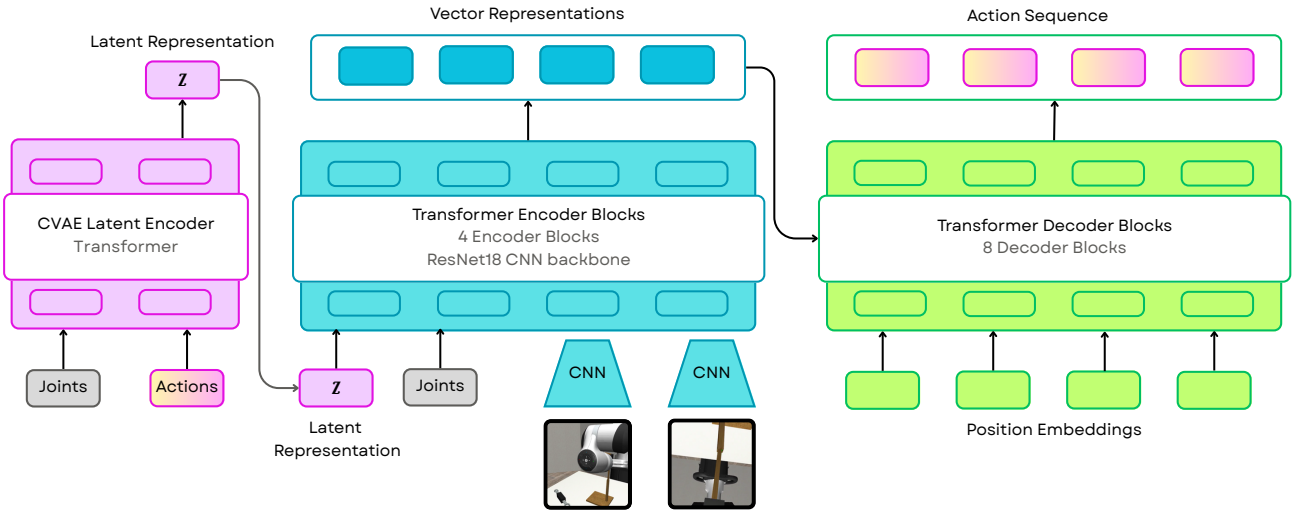


Fig. 1: Architecture of the overall ACT model.

stration datasets, highlighting key challenges like algorithm sensitivity, demonstration quality, and evaluation variability, while emphasizing the potential to learn complex policies from raw sensory data; all datasets and code are open-sourced to support future research. Further developed on Robomimic, MimicGen [25] is a system that automatically synthesizes large-scale, diverse robot demonstration datasets from a small set of human examples by adapting them to new contexts, enabling effective imitation learning for complex, long-horizon tasks and offering a cost-efficient alternative to collecting extensive human demonstrations.

III. Approach

A. Problem Formulation

We aim to learn a policy that predicts chunks of consecutive actions based on the current observation and a latent variable inferred from the demonstration. Given a dataset of trajectories $\mathcal{D}$, at each timestep the policy outputs a sequence of k future actions conditioned on the observation o_t and a latent embedding z. The latent z is obtained by encoding the ground-truth action chunk and observation history, while the policy decodes z and o_t to reconstruct the action chunk. The training objective combines a reconstruction loss that minimizes the mean squared error between the predicted and demonstrated action chunks:

$$\mathcal{L}_{\text{reconst}} = \mathbb{E}_{\mathcal{D}}\big[\text{MSE}(\hat{a}_{t:t+k}, a_{t:t+k})\big] \tag{1}$$

and a regularization loss that encourages the latent variable to follow a standard Gaussian prior:

$$\mathcal{L}_{\text{reg}} = \mathbb{E}_{\mathcal{D}}\big[D_{\text{KL}}\big(q_\phi(z|a_{t:t+k}, \bar{o}_t) \,\|\, \mathcal{N}(0, I)\big)\big]. \tag{2}$$

B. Action Chunking Transformer Policy

To mitigate the compounding error problem inherent in imitation learning, particularly for high-frequency, long-horizon tasks, ACT adopt the concept of *action chunking*, inspired by findings in neuroscience [26], where sequences of actions are grouped and executed as a single unit. This approach reduces the effective task horizon by a factor of k, which improves robustness and allows the policy to handle non-Markovian behaviors observed in human demonstrations, such as pauses or temporally correlated noise [1].

Specifically, rather than predicting a single action a_t at each timestep given the current observation s_t, the ACT policy predicts a chunk of k future actions jointly:

$$\pi_\theta(a_{t:t+k-1} \mid s_t), \tag{3}$$

where $a_{t:t+k-1} = (a_t, a_{t+1}, \ldots, a_{t+k-1})$ denotes the predicted chunk. At inference time, every k steps, the agent observes s_t, samples z from the encoder, and generates the next k actions, which are then executed sequentially.

However, simply updating the observation every k steps can result in abrupt and jerky motion. To alleviate this, ACT employs a *temporal ensembling* strategy. Instead of discarding intermediate observations, the policy is queried at every timestep, producing overlapping predictions for the same future timestep from multiple chunks. These predictions are then aggregated through a weighted average:

$$\hat{a}_t = \frac{\sum_{i=0}^{k-1} w_i A_i[t]}{\sum_{i=0}^{k-1} w_i}, \tag{4}$$

where $A_i[t]$ is the i-th prediction of the action at timestep t from the overlapping chunks, and w_i is an exponentially decaying weight given by:

$$w_i = \exp(-m \cdot i), \tag{5}$$

where m controls how quickly newer observations are incorporated smaller m allows faster adaptation to new observations.

This temporal ensembling ensures that action predictions are smooth and continuously updated, while maintaining the

benefits of reduced horizon and robustness to temporal confounders. Importantly, this ensemble operates only at inference time and does not introduce additional training cost.

