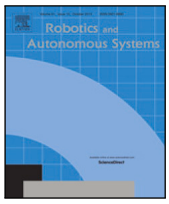
Robotics and Autonomous Systems 205 (2026) 105616

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot



Mixture-of-experts action chunking transformers for high-precision robot imitation learning

Hung Phan Quoc Mai [a,b], Naeem Ul Islam [b,*]

[a] National Economics University, Viet Nam
[b] Yuan Ze University, Taiwan

ARTICLE INFO

Keywords:
Imitation learning
Human teaching robot
Robot control policy

ABSTRACT

Learning visuomotor manipulation policies from demonstrations is often formulated as behavior cloning, where a policy maps visual observations and robot states to future actions. Recent methods such as Action Chunking Transformer (ACT) and diffusion-based policies have improved this paradigm by modeling temporal structure and multimodal action distributions. However, these approaches still face limitations in precision-critical, long-horizon tasks. ACT reduces compounding error by predicting action chunks, but its dense Transformer layers must use the same feed-forward capacity for all manipulation regimes, including visual approach, contact acquisition, transport, alignment, and insertion. This can cause interference between short-term reactive control and longer-horizon goal-directed action prediction. Diffusion policies better represent multimodal actions, but their iterative denoising process increases inference cost, which is undesirable for real-time robot deployment.

We propose MEAT, a Mixture-of-Experts Action Chunking Transformer for high-precision imitation learning. MEAT introduces sparse expert routing into the ACT architecture, allowing different expert modules to specialize for different tokens, task phases, and temporal regimes within an action chunk. This provides conditional model capacity without activating the full parameter set at every step. We further study the role of expert balancing and show that stable routing is important for effective specialization. Across simulated and real-world manipulation tasks, MEAT improves over standard ACT by 5%–30%, while maintaining efficient inference compared with diffusion-based policies. Our results suggest that expert specialization is a targeted architectural mechanism for improving action-chunking policies in precision-critical robot imitation learning.

Note to Practitioners

This paper introduces a Mixture of Experts (MoE)-enhanced Action Chunking Transformer (ACT), which predicts multiple future actions jointly while leveraging specialized expert modules for different contexts. For practitioners, this means a more robust and adaptable policy that can reduce noise in execution and improve accuracy in both simulation and real-world deployments. The approach is especially relevant for applications that require long-horizon decision-making, such as collaborative robots in manufacturing, autonomous warehouse systems, or precision surgical tools. Although the current evaluation demonstrates significant accuracy improvements, further work is needed to validate performance across diverse robot platforms and integrate the method into end-to-end training pipelines for real-time deployment. This work presents a Mixture of Experts (MoE)-enhanced Action Chunking Transformer (ACT) that jointly predicts multiple future actions by utilizing specialized expert modules for different contexts. This provides a more resilient and flexible strategy that can increase accuracy in simulation and real-world deployments while reducing execution noise to meet the demands of practitioners. This methodology is particularly applicable to applications that require long-term decision-making, such as autonomous warehousing systems, precision surgical instruments, or collaborative robots in manufacturing.

1. Introduction

Imitation learning for robotic manipulation is commonly studied under two major paradigms: behavior cloning (BC) and inverse reinforcement learning (IRL). Behavior cloning learns a direct mapping from observations to actions using supervised learning, while IRL infers a reward function from demonstrations and then optimizes a policy with reinforcement learning. In this work, we focus on the behavior-cloning family of methods, which has proven effective for learning robotic skills directly from demonstrations [1–3]. This distinction is important because our goal is not to replace IRL-based imitation learning, but to improve the supervised visuomotor policy architecture

* Corresponding author.
E-mail address: naeem@saturn.yzu.edu.tw (N.U. Islam).

https://doi.org/10.1016/j.robot.2026.105616
Received 5 October 2025; Received in revised form 23 May 2026; Accepted 30 June 2026
Available online 22 July 2026

used in BC. IRL and adversarial imitation learning methods, including trajectory-level approaches such as TrajGAIL [4], can reduce compounding error by learning a reward or trajectory distribution and optimizing a policy against it. However, they typically require reinforcement learning or adversarial optimization in the loop, which can be expensive and difficult to scale for high-frequency visuomotor control. By contrast, BC enables stable and efficient supervised training, but must directly address multimodal demonstrations, temporal consistency, and accumulated prediction errors.

Recent robot policies have addressed these BC challenges using increasingly expressive architectures such as Transformers [5], diffusion models [6,7], GANs [8], and VAEs [9]. Transformer-based policies are especially attractive because they can integrate visual observations, robot states, and language-like instructions using flexible sequence representations. This has led to their widespread use in robot control models such as the Action Chunking Transformer (ACT) [5], Octo [10], and π_0 [11]. More broadly, recent advances in reinforcement learning and robotic manipulation have further improved robustness and decision-making in complex control settings [12,13]. Among these, ACT directly targets a core weakness of one-step BC: small action errors can accumulate over time and push the robot into states that are not well covered by the demonstrations. Instead of predicting only the next action, ACT predicts a sequence of k future actions as an *action chunk*. This reduces the effective control horizon and makes the policy more robust to short-term pauses, corrections, and temporal noise in human demonstrations.

However, action chunking also introduces a new architectural challenge. A single forward pass must predict both near-term actions that are strongly conditioned on the current observation and later actions that depend more on task intent, object interaction, and longer-horizon dynamics. In manipulation tasks, these temporal regimes often correspond to different task phases, such as visual approach, grasping, transport, handoff, contact-rich alignment, and insertion. Standard ACT uses dense Transformer feed-forward layers shared across all tokens, phases, and temporal positions. Therefore, the same parameters must model both reactive feedback control and more distal goal-directed action prediction. Simply increasing the size of this dense model can add capacity, but it does not explicitly reduce interference between these different action-generation regimes and may increase computation uniformly for every token.

Diffusion-based policies provide another strong solution for visuomotor imitation learning because they can represent complex and multimodal action distributions [6,7]. Nevertheless, their inference typically requires iterative denoising, which can be costly for real-time robot deployment. This creates a practical trade-off: diffusion policies can improve expressiveness, but direct action-chunking policies remain attractive when low latency and efficient deployment are important. The open question is therefore how to increase the conditional capacity and specialization of an action-chunking policy without losing the efficient inference structure that makes ACT practical.

We propose MEAT, a Mixture-of-Experts Action Transformer policy, to address this limitation. Rather than introducing Mixture of Experts (MoE) only as a generic scaling technique from large language models, we use it as a targeted mechanism for conditional specialization in visuomotor policy learning. MoE [14] activates only a sparse subset of expert modules for each token, allowing the model to increase representational capacity without activating all parameters at every step. While MoE has shown strong effectiveness in large-scale models such as DeepSeek [15] and Mixtral [14], its role in robot imitation learning is different: the goal is not only parameter scaling, but also to let different experts specialize for different task contexts, temporal positions, and manipulation phases.

This design is especially compatible with action chunking. Since an action chunk contains both immediate and distal future actions, the policy must represent multiple control regimes within the same prediction. Sparse expert routing provides a natural way to allocate different feed-forward capacity to these regimes. For example, some experts may specialize in visually guided approach and grasping, while others may specialize in transport, alignment, or contact-rich insertion. We hypothesize that MoE is particularly helpful for the distal part of a predicted chunk, where prediction depends less on the instantaneous observation and more on inferred task progress and longer-term dynamics. At the same time, the same routing mechanism can preserve reactive near-term control by selecting experts suited to the current observation. Thus, MoE directly addresses the interference problem in dense ACT layers by providing conditional capacity and phase-dependent specialization.

In this study, we integrate MoE layers into the ACT architecture and study the effect of expert balancing during training. We evaluate MEAT on simulated manipulation benchmarks from ALOHA and RoboMimic [16], as well as on real-world manipulation tasks using UFactory Lite 6 robot arms (Fig. 1). We also analyze expert routing behavior to understand whether the learned experts are used in a structured way. Our results show that MEAT improves over standard ACT across multiple simulated and real-world tasks while maintaining efficient inference. At the same time, we view this work as an initial step toward scaling MoE-based robot policies: our experiments study the feasibility, performance, and routing behavior of MoE-enhanced action chunking, while larger-scale datasets and broader scaling curves remain important directions for future work.

In summary, our main contributions are as follows:

- We identify a key limitation of dense action-chunking policies: although ACT reduces the effective control horizon, its shared Transformer layers must model multiple manipulation regimes with the same parameters, which can cause interference in high-precision long-horizon tasks.
- We propose MEAT, a Mixture-of-Experts Action Transformer policy that integrates sparse expert routing into ACT to provide conditional capacity and encourage specialization across task phases, temporal positions, and action-generation regimes.
- We study expert balancing for MoE-based action-chunking policies and show that stable routing is important for effective specialization; simply spreading load across experts is insufficient if routing becomes noisy or weakly aligned with task structure.
- We empirically demonstrate that MEAT improves over the original ACT across multiple simulated and real-world manipulation tasks, including ALOHA and RoboMimic [16] benchmarks, while preserving efficient inference compared with more expensive generative policies.

2. Related works

Inverse Reinforcement Learning and Adversarial Imitation Learning. Beyond behavior cloning, another major direction in imitation learning focuses on recovering a reward function from demonstrations through inverse reinforcement learning (IRL) [17–19]. Adversarial imitation learning methods such as GAIL [20] extend this idea by learning policies that match expert trajectory distributions while optimizing a learned reward. Recent work has further connected IRL with generative modeling and diffusion models via maximum-entropy formulations [4,21]. These approaches often improve robustness to compounding error and better capture long-horizon objectives. For example, TrajGAIL [4] models sequential decision-making with variable-length dependencies to capture long-term trajectory structure. However, IRL-based methods typically require reinforcement learning in the loop, making them computationally expensive and difficult to scale for high-frequency visuomotor control. In contrast, behavior cloning methods such as ACT [5] enable stable and efficient supervised training, which is particularly suitable for robotic manipulation with large demonstration datasets. Our work therefore focuses on advancing the behavior cloning paradigm.

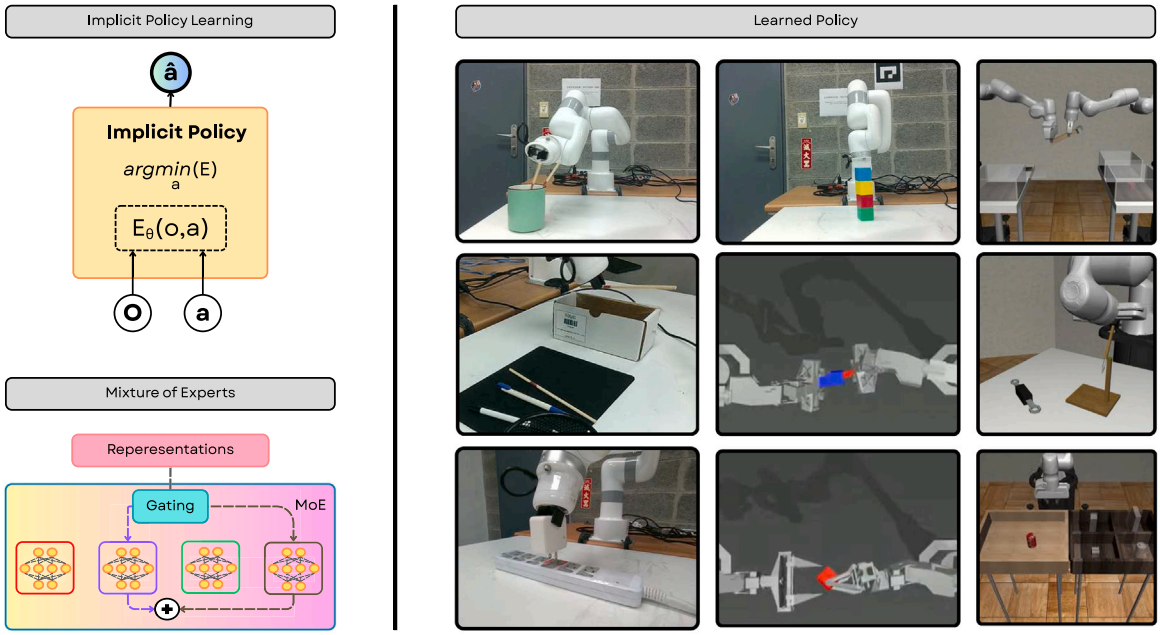


Fig. 1. MEAT: *M*ixture of *E*xperts in *A*ction *T*ransformer Policy. *Upper Left*: Implicit policy learns a function conditioned on both action and observation and optimizes for actions that minimize the error. *Bottom Left*: The visualization of Mixture of Experts routing in our new model, a part of the Transformer Policy. *Right*: Examples of learned policy and deployed to the real robot. We will showcase of both simulation and real robots.

