Decoupled Action Head: Confining Task Knowledge to Conditioning Layers

Jian Zhou[1], Sihao Lin[1], Shuai Fu[1], and Qi Wu[1]

Abstract— **Behavior Cloning (BC) is a data-driven supervised learning approach that has gained increasing attention with the success of scaling laws in language and vision domains. Among its implementations in robotic manipulation, Diffusion Policy (DP), with its two variants DP-CNN (DP-C) and DP-Transformer (DP-T), is one of the most effective and widely adopted models, demonstrating the advantages of predicting continuous action sequences. However, both DP and other BC methods remain constrained by the scarcity of paired training data, and the internal mechanisms underlying DP's effectiveness remain insufficiently understood, leading to limited generalization and a lack of principled design in model development. In this work, we propose a decoupled training recipe that leverages nearly cost-free kinematics-generated trajectories as observation-free data to pretrain a general action head (action generator). The pretrained action head is then frozen and adapted to novel tasks through feature modulation. Our experiments demonstrate the feasibility of this approach in both in-distribution and out-of-distribution scenarios. As an additional benefit, decoupling improves training efficiency; for instance, DP-C achieves up to a 41% speedup. Furthermore, the confinement of task-specific knowledge to the conditioning components under decoupling, combined with the near-identical performance of DP-C in both normal and decoupled training, indicates that the action generation backbone plays a limited role in robotic manipulation. Motivated by this observation, we introduce DP-MLP, which replaces the 244M-parameter U-Net backbone of DP-C with only 4M parameters of simple MLP blocks, achieving a 83.9% faster training speed under normal training and 89.1% under decoupling.**

I. INTRODUCTION

In the conventional formulation of Behavior Cloning (BC), robot manipulation tasks are modeled as a supervised learning problem, where a policy seeks to learn a direct mapping from observations (states S) to actions (A) [1]. Recent research in robotic manipulation—including action policy learning [2], [3], [4], observation encoder exploration [5], [6], [7], [8] and vision-language action (VLA) models [9], [10], [11], [12]—generally follows this paradigm.

However, this direct mapping approach is inherently constrained by the high cost of manually collecting high-quality demonstration data. Unlike domains such as natural language processing or computer vision—where abundant data is available from the internet—robot manipulation requires precise and physically grounded demonstrations, often collected manually and in controlled environments. The largest manipulation dataset to date is open x-embodiment dataset which contains 1M+ real robot trajectories pooled from 60 existing robot datasets, while, by contrast, stable diffusion v1 [13] uses 2.3B images filtered from LAION-5B [14] and the filtered common crawl dataset [15] that is used in gpt-3 [16] contains 410B tokens. Although originating from different domains, the orders-of-magnitude disparity in dataset size highlights the pronounced scarcity of robot demonstration data for training a general purpose manipulation policy.

Apart from data availability, an open technical question is which backbone architectures are best suited for modeling continuous action sequences. This aspect has not been thoroughly explored, and as a result, many VLA models that adapt Diffusion Policy as an action expert directly inherit architectures from the image generation domain, resulting in unnecessarily large action generation part [17] [18] [19] [20]. In addition, recent work reveals that uninitialized action experts' gradients are harmful for both VLA's reasoning backbone and the action expert itself, leading to an unstable training process and undermining the sbusequent performance [21]. This reflects a limited understanding of the underlying mechanisms that govern the effectiveness of Diffusion-Like Policy.

This paper introduces a training recipe that could make full use of observation-free data and the decoupling recipe could become a research tool to unveil the underlying mechanisms of Behavior Cloning Models. Firstly, to mitigate the challenge of limited data availability, we propose a training recipe that decouples action generation from Behavior Cloning (BC) models. Our key insight is that while observation-action pairs are expensive to acquire, continuous action sequences themselves—owing to their physical and kinematic structure—can be generated more easily. This motivates a decoupled formulation where we first pretrain a general-purpose **Action Head** (Action Generator), independent of any specific task, and later freeze it and train only conditioning part for downstream tasks. Concretely, we introduce the **Decoupled Action Head** training recipe. The method proceeds in two stages:

- **Stage 1:** Joint Position (JP) is introduced as conditioning signal to train a general purpose Action Head.
- **Stage 2:** With the pretrained frozen action generation backbone, conditioning layers are replaced to handle features from normal observation encoders.

Decoupling training recipe has some obvious benefits.

1) Task specific knowledge can be fully confined to the conditioning network, reducing reliance on a large action-generation backbone.
2) Training speed is improved. For instance, Diffusion

[1] Jian Zhou, Sihao Lin, Shuai Fu and Qi Wu are with the Australian Institute for Machine Learning, University of Adelaide, SA, Australia. {j.zhou, sihao.lin, shuai.fu, qi.wu01}@adelaide.edu.au

Policy-CNN, which is commonly adopted by other methods, achieves 41% faster training speed under decoupling as the majority parameters of the model is frozen.

Secondly, to explore the underlying mechanisms, we further compared the effects of conditioning methods and action-generation backbones within the decoupled training recipe. Our results show that feature modulation is more suitable than cross-attention for this paradigm, while the specific choice of action-generation backbone is comparatively less critical and can even replaced by MLP Blocks.

In summary, this work has the following contributions:

1) We validate the feasibility of an observation-free decoupled action head, showing that task-specific knowledge can be confined to the conditioning module.
2) We identify feature modulation as the effective conditioning method under decoupling, explain why cross-attention fails, and propose DP-T-FiLM.
3) We demonstrate that the action head is relatively less critical, and design a lightweight DP-MLP that achieves substantial speedups while preserving performance.

Beyond contribution on training recipe and model architecture above, the DP-MLP designed to validate observation in the third contribution also achieve great efficiency gain in normal training compared with vanilla Diffusion Policy, where it have marginally better performance but 83.9% faster in training speed.