Figure 1 illustrates the architecture of a Transformer policy used for visuomotor robotic manipulation. The model follows an encoder-decoder Transformer structure originated from the idea of [27], where multimodal sensory inputs, which include images processed by a ResNet18 CNN, joint positions, and a latent vector, are encoded into a vector representation using a stack of 4 Transformer encoder blocks. The latent vector representation produced by CVAE Latent Encoder, and only be used during training, for inference time it is set to $\mathbf{0}$. These encoded features are then passed to 8 Transformer decoder blocks to autoregressively generate the action sequence, conditioned on learned position embeddings.

C. Mixture of Experts

To evaluate the efficiency of advanced Transformer architectures, which are widely adopted in NLP for robotic arm manipulation, we introduce a Mixture-of-Experts (MoE) design into the Transformer policy (MEAT). The key innovation lies in the incorporation of MoE within the Transformer blocks (as detailed on the right side of the figure). Each Transformer block contains a MoE sub-layer consisting of multiple expert networks and a gating mechanism that dynamically selects the top-k experts per input token. This design increases model capacity without proportionally increasing computation, allowing more expressive policies while maintaining efficiency. This MoE-enhanced Transformer policy significantly boosts the capability of robotic arms in complex tasks by enabling specialization across experts for different manipulation subtasks or observation types.

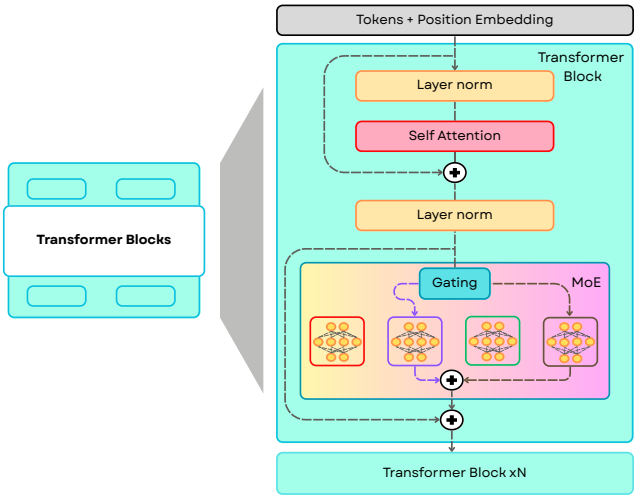


Fig. 2: Transformer Block architecture composition including Mixture of Experts

Figure 2 describes the Transformer block augmented with a Mixture-of-Experts (MoE) layer. After the standard self-attention and normalization layers, tokens are passed through a gating mechanism that routes them to a sparse subset of expert networks within the MoE module. Each expert is a small feedforward network, and only a few are activated per input, improving model capacity without increasing computational cost proportionally. In our experiment, we set 4 experts in each Transformer block, of which 2 experts are activated.

To preserve gradient flow through the non-differentiable sampling step, we scale each expert's output by its routing probability and renormalize the selected probabilities. To mitigate expert collapse, we add a load balancing loss that encourages more uniform expert utilization [14]:

$$LB(\sigma_t) = N \sum_{n=1}^{N} \frac{1}{|B|} \left(\sum_{i=1}^{|B|} \mathbb{1}\big(\phi(\sigma_t)_i\big)_n > 0 \big) \right) \times \frac{1}{|B|} \left(\sum_{i=1}^{|B|} i = \mathbb{1}^{|B|} \operatorname{softmax}\big(\phi(\sigma_t) W_R\big)_n \right) \quad (6)$$

D. Auxiliary-Loss-Free Load Balancing Strategy

In addition to employing the standard MoE architecture, we also evaluate the Auxiliary-Loss-Free Balancing Strategy proposed in [20]. Specifically, instead of relying solely on naive top-K gating, [20] introduce an expert-specific bias term $\{b_i\}_{i=1}^{N}$ to the gating scores $s_{i,t}$. These biased scores, $s_{i,t}+b_i$, are used to determine the top-K experts for routing, thereby promoting a more balanced expert utilization:

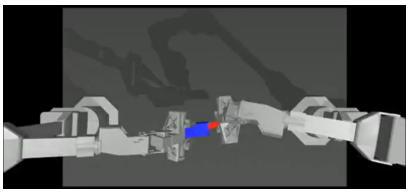
$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \text{TopK}(\{s_{j,t} + b_j \mid 1 \le j \le N\}, K) \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

It is important to note that the expert bias term b_i is used solely to influence the top-K routing decision. It does not affect the final weighted combination of expert outputs—i.e., $g_{i,t}$ is not modified by b_i during the actual MoE output computation.

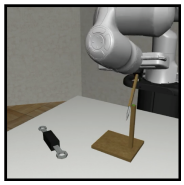
IV. Simulation Experiments

A. Dataset

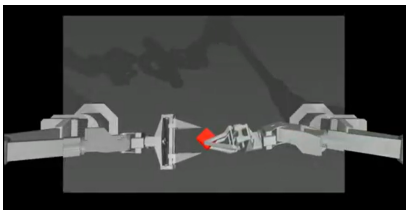
For our experiments, we evaluate on 4 simulation tasks derived from publicly available datasets: 2 simulated tasks Transfer Cube and Bimanual Insertion introduced by ALOHA [1], and 2 RoboMimmic tasks Tool Hang and Tool Transfer proposed by [8] as shown in **Figure 3**. Among Robomimic tasks, we chose 2 tasks, Tool Hang and Tool Transfer, as they are the most challenging since they require many steps and dexterous to complete. The simulated dataset from ALOHA contains two types of demonstrations: scripted and human demonstrations, whereas RoboMimic provides only demonstrations performed by proficient humans. Human demonstrations are often noisier, less consistent, and harder for the robot to learn from compared to scripted ones. This is because humans may make small mistakes and immediately correct them, which behavior that can confuse the robot during imitation learning.