Advances in Imitation Learning. Many of recent research has been directed toward imitation learning for robots, and many approaches have been presented. Since the success of the Diffusion model [22] and the Transformer model [23], these models are widely studied and applied for use as a policy for robotics manipulation. [6] 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 [24] 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 h 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. [5] 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. [10] 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 [11] proposes a generalist robot policy using a novel flow matching architecture built on a pretrained 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 [25] 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 adopted by [26], which introduced conditional routing via a gating network. With the success of Transformers, MoE architectures evolved most notably with Switch Transformers [27], which incorporated sparse expert feed-forward layers to scale model capacity while keeping computation efficient, improving model capacity and efficiency. This approach enables scaling to larger models while maintaining manageable computational costs. This design became foundational for many recent Large Language Models: Mixtral [14], Deepseek-v3 [15], GLaM [28], and gpt-oss [29]. Notably, at the time Mixtral 8 × 7B – Instruct [14] 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 [26,27] address this by encouraging more uniform expert usage. Various routing strategies have since been proposed, including expert choice routing [30], differential k-selection [31], frozen hashing [32], and linear assignment [33]. Instead of learning an auxiliary loss for gating expert balancing, DeepSeek [34] 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. One such approach is MoDE [35] which 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 [9] 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 [36] 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 [37] with 76k demonstration trajectories or 350 h of interaction data, collected across 564 scenes and 86 tasks by 50 data collectors; and Mobile-ALOHA [38] which is a low-cost and whole-body teleoperation system for data collection. In simulation, Robomimic [16] presents an extensive evaluation of six offline learning algorithms on diverse simulated and real-world multi-stage manipulation tasks using human demonstration 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 [39] 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.

3. Approach

3.1. 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 D, at each timestep the policy outputs a sequence of k future actions $\hat{a}_{t:t+k}$ 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:

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

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

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

3.2. 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 [40], 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 [5].

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 ith 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.

Fig. 2 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 [41], 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 **0**. These encoded features are then passed to 8 Transformer decoder blocks to autoregressively generate the action sequence, conditioned on learned position embeddings.

3.3. MoE-enhanced action transformer policy

To evaluate the efficiency of advanced Transformer architectures, we introduce the **MEAT** (MoE-Enhanced Action Transformer) policy, which integrates a Mixture-of-Experts (MoE) module into the Transformer architecture. 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. 3 describes the Transformer block augmented with a 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 [27]:

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

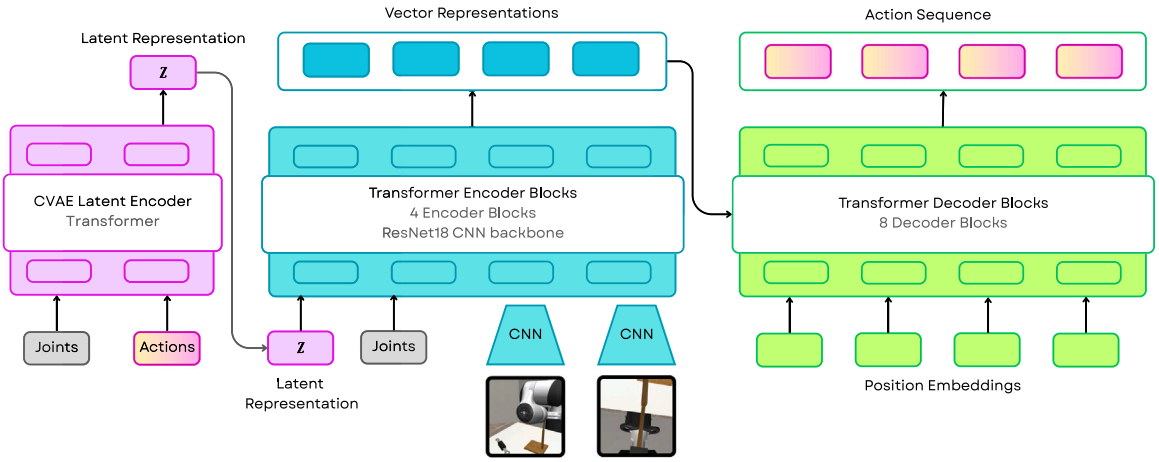


Fig. 2. Architecture of the overall MEAT (ACT) model. MEAT (ACT) is trained under a Conditional VAE (CVAE) framework with an encoder–decoder architecture. The encoder maps the action sequence and joint observations into a latent style variable z, which is discarded during inference. The decoder (policy) employs a transformer encoder to synthesize images from multiple viewpoints, joint positions, and z, and a transformer decoder to predict the corresponding sequence of actions. At test time, z is set to the mean of the prior distribution (i.e., zero).

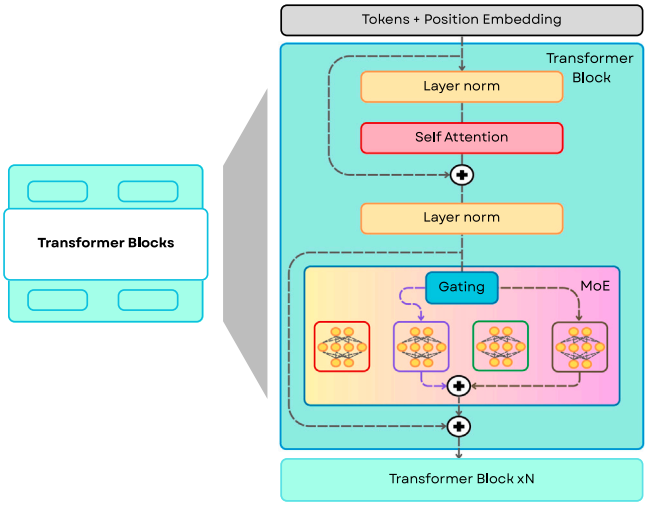


Fig. 3. Transformer Block architecture composition including Mixture of Experts. Following processing through the Self-Attention mechanism and Layer Normalization, the embedding vectors are subsequently input into a gating function, which adaptively selects two experts from a pool of four for each token.

where N denotes the number of experts and B is the mini-batch of tokens processed by the router. The load-balancing loss is defined using two complementary statistics computed over the batch:

$$f_n = \frac{1}{|B|} \sum_{i=1}^{|B|} \mathbb{1}\left[n \in \mathrm{TopK}\left(\phi(\sigma_t)_i\right)\right], \tag{7}$$

$$p_n = \frac{1}{|B|} \sum_{i=1}^{|B|} \mathrm{softmax}\left(\phi(\sigma_t)_i W_R\right)_n. \tag{8}$$

Here, $\sigma_t \in \mathbb{R}^{|B| \times d}$ denotes the token representations at Transformer layer t, where d is the hidden dimension. The routing network $\phi(\cdot)$ maps each token representation to routing logits over the N experts. The matrix $W_R \in \mathbb{R}^{d \times N}$ is the router projection that produces expert-selection logits, which are converted into routing probabilities via a softmax. The indicator function $\mathbb{1}[\cdot]$ evaluates to 1 when expert n is selected among the Top-k experts for token i, and 0 otherwise. Consequently, f_n measures the fraction of tokens that are actually routed to expert n (hard assignment), while p_n represents the average routing probability assigned to expert n across the batch (soft assignment). Minimizing LB encourages both quantities to approach a uniform distribution over experts, thereby preventing expert collapse and promoting balanced expert utilization.

An important consideration is the interaction between the action-chunking horizon and the role of the MoE layer. While action chunking reduces compounding error by predicting multiple future actions jointly, it also increases the difficulty of long-horizon prediction within a single forward pass. The model must simultaneously represent immediate reactive control and longer-term goal-directed motion across the predicted chunk. We hypothesize that the MoE layer complements action chunking by enabling implicit temporal specialization across different phases of the predicted action sequence.

In manipulation tasks, early actions in a chunk are typically observation-driven and reactive, whereas later actions depend more heavily on inferred task intent and longer-term dynamics. A single shared feed-forward network must otherwise model both regimes, which can limit representation capacity. Sparse expert routing allows different experts to specialize in these distinct temporal behaviors, reducing interference between short-term feedback control and longer-horizon planning. Evidence supporting this interpretation can be observed in the gating visualizations (Fig. 6), which show consistent yet diverse expert activation patterns across timesteps and layers.

Although our experiments do not explicitly isolate prediction accuracy at different positions within the chunk, the consistent improvements on long-horizon manipulation tasks suggest that MoE primarily

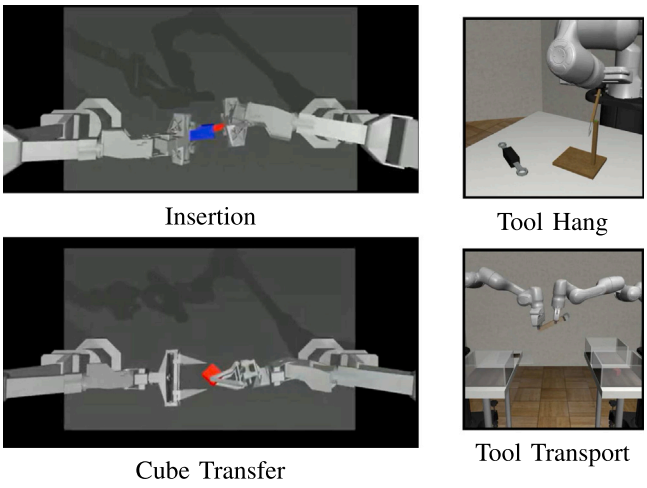


Fig. 4. Illustrations of 4 Simulated tasks. Insertion and Cube Transfer from ALOHA (Left), Tool Hang and Tool Transport from RoboMimic (Right).

benefits the distal portion of predicted action sequences, where model capacity and specialization become increasingly important.

3.4. Auxiliary-loss-free load balancing strategy

In addition to employing the standard MoE architecture, we also evaluate the Auxiliary-Loss-Free Balancing Strategy proposed by DeepSeek [34]. Specifically, instead of relying solely on naive top-K gating, [34] 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:

$$g_{i,t} = \begin{cases} s_{i,t}, & s_{i,t} + b_i \in \text{TopK}(\{s_{j,t} + b_j \mid 1 \le j \le N\}, K) \\ 0, & \text{otherwise.} \end{cases} \tag{9}$$

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.

4. Simulation experiments

The simulation experiments are designed to validate MEAT under controlled and repeatable conditions. We evaluate three main questions. First, does sparse expert specialization improve over dense ACT on precision-critical action-chunking tasks? Second, can MEAT improve task success while preserving the direct-inference efficiency of ACT, especially compared with diffusion-based policies? Third, does expert balancing affect whether MoE routing becomes structured and useful rather than noisy and unstable? The selected tasks stress different aspects of these questions: ALOHA Transfer Cube and Bimanual Insertion require high-precision bimanual coordination with small spatial clearances, while RoboMimic Tool Hang and Tool Transport require longer-horizon, multi-stage manipulation from human demonstrations.

4.1. 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 [5], and 2 RoboMimmic tasks [16] Tool Hang and Tool Transfer as shown in Fig. 4. Among Robomimic tasks, we chose 2 tasks, Tool Hang and Tool Transfer, as they are the most challenging since they require many steps and dexterity 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, a behavior that can confuse the robot during imitation learning.

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.

4.2. Experiment setup

We conduct all training and inference on an RTX 4080 GPU with 16 GB of memory. Each task is trained using 3 random seeds. When training ACT models, for ALOHA tasks, training takes approximately 40 min per run for 3000 epochs. In contrast, RoboMimic tasks require around 1.5 h 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 480×640. 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.

4.3. Evaluation method

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.

4.3.1. 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 [42] 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 [43] 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 continuous offset from

Table 1
Success rate (%) for 2 simulated tasks from ALOHA paper. The bold Insert and Transfer means completion rate. MEAT outperforms the original ACT and other models.

