Flexible Multitask Learning with Factorized Diffusion Policy

Chaoqi Liu[1], Haonan Chen[1], Sigmund H. Høeg[2*], Shaoxiong Yao[1*], Yunzhu Li[3], Kris Hauser[1], Yilun Du[4]

[1]University of Illinois at Urbana-Champaign

[2]Norwegian University of Science and Technology

[3]Columbia University [4]Harvard University

***Abstract*—We present *Factorized Diffusion Policy* (`FDP`), a modular framework that models complex action distributions as compositions of specialized diffusion components. Each component captures a distinct behavioral mode, and together they represent the full multimodal distribution. This structure enables interpretable sub-skill learning and supports flexible task adaptation by fine-tuning or adding new components. Across simulated and real-world robotic tasks, `FDP` outperforms monolithic and MoE-based baselines, achieving 24% higher multitask success and 34% improvement in adaptation. Website: factorized-diffusion-policy.github.io.**

I. Introduction

Recent progress in large-scale imitation learning has enabled training general-purpose robot policies from diverse demonstrations. Yet, fitting a single model to complex, multimodal tasks remains a challenge – monolithic diffusion policies struggle to generalize and adapt without forgetting.

Modular methods, such as Mixture-of-Experts (MoE), offer scalability by distributing skills across components. However, existing approaches often suffer from routing instability, overlapping expert roles, and lack of probabilistic grounding.

We propose *Factorized Diffusion Policy* (`FDP`), which represents the policy as a composition of diffusion components. Each module specializes in a behavioral mode and is softly combined through observation-dependent score aggregation—avoiding hard selection and improving training stability. `FDP` builds on compositional diffusion modeling, where sampling from the product of distributions provides both a principled foundation and interpretable constraint satisfaction.

The modular design enables efficient adaptation to new tasks by adding or fine-tuning selected components. We demonstrate strong multitask and adaptation performance across Meta-World, RLBench, and real-world settings.

II. Related Works

Diffusion Models for Robotics. Diffusion models offer stable training and flexible generation, and have recently been applied to policy learning [1], grasp synthesis [2], and planning [3]. Diffusion Policy (DP) [1] showed that these models can learn effective single-task visuomotor control from demonstrations.

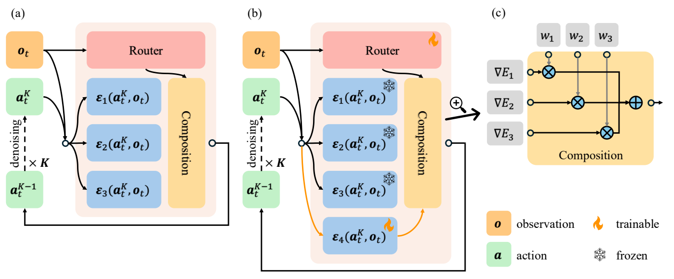


Fig. 1: **Overview of `FDP`. (a)** Given an observation $\mathbf{o}_t$, multiple diffusion experts predict score estimates $\boldsymbol{\varepsilon}_i(\mathbf{a}_t^K, \mathbf{o}_t)$ at each denoising step. A lightweight router network computes observation-dependent weights $\{w_i\}$, which are used to compose the final score as a weighted sum (see **(c)**). The composed score guides the iterative denoising process over K steps to generate an action $\mathbf{a}_t$. **(b)** This compositional structure enables `FDP` to model complex multimodal distributions and supports modular adaptation via selective tuning or addition of diffusion components.

Multitask Imitation and Modularity. Multitask policies often rely on monolithic architectures [4] or language-conditioning [5], which limit scalability and adaptability. MoE-based methods such as SDP [6] and MoDE [7] improve modularity but suffer from expert imbalance and reduced interpretability.

In contrast, `FDP` uses continuous score aggregation to combine component outputs, ensuring balanced optimization and clearer specialization. `FDP` supports modular growth through new components while preserving previously learned skills – enabling scalable and interpretable multitask policy learning.

III. `FDP`: Factorized Diffusion Policy

We propose `FDP`, a modular policy architecture designed to scale to diverse manipulation tasks and enable efficient adaptation to new ones. Instead of relying on a monolithic diffusion model, we factorize the policy into multiple diffusion components, each modeling a distinct sub-mode of the multimodal action distribution. At inference time, these components are composed via a lightweight observation-conditioned router (Fig. 1).