Taken together, these findings contribute new insights into the design of neural architectures for Behavior Cloning, and suggest that lightweight backbones paired with effective modulation mechanisms can achieve efficient policy learning. It also give insight on where to scale for large general BC policy learning, since a diffusion policy model with a 4M frozen MLP could do various tasks.

II. Background

A. Behavior Cloning (BC) and Diffusion Policy (DP)

Within Behavior Cloning (BC), a variety of approaches have been explored prior to Diffusion-based methods. Unlike Diffusion Policy (DP), which emphasizes predicting continuous action sequences, earlier works adopt alternative strategies. For example, methods such as BeT [4], PerAct [6], and RVT-2 [22] formulate action prediction as classification over categorical bins; ACT employs weighted averaging across multiple frames; and models such as RT-1 [9] tokenize actions in different ways. DP differs from these approaches by demonstrating that predicting continuous action sequences leads to superior performance. Building on this insight, we further show that the ability to generate continuous action sequences can be pretrained independently, thereby enabling the DP network to function as a decoupled action head.

B. Data Generation for Manipulation

Data generation has become one of the central themes in behavior cloning (BC). In the real world setting, many labs collect and clean lab-specific datasets tailored to their robots and environments, and Open-X Embodiment [23] brings many of these collections under a common interface. In simulation, pipelines typically follow one of two routes: (i) converting existing human demonstrations into new tasks or domains (e.g., MimicGen [24]), or (ii) scripting object-centric rollouts with task programs (e.g., RLBench [25]). More recently, several works aim to synthesize data at larger scale via *video generation* (e.g., Rebo [26]; TASTE-Rob [27]; RoboMaster [28]) or Gaussian-based scene and trajectory reconstruction [29], seeking to expand diversity without the full cost of human involving.

Despite this progress, practical constraints remain. First, generating long-horizon, contact-rich interactions is still expensive, leading to small-sample regimes even in simulation. Second, coverage is often narrow: datasets concentrate on a single robot, a single lab layout, or a limited set of objects, which limits generalization. Given these challenges, we observe that although observation–action pairs are costly to obtain, trajectories can be generated at nearly zero cost with appropriate design.This raises a natural question: **Can a policy learn continuous action generation in advance?** Our proposed pretraining strategy based on observation-free data provides an alternative direction for policy learning. This approach reduces reliance on massive fully supervised datasets to train a model that is able to generating high-fidelity action sequences while retaining the strengths of BC at task-specific stage2 training.

C. Model Design for Action Generator Backbone

Many BC systems follow a common architecture: an observation encoder (images and, optionally, low-dimensional state) produces features or tokens that condition an action generator via cross-attention, feature modulation (e.g., FiLM), or concatenation. Representative examples include ALOHA-Unleashed [30], Diffusion Policy [2], 3D Diffusion Actor [8], and ACT [3]. A smaller line of work formulates manipulation as discrete spatial classification with point transformers (e.g., 3D-LOTUS [31]).

1) Action Generator: As the backbone that produces actions, transformers are widely adopted (e.g., DP-Transformer [2], ALOHA-Unleashed [30], RT-1 [9], Octo [12], OpenVLA [11]) due to their modeling capacity and natural compatibility with LLM/VLM. In parallel, U-Net–style backbones are also commonly used, with a detailed comparison provided in the original DP paper [2]. In our experiments, however, we find that once the action head is decoupled the backbone type is not critical. UNet, Transformer, and even a lightweight MLP can achieve comparable performance, suggesting that, for training a large general manipulation policy, action generation backbone may not the place to scale.

2) Observation Encoder: Given the limited availability of observation–action data, observation encoders have received substantial attention. The original DP employs a straightforward image encoder plus low-dimensional state. DP3 [32] replaces images with 3D point clouds, showing that even simple 3D encoders can improve performance. Other work, such as PerAct [6], lifts 2D CLIP features into 3D, providing a strong object-centric prior. Meanwhile, pretrained encoders (e.g., SUGAR [33], SPA [34]) aim to produce general-purpose representations via large-scale pretraining. To assess feasibility under controlled conditions, we retain the original DP image-based inputs, ensuring a direct, comparable assessment.

III. Preliminaries

A. Diffusion Policy

Diffusion Policy (DP) is a special implementation of Behavior Cloning (BC) where the objective is to learn a policy directly from human demonstrations. A dataset $\mathcal{D}_{\text{flat}} = \{(o_n, a_n)\}_{n=1}^{M},\ M = \sum_{i=1}^{N} T_i$ (T denotes the number of frames in each demo and N denotes the number of demo) consists of paired observations and actions collected from expert rollouts. In its conventional form, BC treats control as a direct supervised mapping from observations to actions:

$$a_t \;=\; \pi_\theta(o_t). \tag{1}$$

Diffusion Policy extends this paradigm by formulating action prediction as a conditional generative denoising process.

Let o_t^{img} and o_t^{low} denote the high-dimensional image observation and the low-dimensional state at demo's time t, respectively. For single diffusion timestep τ, each input is encoded by a separate encoder ϕ_{img} and ϕ_{low}, and concatenated with the diffusion timestep embedding to form a global conditioning vector:

$$c_{t,\tau} \;=\; \big[\,\phi_{\text{img}}(o_t^{\text{img}}),\ \phi_{\text{low}}(o_t^{\text{low}}),\ f_\tau(\tau)\,\big]. \tag{2}$$

Diffusion Policy in his paper introduced two variants, which are Diffusion Policy CNN and Diffusion Policy Transformers. They share the same encoder design but differs from both backbone networks and conditioning.