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 [42]	0	0	0	21	4	3	16	13	1	60	51	27
RT-1 [43]	0	0	0	2	0	1	4	2	0	44	33	2
VINN [44]	0	0	0	6	1	1	17	11	0	13	9	3
ACT [5]	73	55	11	96	88	33	97	55	55	97	79	80
MEAT	**85**	**60**	**21**	**97**	**88**	**65**	**89**	**57**	**64**	**99**	**93**	**85**
MEAT Aux-free	81	61	16	99	81	37	97	62	61	100	78	77

the bin center. In contrast, ACT directly predicts continuous actions, which is better suited for precise manipulation tasks. Finally, VINN [44] 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.

4.3.2. RoboMimic tasks

We compare RoboMimic data result with 2 models as benchmarks. Implicit Behavior Cloning (IBC) [45] 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 [16], a recurrent variant of BC 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.

4.4. Results

The experiment result of tasks from ALOHA dataset is represented in Table 1. Unfortunately, we cannot reproduce exactly the reported result in ALOHA paper [5], however some tasks achieve higher results (insertion scripted, transfer cube human) while the other achieve lower results (insertion human, transfer cube scripted). From results in Table 1, it is clear that MEAT consistently outperforms all other models, achieving the highest scores in most categories. In particular, in the Insertion human task, MEAT achieves 85% in grasp, 60% in contact, and 21% in insert, significantly surpassing ACT and other baselines. Similarly, in Insertion scripted, MEAT reaches near-perfect scores (97% grasp, 88% contact, 97% insert). On the Transfer cube tasks, MEAT again leads with 65%–97% success in the human setting and 99–93%–85% in the scripted setting. Other models, such as ACT and MEAT Aux-free, also perform competitively but generally lower than MEAT. Notably, baseline models like BC-ConvMLP, BeT, RT-1, and VINN exhibit much lower success rates, often close to 0% on several sub-tasks. These results demonstrate that MEAT significantly improves task performance, particularly in challenging manipulation scenarios, and outperforms both the original ACT and all other compared methods.

These results support the first hypothesis: adding MoE to ACT improves precision-critical action-chunking performance. The largest gains appear in completion-stage metrics such as Insert and Transfer, which are the stages most sensitive to accumulated errors and precise coordination.

Table 2 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 Fig. 5, 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 [46,47], 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.

Table 2
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 [45]	0 | 0	0 | 0
BC-RNN [16]	24 | 18	38 | 28
DP [6]	40 | 34	**50** | **40**
ACT [5]	46 | 38	44 | 34
MEAT	**52** | **46**	44 |38
MEAT Aux-free	48 | 28	24 | 12

4.5. Discussion

Although BC-RNN can yield good performance, we found that training it is often slow and requires a large amount of RAM. Specifically, we had to reduce the number of demonstrations from 200 to 150 in order to fit the entire dataset into 120 GB 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 h for Tool Transport and 10 h for Tool Hang, which, compared to ACT completes training in only 1.4 h while still achieving better results. This further highlights the efficiency and effectiveness of MEAT.

Regarding Fig. 5, 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.

These results support the second hypothesis. MEAT remains competitive with Diffusion Policy on long-horizon RoboMimic tasks, while retaining the efficient single-pass inference structure of ACT. Thus, the benefit of MEAT is not only higher success over dense ACT, but also a more favorable accuracy–efficiency trade-off compared with iterative generative policies.

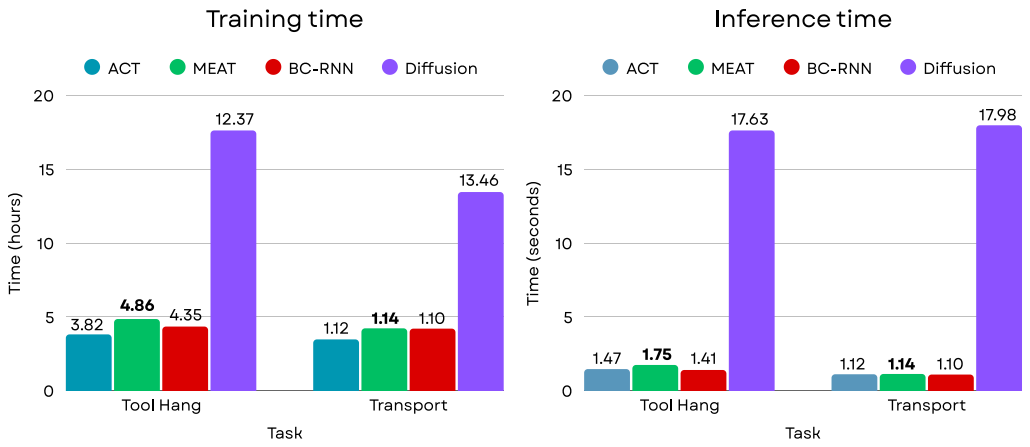


Fig. 5. Training and Inference Time Benchmark. Inference time is measured as the total time the model takes to predict an action, excluding the time taken to execute the action step. Training time is reported in hours, inference time is reported in seconds.

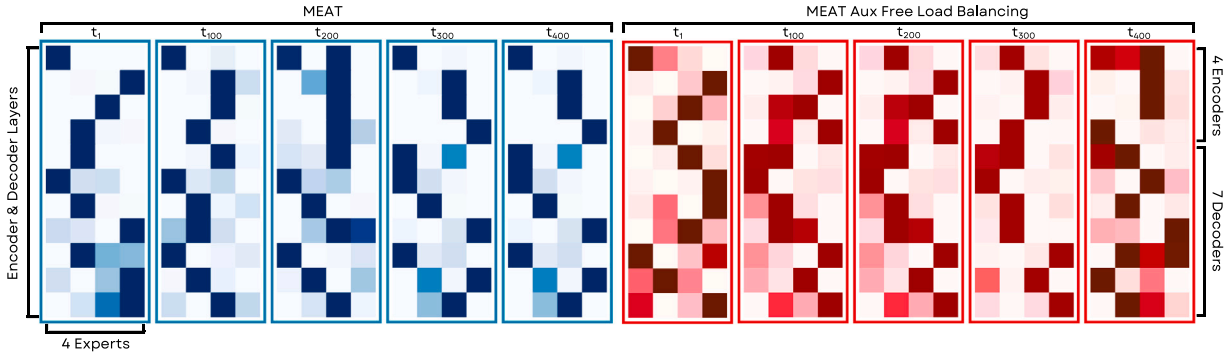


Fig. 6. Visualized Expert Gating Samples. The average usage of all experts in MEAT across all layers in some sampled timestep. Brighter blue/red corresponds to low usage and bolder to high one, and each image is separately normalized. Each matrix corresponds to 1 timestep during inference and has a size of 4×11 (which is 4 experts within 11 decoder and encoder layers). The activation shows that the Aux-Free load balancing strategy produces more noisy gating probabilities.

4.6. MoE analysis

4.6.1. Auxiliary-free load balancing

To better understand why MEAT with aux-free load balancing performs poorly, we analyze the expert utilization by visualizing the gating probabilities for each expert, as shown in Fig. 6. The results clearly demonstrate that the standard MEAT model exhibits more focused and consistent expert selection, leading to cleaner and more concentrated gating decisions. In contrast, the aux-free load balancing variant displays significantly noisier and more scattered gating behavior, suggesting that the model has difficulty in confidently selecting which experts to activate. This noisy expert selection results in less specialization among the experts, meaning that each expert is not sufficiently or consistently trained. Consequently, the overall model performance suffers due to the lack of clear routing and expert specialization, which are critical for the effectiveness of MoE architectures. These observations indicate that MEAT with aux-free load balancing is currently ineffective, and further investigation is necessary to improve its expert routing mechanism, either by designing better load balancing strategies or by introducing alternative forms of regularization to promote expert diversity and specialization.

4.6.2. Expert statistical analysis

We next quantify routing behavior to complement the qualitative timelines. Table 3 compares MEAT with the auxiliary-free routing variant across load balance, similar-input consistency, phase dependence, and temporal stability. Formal metric definitions are provided in Appendix A. The auxiliary-free variant achieves higher routing entropy and a larger effective number of experts, indicating that it spreads assignments more evenly across experts. However, better load spread does not necessarily imply better routing structure. On both Transfer Cube and Bimanual Insertion, auxiliary-free routing substantially reduces nearest-neighbor agreement, from 97.3% to 72.0% on Transfer Cube and from 97.5% to 73.1% on Bimanual Insertion. This suggests that similar policy states are less consistently assigned to the same dominant expert. The same trend appears in phase dependence and temporal stability. Compared with MEAT, the auxiliary-free variant has lower phase NMI on both tasks (0.203 to 0.081 for Transfer Cube and 0.509 to 0.306 for Bimanual Insertion), indicating weaker alignment between expert assignment and manipulation phase. It also has much higher switch rates, increasing from 0.9% to 13.5% on Transfer Cube and from 0.5% to 8.4% on Bimanual Insertion. Thus, while auxiliary-free balancing encourages more uniform expert usage, it produces less stable and less phase-structured routing. In contrast, MEAT maintains sufficient load balance while preserving high similar-input consistency, low temporal switching, and stronger phase-dependent specialization. These results support the interpretation that the auxiliary load-balancing objective does not merely balance expert usage, but also helps the router learn more stable and interpretable expert assignments. Layer-level routing statistics are provided in Appendix B.

4.6.3. Expert usage over time

We first visualize expert usage over time for two individual rollout examples, one from Bimanual Insertion and one from Transfer Cube. The sampled Transfer Cube rollout in Fig. 7 shows a similar phase-dependent pattern. During approach and grasp, the decoder uses a mixture of E1/E2/E3, while after the cube is grasped, routing shifts strongly toward E0 and remains concentrated during lift, handoff,

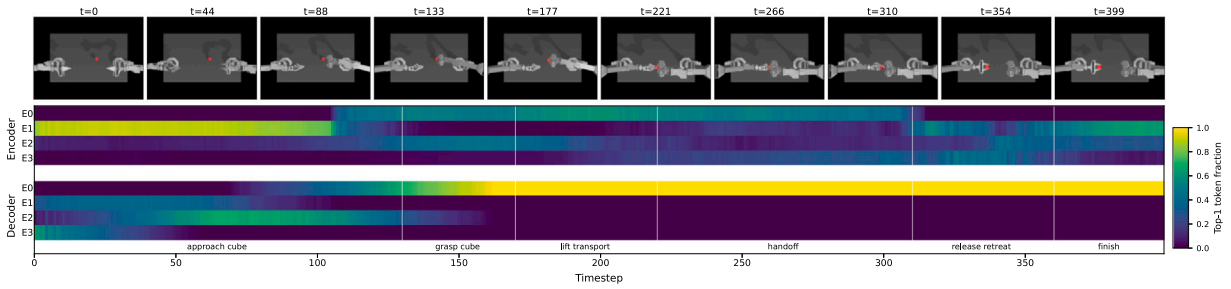


Fig. 7. Visualized Expert Gating over Time of Cube Transfer task. Expert-gating patterns in a sampled layer across the rollout timeline. Each row shows an expert, and each column corresponds to a time chunk where the expert is activated.

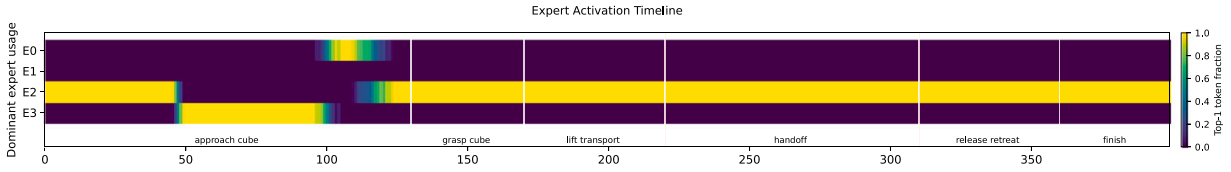


Fig. 8. Aggregated expert activation timeline for Transfer Cube. We average the top-1 expert-token fraction over 50 rollouts for a selected MoE layer. Sampled frames on top show the corresponding task progression, while the heatmap below shows dominant expert usage across time. The averaged routing remains sparse and phase-structured: E3 is mainly activated during early approach, E0 appears near the grasp/transport transition, and E2 dominates the later grasp, lift, handoff, release, and finish phases.