We represent the policy distribution as a product of component distributions: $p(\mathbf{a}_t|\mathbf{o}_t) \propto \prod_i p_i(\mathbf{a}_t|\mathbf{o}_t)^{w_{t,i}}$, where p_i is parameterized by a diffusion model, and $w_{t,i}$ are observation-dependent weights predicted by the router. This formulation enables soft composition: rather than selecting a discrete

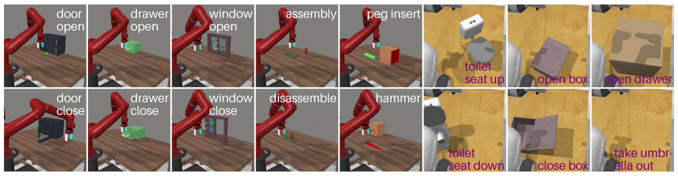
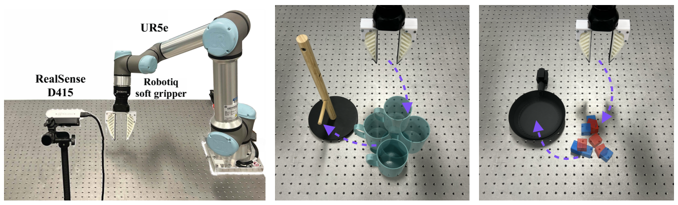


Fig. 2: **Simulation environments**. We evaluate `FDP` on 10 MetaWorld tasks and 6 RLBench tasks.



(a) Real-world workspace. (b) Illustration of *hang-X* (c) Illustration of *cube-X*

Fig. 3: **Real-world setup and task illustrations**. (a) Workspace setup with a UR5e arm, Robotiq gripper, and RealSense D415 camera. (b) High-level illustrations. Example rollouts can be found in Fig. 4.

subset of experts as in MoE-based policies, all components contribute to the final output in a differentiable way. As a result, training is more stable, and each component specializes through consistent optimization.

Each p_i is trained using standard DDPM objectives [8], where the noise prediction networks are optimized jointly using a combined score-matching loss. The router is trained end-to-end alongside the diffusion modules to predict the mixing weights.

This modular structure also enables data-efficient task adaptation. To adapt to a new task, we freeze the original components and introduce a new diffusion module, initialized via upcycling [9]. Only the new module and the router are updated during adaptation, preserving prior capabilities while extending the policy to new tasks without catastrophic forgetting.

Together, this design offers a stable, interpretable, and scalable approach to multitask policy learning and adaptation. We show that `FDP` outperforms both monolithic and MoE-based diffusion baselines across a wide range of tasks in simulation and the real world. Full details can be found in Appendix C.

IV. Experiments

In this section, we aim to empirically investigate several key questions regarding our proposed policy architecture: (1) Whether factorizing the complex action distribution into simpler distributions captured by smaller diffusion models can improve overall policy learning and performance. (2) Whether the modular structure of `FDP`, composed of multiple diffusion-based expert modules, facilitates more efficient and effective task transfer and adaptation. (3) How different adaptation strategies compare, highlighting trade-offs such as data efficiency, policy performance, and compute.

A. Experiments Setup

We evaluate policies on 16 tasks (Fig. 2) across MetaWorld [10] and RLBench [11]. Demonstrations are generated

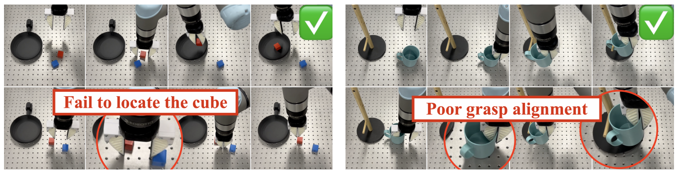


Fig. 4: **Qualitative real-world rollouts.** Each row shows execution for *cube-X* (left) and *hang-X* (right) tasks. Successes are shown in the top row; failure cases (bottom) illustrate challenges faced by baselines. Enlarged version can be found in Fig. 7.

by benchmark-provided scripted experts. Real-world experiments use a UR5e arm with a Robotiq gripper and a RealSense D415 camera (Fig. 3a). We evaluate policies on four distinct tasks: *cube red*, *cube blue*, *hang low*, and *hang high*. The tasks *cube-X* involve picking up a cube of color *X* from the tabletop and placing it into a designated bowl. The *hang-X* tasks require the robot to grasp a mug from the tabletop and precisely hang it on the *X* branch of a mug stand positioned on the table. Illustrations and setups of these real-world tasks are shown in Fig. 3. More details are provided in Appendix D.

B. Implementations

All policies take RGB images and joint angles as input and predict absolute joint angle trajectories. A history window of size 2 is used, with 16-step trajectories predicted and 8 steps executed. We use DDPM [8] with 100 diffusion steps during training and inference. We compare `FDP` against three baselines: DP [1], a monolithic diffusion policy; SDP [6], a MoE-based diffusion policy with observation-conditioned routing; and MoDE [7], a MoE variant with routing based on noise levels. Task-specific routing and goal conditioning are removed for fair comparison. We follow the original configurations used in each baselines, and proportionally reduce the model size of MoDE to match others. We refer readers to the original papers for more details on architecture and training. In `FDP`, four U-Net diffusion modules are composed. For adaptation, we adopt the upcycling strategy [9] to initialize new MoE experts or diffusion components from existing ones. More details are available in Appendix E.