1) DP-C: Diffusion Policy CNN (DP-C) receives global conditioning vector as input, using a feature modulation method (FiLM) to inject conditioning information to a 1D-CNN based U-net backbone network. Formally, a set of linear projections maps the global conditioning vector to FiLM parameters, one pair per block:

$$\{(\boldsymbol{\gamma}_r, \boldsymbol{\beta}_r)\}_{r=1}^{R} \;=\; f_\theta(c_{t,\tau})\,, \tag{3}$$

where R is the number of blocks in action head backbone. Diffusion target for the single demo frame t and diffusion timestep τ is then calculated as

$$\text{DiffusionTarget}_{t,\tau} = g_\theta\Big(\{(\boldsymbol{\gamma}_r, \boldsymbol{\beta}_r)\}_{r=1}^{R},\, a_t^{(\tau)}\Big). \tag{4}$$

2) DP-T: Similar to DP-C, the Diffusion Policy Transformer (DP-T) also receives the global conditioning vector as input, but processes it differently. DP-T constructs a token sequence by combining the observation embedding with the diffusion timestep embedding. Since DP-T conditions on the two most recent observations, this results in a sequence of three tokens ($n_{\text{obs}} = 2$ plus the timestep token), resulting a sequence of shape $(B, 3, d_{model})$. Positional encodings are added to the token sequence to preserve temporal order. The sequence is then projected into the model dimension through an MLP and further processed by an MLP encoder to form the memory representation for the Transformer decoder. The decoder integrates this conditioning information into the denoising process via cross-attention.

Beyond model architecture design (DP-C, DP-T), factors such as the number of input and output frames and the representation of actions also play an important role in the Diffusion Policy paradigm. To maintain controllability in our study, we primarily focus on the setting of DP-C variant. The issues associated with DP-T and corresponding mitigation strategies will be elaborated in the methodology and experimental sections.

B. Forward Kinematics of the Franka Emika Panda

Franka Emika Panda is used as the robot equipment in our experiments. For clarity, we present the one-to-one mapping from each joint position (JP) to a unique end-effector pose (eePose) here to demonstrate and emphasize that JP can be used as conditioning signal.

The Franka Emika Panda is a 7-DoF robotic manipulator composed entirely of revolute joints. The forward kinematics (FK) problem concerns mapping a set of joint angles $q = [q_1, \ldots, q_7]^\top \in \mathbb{R}^7$ to the pose of the end-effector in the robot's base frame. Formally, the pose is expressed as a homogeneous transformation matrix

$$T_{ee}(q) \;=\; T_{0,1}(q_1)\, T_{1,2}(q_2)\, \cdots\, T_{6,7}(q_7)\, T_{7,ee}, \tag{5}$$

where $T_{i,i+1}(q_i) \in SE(3)$ denotes the rigid transformation introduced by joint q_i, and $T_{7,ee}$ is the fixed transformation from the last joint to the end-effector frame. Each $T_{i,i+1}(q_i)$ can be parameterized using the Denavit–Hartenberg convention (D-H convention) [35] or Modified D-H convention such as that used in this paper [36]. The resulting transformation matrix

$$T_{ee}(q) \;=\; \begin{bmatrix} R(q) & p(q) \\ 0 & 1 \end{bmatrix}, \tag{6}$$

encodes both the rotation $R(q) \in SO(3)$ and position $p(q) \in \mathbb{R}^3$ of the end-effector. This formulation serves as the basis for motion planning, control, and policy learning, where accurate forward kinematics is required to relate the robot's joint configuration to its task-space behavior. We leverage the one-to-one relationship of forward kinematics as a conditioning signal for training on observation-free data.

IV. Method

A. Utilizing Observation-Free Data

A key challenge in behavior cloning (BC) for robotic manipulation is the pure reliance on costly human demonstrations, unlike NLP or CV where internet-scale data are abundant. Under the situation that demonstration collection limits scalability, we notice continuous action sequences themselves are comparatively easy to generate. This raises a significant question - whether observation-free data can be effectively leveraged?

To answer this question, we propose a potential formulation: training continuous action sequence generators without

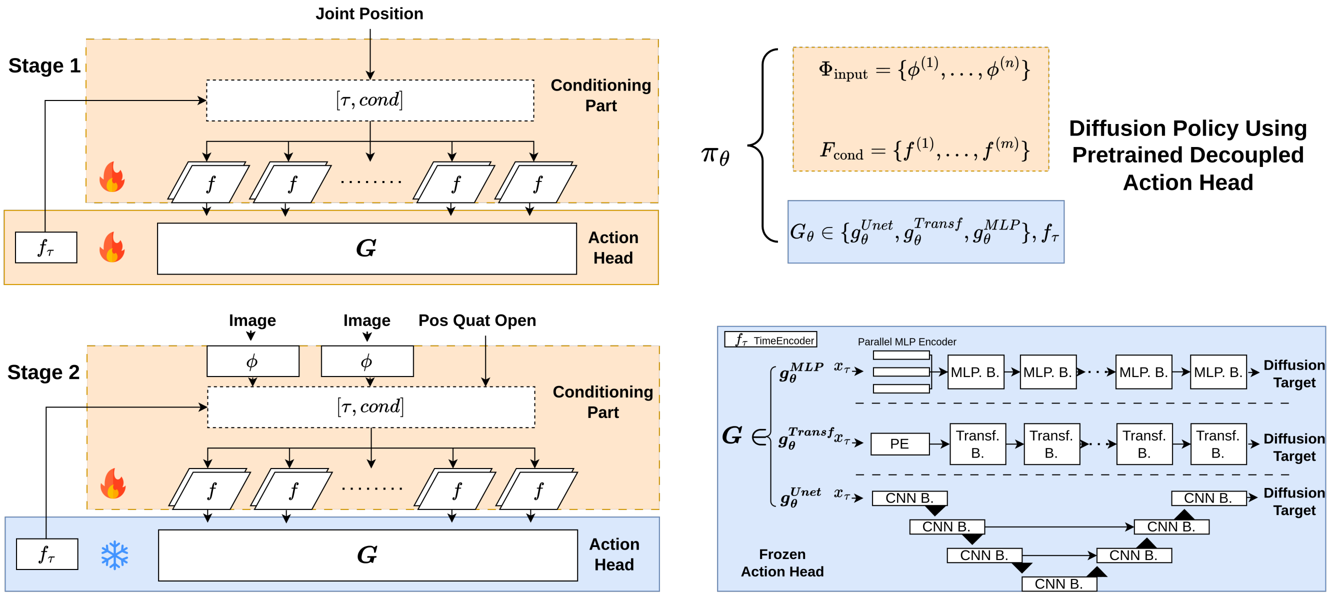


Fig. 1: The application of Decoupled Action Head recipe on Diffusion Policy and our proposed backbone.