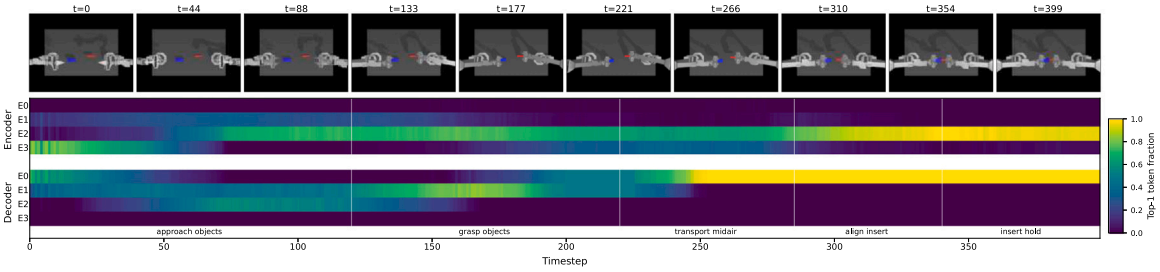
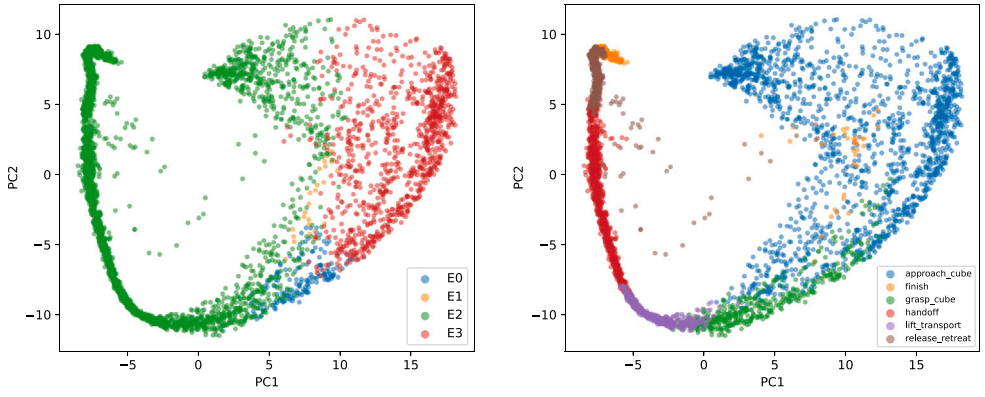


Fig. 9. Visualized Expert Gating over Time of Insertion task. Expert-gating patterns in a sampled layer across the rollout timeline. Each row shows an expert, and each column corresponds to a time chunk where the expert is activated.



(a) Colored by expert. (b) Colored by phase.

Fig. 10. PCA visualization of routed hidden states on Transfer Cube. We visualize hidden states before MoE routing in a sampled decoder layer and show the same projection colored by selected expert and by task phase. The phase-colored view reveals a task-progress manifold, while the expert-colored view shows that routing aligns with different regions of this manifold, indicating phase-dependent expert specialization.

release, and finish. This suggests a transition from perception/contact acquisition to sustained transport and control. To verify that this pattern is not only specific to one sampled rollout, Fig. 8 reports the routing timeline averaged over 50 Transfer Cube rollouts for one selected layer. The averaged result remains sparse and phase-structured: E2 dominates most of the rollout, E3 is selected mainly during an early approach interval, and E0 briefly activates near the transition into grasp/transport. Further, as shown in Fig. 9, routing changes systematically across manipulation phases rather than remaining uniform throughout the episode. In the sampled Bimanual Insertion rollout,

Table 3
MoE routing stability under auxiliary-free balancing. Statistics are computed over 100 scripted-task inference rollouts and compare MEAT with an auxiliary-free routing variant. Auxiliary-free balancing increases routing entropy and the effective number of experts, but NN agreement, Phase NMI, and Switch show that this load spread comes with less structured and less temporally stable routing.

	Variant	Entropy ↑	Eff. exp. ↑	KL ↓	NN agree ↑	Phase NMI ↑	Switch ↓
Trans.	MEAT	0.800	3.080	0.278	97.3%	0.203	0.9%
	Aux-free	0.930	3.620	0.294	72.0%	0.081	13.5%
Ins.	MEAT	0.870	3.373	0.180	97.5%	0.509	0.5%
	Aux-free	0.954	3.810	0.161	73.1%	0.306	8.4%

the encoder gradually shifts from early E3/E1 activity during object approach toward stronger E2 usage during later grasping, transport, alignment, and insert-hold phases. This suggests that encoder routing adapts as the observation regime changes from free-space approach to contact-rich alignment. The decoder shows an even clearer action-phase structure: E1/E2 are more active during approach and grasp, while E0 becomes dominant during mid-air transport, alignment, and insertion hold. Since decoder tokens are closer to action prediction, this indicates that the MoE layer may allocate different action-generation regimes to different experts. These results support the interpretation that MEAT learns temporally organized expert usage, where experts are preferentially activated during specific manipulation phases rather than being uniformly mixed across the full trajectory.

4.6.4. PCA analysis of routed hidden states

We further examine whether expert routing reflects meaningful internal structure by visualizing hidden states before MoE routing in decoder layer 6 on Transfer Cube. Fig. 10 shows the same PCA projection with two colorings: task phase and selected expert. The phase-colored view reveals a structured task-progress manifold: *approach_cube* occupies a broad right-side region, while later phases such as *grasp_cube*, *lift_transport*, *handoff*, and *release_retreat* follow a curved trajectory toward the left. This suggests that the decoder representation captures task progression. The expert-colored view shows that routing aligns with this manifold. E3 is mainly concentrated in the approach region, E2 dominates the later post-contact trajectory, and E0/E1 appear more selectively near transition regions. Together with Table 3, this indicates that MEAT learns phase-dependent expert routing rather than arbitrary or uniformly mixed gating.

Overall, the temporal visualizations, PCA plots, and routing statistics show that MEAT maintains reasonable model-level load balance while learning structured expert assignments. Expert usage changes systematically across approach, grasp, transport, handoff, alignment, and insertion phases, and nearby policy states receive consistent expert assignments. These results suggest that MEAT decomposes manipulation behavior into stable expert-specific regimes.

5. Realworld experiments

5.1. Setup

The real-world experiments test whether the improvements observed in simulation transfer to physical robot deployment. Unlike simulation, real robot execution includes perception noise, calibration error, object variation, latency, and imperfect contact dynamics. We therefore evaluate MEAT across tasks that isolate different practical requirements: sequential multi-object manipulation, contact-rich insertion, long-horizon stacking, object-conditioned sorting, and robustness to perturbations. The main validation question is whether conditional expert capacity improves precision and robustness over ACT while still allowing real-time deployment on limited hardware.

Table 4
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**

Teleoperation. 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.

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 an estimated 12,000 to 30,000 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.

5.2. Task 1: 'Chopsticks'

Our first experiment is named Chopsticks. This task tests sequential multi-object manipulation and recovery from failed attempts. It is useful for evaluating whether MEAT can maintain consistent behavior across repeated pick-and-place stages, rather than succeeding only on a single short manipulation.

In this task, the robot is equipped with a gripper as its end-effector. As shown in Fig. 11, 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 h 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 4 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 ACT and BC-ConvMLP across all stages of the Chopsticks task. MEAT achieves the highest success rates in all sub-tasks, 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

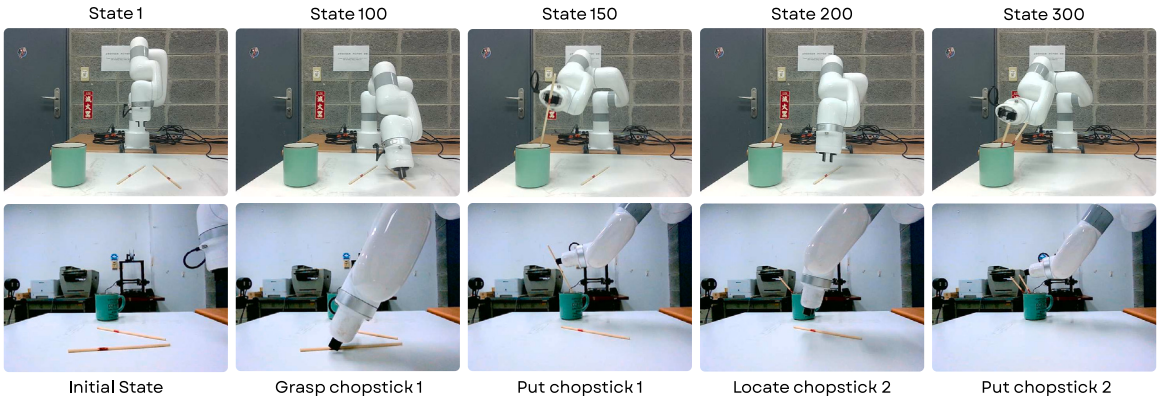


Fig. 11. Chopsticks task demonstrations. The steps in this illustration correspond to the robot states from state 1 to state 300.

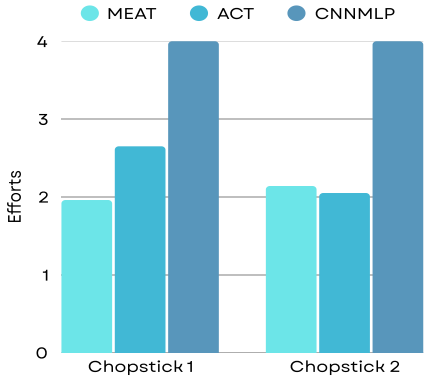


Fig. 12. Chopsticks task average efforts until success or fail. If the policy cannot succeed after a maximum of 4 efforts, it will be marked as FAIL.

Table 5
Success rate (%) for the 3 real-world tasks: 'Plug Insertion', 'Stack 4', and 'Sort Pens & Chopsticks'.

Task	Plug Insertion		Stack 4			Sorting	
	Pick Up	Insertion	Stack 2	Stack 3	Stack 4	Pen	Chopsticks
Human	98	90	96	94	94	98	100
ConvMLP	0	0	0	0	0	0	0
ACT	94	46	22	10	2	69	59
MEAT	**96**	**62**	**26**	**14**	**4**	**84**	**82**

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.

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 Fig. 12, 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 retries, this count only applies if Chopstick 1 was successfully completed. Referring to Table 4, we note that fewer ACT trials reached Chopstick 2, which influences the average and slightly inflates the apparent efficiency.

The Chopsticks results support this hypothesis: MEAT improves over ACT at every stage and degrades less severely from the first object to the second object, indicating better stability across repeated sequential manipulation.

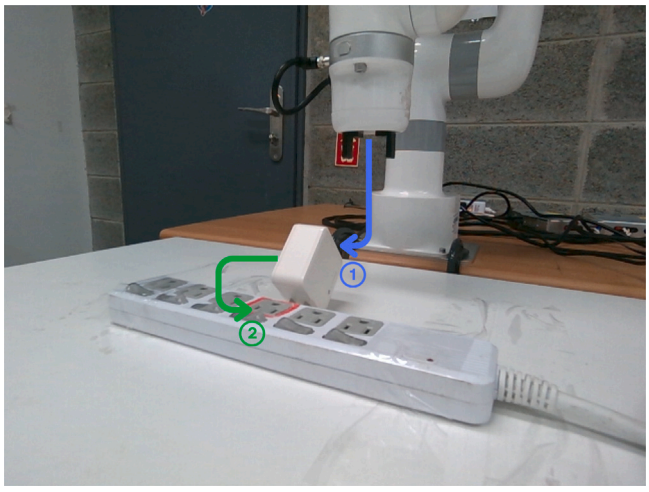


Fig. 13. Plug Insertion task. This task involves: (1) picking up a power adapter, (2) bringing it to a plug and inserting it completely.

5.3. Task 2: 'Plug insertion'

The second experiment, inspired by Consistency Policy [46], is called 'Plug Insertion'. This task tests contact-rich precision. Picking up the plug mainly evaluates object localization and grasping, while insertion evaluates whether the policy can maintain accurate alignment under tight contact constraints.

As shown in the Fig. 13, a plug is placed in front of a socket, and the robot must pick up the plug and insert it into the socket. This task is more contact-rich and requires greater precision. We trained the ACT and MEAT models for 25,000 steps, which took approximately six hours using 50 demonstrations. All demonstration trajectories were padded to 400 steps.