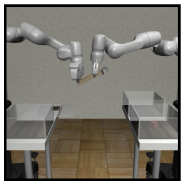
Insertion



Tool Hang



Cube Transfer



Tool Transport

Fig. 3: Illustrations of 4 Simulated tasks: Insertion and Cube Transfer from ALOHA (Left), Tool Hang and Tool Transport from RoboMimic (Right).

In the **Transfer Cube** task, the right robotic arm must first grasp a red cube from the table and accurately place it into the gripper of the left arm. This task is highly sensitive to precision, as the clearance between the cube and the left gripper is approximately 1 cm; even small errors can lead to collisions or task failure. In the **Bimanual Insertion** task, both arms are required to collaborate: one grasping a socket and the other a peg, and perform a mid-air insertion such that the peg aligns with the pins inside the socket. This task demands even higher precision, with a clearance of roughly 5 mm during the insertion phase.

For RoboMimic tasks, **Transport Tool** involves two robotic arms coordinating to move a hammer from a closed container to a target bin on a shelf. One arm retrieves the hammer, while the other clears trash from the target bin before completing a mid-air handover. And **Tool Hang** requires a robot to assemble a hook-and-base frame and hang a wrench on the hook, which demanding precise, multi-stage, rotation-intensive manipulation, making it the most challenging task.

B. Experiment Setup

We conduct all training and inference on an RTX 4080 GPU with 16GB of memory. Each task is trained using 3 random seeds. When training ACT models, for ALOHA tasks, training takes approximately 40 minutes per run for 3000 epochs. In contrast, RoboMimic tasks require around 1.5 hours per run with 4000 epochs due to their higher complexity and larger dataset size. We trained ACT with only one camera view: top view for ALOHA tasks and agent view for RoboMimic tasks, both using an image size of 480x640. We use full of 50 demos of ALOHA tasks, but only 150 of 200 demos of RoboMimic tasks due to resource restrictions. The ACT model has 84M parameters, while the MoE model contains 281M parameters, of which only 189M are activated per token during inference. Refer to **Appendix B** for detail hyperparameters configuration for each task and model.

C. Evaluation Method

We select 6 imitation learning methods (BC-ConVMLP, BeT, RT-1, VINN, IBC, BC-RNN) as baselines because they represent the most relevant and diverse approaches in recent robotic manipulation research. BC-ConvMLP serves as a simple baseline that maps visual and proprioceptive inputs directly to actions. BeT [28] and RT-1 [29] extend this idea using Transformer architectures, capturing temporal context but relying on discretized action spaces. BC-RNN from the RoboMimic benchmark [8] is a behavior cloning baseline that models temporal dependencies in demonstrations by applying a recurrent neural network (typically an LSTM-GMM) to sequential state–action data for improved long-horizon imitation performance. Implicit Behavioral Cloning (IBC) [30] learns a policy by modeling an implicit energy function over actions conditioned on observations, allowing continuous control without explicitly parameterizing the action distribution and enabling more expressive, multimodal imitation behaviors. In contrast, VINN [31] offers a non-parametric perspective, retrieving actions from similar visual demonstrations. Together, these baselines cover a spectrum of design philosophies, feed-forward, sequential Transformer-based, and retrieval-based, providing a comprehensive benchmark for evaluating how ACT and MEAT improve continuous, high-precision control in imitation learning.

We present the best-performing method for each baseline on each benchmark, sourced from all possible sources: our reproduced results (ACT, BC-RNN, Diffusion Policy); or the results reported in the original papers (BC-ConVMLP, BeT, RT-1, VINN, IBC) since these models are reported as inefficient, so there is no need to reproduce. For BC-RNN, to align with the training setup used in ACT and the original Robomimic paper, we train the model using only one camera view (agent view), and increase the batch size as well as the number of training epochs from 600 to 1000. Evaluation is performed over 50 rollouts per seed among 3 seeds, resulting in roughly 3000 total evaluation rollouts after 18 training runs per model architecture.

1) ALOHA tasks: We compare ACT models against four prior imitation learning methods. BC-ConvMLP is a simple yet widely adopted baseline with vanilla Convolution and MLP network, which processes current image observations using a convolutional network, concatenates its output with joint positions, and predicts the action. BeT [28] also employs a Transformer architecture but differs from ACT in two key ways: (1) it does not use action chunking, instead predicting a single action based on the observation history; and (2) it uses a separately trained, frozen visual encoder for image features, meaning perception and control are not jointly optimized. RT-1 [29] is another Transformer-based model that predicts one action from a fixed-length observation history. Both BeT and RT-1 discretize the action space, outputting a categorical distribution over discrete bins, with BeT adding a

TABLE I: Mean success rate (%) ± standard deviation over 3 seeds for two simulated tasks from the ALOHA benchmark. Bold numbers in **Insert** and **Transfer** indicate completion rates. MEAT consistently outperforms ACT and other baselines across both human and scripted settings.

Model	Insertion human			Insertion Scripted			Transfer cube human			Transfer cube scripted		
	Grasp	Contact	**Insert**	Grasp	Contact	**Insert**	Touch	Lift	**Transfer**	Touch	Lift	**Transfer**
BC-ConvMLP	0	0	0	5	1	1	3	1	0	34	17	1
BeT [28]	0	0	0	21	4	3	16	13	1	60	51	27
RT-1 [29]	0	0	0	2	0	1	4	2	0	44	33	2
VINN [31]	0	0	0	6	1	1	17	11	0	13	9	3
ACT [1]	73±4.5	55±4.6	13±3.8	96±4.6	88±8.4	33±12.3	97±1.7	55±3.8	55±3.0	97±4.6	89±8.8	80±6.2
MEAT	**85±2.0**	**60±4.8**	**21±3.1**	**97±1.6**	**88±6.4**	**65±13.3**	**89±3.4**	**57±8.8**	**64±4.2**	**99±2.4**	**93±5.3**	**85±7.8**
MEAT Aux-free	81±4.2	61±5.3	16±2.0	99±4.4	81±8.9	37±22.0	97±1.5	62±3.4	61±8.1	100±0.0	78±12.3	77±17.0

continuous offset from the bin center. In contrast, ACT directly predicts continuous actions, which is better suited for precise manipulation tasks. Finally, VINN [31] is a non-parametric method that requires access to the demonstration data at test time: it retrieves the k most visually similar observations from the demonstrations and predicts the action via weighted k-nearest-neighbors.