explicit observation data. The key insight is that robotic manipulation admits a structural property absent in domains such as image generation—the existence of a deterministic mapping between joint positions and end-effector poses via forward kinematics. This mapping can be written as

$$T_{ee} = T_{0,1}(q_1)\, T_{1,2}(q_2) \cdots T_{n-1,n}(q_n) T_{n,ee}, \tag{7}$$

which provides a one-to-one correspondence between a joint configuration and its resulting end-effector pose. By leveraging this relation, one action representation (joint positions) can substitute for the observation state (S in BC theory). This enables the construction of observation-free training data, where pure action pairs (JP-eePose pairs) serve as substitutes for costly observation–action pairs.

B. Two-Stage Decoupling Training Recipe

The main concept of the decoupled action head is to first leverage observation-free data to train an action generation backbone (Action Head) that is able to generate continuous action sequences, and then replace its conditioning components when adapting to specific tasks. We formalize this procedure as a two-stage training recipe. Recall the Diffusion Policy-CNN that uses feature modulation (FiLM). The inner components of it are encoders, feature modulation modules and single generator backbone:

$$\pi_\theta \begin{cases} \Phi_{\text{input}} = \left\{ \phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(n)} \right\}, \\ F_{\text{cond}} = \left\{ f^{(1)}, f^{(2)}, \ldots, f^{(m)} \right\}, \\ G_\theta \in \left\{ g_\theta^{\text{Unet}},\, g_\theta^{\text{Transf}},\, g_\theta^{\text{MLP}} \right\}, f_\tau, \end{cases} \tag{8}$$

where Φ_{input} denotes the encoders for input sources, F_{cond} refers to feature modulation modules for each block in the generator backbone. f_τ is the timestep embedding layer of diffusion. The relationship between them please refer to III-A and III-A.1.

1) Stage 1: Through the one-to-one correspondence from joint positions (JP) to end-effector pose (eePose), we exploit forward kinematics to provide a conditioning signal. This allows us to train an action head that accepts conditioning using only inexpensive trajectory information. The network architecture are shown in figure 1. In the formula, we train a policy that accepts joint positions and generates continuous action sequences. Since the JP is directly sent to the model as one part of the global conditioning vector, the actual neural networks that are trainable in stage 1 are as follows

$$Stage1 : \pi_\theta \begin{cases} F_{\text{cond}} = \left\{ f^{(1)}, f^{(2)}, \ldots, f^{(m)} \right\}, \\ G_\theta \in \left\{ g_\theta^{\text{Unet}},\, g_\theta^{\text{Transf}},\, g_\theta^{\text{MLP}} \right\}, f_\tau. \end{cases} \tag{9}$$

2) Stage 2: Then, at the task specific training, we replace the conditioning modules and freeze the pretrained action head, and only update the new conditioning parameters: Φ and F.

$$Stage2 : \pi_\theta \begin{cases} \Phi_{\text{input}} = \left\{ \phi^{(1)}, \phi^{(2)}, \ldots, \phi^{(n)} \right\}, \\ F_{\text{cond}} = \left\{ f^{(1)}, f^{(2)}, \ldots, f^{(m)} \right\}, \\ (G_\theta \in \left\{ g_\theta^{\text{Unet}}, g_\theta^{\text{Transf}}, g_\theta^{\text{MLP}} \right\},\ f_\tau)_{\text{frozen}} \end{cases} \tag{10}$$

Note that the encoder Φ is independent of the DP module and can be substituted with any pretrained encoder (CLIP [37], SPA[34]), or with encoders from other modalities (e.g. Point Transformer [38]).

C. Backbone Design

DP-T-FiLM. In addition to the proposed Decoupled Action Head training paradigm, we further investigate the role of backbone design in Diffusion Policy (DP). Our analysis shows that for both decoupled and standard DP training, *feature modulation* plays a critical role in convergence and performance, which motivates the design of a DP-T-FiLM variant. In this variant, we employ a transformer decoder while injecting conditioning information only into the output of its feed-forward network (FFN) and replacing cross-attn with self-attn, so that original conditioning method is replaced and also keep network aligned with DP-T for fair comparison.

DP-MLP. The DP-MLP architecture consists of a simple stack of MLP blocks together with an additional parallel MLP encoder. Each block contains two linear layers, one activation function, two dropout layers, and a normalisation layer with a residual connection. FiLM parameters are injected in the middle of each block. This architecture represents one of the simplest implementation of the Action Head, with many hyperparameters inherited from DP-T. While there remains room for reduction or further exploration, our design primarily serves to validate a hypothesis: since decoupled training of DP-C and DP-T-FiLM under the same data achieves comparable performance to standard training, and task-specific knowledge in the decoupled setting is confined to the conditioning module, the action head may not require a large number of parameters. Details are provided in the experimental section V-C.