C. Multitask Learning

We first investigate whether decomposing complex motion distributions into simpler, behavior-specialized components can improve policy performance in multitask settings.

Policy	MetaWorld	RLBench
DP	0.725	0.569
SDP	0.717	0.319
MoDE	0.733	0.394
`FDP`	**0.750**	**0.638**

TABLE I: **Multitask learning in simulation.** Average success rate on MetaWorld (*door open*, *drawer open*, *assembly*, *window close*, *peg insert*, *hammer*) and RLBench (*toilet seat up*, *open box*, *open drawer*, *take umbrella out*) tasks. Evaluated over 40 episodes per task. See Table V for full results.

Simulation. We evaluate `FDP` on six MetaWorld tasks (25 demonstrations each) and four RLBench tasks (50 demonstrations each). All methods are evaluated over 40 rollouts per task, with results shown in Table I. The DP baseline performs surprisingly well, particularly on tasks like *drawer open*, *assembly*, and *hammer*, which primarily involve reaching and grasping and exhibit fewer multimodal behaviors – making them easier to solve with a single model. Among modular baselines, SDP underperforms due to instability common in training MoE architectures [9]: too few experts limit expressiveness, while too many can cause overfitting and noisy routing. MoDE performs reasonably by routing based on the noise level, but still inherits instability from MoE training [7]. In contrast, `FDP`'s compositional structure avoids abrupt routing decisions by continuously composing diffusion component outputs via score-weighted aggregation, which enables stable training and more balanced component specialization of the multimodal action distributions.

Policy	Cube Red	Hang Low	Avg.
DP	0.700	0.800	0.750
SDP	0.750	0.650	0.700
MoDE	0.700	0.800	0.750
`FDP`	**0.750**	**0.850**	**0.800**

TABLE II: **Real-world multitask success rates.** Average over 20 trials. Tasks: *cube red* and *hang low*.

Real-world. We further evaluate our method in real-world settings on two tasks: *cube red* (300 demonstrations) and *hang low* (200 demonstrations). 20 samples are used for evaluation, and results are summarized in Table II. The DP baseline often overfits to specific joint trajectories, failing to attend to RGB inputs due to the multimodal and perceptually complex nature of the tasks. By contrast, `FDP` captures diverse behavior patterns more effectively by decomposing the action distribution across interpretable sub-modules. This results in higher success rates. Fig. 4 shows qualitative failure cases from baseline methods, which struggle to capture the complex distribution, resulting in imprecise end-effector poses and frequent task failures.

D. Task Transfer and Adaptation

In this section, we evaluate the adaptability of `FDP` in adapting to novel tasks under limited data. We compare several adaptation strategies: full-parameter fine-tuning, partial fine-tuning of the router, observation encoder, and selective module expansion via new expert components.

Simulation. We evaluate adaptation performance on four MetaWorld tasks and two RLBench tasks, using 10 and 25 demonstrations per task, respectively. We run 60 evaluations for MetaWorld and 40 evaluations for RLBench. As shown in Table III, full-parameter fine-tuning achieves strong performance but is computationally intensive. Partial fine-tuning – modifying only the router or including the observation encoder – offers limited gains. In contrast, adding new modules (two expert blocks per layer for MoE-based methods and a new diffusion component for `FDP`) consistently improves performance. `FDP` benefits most from this strategy, leveraging its compositional structure to reuse prior knowledge while efficiently learning new behaviors.

Method	Policy	MetaWorld	RLBench
Full Param.	DP	0.900	0.800
	SDP	0.892	0.763
	MoDE	**0.917**	0.813
	`FDP`	**0.917**	**0.825**
Router + Obs. Enc.	SDP	0.833	0.338
	MoDE	0.833	0.438
	`FDP`	**0.858**	**0.463**
+ New Module	SDP	0.900	0.450
	MoDE	0.908	0.600
	`FDP`	**0.925**	**0.850**

TABLE III: **Adaptation in MetaWorld** (*door open*, *drawer open*, *assembly*, *window close*, *peg insert*, *hammer*) **and RLBench** (*toilet seat up*, *open box*, *open drawer*, *take umbrella out*). Pretrained on tasks in Table I. Full results in Table VI.