2) RoboMimic tasks: For RoboMimic data, we compare our result with 2 models as benchmarks. Implicit Behavior Cloning (IBC) [30] leverages implicit energy-based models to learn policies that better capture complex, multimodal, and discontinuous behaviors from demonstrations, often outperforming explicit methods and achieving competitive or superior performance even on challenging, high-dimensional robot learning tasks without using rewards. BC-RNN [8], a recurrent variant of Behavioral Cloning that models temporal dependencies in demonstrations via an RNN policy, is the best-performing method reported in the RoboMimic paper, achieving strong results by effectively leveraging sequential structure in the data.

D. Results

The experimental results on the ALOHA benchmark are summarized in Table I. While we were unable to exactly reproduce the results reported in the original ALOHA paper [1], our reproduced values show comparable trends: certain tasks, such as Insertion (scripted) and Transfer cube (human), achieve slightly higher success rates, whereas Insertion (human) and Transfer cube (scripted) show lower outcomes. Overall, MEAT consistently outperforms all baseline models across both human and scripted settings, achieving the highest mean success rates in nearly every subtask. For example, in the Insertion (human) task, MEAT attains 85 ± 2.% for grasp, 60 ± 4.8% for contact, and 21 ± 3.1% for insert—clearly surpassing ACT (73 ± 4.5%, 55 ± 4.6%, 13 ± 3.8%) and other baselines. Likewise, in Insertion (scripted), MEAT approaches near-perfect performance (97 ± 1.6%, 88 ± 6.4%, 65 ± 13.3%), showing not only high accuracy but also relatively low variability across runs. On the Transfer cube tasks, MEAT maintains strong generalization with success rates ranging from 89–99% and moderate deviations (typically below ± 8%), indicating stable behavior across trials. Although the MEAT Aux-free variant achieves competitive results, its higher standard deviations (up to ± 17%) suggest greater performance fluctuation, especially on more complex scripted tasks. In contrast, MEAT demonstrates both high success rates and low variance, reflecting robust policy learning and consistent execution. Baseline models such as BC-ConvMLP, BeT, RT-1, and VINN remain substantially weaker, often yielding near-zero success on key subtasks. These findings collectively indicate that MEAT not only improves average task performance but also enhances stability across independent seeds, outperforming both the original ACT and all other compared approaches, particularly in challenging robotic manipulation scenarios.

TABLE II: Success rate (%) (Highest | Average) for 2 simulated tasks from RoboMimic Proficient Human (ph) dataset. MEAT outperforms the original ACT and having competitive results with Diffusion Policy.

Model	**Tool hang, ph**	**Tool transport, ph**
IBC [30]	0 | 0	0 | 0
BC-RNN [8]	24 | 18	38 | 28
DP [2]	40 | 34	**50** | **40**
ACT [1]	46 | 38	44 | 34
MEAT	**52** | **46**	44 |38
MEAT Aux-free	48 | 28	24 | 12

Table II reports the highest and average success rates on the Tool Hang and Tool Transport tasks from the RoboMimic dataset. For reproducing BC-RNN, as mentioned, we use only 1 camera view and increase the number of training epochs to align with the experimental setup in ACT. However, the performance is significantly lower under this configuration. Our proposed MEAT consistently outperforms the original ACT in both tasks and has a competitive result compared with Diffusion Policy. On Tool Hang, MEAT achieves the highest | average success rate of 52 | 46, compared to ACT's 46 | 38, showing improvements in both peak and overall performance. On the Tool Transport task, MEAT did not outperform DP in terms of overall accuracy. However, it still achieves a competitive average performance compared to DP. Importantly, MEAT's strengths lie in its efficient inference and training

times, as well as its lightweight architecture. As shown in **Figure 4**, MEAT's inference time is comparable to that of lighter models such as CNNMLP and ACT, and is approximately 16 times faster than that of DP. This is a significant advantage, considering that DP is known for its slow inference [32], [33], primarily due to its time-consuming denoising process. In our real-world experiments, we demonstrate that MEAT can be efficiently deployed on a laptop GPU, delivering real-time performance while maintaining competitive predictive accuracy.

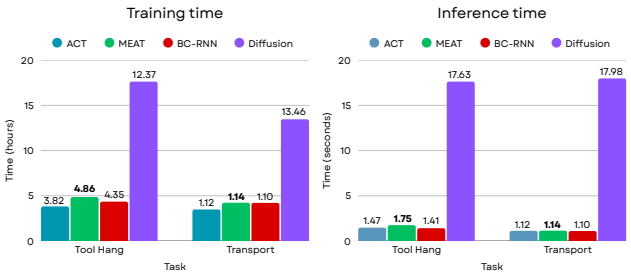


Fig. 4: **Training and Inference Time Benchmark.** Inference time is measured as the total time the model takes to predict an action, excluding the time taken to execute the action step. Inference time is reported in seconds, while training time is reported in hours. Apparently, Diffusion Policy requires significantly more time for both inference and training compared to other methods.

E. Discussion & Analysis

Although BC-RNN can yield good performance, we found that training it is often slow and requires a large amount of Random Access Memory (RAM). Specifically, we had to reduce the number of demonstrations from 200 to 150 in order to fit the entire dataset into 120GB of RAM and run training successfully. This is why, as mentioned, we used only 150 demonstrations for the RoboMimic tasks. Furthermore, BC-RNN takes significantly longer to train completely, approximately 6 hours for Tool Transport and 10 hours for Tool Hang, which, compared to ACT completes training in only 1.4 hours while still achieving better results. This further highlights the efficiency and effectiveness of MEAT.