V. EXPERIMENTS

Key Concerns. The first major concern regarding the proposed decoupled action head is whether freezing the action generation component, which accounts for the majority of parameters in an action generation backbone, might degrade overall policy performance severely. Another natural concern is whether it has cross-task adaptability, as an action head trained on one distribution of trajectories may not generalize to action sequence distributions from other tasks. To evaluate whether the decoupled action head can preserve the original performance and its adaptability, we have done some experiments on feasibility in section V-A. Another two supporting experiments for two model design insights are shown in IV-C and V-B. Then, a comparison of the training speed of Normal training and using a decoupled action head is shown in V-D.

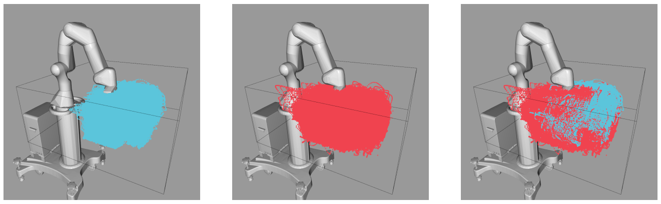
Experiments Setting. Our experiments are conducted in the *MimicGen* environment that was used in the Equivariant Diffusion Policy (EDP) [39]. MimicGen is an extension of the *robomimic* benchmark originally used in vanilla Diffusion Policy. Following EDP, we select eight tasks according to their average demonstration length. Observations consist of two camera pictures: a front-view and a hand-view. Each image stream is encoded using a ResNet-18 backbone, and the resulting visual embeddings are concatenated with low-dimensional state features (end-effector position, orientation via quaternion, and gripper openness). This combined representation serves as the global conditioning vector for policy generation.

The training data are generated by MimicGen, with 1000 demos provided per task. For evaluation, we sample scenes in MuJoCo beyond the training set and execute the learned policies in these environments. For each task, model, and epoch, we evaluate 50 rollouts and compute the success rate. To account for stochasticity, each experiment is repeated with **three seeds**, and results are reported as the average across seeds. This setup provides a reasonable approximation of each model's fitting capability.

A. Feasibility

1) In-distribution Feasibility: To evaluate the general feasibility of the decoupled action head, where an action generator is pretrained using observation-free data, we design an in-distribution experiment. Specifically, we first train an action head solely on the actions from a single task's observation–action pairs. The pretrained action head is then frozen, and the conditioning layers, which include FiLM projection layers and observation encoders, are subsequently trained using the full observation–action pairs.

As shown in Table I, the decoupled action head achieves nearly identical performance to the standard training pipeline in the convolutional variant (DP-C) [2]. In contrast, the transformer variant (DP-T), shown in figure 3b, suffers a large performance drop. This discrepancy can be explained by differences not only in backbone architecture but also in the conditioning mechanism, which we analyze in the subsection V-B.



(a) ACG (b) BDEFH (c) Mixed

Fig. 2: Visualizations of trajectories for the task groups.

2) Out-of-distribution Feasibility: The decoupled action head is designed with the expectation that inexpensive kinematics-generated action data can be leveraged for pretraining. In this context, cross-distribution adaptability becomes particularly important. To investigate this, we empirically select trajectories from three tasks (A, C, and G) to train the action head, and then apply the pretrained head to tasks B, D, E, F, and H. The trajectory distributions are illustrated in Fig. 2, where it can be seen that the trajectories of B, D, E, F, and H do not fully overlap with those of A, C, and G, and many lie outside the ACG distribution.

The results of this cross-distribution experiment are reported in Table III, averaged over three seeds. As shown in the table, while out-of-distribution tasks exhibit a modest decrease in performance compared to in-distribution and normal training, the results clearly demonstrate the feasibility of applying the decoupled action head to unseen task distributions. Moreover, these results are obtained with an action head trained on only a few thousand trajectories. We expect that training on millions or even billions of trajectories would yield a more generalizable feature space. Designing such large-scale observation-free datasets is an important direction for future work.

TABLE I: Performance and speed comparison between normal (Norm) and decoupled action heads (Dec) across tasks A–H.

	A		B		C		D		E		F		G		H		**Avg**		**Speed**	
	Norm	Dec	Norm	Dec	Norm	Dec	Norm	Dec	Norm	Dec	Norm	Dec	Norm	Dec	Norm	Dec	Norm	Dec	Norm	Dec
DP-C[2]	100	100	51.3	47.3	70.7	73.3	44.7	46.7	92.7	87.3	58.7	60.0	24.0	25.3	70.0	67.3	64.0	63.4	1.00	1.41
DP-MLP	100	100	49.3	36.0	71.3	72.7	42.0	40.7	85.3	80.7	58.0	58.0	40.7	28.0	68.0	72.7	64.3	61.1	**1.83**	**1.89**
DP-T[2]	98.0	86.0	39.3	19.3	66.7	35.3	24.7	18.0	78.0	44.0	63.3	52.7	26.0	8.7	70.0	45.3	59.8	38.7	1.00	1.17
DP-T-FiLM	98.0	98.7	48.7	41.3	61.3	54.0	30.0	28.0	85.3	76.0	58.7	57.3	23.3	14.0	63.3	58.0	58.6	53.4	1.06	1.16

TABLE II: Mapping between task codes and task names.

Code	Task Name	Code	Task Name
A	Stack D1	E	Stack Three D1
B	Square D2	F	Hammer Cleanup D1
C	Coffee D2	G	Three Piece Assembly D2
D	Threading D2	H	Mug Cleanup D1

TABLE III: Comparison on performance of tasks using action head pretrained from pure actions of ACG.