Method	Policy	Cube Blue	Hang High	Avg.
Full Param.	DP	0.750	0.850	0.800
	SDP	0.700	0.800	0.750
	MoDE	0.750	0.800	0.775
	`FDP`	**0.850**	0.750	**0.800**
Router + Obs. Enc.	SDP	0.500	0.450	0.475
	MoDE	0.500	0.550	0.525
	`FDP`	**0.550**	**0.550**	**0.550**
+ New Module	SDP	0.650	0.550	0.600
	MoDE	0.700	0.650	0.675
	`FDP`	**0.850**	**0.850**	**0.850**

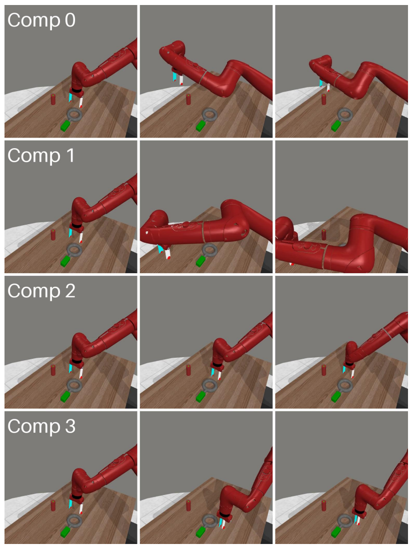
TABLE IV: **Adaptation in real-world**. Evaluated on *cube blue* and *hang high*. Pretrained on *cube red* and *hang low*. See Table VII for full results.

Real-world. We further evaluate adaptation on two real-world tasks, each with 100 demonstrations. 20 samples are used for evaluation. Results in Table IV echo the simulation trends. While full-parameter fine-tuning performs reasonably well, it is resource-intensive. Partial fine-tuning yields modest improvements. The most effective strategy across all methods involves introducing new modules. Under this setting, `FDP` achieves the best performance, highlighting the advantage of its modular design for rapid and robust adaptation even in complex, real-world scenarios.

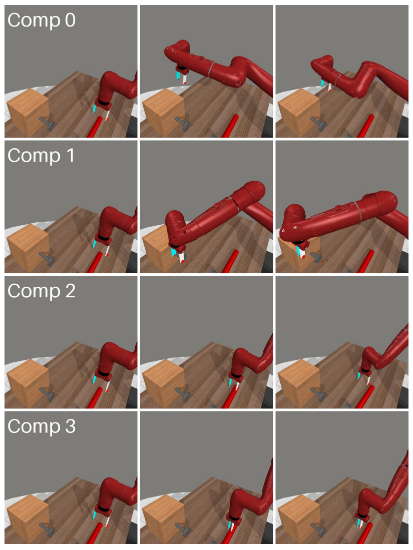
E. Diffusion Components Analysis

To better understand how modularity manifests in `FDP`, we analyze the behavior and specialization of individual diffusion components. Fig. 5 shows rollout trajectories produced by each component in two representative MetaWorld tasks: *assembly* and *hammer*. Across both tasks, we observe that different components specialize in distinct functional stages, such as alignment, approach, and grasp execution. This consistent division of responsibility indicates that `FDP` naturally decomposes complex behaviors into distinct, interpretable sub-skills across its components.

To complement the qualitative analysis, we compute the pairwise cosine similarity between the score outputs, shown in Fig. 6, visualize how the learned components relate to each other during inference. While the components are not completely orthogonal, we observe noticeable variation between different pairs, indicating that diffusion components capture



(a) Assembly



(b) Hammer

Fig. 5: **Rollout trajectories of individual diffusion components** in FDP. **(a)** In *assembly*, components 0 and 1 align the robot with the stand, component 2 aligns with the ring, and component 3 executes the grasp. **(b)** In *hammer*, components 0 and 1 align and approach the pin, component 2 approaches the hammer, and component 3 performs the grasp. Enlarged version in Fig. 8.

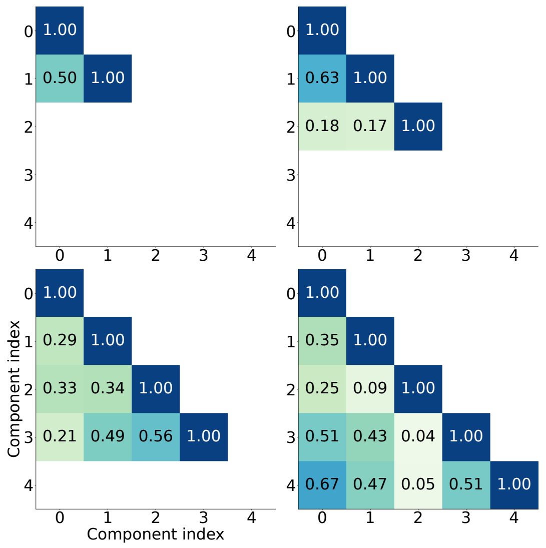


Fig. 6: **Cosine similarity between diffusion component scores.** Each heatmap shows average pairwise similarity for models with 2–5 components, computed over four RLBench tasks. Lower similarity indicates more distinct component behaviors.

distinct, though partially overlapping, aspects of the behavior distribution.