Regarding **Figure 4**, MEAT, ACT, and BC-RNN show no significant differences in inference time. In some cases, MEAT even infers faster than ACT or BC-RNN, particularly in the tool transfer task. Notably, the Diffusion Policy model exhibits extremely high inference and training times, which is up to 16 times longer in inference and 3 times longer in training compared to other models. While Diffusion Policy may offer slightly better accuracy, the gain is not substantial enough to justify its high computational cost, making it unclear whether it truly outperforms MEAT overall.

V. Realworld Experiment

A. Setup

Teloperation. To record human demonstration data for training models, we used two robotic arms: one as the leader and the other as the follower. Both are UFactory Lite 6 models, each with 6 degrees of freedom (DoF). During teleoperation, the human operator manually moves the leader robot arm. The system then transmits the current state of the leader to the follower, allowing the follower to replicate the leader's movements in real time. Refer to **Appendix A** for the detailed description of teleoperation.

Recording data. As the follower robot executes the task, we simultaneously record the leader's state (used as the action), and capture the follower's joint positions, joint angles, and images of the scene involving the robot and the objects. Teleoperation and data recording are handled in parallel using two separate threads. Due to hardware limitations, we record robot states at a frequency of 20 Hz. For each task in real-world experiments, the robot's state is represented as a 6-dimensional vector. We extend both the action and observation vectors to 7 dimensions to include the state of the end-effector. The end-effectors used in experiments are grippers and vacuums, which can operate in two basic states: open or closed.

Policy Training. Similar to the simulation tasks, we trained the policy on a single NVIDIA RTX 4080 GPU using data collected from the real robot. Training Transformer-based models on real-robot data typically requires more time, with 15,000 steps needed to achieve stable and reliable performance in real-world scenarios. During inference, the trained policy was deployed on a laptop equipped with an NVIDIA RTX 4060 GPU. The model takes real-time inputs, including RGB images and robot states, and outputs the corresponding actions, specifically, the predicted next-step position states for the robot.

B. Task: 'Chopsticks'

Our experiment is named **Chopsticks**. In this task, the robot is equipped with a gripper as its end-effector. As shown in **Figure 5**, the evaluation setup is as follows: two chopsticks are placed randomly within a designated box area on the table, and a cup is positioned beside them. The robot's objective is to pick up each chopstick individually and place it into the cup. We trained ACT and MEAT models for 15,000 steps, which took approximately 4.5 hours on 50 demonstrations. Each raw trajectory record of Chopsticks tasks includes 390 to 430 timesteps; after data cleaning we refine all to fit into 400 timesteps.

Result Analysis. Table III reports the 50 experiments result for chopsticks and comparison among 3 policies, in the table *locate* means the gripper is correctly positioned over a chopstick, *lift* is that the chopstick is successfully grasped, and *put* is the chopstick is placed into the cup. The robot must complete all three stages for each of the 2 chopsticks. our model MEAT significantly outperforms both

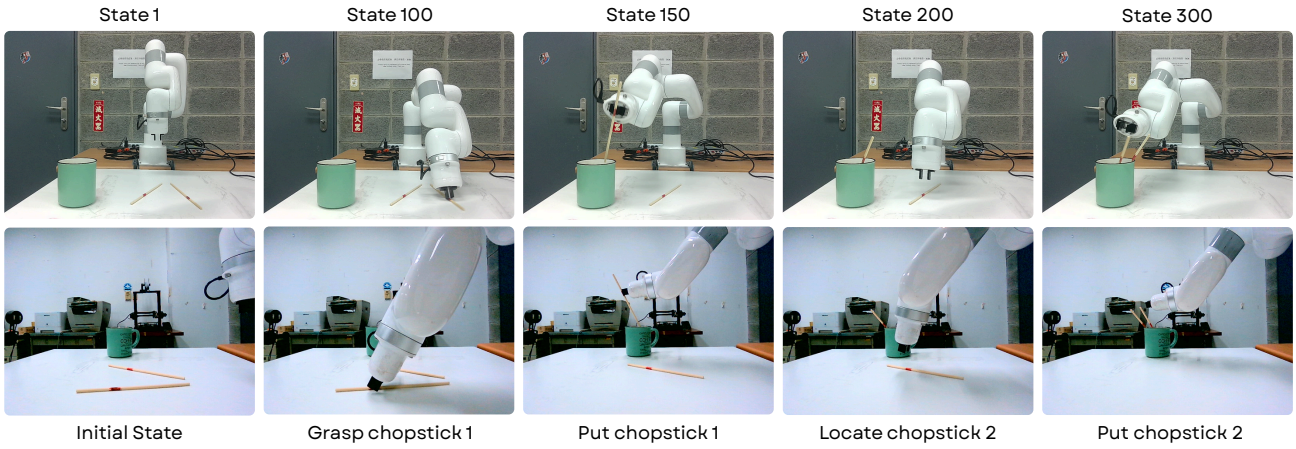


Fig. 5: Chopsticks task demonstrations.

TABLE III: Success rate (%) for the Chopsticks task is evaluated in three stages.

	1st Chopstick			2nd Chopstick		
	Locate	**Lift**	**Put**	**Locate**	**Lift**	**Put**
Proficient Human	96	96	94	94	94	92
BC-ConvMLP	0	0	0	0	0	0
ACT	54	50	44	44	40	34
MEAT*	**88**	**74**	**66**	**56**	**56**	**48**

ACT and BC-ConvMLP across all stages of the Chopsticks task. MEAT achieves the highest success rates in all subtasks, with 88% success in locating the first chopstick, 74% in lifting, and 66% in placing it into the cup. The trend continues for the second chopstick, with MEAT achieving 56%, 56%, and 48% respectively, demonstrating strong consistency. In contrast, ACT shows a noticeable drop in performance, especially in the second chopstick phase, and BC-ConvMLP fails entirely in all stages. These results highlight the robustness and generalization capabilities of MEAT in sequential multi-step manipulation tasks under real-world constraints.