Method	**B**	**D**	**E**	**F**	**H**	**Avg.**
Normal	51.3	44.7	92.7	58.7	70.0	63.48
In-Distribution	47.3	46.7	87.3	60.0	67.3	61.72
Out-of-Distribution	47.3	39.3	83.3	56.0	68.7	58.92

3) Multi-Task Feasibility: The multi-task experimental setting involves training a single policy jointly on multiple tasks. Although the feasibility of the decoupled action head in multi-task settings does not require explicit verification, we include such experiments here for completeness. We select three tasks (A, C, and G) for evaluation, and the results are reported in Table IV. The choice of only A, C, and G as training tasks was purely due to computational constraints. Although the results suggest a modest improvement, we note cautiously that larger-scale experiments are required to more thoroughly investigate how the concentration of task-specific knowledge in the decoupled setting affects policy performance.

Feasibility Summary. Taken together, the three feasibility experiments above demonstrate that an action generation backbone trained purely on trajectories (observation-free data) can be applied to both in-distribution tasks and unseen (out-of-distribution) tasks, exhibiting strong cross-task generalization as well as multi-task adaptability.

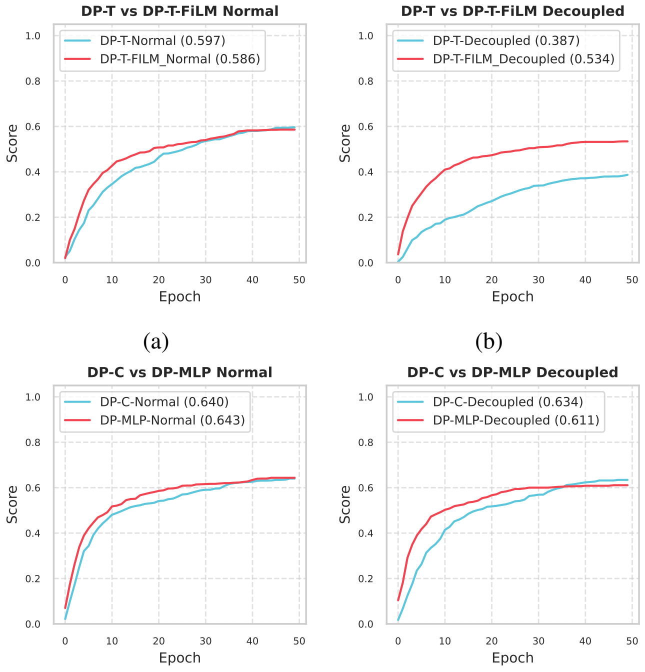
B. Feature Modulation Matters

As shown in Table I, the performance of DP-C remains nearly unchanged under decoupling, whereas DP-T exhibits a significant decline. This discrepancy arises from the difference between cross-attention and feature modulation when applied in the decoupled setting. DP-C relies on feature-wise linear modulation (FiLM), a feature modulation method, whereas DP-T employs cross-attention.

$$\gamma_{t,\tau}, \beta_{t,\tau} = f_\theta(c_{t,\tau}), \qquad h' = \gamma_{t,\tau} \odot h + \beta_{t,\tau}, \qquad (11)$$

TABLE IV: Multitasks performance between Normal and in-distribution decoupled training.

Method	**A**	**C**	**G**	**Avg.**
Normal	96.0	72.0	16.0	61.3
Decoupled	98.7	63.3	27.3	63.1



(a) (b)

(c) (d)

Fig. 3: (a) and (b) are comparison between DP-T-FiLM and DP-T [2] for section V-B; (c) and (d) are comparison between DP-MLP and DP-C [2] for section V-C.

The FiLM above or other related feature modulation methods, such as AdaLN, can be interpreted as dynamic parameterizations of the feature space. At stage 1, conditioning vectors adjust the inner middle tensor of backbone into distinct subspaces through scaling and shifting operations, thereby enabling condition-dependent representation modulation. At stage 2, the replaced conditioning part uses another untrained network to fit the γ, β that required by the pretrained backbone. In other words, feature modulation defines a representational space parameterized by sets of γ and β, and decoupled task-specific training then aligns the observation features with this modulated space. In contrast, cross-attention has another mechanism under decoupling. Consider the attention formula:

$$\mathrm{Attn}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^{\top}}{\sqrt{d_k}}\right) V, \tag{12}$$

(where $Q = XW_q$, $K = XW_k$, $V = XW_v$). When the action head is frozen, W_q, W_k, W_v are fixed, forcing the model to align observation features with a pretrained JP token representation. The decoupled recipe in this situation does not parameterize the continuous action sequences but making fitting eePose sequences problem a fitting JP sequences problem with a reduced quantity of network parameters and lower structural complexity. This mismatch explains the degraded performance in DP-T.

To address this issue, we replace the cross-attention module with FiLM. As shown in Figure 3a and 3b, this substantially reduces the performance drop to acceptable level and normal training DP-T-FILM share nearly identical performance with normal DP-T.

C. Decoupling Recipe as Knowledge Confinement

Beyond the ability to leverage observation-free data, the decoupling recipe has another important property: it confines all task-specific knowledge to the conditioning component, including the encoder and the FiLM linear projection layer. This property leads to a direct implication. That is the action generation backbone only needs the capacity to be conditioned to generate continuous action sequences. In other words, its role is analogous to generating horizon-length sequences of action tokens, similar in form to producing pixels with multiple channels. In our setting, this corresponds to a patch of 16 pixels with 10 channels each, where the horizon length is 16, and the 10 channels consist of 3 for position, 6 for rotation represented in the continuous 6D formulation [40], and 1 for gripper openness. This suggests a hypothesis that only a small number of parameters are needed to model these 16 pixels.

To validate this hypothesis, we naively designed a simplified backbone consisting only of MLP Blocks with basic activation and dropout layers. Actions are treated as tokens processed in the d_{model} dimension, with positional information injected by a parallel MLP block. As shown in Table I, this lightweight architecture exhibits even slightly better performance compared to DP-C at normal training and slight performance drop at decoupled setting, under the situation that DP-C 's action backbone requires 244M parameters and DP-MLP here only require 4M parameters. Note the MLP-based DP is only a naive implementation, we believe a more reasonable implementation could perform even better in efficiency. Epoch-wise comparison can be found in Figure 3c and 3d.