Result Analysis. Table 5's first column presents the success rates for the plug insertion task. Both MEAT and ACT demonstrate high performance in the Pick up stage, achieving success rates of 96% and 94%, respectively–close to the human benchmark of 98%. The key difference lies in the Insertion stage: MEAT achieves a 62% success rate, notably higher than ACT's 46%, highlighting MEAT's superior efficiency. During experiments, we observed that ACT exhibited specific failure modes. For example, in the first training seed, the policy could guide the robot arm to the plug but failed to close the gripper. In the second seed, the policy navigated near the socket but took an excessively long time to lower the arm for insertion. These results suggest that, despite its competence in grasping, ACT suffers from stability issues during insertion.

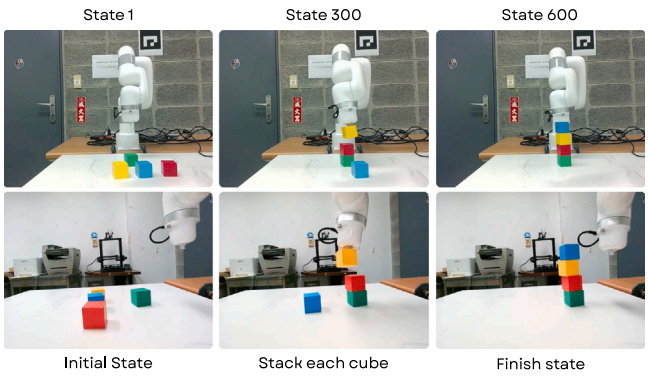


Fig. 14. Task Stack 4 demonstrations.

The results support the contact-precision hypothesis. ACT and MEAT perform similarly in the Pick Up stage, but MEAT is substantially better in the Insertion stage, suggesting that the main advantage appears in the more precision-sensitive part of the task.

5.4. Task 3: 'Stack 4'

The third experiment is Stack 4, a classic robotic manipulation benchmark involving multi-step stacking. This task tests long-horizon error accumulation. Each additional cube increases the number of required perception, grasping, and placement decisions, making Stack 4 a direct stress test for temporal stability in real-world action-chunking policies.

Such tasks are widely studied due to their relevance to long-horizon control and precise manipulation. For example, [48] investigates stacking cubes of different shapes, while [49] studies building stone towers from irregular objects.

In our setup, four cubes (Red, Yellow, Blue, and Green) are placed on the table, with the Green cube positioned near the robot arm's center. The objective is to stack the Red, Yellow, and Blue cubes sequentially on top of the Green cube to form a stable tower. The robot uses a vacuum end-effector (Fig. 14). Compared to the Sorting task, Stack 4 is significantly more challenging: the robot must accurately locate and grasp each cube and place it precisely onto the growing stack using only two camera views. For training, we collected 65 demonstrations from proficient human operators and trained the model for 30,000 epochs (approximately 7 h). Each trajectory contains 600 timesteps.

Result Analysis. The Stack 4 task is particularly challenging, as shown in Table 5. Although MEAT consistently outperforms ACT, the overall success rates remain low due to the difficulty of long-horizon manipulation. MEAT achieves 26% success on stacking two blocks, 14% on stacking three blocks, and 4% on completing the full four-block stack, compared to ACT's 22%, 10%, and 2%, respectively. These results indicate that errors accumulate rapidly as the task horizon increases. We observed that the primary difficulty stems from accurately locating and grasping each cube, while placement after grasping is comparatively more reliable.

Limitations. The difficulty in locating cubes may be partly attributed to the visual encoder and the restricted camera setup, as only two viewpoints are used. Future work will explore algorithmic improvements and additional camera perspectives to enhance perception and robustness.

The Stack 4 results partially support previous stated hypothesis. MEAT consistently improves over ACT at each stacking depth, but the absolute success rate remains low as the horizon increases. This indicates that MoE specialization reduces, but does not eliminate, long-horizon error accumulation in real-world stacking.

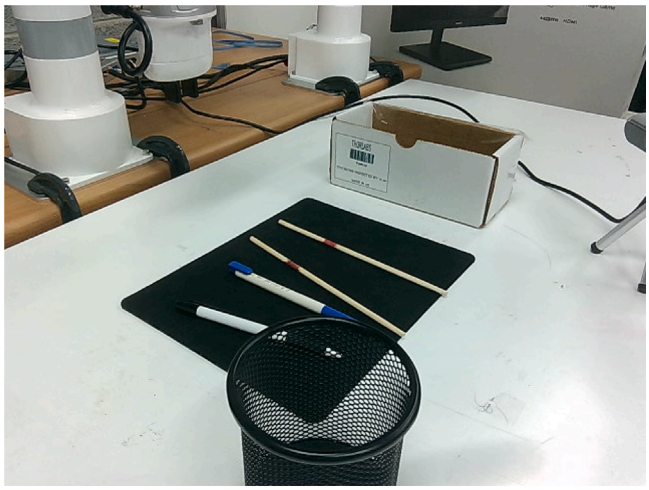


Fig. 15. Sorting Pens and Chopsticks task setup. Two pens and two chopsticks are placed in front of the robot; the goal is to place the chopsticks in the white box and the pens in the black pen holder.

5.5. Task 4: 'Sorting pen & chopsticks'

We want to evaluate whether the policy can reliably distinguish between different objects. Hence, this task tests object-conditioned behavior. Unlike the previous tasks, success requires not only reaching and grasping objects, but also distinguishing object categories and selecting the correct placement target for each category.

While this ability largely depends on the vision backbone of the policy, it is still an important factor to observe. In this task, two ink pens and two chopsticks will be placed in front of the robot (Fig. 15). The robot's objective is to pick up the two chopsticks and place them in the white box on the right, and to pick up the two pens and place them in the black pen holder on the left. As discussed in the previous section, the model's main weakness lies in accurately localizing objects. To address this, our implementation provides additional support to help the model localize the objects more precisely. Therefore, the focus of this experiment is not on localization performance but rather on testing whether the policy can correctly distinguish the objects and place them in the appropriate locations.

Result Analysis. For the Sorting task (pens and chopsticks), MEAT performs the best with 84% on pens and 82% on chopsticks, showing strong generalization and robustness in real-world object sorting. In contrast, ACT achieves only 69% on pens and 59% on chopsticks, indicating clear difficulties in reliably distinguishing and placing objects. Meanwhile, ConvMLP completely fails, with a 0% success rate in both categories, suggesting that simple architectures without temporal modeling or richer representations are insufficient for this task. Overall, the results highlight that while MEAT narrows the gap toward human-level sorting, there is still room for improvement, especially in achieving consistent accuracy comparable to human performance. It is interesting to note that when the robot about to drop the chopstick or pen in wrong side, it will replanning to put it in other side, however we would see some confusion during the planning, the problem is that MEAT show less confusion while replanning while ACT is more confusion, and some time it drop the object right at the middle without choosing any box to put it in.

The Sorting results support the object-conditioned behavior hypothesis: MEAT achieves higher success than ACT for both pens and chopsticks, suggesting that the MoE policy better preserves task-dependent action choices during real-world execution.

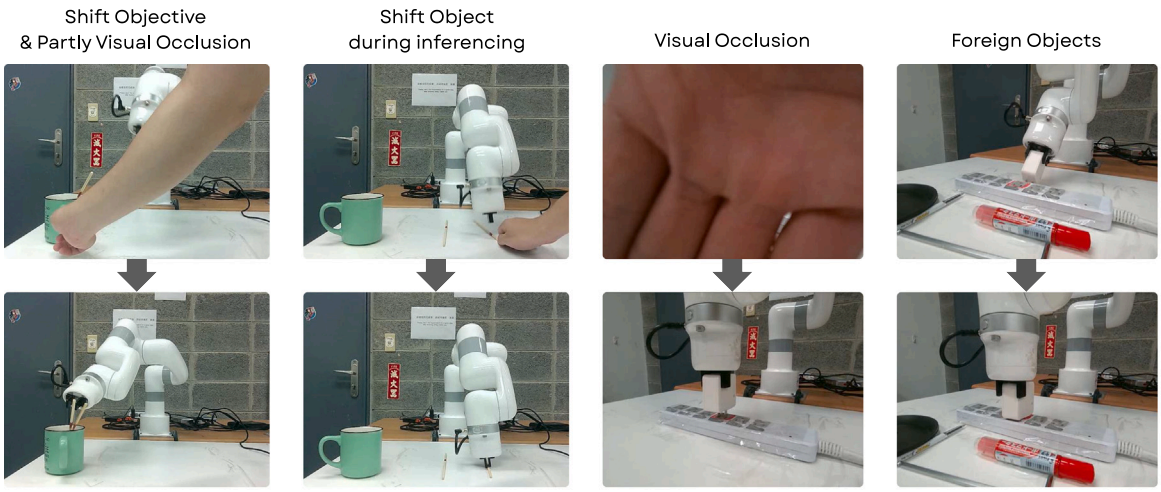


Fig. 16. Robustness Test for MEAT.

5.6. Robustness against perturbation

This experiment tests robustness under visual and physical disturbances rather than nominal task execution. The purpose is to examine whether MEAT can continue producing reasonable corrective actions when the observation stream or object configuration changes during inference.

MEAT's robustness against visual and physical perturbations was evaluated in the experiments shown in Fig. 16, where we introduced disturbances such as partially covering the camera and shifting the object, moving the objects during inference, fully covering one camera view for three seconds, and throwing foreign objects during policy execution. The MEAT policy promptly adapted by adjusting the robot arm's trajectory to compensate for these perturbations.

The perturbation experiments provide qualitative support for the robustness hypothesis. MEAT adapts its trajectory after camera occlusion, object motion, and external disturbances, suggesting that the learned policy is not limited to fixed open-loop replay of the demonstrations.

6. Ablations

All ablation studies in this section are conducted on the two scripted ACT benchmark tasks from ALOHA: *Bimanual Insertion Scripted* and *Transfer Cube Scripted*. We use the scripted versions because they provide controlled and repeatable evaluation conditions, allowing us to isolate the effect of each architectural or training choice without additional noise from human demonstrations or real-world hardware. Unless explicitly varied by the ablation, all experiments follow the same simulation setup as in Section 4, including the same observation setting, training budget, and evaluation protocol, and three random seeds. Therefore, these ablations should be interpreted as controlled diagnostic studies that explain the design choices used in the main MEAT experiments, rather than as additional independent benchmarks.

In this section, we perform three ablations to analyze auxiliary loss, backbone depth, and normalization strategy in our setup.

6.1. Impact of the auxiliary loss term

This ablation uses the same MEAT architecture and training protocol as the main scripted ALOHA experiments, but removes the auxiliary load-balancing loss from the MoE router. The purpose is to test whether the auxiliary loss is necessary for stable expert utilization and whether it contributes to the main success improvements reported in Section 4.

In this ablation study, we remove the auxiliary loss for expert load balancing to test its effect on performance. The results in Fig. 17.1 show only a small change: accuracy in the Insertion Scripted task drops by about 2%, while the Transfer Scripted task drops by around 1%. This indicates that the auxiliary loss does provide a positive effect, but the difference is relatively minor in this setup.

The limited impact can be explained by the fact that this experiment uses a small model, where the benefit of expert load balancing is less pronounced. In smaller models, the number of experts and their capacity are already constrained, so the imbalance between experts is not as severe. As a result, removing the auxiliary loss does not significantly harm accuracy, though it still offers slight improvements, particularly in tasks that demand higher precision.

6.2. Backbone sensitivity study

This ablation keeps the MEAT policy and scripted ALOHA task setup fixed, while only replacing the visual backbone. The purpose is to test whether the main performance gain comes from the MoE policy architecture or can instead be explained by using a deeper visual encoder. Because deeper backbones change memory usage and feasible batch size, we report both accuracy and maximum batch size under the same hardware constraints.

We conduct an ablation study by replacing the original ResNet18 backbone with deeper ResNet variants (ResNet34, ResNet50, ResNet101, ResNet152) from the original ResNet paper [50] to assess their impact on the Insertion Scripted and Transfer Scripted tasks. ResNet, with its residual connections, enables training of deep convolutional networks by mitigating vanishing gradients. While deeper models can capture more complex features, they also increase memory usage and computational cost, which may limit batch size and, in some cases, lead to diminishing returns in performance.