Robustness against perturbation. MEAT's robustness against visual and physical perturbations was evaluated in experiments as shown in **Figure 6**. During experiments, we test robustness with perturbations such as: 1) covering 1 camera view for seconds, or 2) shifting the objects (cup or chopsticks), or 3) taking the chopsticks out of the cup and placing them on the table again while the robot was en route to the end-zone after the completion. The MEAT policy immediately changed course to adjust the robot arm after those perturbations.

Retrying efforts. As explained, when the policy fails to locate the gripper over a chopstick or fails to grasp it, it automatically attempts to retry the action. Due to time constraints, we allow a maximum of four retries, as failure beyond this point typically results in continued failure for the remainder of the trial. As shown in **Figure 7**, we report the average number of retries per model. Fewer retries indicate more efficient and successful policy behavior. For Chopstick 1, MEAT averages 1.96 retries, compared to ACT's 2.65. For Chopstick 2, while ACT appears to require slightly fewer

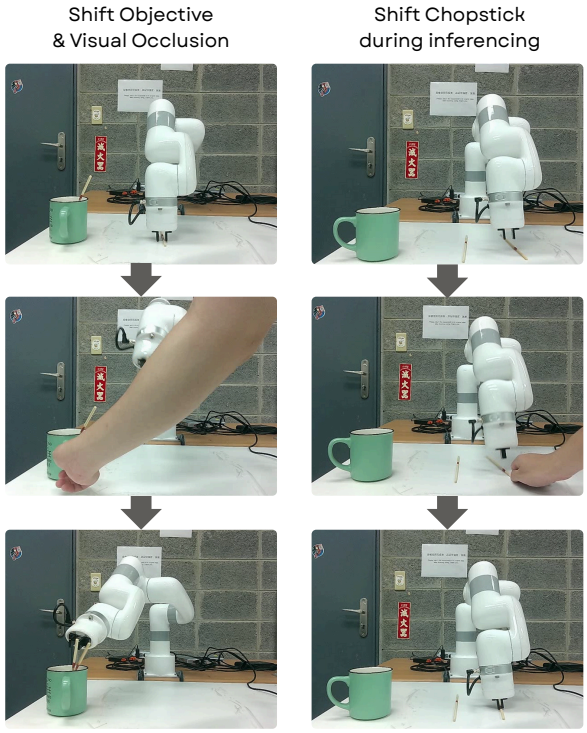


Fig. 6: **Robustness Test for MEAT**. **Left**: Shift the cup and partly cover the front view camera for 3 seconds, but the predicted actions still function as expected. **Right**: shifting the chopstick when the robot nearly reaches the chopstick, the robot still successfully grasp it after replanning.

retries, this count only applies if Chopstick 1 was successfully completed. Referring to **Table III**, we note that fewer ACT trials reached Chopstick 2, which influences the average and slightly inflates the apparent efficiency.

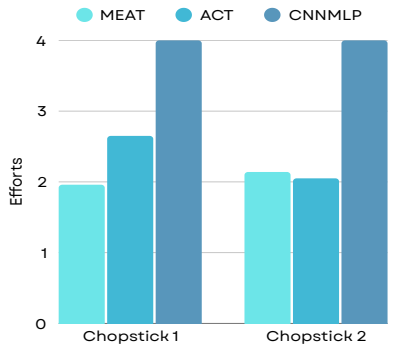


Fig. 7: **Chopsticks task average efforts until SUCCESS or FAIL**. If the policy can not succeed after a maximum of 4 efforts, it will be marked as FAIL.

VI. Limitations & Conclusion

Limitations. This research includes only one real-world experiment. In future work, we will conduct additional real-world experiments to further demonstrate the efficiency of MEAT. The auxiliary-loss-free Mixture-of-Experts (MoE) routing mechanism did not yield successful results in this study, highlighting the need for further research into more effective balancing strategies and loss function design. While the model achieves strong performance, training remains unstable across different random seeds, with some runs producing significantly better outcomes than others; however, a similar issue is also observed in prior work. Additionally, this study focuses primarily on detailed architectural improvements; future research will explore integrating the model into more general frameworks, such as those involving text goal actions or vision-language models.

Conclusion. In this paper, we propose a MOE architecture integrated into a Transformer-based policy to control a robotic arm. Through 6 simulation and 1 realworld experiments across various tasks, our model consistently outperformed the baseline ACT model by margins ranging from 5-20%, demonstrating improved learning efficiency and control precision. Furthermore, when evaluated on the RoboMimic dataset, our approach also achieved higher average performance compared to the highest-performing models from original papers on average, highlighting its robustness and generalization capability. These results suggest that incorporating MoE into policy learning holds strong potential and offers a promising direction for future advancements in robotic control.

References

[1] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, Jul. 2023. DOI: 10.15607/RSS.2023.XIX.016.

[2] C. Chi, Z. Xu, S. Feng, *et al.*, "Diffusion policy: Visuomotor policy learning via action diffusion," *The International Journal of Robotics Research*, vol. 0, no. 0, p. 02 783 649 241 273 668, 0. DOI: 10.1177/02783649241273668.

[3] V. Prasad, A. Kshirsagar, D. Koert, R. Stock-Homburg, J. Peters, and G. Chalvatzaki, "Moveint: Mixture of variational experts for learning human–robot interactions from demonstrations," *IEEE Robotics and Automation Letters*, vol. 9, no. 7, pp. 6043–6050, 2024. DOI: 10.1109/LRA.2024.3396074.

[4] D. Ghosh, H. R. Walke, K. Pertsch, *et al.*, "Octo: An Open-Source Generalist Robot Policy," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, Jul. 2024. DOI: 10.15607/RSS.2024.XX.090.