D. Significant Training Efficiency

As an additional benefit, both the decoupled training recipe and the proposed MLP backbone improve training efficiency compared to DP-C. In the decoupled setting, the action generation backbone is frozen during Stage-2 training. It means a portion of the model parameters does not require parameters updates when training on observation–action pairs, which improves training efficiency. The exact speedup is reported in Table V. The table reports training speeds, measured in iterations per second, on both RTX 3090 and RTX 4090 GPUs under normal training and training with a pretrained action head. As shown, the two original DP variants achieve speedups of 41.1% and 10.2%, respectively. Our proposed DP-T-FiLM and DP-MLP exhibit some modest gains relative to their own baselines, with improvements of 7.2% and 2.9%. However, when comparing DP-MLP under normal training with DP-C, DP-MLP achieves substantially higher efficiency, improving training speed by 83.9% on Normal training and 89.1% on decoupled training (using RTX 4090).

TABLE V: Training speed of DP variants under normal and decoupled training on 3090 and 4090 GPUs. Values represent **iterations/s**.

Method	**3090**			**4090**		
	Norm	Dec	Dec/Norm	Norm	Dec	Dec/Norm
DP-C[2]	5.21	7.35	41.1%	10.67	15.06	41.1%
DP-T[2]	7.45	8.01	7.5%	17.41	19.19	10.2%
DP-T-FiLM	7.93	8.70	9.7%	18.28	19.60	7.2%
DP-MLP	9.11	9.82	7.7%	19.62	20.18	2.9%
DP-MLP / DP-C(Norm)	74.8%	88.4%	–	83.9%	89.1%	–

VI. Conclusion

This work introduces a decoupled action head training recipe that enables the use of observation-free data. We further identify the critical role of feature modulation in the decoupled setting. Finally, the insight from knowledge confinement reveals that a lightweight 4M-parameter MLP backbone can replace a 244M-parameter CNN U-Net, achieving substantial speedups while even slightly surpassing performance.

However, alongside these contributions, the limitations of our experiments give rise to several open problems. First, while we demonstrate the feasibility of using observation-free data, the design and generation of such low-cost data require careful consideration, and the effort involved could itself constitute an independent line of research. Moreover, the upper bound of this approach remains unknown. Second, although we show the effectiveness of an MLP backbone, the limits of knowledge confinement have not been fully explored. In particular, it remains unclear how simple a neural network can be while still generating continuous action sequences effectively, and what trade-offs such simplification may introduce.

References

[1] M. Zare, P. M. Kebria, A. Khosravi, and S. Nahavandi, “A survey of imitation learning: Algorithms, recent developments, and challenges,” *IEEE Transactions on Cybernetics*, 2024.

[2] C. Chi, Z. Xu, S. Feng, E. Cousineau, Y. Du, B. Burchfiel, R. Tedrake, and S. Song, “Diffusion policy: Visuomotor policy learning via action diffusion,” *The International Journal of Robotics Research*, 2024.

[3] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn, "Learning fine-grained bimanual manipulation with low-cost hardware," 2023. [Online]. Available: https://arxiv.org/abs/2304.13705

[4] N. M. M. Shafiullah, Z. J. Cui, A. Altanzaya, and L. Pinto, "Behavior transformers: Cloning k modes with one stone," 2022. [Online]. Available: https://arxiv.org/abs/2206.11251

[5] T. Gervet, Z. Xian, N. Gkanatsios, and K. Fragkiadaki, "Act3d: 3d feature field transformers for multi-task robotic manipulation." [Online]. Available: http://arxiv.org/abs/2306.17817

[6] M. Shridhar, L. Manuelli, and D. Fox, "Perceiver-actor: A multi-task transformer for robotic manipulation." [Online]. Available: http://arxiv.org/abs/2209.05451

[7] S. Chen, R. Garcia, C. Schmid, and I. Laptev, "PolarNet: 3d point clouds for language-guided robotic manipulation." [Online]. Available: http://arxiv.org/abs/2309.15596

[8] T.-W. Ke, N. Gkanatsios, and K. Fragkiadaki, "3d diffuser actor: Policy diffusion with 3d scene representations." [Online]. Available: http://arxiv.org/abs/2402.10885

[9] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, T. Jackson, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, I. Leal, K.-H. Lee, S. Levine, Y. Lu, U. Malla, D. Manjunath, I. Mordatch, O. Nachum, C. Parada, J. Peralta, E. Perez, K. Pertsch, J. Quiambao, K. Rao, M. Ryoo, G. Salazar, P. Sanketi, K. Sayed, J. Singh, S. Sontakke, A. Stone, C. Tan, H. Tran, V. Vanhoucke, S. Vega, Q. Vuong, F. Xia, T. Xiao, P. Xu, S. Xu, T. Yu, and B. Zitkovich, "Rt-1: Robotics transformer for real-world control at scale," in *arXiv preprint arXiv:2212.06817*, 2022.

[10] Google DeepMind RT-2 Team, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," in *arXiv preprint arXiv:2307.15818*, 2023.

[11] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn, "Openvla: An open-source vision-language-action model," *arXiv preprint arXiv:2406.09246*, 2024.

[12] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine, "Octo: An open-source generalist robot policy," in *Proceedings of Robotics: Science and Systems*, Delft, Netherlands, 2024.

[13] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2021.