Fig. 17.2 investigates the impact of different ResNet backbones on both batch size efficiency and task accuracy. The original implementation uses ResNet18, a relatively shallow architecture that is computationally lighter and more memory-efficient. As shown in Fig. 17.2a, where we capture the maximum bathsize of each resnet on our setup, ResNet18 supports the largest batch size, while deeper networks such as ResNet101 and ResNet152 significantly reduce the feasible batch size due to higher parameter counts and memory requirements. Since all models were trained for the same number of epochs, a smaller batch size effectively means fewer overall training steps, which may limit convergence for deeper backbones.

From the accuracy perspective (Fig. 17.2b), deeper ResNet variants do not consistently improve performance across the Insertion Scripted and Transfer Scripted tasks. In fact, ResNet18 performs competitively, achieving the highest or near-highest accuracy. This suggests that

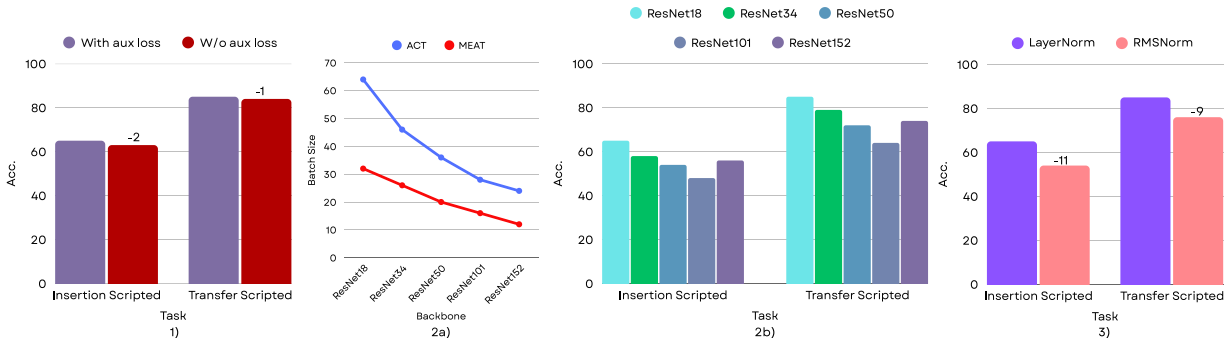


Fig. 17. Ablation study results. (1) Effect of auxiliary loss on accuracy for 2 tasks. (2) Impact of different ResNet backbones on batch size and accuracy. (3) Comparison of LayerNorm and RMSNorm on both tasks.

deeper models may require longer training schedules to fully utilize their capacity, while under the current setting, their reduced effective batch size and increased complexity can hinder optimization.

Overall, these results show that deeper is not always better for visual backbones in robotic manipulation. ResNet18 offers a favorable trade-off between efficiency and performance, highlighting the importance of aligning model capacity with task complexity and available training budget.

6.3. Replacing LayerNorm with RMSNorm

This ablation keeps the same two scripted ALOHA tasks and replaces only the normalization layer inside the Transformer blocks. The purpose is to test whether a cheaper normalization strategy can preserve the stability of MEAT in the low-data, small-model regime used in our benchmark experiments.

We compared RMSNorm [51] and LayerNorm [52] on two small-scale experiments from ALOHA dataset (*Insertion Scripted* and *Transfer Cube Scripted*), each averaged over 3 seeds.

Formally, LayerNorm standardizes activations by both mean and variance,

$$\hat{a}_i = \frac{a_i - \mu(a)}{\sigma(a)}, \qquad y_i = \gamma_i \hat{a}_i + \beta_i, \tag{10}$$

whereas RMSNorm discards the mean and rescales only by the root mean square,

$$\tilde{a}_i = \frac{a_i}{\sqrt{\frac{1}{d}\sum_{i=1}^{d} a_i^2 + \epsilon}}, \qquad y_i = \gamma_i \tilde{a}_i + \beta_i, \tag{11}$$

As shown in the Fig. 17.3, replacing LayerNorm with RMSNorm consistently reduces final performance: accuracy dropped by 11% on *Insertion Scripted* and 9% on *Transfer Scripted*. While RMSNorm is computationally cheaper by removing mean-centering and normalizes only by the root mean square, our results suggest that the re-centering provided by LayerNorm plays a stabilizing role in low-data or small-model regimes. Prior works [52,53] show that RMSNorm yields 5% $\sim$ 64% speedups with generally comparable quality, and in large-scale settings, this trade-off can be negligible. However, our findings highlight that in smaller setups, mean-centering contributes noticeably to optimization stability and generalization, making LayerNorm the more robust choice despite its higher cost.

7. Conclusion & limitations

In this paper, we propose MEAT, where the MoE architecture is integrated into a Transformer-based policy to control a robotic arm. Through 6 simulation experiments and 4 real experiments across various tasks, our model consistently outperformed the baseline Transformer by margins ranging from 5% to 30%, demonstrating improved learning efficiency and control precision. Furthermore, we tested the proposed approach with auxiliary-loss-free MoE routing mechanism as an additional experiment, but it did not yield prominent results, highlighting the need for further investigation into more effective balancing strategies and loss function design. While our approach achieves state-of-the-art performance, we observed that training stability can be sensitive to the choice of random seed, with some seeds producing significantly better outcomes than others, a phenomenon also reported in prior work. Finally, 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.

CRediT authorship contribution statement

Hung Phan Quoc Mai: Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Naeem Ul Islam:** Writing – review & editing, Validation, Resources, Project administration, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by the National Science and Technology Council (NSTC), Taiwan (Project No. 115-2221-E-155-048-), and the UK–Taiwan Innovative Industries Programme (I^2P Programme).

Appendix A. Formal definitions of expert-routing metrics

We compute expert-routing statistics from inference rollouts. Let N be the number of experts and K be the number of selected experts per token. In our experiments, $N = 4$ and $K = 2$. For a fixed MoE layer ℓ, let $\mathcal{B}_\ell$ denote the set of routed token instances collected across all evaluated rollouts and timesteps. For each token $b \in \mathcal{B}_\ell$, the router outputs probabilities $p_{\ell,b,n}$ over expert $n \in \{1, \ldots, N\}$ and selects a top-K expert set $S_{\ell,b}$.

Top-K expert usage. The top-K usage of expert n in layer ℓ is defined as

$$u_{\ell,n} = \frac{1}{K|\mathcal{B}_\ell|} \sum_{b \in \mathcal{B}_\ell} \mathbf{1}\left[n \in S_{\ell,b}\right]. \tag{12}$$

Thus, $\sum_{n=1}^{N} u_{\ell,n} = 1$. A uniform routing distribution has $u_{\ell,n} = 1/N$ for all experts.

Normalized entropy. We measure load balance using normalized entropy:

$$H_\ell^{\text{norm}} = -\frac{1}{\log N}\sum_{n=1}^{N} u_{\ell,n}\log(u_{\ell,n}+\epsilon), \tag{13}$$

where ϵ is a small numerical constant. Higher values indicate more balanced expert usage, with $H_\ell^{\text{norm}} = 1$ under perfectly uniform routing.

Effective number of experts. The effective number of used experts is computed as the exponential of the unnormalized entropy:

$$E_\ell^{\text{eff}} = \exp\left(-\sum_{n=1}^{N} u_{\ell,n}\log(u_{\ell,n}+\epsilon)\right). \tag{14}$$

This value ranges from 1, when all assignments go to one expert, to N, when usage is perfectly uniform.

KL divergence to uniform. We also report the KL divergence from the empirical routing distribution to the uniform distribution:

$$D_{\text{KL}}(u_\ell \parallel \text{Unif}) = \sum_{n=1}^{N} u_{\ell,n}\log\frac{u_{\ell,n}+\epsilon}{1/N}. \tag{15}$$

Lower values indicate better load balance.

Coefficient of variation. The coefficient of variation measures relative dispersion in expert usage:

$$\text{CV}_\ell = \frac{\sqrt{\frac{1}{N}\sum_{n=1}^{N}(u_{\ell,n}-\bar{u}_\ell)^2}}{\bar{u}_\ell}, \qquad \bar{u}_\ell = \frac{1}{N}\sum_{n=1}^{N} u_{\ell,n}. \tag{16}$$

Since $\sum_n u_{\ell,n} = 1$, $\bar{u}_\ell = 1/N$. Lower CV indicates more balanced usage.

Gini coefficient. Let $u_{\ell,(1)} \le \cdots \le u_{\ell,(N)}$ be the sorted expert usage values. The Gini coefficient is

$$G_\ell = \frac{2\sum_{i=1}^{N} i\, u_{\ell,(i)}}{N\sum_{i=1}^{N} u_{\ell,(i)}} - \frac{N+1}{N}. \tag{17}$$

Lower Gini indicates more equal expert usage.

Nearest-neighbor expert agreement. To test whether similar policy states receive similar expert assignments, we define a per-step routing record r. Each record contains a normalized policy-state feature vector x_r, constructed from robot state, predicted action, and rollout progress. Let $v_r \in \mathbb{R}^N$ be the top-1 expert-token fraction for that record, and let

$$d_r = \arg\max_n v_{r,n} \tag{18}$$

be its dominant expert. For each record r, let $\mathcal{N}_k(r)$ denote its k nearest neighbors in the normalized feature space. The nearest-neighbor agreement is

$$A_{\text{NN}} = \frac{1}{|\mathcal{R}|k}\sum_{r\in\mathcal{R}}\sum_{r'\in\mathcal{N}_k(r)} \mathbf{1}\left[d_r = d_{r'}\right], \tag{19}$$

where $\mathcal{R}$ is the set of sampled routing records.

Random marginal baseline and lift. The random baseline is the expected agreement if dominant experts were sampled independently from the marginal expert distribution. Let

$$\pi_n = \frac{1}{|\mathcal{R}|}\sum_{r\in\mathcal{R}} \mathbf{1}\left[d_r = n\right]. \tag{20}$$

Then the random marginal agreement is

$$A_{\text{rand}} = \sum_{n=1}^{N} \pi_n^2. \tag{21}$$

The agreement lift over random is

$$\Delta A = A_{\text{NN}} - A_{\text{rand}}. \tag{22}$$

A positive lift indicates that similar policy states are routed to the same dominant expert more often than expected from marginal expert frequencies alone.

Nearest-neighbor usage cosine. In addition to hard dominant-expert agreement, we compare the full top-1 usage vectors using cosine similarity:

$$C_{\text{NN}} = \frac{1}{|\mathcal{R}|k}\sum_{r\in\mathcal{R}}\sum_{r'\in\mathcal{N}_k(r)} \frac{v_r^{\top} v_{r'}}{\|v_r\|_2\|v_{r'}\|_2+\epsilon}. \tag{23}$$

Higher values indicate that nearby policy states have similar full routing distributions, not only the same dominant expert.

Appendix B. Layer-level expert routing statistics

We additionally report representative layer-level routing statistics for the two scripted simulation tasks. These results show that MEAT is balanced at the model level, but not uniformly mixed at every layer. Encoder layers tend to preserve broader expert diversity, while decoder layers often become more selective, which is consistent with phase-dependent action generation (see Table 6).

For Transfer Cube, the aggregate expert usage is moderately balanced, with 3.080 effective experts on average. Encoder layers are close to uniform routing: encoder 0 and encoder 1 reach 3.927 and 3.960 effective experts, respectively. Decoder layers are more selective. For example, decoder 0 mainly routes to E0/E1, while decoder 6 emphasizes E0/E2. This supports the view that early visual-state encoding uses broader expert diversity, whereas action-generation layers route more selectively according to the control regime (see Table 7).

For Bimanual Insertion, the aggregate routing is also balanced, with 3.373 effective experts and normalized entropy of 0.870. Encoder layers again show broad expert usage, while decoder layers show stronger specialization. Decoder 0 relies mainly on E0/E1, whereas decoder 6 uses E0/E1/E2 and suppresses E3. The nearest-neighbor agreement remains high across representative layers, indicating that similar policy states are routed consistently. We exclude saturated layers with both 100% nearest-neighbor agreement and 100% random baseline from the representative table, since those cases indicate deterministic dominant routing rather than meaningful similarity-sensitive consistency.