[5] K. Black, N. Brown, D. Driess, *et al.*, π_0*: A vision-language-action flow model for general robot control*, 2024. arXiv: 2410.24164 [cs.LG].

[6] A. Q. Jiang, A. Sablayrolles, A. Roux, *et al.*, *Mixtral of experts*, 2024. arXiv: 2401.04088 [cs.LG].

[7] DeepSeek-AI, *Deepseek-v3 technical report*, 2025. arXiv: 2412.19437 [cs.CL].

[8] A. Mandlekar, D. Xu, J. Wong, *et al.*, "What matters in learning from offline human demonstrations for robot manipulation," in *5th Annual Conference on Robot Learning*, 2021.

[9] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," in *Proceedings of the 34th International Conference on Neural Information Processing Systems*, ser. NIPS '20, Vancouver, BC, Canada: Curran Associates Inc., 2020, ISBN: 9781713829546.

[10] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010, ISBN: 9781510860964.

[11] S. Dasari, O. Mees, S. Zhao, M. K. Srirama, and S. Levine, "The ingredients for robotic diffusion transformers," *arXiv preprint arXiv:2410.10088*, 2024.

[12] "Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions," in *7th Annual Conference on Robot Learning*, 2023.

[13] N. Shazeer, A. Mirhoseini, K. Maziarz, *et al.*, "Outrageously large neural networks: The sparsely-gated mixture-of-experts layer," *CoRR*, vol. abs/1701.06538, 2017. arXiv: 1701.06538.

[14] W. Fedus, B. Zoph, and N. Shazeer, "Switch transformers: Scaling to trillion parameter models with simple

and efficient sparsity," *J. Mach. Learn. Res.*, vol. 23, no. 1, Jan. 2022, ISSN: 1532-4435.

[15] N. Du, Y. Huang, A. M. Dai, *et al.*, "Glam: Efficient scaling of language models with mixture-of-experts," in *International Conference on Machine Learning*, PMLR, 2022, pp. 5356–5368.

[16] Y. Zhou, T. Lei, H. Liu, *et al.*, "Mixture-of-experts with expert choice routing," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22, New Orleans, LA, USA: Curran Associates Inc., 2022, ISBN: 9781713871088.

[17] H. Hazimeh, Z. Zhao, A. Chowdhery, *et al.*, "Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning," in *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P. Liang, and J. W. Vaughan, Eds., vol. 34, Curran Associates, Inc., 2021, pp. 29 335–29 347.

[18] S. Roller, S. Sukhbaatar, A. Szlam, and J. Weston, "Hash layers for large sparse models," in *Proceedings of the 35th International Conference on Neural Information Processing Systems*, ser. NIPS '21, Red Hook, NY, USA: Curran Associates Inc., 2021, ISBN: 9781713845393.

[19] M. Lewis, S. Bhosale, T. Dettmers, N. Goyal, and L. Zettlemoyer, "Base layers: Simplifying training of large, sparse models," in *Proceedings of the 38th International Conference on Machine Learning*, M. Meila and T. Zhang, Eds., ser. Proceedings of Machine Learning Research, vol. 139, PMLR, 18–24 Jul 2021, pp. 6265–6274.

[20] L. Wang, H. Gao, C. Zhao, X. Sun, and D. Dai, *Auxiliary-loss-free load balancing strategy for mixture-of-experts*, 2024. arXiv: 2408.15664 [cs.LG].

[21] M. Reuss, J. Pari, P. Agrawal, and R. Lioutikov, "Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning," in *The Thirteenth International Conference on Learning Representations*, 2025.

[22] A. O'Neill, A. Rehman, A. Maddukuri, *et al.*, "Open x-embodiment: Robotic learning datasets and rt-x models : Open x-embodiment collaboration," in *ICRA*, 2024, pp. 6892–6903.

[23] A. Khazatsky, K. Pertsch, S. Nair, *et al.*, "DROID: A large-scale in-the-wild robot manipulation dataset," in *RSS 2024 Workshop: Data Generation for Robotics*, 2024.

[24] Z. Fu, T. Z. Zhao, and C. Finn, "Mobile ALOHA: Learning bimanual mobile manipulation using low-cost whole-body teleoperation," in *8th Annual Conference on Robot Learning*, 2024.

[25] A. Mandlekar, S. Nasiriany, B. Wen, *et al.*, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition @ CoRL2023*, 2023.

[26] L. Lai, A. Z. Huang, and S. J. Gershman, "Action chunking as conditional policy compression," *Cognition*, vol. 264, p. 106 201, 2025, ISSN: 0010-0277. DOI: https://doi.org/10.1016/j.cognition.2025.106201.

[27] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *Computer Vision – ECCV 2020*, A. Vedaldi, H. Bischof, T. Brox, and J.-M. Frahm, Eds., Cham: Springer International Publishing, 2020, pp. 213–229, ISBN: 978-3-030-58452-8.

[28] N. M. (Shafiullah, Z. J. Cui, A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," in *Proceedings of the 36th International Conference on Neural Information Processing Systems*, ser. NIPS '22, New Orleans, LA, USA: Curran Associates Inc., 2022, ISBN: 9781713871088.

[29] A. Brohan, N. Brown, J. Carbajal, *et al.*, "RT-1: Robotics Transformer for Real-World Control at Scale," in *Proceedings of Robotics: Science and Systems*, Daegu, Republic of Korea, Jul. 2023. DOI: 10.15607/RSS.2023.XIX.025.

[30] P. Florence, C. Lynch, A. Zeng, *et al.*, "Implicit behavioral cloning," in *Proceedings of the 5th Conference on Robot Learning*, A. Faust, D. Hsu, and G. Neumann, Eds., ser. Proceedings of Machine Learning Research, vol. 164, PMLR, Aug. 2022, pp. 158–168.