FDP's structure contrasts with baseline MoE-based policies. In MoDE, experts specialize according to diffusion noise levels rather than task semantics, leading to noise-level specialization that lacks behavioral interpretability. In SDP, sub-skills emerge from sets of experts selected across layers, making it difficult to assign functionality to any single expert. Experts can be reused across different combinations or ignored altogether. Furthermore, SDP routers tend to favor a small subset of experts, leading to poor load balancing and limited diversity [6]. FDP assigns each behavioral mode to a distinct, standalone diffusion component. This enables clean semantic separation between modules, avoiding routing instability or expert redundancy commonly seen in MoE. This modular design facilitates straightforward analysis, interpretability, and reuse, contributing to better training stability and more coherent specialization.

V. Real-World Deployability and Generalization

FDP is explicitly designed to enhance generalization and real-world deployability of robot policies through modularity, interpretability, and adaptability. Below, we highlight several ways our design and evaluations align with this goal.

Multitask generalization. FDP demonstrates strong multitask performance across diverse tasks in MetaWorld and RLBench, including object reconfiguration, tool use, and contact-rich manipulation. These tasks require generalizable behavior representations, and our model achieves higher success rates than monolithic and MoE-based baselines, indicating improved generalization to a range of task geometries and dynamics.

Real-world deployment. We evaluate FDP on real-world tasks using a UR5e arm across four task variants with varied object shapes, positions, and constraints. FDP successfully transfers to these settings with minimal tuning, outperforming baselines and showcasing robustness to real-world perception and control noise.

Modular adaptation. FDP supports efficient adaptation to novel tasks by fine-tuning or adding new components without modifying previously learned ones. This enables real-time policy updates and skill expansion on deployed systems without catastrophic forgetting—an essential capability for long-term, continuously learning robots.

Interpretable structure. The interpretable decomposition into behavior-specialized modules facilitates debugging, reuse, and safety analysis in deployed systems. Compared to MoE-based policies with unclear expert roles, FDP's design improves transparency and supports modular verification in safety-critical environments.

Together, these capabilities make FDP a practical framework for scalable, generalizable, and robust robot control in real-world applications.

References

[1] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2024.

[2] Julen Urain, Niklas Funk, Jan Peters, and Georgia Chalvatzaki. SE(3)-DiffusionFields: Learning smooth cost functions for joint grasp and motion optimization through diffusion, June 2023. URL http://arxiv.org/abs/2209.03855. arXiv:2209.03855 [cs].

[3] Michael Janner, Yilun Du, Joshua Tenenbaum, and Sergey Levine. Planning with Diffusion for Flexible Behavior Synthesis. In *Proceedings of the 39th International Conference on Machine Learning*, pages 9902–9915. PMLR, June 2022. URL https://proceedings.mlr.press/v162/janner22a.html. ISSN: 2640-3498.

[4] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. OpenVLA: An Open-Source Vision-Language-Action Model, June 2024. URL http://arxiv.org/abs/2406.09246. arXiv:2406.09246 [cs].

[5] Huy Ha, Pete Florence, and Shuran Song. Scaling Up and Distilling Down: Language-Guided Robot Skill Acquisition. In *Proceedings of The 7th Conference on Robot Learning*, pages 3766–3777. PMLR, December 2023. URL https://proceedings.mlr.press/v229/ha23a.html. ISSN: 2640-3498.

[6] Yixiao Wang, Yifei Zhang, Mingxiao Huo, Ran Tian, Xiang Zhang, Yichen Xie, Chenfeng Xu, Pengliang Ji, Wei Zhan, Mingyu Ding, and Masayoshi Tomizuka. Sparse diffusion policy: A sparse, reusable, and flexible policy for robot learning, 2024. URL https://arxiv.org/abs/2407.01531.

[7] Moritz Reuss, Jyothish Pari, Pulkit Agrawal, and Rudolf Lioutikov. Efficient diffusion transformer policies with mixture of expert denoisers for multitask learning, 2024. URL https://arxiv.org/abs/2412.12953.

[8] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

[9] Xi Victoria Lin, Akshat Shrivastava, Liang Luo, Srinivasan Iyer, Mike Lewis, Gargi Ghosh, Luke Zettlemoyer, and Armen Aghajanyan. Moma: Efficient early-fusion pre-training with mixture of modality-aware experts, 2024. URL https://arxiv.org/abs/2407.21770.

[10] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Avnish Narayan, Hayden Shively, Adithya Bellathur, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning, 2021. URL https://arxiv.org/abs/1910.10897.

[11] Stephen James, Zicong Ma, David Rovick Arrojo, and Andrew J. Davison. Rlbench: The robot learning benchmark & learning environment, 2019. URL https://arxiv.org/abs/1909.12271.

[12] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion, 2024. URL https://arxiv.org/abs/2303.04137.

[13] Max Welling and Yee W Teh. Bayesian learning via stochastic gradient langevin dynamics. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 681–688. Citeseer, 2011.

[14] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7): 1661–1674, 2011.