[14] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev, "Laion-5b: An open large-scale dataset for training next generation image-text models," 2022. [Online]. Available: https://arxiv.org/abs/2210.08402

[15] Common Crawl, "Common crawl corpus," 2008–2025, a regularly updated web-scale dataset of crawled web pages. [Online]. Available: https://commoncrawl.org

[16] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, Eds., vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[17] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, S. Jakubczak, T. Jones, L. Ke, S. Levine, A. Li-Bell, M. Mothukuri, S. Nair, K. Pertsch, L. X. Shi, J. Tanner, Q. Vuong, A. Walling, H. Wang, and U. Zhilinsky, "π_0: A vision-language-action flow model for general robot control," 2024. [Online]. Available: https://arxiv.org/abs/2410.24164

[18] Q. Li, Y. Liang, Z. Wang, L. Luo, X. Chen, M. Liao, F. Wei, Y. Deng, S. Xu, Y. Zhang, X. Wang, B. Liu, J. Fu, J. Bao, D. Chen, Y. Shi, J. Yang, and B. Guo, "Cogact: A foundational vision-language-action model for synergizing cognition and action in robotic manipulation," 2024. [Online]. Available: https://arxiv.org/abs/2411.19650

[19] S. Liu, L. Wu, B. Li, H. Tan, H. Chen, Z. Wang, K. Xu, H. Su, and J. Zhu, "Rdt-1b: a diffusion foundation model for bimanual manipulation," *arXiv preprint arXiv:2410.07864*, 2024.

[20] Z. Hou, T. Zhang, Y. Xiong, H. Pu, C. Zhao, R. Tong, Y. Qiao, J. Dai, and Y. Chen, "Diffusion transformer policy," *arXiv preprint arXiv:2410.15959*, 2024.

[21] D. Driess, J. T. Springenberg, B. Ichter, L. Yu, A. Li-Bell, K. Pertsch, A. Z. Ren, H. Walke, Q. Vuong, L. X. Shi *et al.*, "Knowledge insulating vision-language-action models: Train fast, run fast, generalize better," *arXiv preprint arXiv:2505.23705*, 2025.

[22] A. Goyal, V. Blukis, J. Xu, Y. Guo, Y.-W. Chao, and D. Fox, "RVT-2: Learning precise manipulation from few demonstrations." [Online]. Available: http://arxiv.org/abs/2406.08545

[23] Open X-Embodiment Collaboration, "Open X-Embodiment: Robotic learning datasets and RT-X models," https://arxiv.org/abs/2310.08864, 2023.

[24] A. Mandlekar, S. Nasiriany, B. Wen, I. Akinola, Y. Narang, L. Fan, Y. Zhu, and D. Fox, "Mimicgen: A data generation system for scalable robot learning using human demonstrations," in *7th Annual Conference on Robot Learning*, 2023.

[25] S. James, Z. Ma, D. Rovick Arrojo, and A. J. Davison, "Rlbench: The robot learning benchmark & learning environment," *IEEE Robotics and Automation Letters*, 2020.

[26] Y. Fang, Y. Yang, X. Zhu, K. Zheng, G. Bertasius, D. Szafir, and M. Ding, "Rebot: Scaling robot learning with real-to-sim-to-real robotic video synthesis," 2025. [Online]. Available: https://arxiv.org/abs/2503.14526

[27] H. Zhao, X. Liu, M. Xu, Y. Hao, W. Chen, and X. Han, "Taste-rob: Advancing video generation of task-oriented hand-object interaction for generalizable robotic manipulation," 2025. [Online]. Available: https://arxiv.org/abs/2503.11423

[28] X. Fu, X. Wang, X. Liu, J. Bai, R. Xu, P. Wan, D. Zhang, and D. Lin, "Learning video generation for robotic manipulation with collaborative trajectory control," 2025. [Online]. Available: https://arxiv.org/abs/2506.01943

[29] S. Yang, W. Yu, J. Zeng, J. Lv, K. Ren, C. Lu, D. Lin, and J. Pang, "Novel demonstration generation with gaussian splatting enables robust one-shot manipulation," *arXiv preprint arXiv:2504.13175*, 2025.

[30] T. Z. Zhao, J. Tompson, D. Driess, P. Florence, K. Ghasemipour, C. Finn, and A. Wahid, "Aloha unleashed: A simple recipe for robot dexterity," 2024. [Online]. Available: https://arxiv.org/abs/2410.13126

[31] R. Garcia, S. Chen, and C. Schmid, "Towards generalizable vision-language robotic manipulation: A benchmark and LLM-guided 3d policy." [Online]. Available: http://arxiv.org/abs/2410.01345

[32] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu, "3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations," in *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[33] S. Chen, R. Garcia, I. Laptev, and C. Schmid, "Sugar: Pre-training 3d visual representations for robotics," in *CVPR*, 2024.

[34] H. Zhu, , H. Yang, Y. Wang, J. Yang, L. Wang, and T. He, "Spa: 3d spatial-awareness enables effective embodied representation," *arXiv preprint arxiv:2410.08208*, 2024.

[35] J. Denavit and R. Hartenberg, "A kinematic notation for lower-pair mechanisms based on matrices," *Journal of Applied Mechanics*, vol. 22, no. 2, pp. 215–221, 1955.

[36] C. Gaz, M. Cognetti, A. Oliva, P. R. Giordano, and A. De Luca, "Dynamic identification of the franka emika panda robot with retrieval of feasible parameters using penalty-based optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4147–4154, 2019.

[37] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," 2021. [Online]. Available: https://arxiv.org/abs/2103.00020

[38] H. Zhao, L. Jiang, J. Jia, P. H. Torr, and V. Koltun, "Point transformer," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2021, pp. 16 259–16 268.

[39] D. Wang, S. Hart, D. Surovik, T. Kelestemur, H. Huang, H. Zhao, M. Yeatman, J. Wang, R. Walters, and R. Platt, "Equivariant diffusion policy," 2024. [Online]. Available: https://arxiv.org/abs/2407.01812

[40] Y. Zhou, C. Barnes, J. Lu, J. Yang, and H. Li, "On the continuity of rotation representations in neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 5745–5753.