[31] J. Pari, N. Shafiullah, S. Arunachalam, and L. Pinto, "The Surprising Effectiveness of Representation Learning for Visual Imitation," in *Proceedings of Robotics: Science and Systems*, New York City, NY, USA, Jun. 2022. DOI: 10.15607/RSS.2022.XVIII.010.

[32] A. Prasad, K. Lin, J. Wu, L. Zhou, and J. Bohg, "Consistency policy: Accelerated visuomotor policies via consistency distillation," in *Robotics: Science and Systems*, 2024.

[33] S. Li, taihang Hu, J. van de Weijer, *et al.*, "Faster diffusion: Rethinking the role of the encoder for diffusion model inference," in *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024.

APPENDIX

A. Teleoperation Method

Algorithm 1 RecordData(arm_1, arm_2, cams)

$t_0 \leftarrow$ current time
while time elapsed $<$ duration **do**
  $a \leftarrow$ servo_angle(arm_1) + eef
  $q \leftarrow$ joint_pos(arm_2), $\theta \leftarrow$ servo_angle(arm_2)
  $I \leftarrow$ capture images from all cameras
  Store a, q, θ, I in HDF5
  if key 'q' **then**
    break
  end if
end while

This system implements a multithreaded teleoperation and data recording pipeline for a dual-arm robot setup. As described in **Algorithm 4**, three concurrent threads are launched to handle key subsystems: end-effector (EEF) control, arm synchronization, and sensor-data logging. Each component operates asynchronously and communicates via shared state (eef_state) and a global termination flag (stop_flag).

The EEFControl thread (see **Algorithm 2**) monitors keyboard input to control the slave robot's end-effector. Pressing '1' opens the end-effector, and '0' closes it. The binary state is stored in a shared list eef_state, making it accessible to both synchronization and logging threads.

Algorithm 2 EEFControl(arm)

while not `stop_flag` **do**
  if key '1' **then**
    open_eef(arm), `eef_state` $\leftarrow 1$
  end if
  if key '0' **then**
    close_eef(arm), `eef_state` $\leftarrow 0$
  end if
end while

Meanwhile, the SyncArms thread **(Algorithm 3**) continuously reads the joint angles of the master arm and appends the eef_state value. These combined control signals are sent to the slave arm using velocity commands. The loop executes at 100Hz (10ms intervals) to maintain smooth and precise motion replication.

Algorithm 3 SyncArms(master, slave)

while not `stop_flag` **do**
  $\theta \leftarrow$ servo_angle(master) + eef
  set_angle(slave, $\theta[:6]$)
  sleep(0.01)
end while

The RecordData function (**Algorithm 1**) logs the master arm's action (servo angles and EEF state), the slave arm's joint states, and image frames from connected USB cameras. The data is serialized and stored in an HDF5 file, structured into action, qpos, qangl, and per-camera image datasets. Recording continues until the defined time limit or user termination.

Algorithm 4 Multithreaded Teleoperation and Data Recording

Init: `stop_flag` $\leftarrow$ false, `eef_state` $\leftarrow 1$
Start thread: `EEFControl(arm_2)`
Start thread: `SyncArms(arm_1, arm_2)`
Start thread: `RecordData(arm_1, arm_2, cams)`
Wait for `RecordData` thread to finish
Signal `stop_flag`
Join all threads
Release cameras, close HDF5, reset arms

Finally, once data recording is complete, all threads are terminated cleanly using stop_flag, resources are released, and the arms are returned to their home configuration. This design achieves real-time synchronized teleoperation with integrated multimodal data collection, suitable for robot learning applications such as behavior cloning or imitation learning.

B. Hyperparameters config

We represent hyperparamerters config for model training on each task in Table IV and V, and a base config using for all tasks as shown in Table VI. We found that for some dexterous RoboMimic and Realword task, decrease chunk size then the policy will be plan better performance. Further for Diffusion model on Tool Transport task, we can not load fully 480x640 images to fit with the RAM on the computer, hence we reduce it to 240x320 then increase the number of epidsode by 50.

TABLE IV: Task specific configs

	Common				Transformer models							
					Common		ACT			MEAT		
Task	Is_real	Views	Chunk size	EpLen	#Episode	ImageRes	BS	LR	Epochs	BS	LR	Epochs
Transfer Cube	F	Top View	100	400	50	480x640	64	5e-5	2500	32	3e-5	2500
Cube Insertion	F	Top View	100	500	50	480x640	64	5e-5	2500	32	3e-5	2500
Tool hang	F	Agent View	20	700	150	480x640	64	5e-5	4000	32	3e-5	4000
Tool transport	F	Agent View	100	600	150	480x640	64	5e-5	4000	32	3e-5	4000
Chopstick	T	Side View, Front View	20	400	49	480x640	48	4e-5	15000	24	2e-5	15000

TABLE V: Task specific configs (cont.)

	BC-RNN					Diffusion				
Task	#Episode	ImageRes	BS	LR	epochs	#Episode	ImageRes	BS	LR	epochs
Tool hang	150	480x640	256	1e-4	1000	150	480x640	128	1e-4	2000
Tool transport	150	480x640	256	1e-4	1000	200	240x320	172	1e-4	2000

TABLE VI: Models base common configs

Transformer models		**Diffusion & BC-RNN**	
Hyper params	**Config**	**Hyper params**	**Config**
#Encoder layers	4	#Layers	8
#Decoder layers	7	#V-params	22
Feedforward dimension	3200	#D-params	9
Hidden dimension	512	Hidden dimension	255
Ctrl	Pos	Ctrl	Pos
KL weight	10	Weight decay	1e-3
# Experts	4	Steps per Epoch	500
Dropout	0.1	Dropout	0.3
Heads	8	Warmup steps	500
Seed	0,42,100	Seeds	0,42,100
Pretrained Encoder	Resnet18	Pretrained Encoder	Resnet18