[15] Lirui Wang, Jialiang Zhao, Yilun Du, Edward H. Adelson, and Russ Tedrake. Poco: Policy composition from and for heterogeneous robot learning, 2024. URL https://arxiv.org/abs/2402.02511.

[16] Yilun Du, Conor Durkan, Robin Strudel, Joshua B. Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc, 2024. URL https://arxiv.org/abs/2302.11552.

[17] Jocelin Su, Nan Liu, Yanbo Wang, Joshua B. Tenenbaum, and Yilun Du. Compositional image decomposition with diffusion models, 2024. URL https://arxiv.org/abs/2406.19298.

[18] Nan Liu, Shuang Li, Yilun Du, Antonio Torralba, and Joshua B. Tenenbaum. Compositional visual generation with composable diffusion models, 2023. URL https://arxiv.org/abs/2206.01714.

[19] Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/378a063b8fdb1db941e34f4bde584c7d-Paper.pdf.

[20] Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B. Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Compositional diffusion-based continuous constraint solvers, 2023. URL https://arxiv.org/abs/2309.00966.

[21] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012. doi: 10.1109/IROS.2012.6386109.

[22] E. Rohmer, S. P. N. Singh, and M. Freese. Coppeliasim (formerly v-rep): a versatile and scalable robot simulation framework. In *Proc. of The International Conference on Intelligent Robots and Systems (IROS)*, 2013. www.coppeliarobotics.com.

Appendix

A. Background on Diffusion Policy

Our method, `FDP`, builds upon the diffusion policy framework [12], leveraging U-Net-based denoising models to represent the conditional action distribution. Diffusion models are a class of generative models that synthesize samples by reversing a predefined noise process. Specifically, we adopt the Denoising Diffusion Probabilistic Model (DDPM) [8] to model the policy $\pi_\theta(\mathbf{a}_t|\mathbf{o}_t)$, where $\mathbf{o}_t$ denotes the observation and $\mathbf{a}_t$ the action.

Sampling begins with a Gaussian noise $\mathbf{a}^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and proceeds through a sequence of denoising steps using a noise prediction network ε_θ:

$$\mathbf{a}^{k-1} = \alpha_k \left(\mathbf{a}^k - \gamma_k\, \varepsilon_\theta(\mathbf{a}^k, \mathbf{o}, k) + \mathcal{N}(\mathbf{0}, \sigma_k^2\mathbf{I})\right), \quad (1)$$

where α_k, γ_k, and σ_k define the noise schedule at timestep k. This denoising process is akin to Stochastic Langevin Dynamics [13], where ε_θ approximates the score function $\nabla \log p(\mathbf{a}|\mathbf{o})$ of an implicit energy-based model [14].

During training, the model learns to predict the added noise. A clean action trajectory $\mathbf{a}^0$ is perturbed by Gaussian noise $\boldsymbol{\epsilon}^k$ to form a noisy sample $\mathbf{a}^k = \mathbf{a}^0 + \boldsymbol{\epsilon}^k$. The network is then supervised to reconstruct the noise via a mean squared error objective:

$$\mathcal{L}_{\text{MSE}} = \left\|\boldsymbol{\epsilon}^k - \varepsilon_\theta(\mathbf{a}^k, \mathbf{o}, k)\right\|_2^2. \quad (2)$$

Minimizing this loss teaches the network to recover the denoising direction, enabling generation of coherent, observation-conditioned action sequences through iterative refinement.

B. Background on Compositional Sampling

Compositional generation with diffusion models has gained increasing attention in robotics and generative modeling due to its ability to produce structured, interpretable outputs. By combining modular distributions, this approach offers enhanced flexibility and expressiveness for tasks such as robot trajectory generation and image synthesis [15, 16, 17, 18].

A core theoretical insight is that aggregating diffusion scores is equivalent to sampling from the product of the underlying probability distributions. Let $p_1, p_2, \ldots, p_n$ be a set of component distributions. Their product distribution is given by:

$$p_{\text{product}}(\mathbf{x}) \propto \prod_{i=1}^{n} p_i(\mathbf{x}), \quad (3)$$

which assigns high probability to samples that are simultaneously likely under all component distributions – i.e., it captures their intersection.

In the energy-based formulation, each $p_i(\mathbf{x}) \propto \exp(-E_i(\mathbf{x}))$, and their product yields a new energy-based model:

$$p_{\text{product}}(\mathbf{x}) \propto \exp\left(-\sum_{i=1}^{n} E_i(\mathbf{x})\right) \text{ [19]}. \quad (4)$$

Sampling from p_{product} can be performed using Langevin dynamics. Starting from an initial noisy sample $\mathbf{x}^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, the iterative update is:

$$\begin{aligned} \mathbf{x}^{k-1} &= \mathbf{x}^k - \gamma_k \nabla_{\mathbf{x}} E_{\text{product}}(\mathbf{x}^k) + \xi_k \quad (5) \\ &= \mathbf{x}^k - \gamma_k \sum_i \nabla_{\mathbf{x}} E_i(\mathbf{x}^k) + \xi_k, \quad (6) \end{aligned}$$

where γ_k is the step size and $\xi_k \sim \mathcal{N}(\mathbf{0}, \sigma_k^2\mathbf{I})$ adds stochasticity to improve exploration. One can bridge EBM and score-matching diffusion [16].