Appendix C. Teleoperation method

Algorithm 1 RecordData(arm_1, arm_2, cams)

```
t_0 ← current time
while time elapsed < duration do
    a ← servo_angle(arm_1) + eef
    q ← joint_pos(arm_2), θ ← servo_angle(arm_2)
    I ← 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.

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

Table 6
Representative layer-level routing statistics on Transfer Cube.

Layer	E0	E1	E2	E3	Entropy ↑	Eff. exp. ↑	NN agree ↑	Lift ↑
Mean	25.8	23.2	29.0	22.1	0.800	3.080	97.3	+22.2
Enc. 0	24.3	33.2	20.5	22.0	0.987	3.927	93.9	+49.2
Enc. 1	22.9	28.0	28.8	20.3	0.993	3.960	94.3	+65.6
Dec. 0	35.4	49.6	12.0	3.1	0.776	2.933	96.2	+44.8
Dec. 6	35.1	8.8	40.1	16.1	0.895	3.458	93.6	+29.2

Expert usage is measured by normalized top-K assignments. Entropy and effective experts measure load balance. NN agreement measures whether similar policy states are assigned to the same dominant expert, and Lift subtracts the random agreement expected from marginal expert usage.

Table 7
Representative layer-level routing statistics on Bimanual Insertion.

Layer	E0	E1	E2	E3	Entropy ↑	Eff. exp. ↑	NN agree ↑	Lift ↑
Mean	25.1	28.1	25.5	21.4	0.870	3.373	97.5	+35.2
Enc. 1	24.9	24.1	32.3	18.7	0.986	3.925	95.8	+34.6
Enc. 3	32.2	29.9	24.4	13.4	0.966	3.818	98.2	+47.0
Dec. 0	37.5	50.0	11.7	0.8	0.724	2.729	89.8	+44.0
Dec. 6	29.5	38.1	23.4	9.0	0.926	3.610	98.5	+47.8

Expert usage is measured by normalized top-K assignments. Entropy and effective experts measure load balance. NN agreement measures whether similar policy states are assigned to the same dominant expert, and Lift subtracts the random agreement expected from marginal expert usage.

Algorithm 2 EEFControl(arm)

```
while not stop_flag do
    if key '1' then
        open_eef(arm), eef_state ← 1
    end if
    if key '0' then
        close_eef(arm), eef_state ← 0
    end if
end while
```

Algorithm 3 SyncArms(master, slave)

```
while not stop_flag do
    θ ← servo_angle(master) + eef
    set_angle(slave, θ[: 6])
    sleep(0.01)
end while
```

commands. The loop executes at 100Hz (10 ms intervals) to maintain smooth and precise motion replication.

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 ← false, eef_state ← 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
```

Table 8
Robot arm specification.

Specification	LITE 6	XARM 6
Payload	600 g	5 kg
Reach	440 mm	700 mm
Degrees of Freedom	6	6
Repeatability	±0.5 mm	±0.1 mm
Maximum Speed	500 mm/s	1 m/s
Weight (arm only)	7.2 kg	12.2 kg
Gripper stroke range	0–16 mm	–
Reverse stroke range	20–38 mm	–

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.

Appendix D. Hardware setup specific

See Table 8.

Appendix E. Training loss

See Fig. 18.

Appendix F. Hyperparameters config

We carefully tune the baselines and include the hyperparameters used in Tables 9, 10, 11.

Data availability

Data will be made available on request.

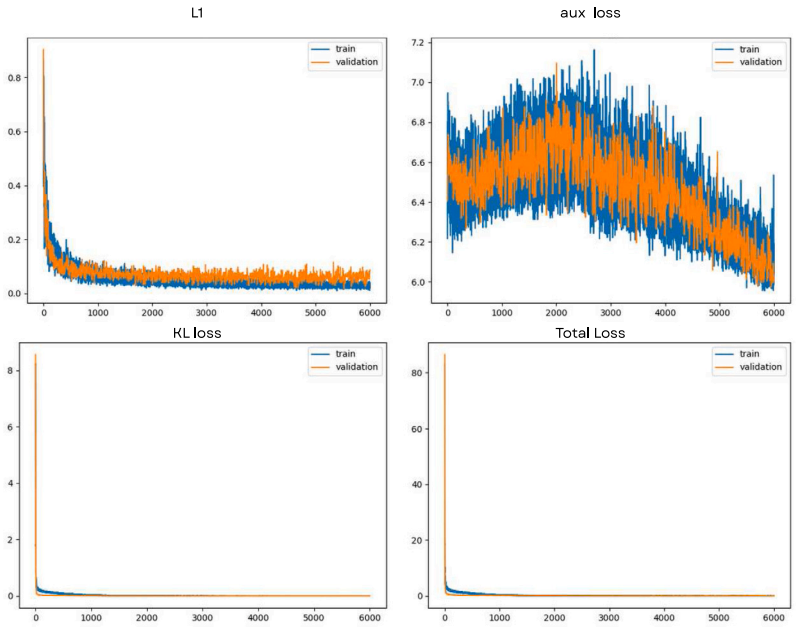


Fig. 18. A sample of training loss.

Table 9
Task specific configs.

Task	Common				Transformer models							
					Common		ACT			MEAT		
	Is_real	Views	Chunk size	EpLen	#Episode	ImageRes	BS	LR	Epochs	BS	LR	Epochs
Transfer Cube	F	Top View	100	400	50	480 × 640	64	5e−5	2500	32	3e−5	2500
Cube Insertion	F	Top View	100	500	50	480 × 640	64	5e−5	2500	32	3e−5	2500
Tool hang	F	Agent View	20	700	150	480 × 640	64	5e−5	4000	32	3e−5	4000
Tool transport	F	Agent View	100	600	150	480 × 640	64	5e−5	4000	32	3e−5	4000
Chopstick	T	Side View, Front View	20	400	49	480 × 640	48	4e−5	15 000	24	2e−5	15 000
Stack 4	T	Side View, Front View	20	700	66	480 × 640	36	3e−5	30 000	22	2e−5	30000

Table 10
Task specific configs (cont.).

Task	BC-RNN					Diffusion				
	#Episode	ImageRes	BS	LR	epochs	#Episode	ImageRes	BS	LR	epochs
Tool hang	150	480 × 640	256	1e−4	1000	150	480 × 640	128	1e−4	2000
Tool transport	150	480 × 640	256	1e−4	1000	200	240 × 320	172	1e−4	2000

Table 11
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
Auxiliary weight	1e−3		

References

[1] Dean A. Pomerleau, Alvinn: An autonomous land vehicle in a neural network, Adv. Neural Inf. Process. Syst. 1 (1988).

[2] Jyothish Pari, Nur Muhammad (Mahi) Shafiullah, Sridhar Pandian Arunachalam, Lerrel Pinto, The Surprising Effectiveness of Representation Learning for Visual Imitation, in: Proceedings of Robotics: Science and Systems, New York City, NY, USA, 2022, http://dx.doi.org/10.15607/RSS.2022.XVIII.010.

[3] Ashvin Nair, Dian Chen, Pulkit Agrawal, Phillip Isola, Pieter Abbeel, Jitendra Malik, Sergey Levine, Combining self-supervised learning and imitation for vision-based rope manipulation, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2017, pp. 2146–2153.

[4] Xin Zhang, Yanhua Li, Xun Zhou, Ziming Zhang, Jun Luo, TrajGAIL: Trajectory generative adversarial imitation learning for long-term decision analysis, in: 2020 IEEE International Conference on Data Mining, ICDM, 2020, pp. 801–810, http://dx.doi.org/10.1109/ICDM50108.2020.00089.

[5] Tony Z. Zhao, Vikash Kumar, Sergey Levine, Chelsea Finn, Learning Fine-Grained Bimanual Manipulation with Low-Cost Hardware, in: Proceedings of Robotics: Science and Systems, Daegu, Republic of Korea, 2023, http://dx.doi.org/10.15607/RSS.2023.XIX.016.

[6] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, Shuran Song, Diffusion policy: Visuomotor policy learning via action diffusion, Int. J. Robot. Res. 02783649241273668, http://dx.doi.org/10.1177/02783649241273668.

[7] Moritz Reuss, Ömer Erdinç Yağmurlu, Fabian Wenzel, Rudolf Lioutikov, Multimodal Diffusion Transformer: Learning Versatile Behavior from Multimodal Goals, in: Proceedings of Robotics: Science and Systems, Delft, Netherlands, 2024, http://dx.doi.org/10.15607/RSS.2024.XX.121.

[8] Sukhan Lee, Naeem Ul Islam, Soojin Lee, Robust image completion and masking with application to robotic bin picking, Robot. Auton. Syst. 131 (2020) 103563.

[9] Vignesh Prasad, Alap Kshirsagar, Dorothea Koert, Ruth Stock-Homburg, Jan Peters, Georgia Chalvatzaki, MoVEInt: Mixture of variational experts for learning human–robot interactions from demonstrations, IEEE Robot. Autom. Lett. 9 (7) (2024) 6043–6050, http://dx.doi.org/10.1109/LRA.2024.3396074.

[10] Dibya Ghosh, Homer Rich Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Tobias Kreiman, Charles Xu, Jianlan Luo, You Liang Tan, Lawrence Yunliang Chen, Quan Vuong, Ted Xiao, Pannag R Sanketi, Dorsa Sadigh, Chelsea Finn, Sergey Levine, Octo: An open-source generalist robot policy, in: Proceedings of Robotics: Science and Systems, Delft, Netherlands, 2024, http://dx.doi.org/10.15607/RSS.2024.XX.090.

[11] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, Szymon Jakubczak, Tim Jones, Liyiming Ke, Sergey Levine, Adrian Li-Bell, Mohith Mothukuri, Suraj Nair, Karl Pertsch, Lucy Xiaoyang Shi, James Tanner, Quan Vuong, Anna Walling, Haohuan Wang, Ury Zhilinsky, π_0: A vision-language-action flow model for general robot control, 2024, CoRR abs/2410.24164.

[12] Naeem Ul Islam, Reducing value estimation uncertainty in continuous control via gated quantile critics, Expert Syst. Appl. (2026) 132517.

[13] Naeem Ul Islam, Chung Tien Dat, Muhammad Khalid, Kolmogorov-arnold inspired convolutional networks for enhancing PPO-based online reinforcement learning, J. Exp. Theor. Artif. Intell. 38 (2) (2026) 155–171.

[14] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, William El Sayed, Mixtral of experts, 2024, arXiv:2401.04088 [cs.LG].

[15] DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J.L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R.J. Chen, R.L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S.S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W.L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X.Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y.K. Li, Y.Q. Wang, Y.X. Wei, Y.X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z.F. Wu, Z.Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, Zizheng Pan, DeepSeek-V3 technical report, 2025, arXiv:2412.19437 [cs.CL].

[16] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, Roberto Martín-Martín, What matters in learning from offline human demonstrations for robot manipulation, in: 5th Annual Conference on Robot Learning, 2021.

[17] Andrew Y. Ng, Stuart J. Russell, Algorithms for inverse reinforcement learning, in: Proceedings of the Seventeenth International Conference on Machine Learning, ICML '00, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ISBN: 1558607072, 2000, pp. 663–670.

[18] Pieter Abbeel, Andrew Y. Ng, Apprenticeship learning via inverse reinforcement learning, in: Proceedings of the Twenty-First International Conference on Machine Learning, ICML '04, Association for Computing Machinery, New York, NY, USA, ISBN: 1581138385, 2004, p. 1, http://dx.doi.org/10.1145/1015330.1015430.

[19] Brian D. Ziebart, Andrew L. Maas, J. Andrew Bagnell, Anind K. Dey, et al., Maximum entropy inverse reinforcement learning, in: Aaai, vol. 8, Chicago, IL, USA, 2008, pp. 1433–1438.

[20] Jonathan Ho, Stefano Ermon, Generative adversarial imitation learning, in: Proceedings of the 30th International Conference on Neural Information Processing Systems, NIPS '16, Curran Associates Inc., Red Hook, NY, USA, ISBN: 9781510838819, 2016, pp. 4572–4580.

[21] Sangwoong Yoon, Himchan Hwang, Dohyun Kwon, Yung-Kyun Noh, Frank C. Park, Maximum entropy inverse reinforcement learning of diffusion models with energy-based models, in: The Thirty-Eighth Annual Conference on Neural Information Processing Systems, 2024.

[22] Jonathan Ho, Ajay Jain, Pieter Abbeel, Denoising diffusion probabilistic models, in: Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS '20, Curran Associates Inc., Red Hook, NY, USA, ISBN: 9781713829546, 2020.

[23] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, Illia Polosukhin, Attention is all you need, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS '17, Curran Associates Inc., Red Hook, NY, USA, ISBN: 9781510860964, 2017, pp. 6000–6010.