Compositional sampling enables several practical benefits. It enhances robustness by enforcing agreement across component models, which helps reduce uncertainty and improves sample fidelity. Moreover, it facilitates modular design – independently trained components can be integrated at inference time, promoting scalability across diverse tasks and data domains [15, 16].

C. `FDP`: Factorized Diffusion Policy

We aim to develop a modular policy architecture that scales to diverse manipulation tasks and supports efficient adaptation to new ones. Traditional monolithic policies struggle with the complexity and multimodality of real-world action distributions, while modular alternatives like Mixture-of-Experts (MoE) suffer from training instability and poor expert interpretability. Our proposed `FDP`, which directly factorizes the policy into a set of composable diffusion models. Each component captures a distinct behavioral mode, and the final action is produced via a weighted aggregation of these modules conditioned on the current observation (Fig. 1).

1) Probabilistic Policy Modeling: We factorize the action distribution as the product of a set of composed distributions

$$p(\mathbf{a}_t|\mathbf{o}_t) \propto \prod_i p_i(\mathbf{a}_{t,i}|\mathbf{o}_t)^{w_{t,i}}, \quad (7)$$

where $\{w_{t,i}\}$ are observation-dependent weights associated with each component distribution. Intuitively, $p(\mathbf{a}_t|\mathbf{o}_t)$ represents the intersection (logical AND) of individual distributions, assigning high likelihood to samples commonly favored by all component distributions. Moreover, each diffusion component $p_i(\mathbf{a}_{t,i}|\mathbf{o}_t)$ can be interpreted as imposing a behavioral constraint (e.g., collision avoidance, precise grasping) [20]. The composed distribution thus captures the intersection of constraints, naturally framing action generation as constraint satisfaction while maintaining a probabilistic interpretation.

Denoising Diffusion Probabilistic Model (DDPM) framework [8] is adopted to model each component distribution $p_i(\mathbf{a}_{t,i}|\mathbf{o}_t)$. To sample from each component, we start from a noisy action sample $\mathbf{a}_{t,i}^K \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, and iteratively refine it using a noise prediction network $\varepsilon_{\theta_i}(\mathbf{a}_{t,i}^k, \mathbf{o}_t, k)$, progressively denoising over k steps:

$$\mathbf{a}_{t,i}^{k-1} = \alpha_k \left(\mathbf{a}_{t,i}^k - \gamma_k\, \varepsilon_{\theta_i}(\mathbf{a}_{t,i}^k, \mathbf{o}_t, k) + \mathcal{N}(\mathbf{0}, \sigma_k^2\mathbf{I})\right), \quad (8)$$

where α_k, γ_k, and σ_k define the noise schedule. This process closely resembles Stochastic Langevin Dynamics [13], with ε_{θ_i} estimating the score function $\nabla \log p_i(\mathbf{a}_{t,i}|\mathbf{o}_t)$ [14].

Training of DDPM minimizes the mean squared error (MSE) between the true added noise $\boldsymbol{\epsilon}^k$ and the network prediction:

$$\mathcal{L}_{\text{MSE}} = \|\boldsymbol{\epsilon}^k - \varepsilon_\theta(\mathbf{a}_{t,i}^0 + \boldsymbol{\epsilon}^k, \mathbf{o}_t, k)\|_2^2, \tag{9}$$

where $\mathbf{a}_{t,i}^0$ is a clean trajectory sample valid under distribution $p_i(\mathbf{a}_{t,i}|\mathbf{o}_t)$. Minimizing this loss teaches the network to progressively denoise noisy actions conditioned on observations.

2) Compositional Sampling and Routing: We next discuss how can we sample from the actual action distribution $p(\mathbf{a}_t|\mathbf{o}_t)$ given DDPM formulation of component distributions $\{p_i(\mathbf{a}_{t,i}|\mathbf{o}_t)\}$, as well as how to automatically discover each component distribution and optimize corresponding diffusion models jointly.