[24] Sudeep Dasari, Oier Mees, Sebastian Zhao, Mohan Kumar Srirama, Sergey Levine, The ingredients for robotic diffusion transformers, in: International Conference on Robotics and Automation, ICRA, 2024.

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

[26] Noam Shazeer, *Azalia Mirhoseini, *Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, Jeff Dean, Outrageously large neural networks: The sparsely-gated mixture-of-experts layer, in: International Conference on Learning Representations, 2017.

[27] William Fedus, Barret Zoph, Noam Shazeer, Switch transformers: scaling to trillion parameter models with simple and efficient sparsity, J. Mach. Learn. Res. (ISSN: 1532-4435) 23 (1) (2022).

[28] Nan Du, Yanping Huang, Andrew M. Dai, Simon Tong, Dmitry Lepikhin, Yuanzhong Xu, Niki Shazeer, Siva Kudugunta, Zhifeng Li, Xiaoqi Chen, et al., GLaM: Efficient scaling of language models with mixture-of-experts, in: International Conference on Machine Learning, PMLR, 2022, pp. 5356–5368.

[29] Eric Wallace, Olivia Watkins, Miles Wang, Kai Chen, Chris Koch, Estimating worst-case frontier risks of open-weight LLMs, 2025, arXiv:2508.03153, URL https://arxiv.org/abs/2508.03153.

[30] Yanqi Zhou, Tao Lei, Hanxiao Liu, Nan Du, Yanping Huang, Vincent Y. Zhao, Andrew Dai, Zhifeng Chen, Quoc Le, James Laudon, Mixture-of-experts with expert choice routing, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., Red Hook, NY, USA, ISBN: 9781713871088, 2022.

[31] Hussein Hazimeh, Zhe Zhao, Aakanksha Chowdhery, Maheswaran Sathiamoorthy, Yihua Chen, Rahul Mazumder, Lichan Hong, Ed Chi, Dselect-k: Differentiable selection in the mixture of experts with applications to multi-task learning, in: M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, J. Wortman Vaughan (Eds.), in: Advances in Neural Information Processing Systems, vol. 34, Curran Associates, Inc., 2021, pp. 29335–29347.

[32] Stephen Roller, Sainbayar Sukhbaatar, Arthur Szlam, Jason Weston, Hash layers for large sparse models, in: Proceedings of the 35th International Conference on Neural Information Processing Systems, NIPS '21, Curran Associates Inc., Red Hook, NY, USA, ISBN: 9781713845393, 2021.

[33] Mike Lewis, Shruti Bhosale, Tim Dettmers, Naman Goyal, Luke Zettlemoyer, BASE layers: Simplifying training of large, sparse models, in: Marina Meila, Tong Zhang (Eds.), Proceedings of the 38th International Conference on Machine Learning, in: Proceedings of Machine Learning Research, vol. 139, PMLR, 2021, pp. 6265–6274.

[34] Lean Wang, Huazuo Gao, Chenggang Zhao, Xu Sun, Damai Dai, Auxiliary-loss-free load balancing strategy for mixture-of-experts, 2024, arXiv:2408.15664 [cs.LG].

[35] Moritz Reuss, Jyothish Pari, Pulkit Agrawal, Rudolf Lioutikov, Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning, in: The Thirteenth International Conference on Learning Representations, 2025.

[36] Abby O'Neill, Abdul Rehman, Abhiram Maddukuri, Abhishek Gupta, Abhishek Padalkar, Abraham Lee, Acorn Pooley, Agrim Gupta, Ajay Mandlekar, Ajinkya Jain, Albert Tung, Alex Bewley, Alexander Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anchit Gupta, Andrew Wang, Anikait Singh, Animesh Garg, Aniruddha Kembhavi, Annie Xie, Anthony Brohan, Antonin Raffin, Archit Sharma, Arefeh Yavary, Arhan Jain, Ashwin Balakrishna, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Blake Wulfe, Brian Ichter, Cewu Lu, Charles Xu, Charlotte Le, Chelsea Finn, Chen Wang, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Christopher Agia, Chuer Pan, Chuyuan Fu, Coline Devin, Danfei Xu, Daniel Morton, Danny Driess, Daphne Chen, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dinesh Jayaraman, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Ethan Paul Foster, Fangchen Liu, Federico Ceola, Fei Xia, Feiyu Zhao, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Gilbert Feng, Giulio Schiavi, Glen Berseth, Gregory Kahn, Guanzhi Wang, Hao Su, Haoshu Fang, Haochen Shi, Henghui Bao, Heni Ben Amor, Henrik I. Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Huy Ha, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jad Abou-Chakra, Jaehyung Kim, Jaimyn Drake, Jan Peters, Jan Schneider, Jasmine

Hsu, Jeannette Bohg, Jeffrey Bingham, Jeffrey Wu, Jensen Gao, Jiaheng Hu, Jiajun Wu, Jialin Wu, Jiankai Sun, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jimmy Wu, Jingpei Lu, Jingyun Yang, Jitendra Malik, João Silvério, Joey Hejna, Jonathan Booher, Jonathan Tompson, Jonathan Yang, Jordi Salvador, Joseph J. Lim, Junhyek Han, Kaiyuan Wang, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Black, Kevin Lin, Kevin Zhang, Kiana Ehsani, Kiran Lekkala, Kirsty Ellis, Krishan Rana, Krishnan Srinivasan, Kuan Fang, Kunal Pratap Singh, Kuo-Hao Zeng, Kyle Hatch, Kyle Hsu, Laurent Itti, Lawrence Yunliang Chen, Lerrel Pinto, Li Fei-Fei, Liam Tan, Linxi Jim Fan, Lionel Ott, Lisa Lee, Luca Weihs, Magnum Chen, Marion Lepert, Marius Memmel, Masayoshi Tomizuka, Masha Itkina, Mateo Guaman Castro, Max Spero, Maximilian Du, Michael Ahn, Michael C. Yip, Mingtong Zhang, Mingyu Ding, Minho Heo, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J. Joshi, Niko Sünderhauf, Ning Liu, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Osbert Bastani, Pannag R. Sanketi, Patrick Tree Miller, Patrick Yin, Paul Wohlhart, Peng Xu, Peter David Fagan, Peter Mitrano, Pierre Sermanet, Pieter Abbeel, Priya Sundaresan, Qiuyu Chen, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Rohan Baijal, Rosario Scalise, Rose Hendrix, Roy Lin, Runjia Qian, Ruohan Zhang, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Shan Lin, Sherry Moore, Shikhar Bahl, Shivin Dass, Shubham D. Sonawani, Shuran Song, Sichun Xu, Siddhant Haldar, Siddharth Karamcheti, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Subramanian Ramamoorthy, Sudeep Dasari, Suneel Belkhale, Sungjae Park, Suraj Nair, Suvir Mirchandani, Takayuki Osa, Tanmay Gupta, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Thomas Kollar, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Trinity Chung, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xinyang Geng, Xiyuan Liu, Liangwei Xu, Xuanlin Li, Yao Lu, Yecheng Jason Ma, Yejin Kim, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Yilin Wu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yue Cao, Yueh-Hua Wu, Yujin Tang, Yuke Zhu, Yunchu Zhang, Yunfan Jiang, Yunshuang Li, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zehan Ma, Zhuo Xu, Zichen Jeff Cui, Zichen Zhang, Zipeng Lin, Open X-Embodiment: Robotic learning datasets and RT-X models : Open X-Embodiment collaboration, in: ICRA, 2024, pp. 6892–6903.

[37] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Xinyu Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Khuong Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, Chelsea Finn, DROID: A large-scale in-the-wild robot manipulation dataset, in: RSS 2024 Workshop: Data Generation for Robotics, 2024.

[38] Zipeng Fu, Tony Z. Zhao, Chelsea Finn, Mobile ALOHA: Learning bimanual mobile manipulation using low-cost whole-body teleoperation, in: 8th Annual Conference on Robot Learning, 2024.

[39] Ajay Mandlekar, Soroush Nasiriany, Bowen Wen, Iretiayo Akinola, Yashraj Narang, Linxi Fan, Yuke Zhu, Dieter Fox, MimicGen: A data generation system for scalable robot learning using human demonstrations, in: Towards Generalist Robots: Learning Paradigms for Scalable Skill Acquisition @ CoRL2023, 2023.

[40] Lucy Lai, Ann Z.X. Huang, Samuel J. Gershman, Action chunking as conditional policy compression, Cognition (ISSN: 0010-0277) 264 (2025) 106201, http://dx.doi.org/10.1016/j.cognition.2025.106201.

[41] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, Sergey Zagoruyko, End-to-end object detection with transformers, in: Andrea Vedaldi, Horst Bischof, Thomas Brox, Jan-Michael Frahm (Eds.), Computer Vision – ECCV 2020, Springer International Publishing, Cham, ISBN: 978-3-030-58452-8, 2020, pp. 213–229.

[42] Nur Muhammad (Mahi) Shafiullah, Ziehen Jeff Cui, Ariuntuya Altanzaya, Lerrel Pinto, Behavior transformers: cloning k modes with one stone, in: Proceedings of the 36th International Conference on Neural Information Processing Systems, NIPS '22, Curran Associates Inc., Red Hook, NY, USA, ISBN: 9781713871088, 2022.

[43] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael S Ryoo, Grecia Salazar, Pannag R Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan H Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, Brianna Zitkovich, RT-1: Robotics Transformer for Real-World Control at Scale, in: Proceedings of Robotics: Science and Systems, Daegu, Republic of Korea, 2023, http://dx.doi.org/10.15607/RSS.2023.XIX.025.

[44] Jyothish Pari, Nur Muhammad (Mahi) Shafiullah, Sridhar Pandian Arunachalam, Lerrel Pinto, The Surprising Effectiveness of Representation Learning for Visual Imitation, in: Proceedings of Robotics: Science and Systems, New York City, NY, USA, 2022, http://dx.doi.org/10.15607/RSS.2022.XVIII.010.

[45] Pete Florence, Corey Lynch, Andy Zeng, Oscar A. Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, Jonathan Tompson, Implicit behavioral cloning, in: Aleksandra Faust, David Hsu, Gerhard Neumann (Eds.), Proceedings of the 5th Conference on Robot Learning, in: Proceedings of Machine Learning Research, vol. 164, PMLR, 2022, pp. 158–168.

[46] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, Jeannette Bohg, Consistency policy: Accelerated visuomotor policies via consistency distillation, in: Robotics: Science and Systems, 2024.

[47] Senmao Li, taihang Hu, Joost van de Weijer, Fahad Khan, Tao Liu, Linxuan Li, Shiqi Yang, Yaxing Wang, Ming-Ming Cheng, jian Yang, Faster diffusion: Rethinking the role of the encoder for diffusion model inference, in: The Thirty-Eighth Annual Conference on Neural Information Processing Systems, 2024.

[48] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gómez Colmenarejo, Alexander Novikov, Gabriel Barth-maron, Mai Giménez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, Nando de Freitas, A generalist agent, Trans. Mach. Learn. Res. (ISSN: 2835-8856) (2022) Featured Certification, Outstanding Certification, URL https://openreview.net/forum?id=1ikK0kHjvj.

[49] Fadri Furrer, Martin Wermelinger, Hironori Yoshida, Fabio Gramazio, Matthias Kohler, Roland Siegwart, Marco Hutter, Autonomous robotic stone stacking with online next best object target pose planning, in: 2017 IEEE International Conference on Robotics and Automation, ICRA, 2017, pp. 2350–2356, http://dx.doi.org/10.1109/ICRA.2017.7989272.

[50] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[51] Jimmy Lei Ba, Jamie Ryan Kiros, Geoffrey E. Hinton, Layer normalization, in: Advances in NIPS Deep Learning Symposium, 2016.

[52] Biao Zhang, Rico Sennrich, Root mean square layer normalization, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, Curran Associates Inc., Red Hook, NY, USA, 2019.

[53] Sharan Narang, Hyung Won Chung, Yi Tay, Liam Fedus, Thibault Fevry, Michael Matena, Karishma Malkan, Noah Fiedel, Noam Shazeer, Zhenzhong Lan, Yanqi Zhou, Wei Li, Nan Ding, Jake Marcus, Adam Roberts, Colin Raffel, Do transformer modifications transfer across implementations and applications? in: Marie-Francine Moens, Xuanjing Huang, Lucia Specia, Scott Wen-tau Yih (Eds.), Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, 2021, pp. 5758–5773.