One way of viewing the composition of distributions is through the lens of energy-based models (EBM) [19]. Assume weights $\{w_{t,i}\}$ are given, and each weighted component distribution is parameterized as $p_i(\mathbf{a}_{t,i}|\mathbf{o}_t) \propto e^{-w_{t,i}E_i}$, then the actual action distribution can be expressed as $p(\mathbf{a}_t|\mathbf{o}_t) \propto e^{-\sum_i w_{t,i}E_i}$ [19]. Therefore, iterative sampling can be performed via Langevin dynamics:

$$\mathbf{a}_t^{k-1} = \mathbf{a}_t^k - \gamma_k \sum_i w_{t,i} \nabla_{\mathbf{a}_t^k} E_i(\mathbf{a}_t^k, \mathbf{o}_t) + \xi_k, \tag{10}$$

where γ_k controls the step size and ξ_k introduces Gaussian noise. Note that we can bridge EBM with score-matching diffusion models [15, 16, 18, 17], which updates Equ. 10 as

$$\mathbf{a}_t^{k-1} = \mathbf{a}_t^k - \gamma_k \sum_i w_{t,i}\, \varepsilon_{\theta_i}(\mathbf{a}_t^k, \mathbf{o}_t, k) + \xi_k. \tag{11}$$

To optimize diffusion components jointly, we update MSE loss in Equ. 9 to

$$\mathcal{L}_{\text{MSE}} = \|\boldsymbol{\epsilon}^k - \sum_i w_{t,i}\, \varepsilon_{\theta_i}(\mathbf{a}_t^0 + \boldsymbol{\epsilon}^k, \mathbf{o}_t, k)\|_2^2, \tag{12}$$

where $\mathbf{a}_t^0$ is a demonstration trajectory sample. Then all diffusion components are optimized jointly end-to-end.

The weights $\{w_{t,i}\}$ are predicted by a lightweight observation-conditioned neural network, referred to as *router*, which is optimized along with other diffusion components. This brings the last piece of `FDP` architecture. The pseudocode for training and inference are provided in Algo. 1 and Algo. 2.

Compared to discrete MoE routing, our compositional approach avoids routing instability and expert imbalance [9] by assigning continuous, observation-dependent weights to all components, rather than selecting a hard subset. In MoE, only a few experts are activated at each step, which can lead to underutilization of some experts and overfitting or saturation in others, especially when routing distributions are sharp or poorly calibrated. In contrast, our method aggregates contributions from all components via soft score-weighted composition, ensuring all modules remain active during optimization. Additionally, because all components participate in every training step, they receive gradient signals consistently, which encourages functional specialization.

3) Multitask Learning and Adaptation: **Multitask Learning**. This factorization is particularly well-suited for multitask imitation learning, where action distributions are inherently multimodal due to diverse object properties, contact dynamics, and task goals. In contrast to monolithic policies that must capture all modes simultaneously, `FDP` distributes complexity evenly across diffusion components, each modeling a coherent subspace of behaviors. Unlike MoE policies, where skills may span combinations of experts across layers, our formulation yields interpretable and disentangled sub-skills.

Adapting to New Tasks. The modularity of `FDP` also enables efficient adaptation to unseen tasks. Instead of retraining the full model, we adapt by introducing a new diffusion component $\varepsilon_{\theta_{\text{new}}}$, initialized via *upcycling* [9] – copying weights from existing components. The updated score function becomes:

$$\varepsilon_{\text{adapt}}(\mathbf{a}_t^k, \mathbf{o}_t, k) = \sum_i w_i\, \varepsilon_{\theta_i}(\mathbf{a}_t^k, \mathbf{o}_t, k) + w_{\text{new}}\, \varepsilon_{\theta_{\text{new}}}(\mathbf{a}_t^k, \mathbf{o}_t, k), \tag{13}$$

where only $\varepsilon_{\theta_{\text{new}}}$ and the new router are updated during adaptation, using the training loss in Equ. 9. All previously trained components $\{\varepsilon_{\theta_i}\}$ are frozen. Freezing existing components ensures that the optimization focuses solely on capturing novel task dynamics without disrupting existing capabilities, thereby mitigating catastrophic forgetting. Such selective adaptation significantly reduces the number of trainable parameters and the amount of supervision required. In contrast, MoE models, where overlapping expert roles make modular reuse and analysis more difficult.

Finally, `FDP` supports heterogeneous architectures – diffusion models can vary in architecture and size, enabling scalable allocation of computation to match task complexity. This extensibility makes `FDP` broadly applicable in diverse and evolving robotic domains.

D. Experiment Setup

Simulation. We evaluate `FDP` on two widely used robotic manipulation benchmarks: MetaWorld [10] and RLBench [11], each based on a distinct physics engine. MetaWorld, built on MuJoCo [21], offers 50 diverse tasks; we select 10 representative ones focused on object reconfiguration and tool use. RLBench, running on CoppeliaSim [22], includes over 100 tasks; we select 6 involving contact-rich and articulated-object interactions. In total, we evaluate across 16 tasks (see Fig. 2). Expert demonstrations are generated using the scripted policies provided by each benchmark.

Real-world. Real-world experiments are conducted using a UR5e robotic arm equipped with a Robotiq soft gripper and a front-facing RealSense D415 camera (Fig. 3a). All objects are custom-designed and 3D printed to ensure controlled and reproducible conditions. We consider four real-world tasks: *cube red*, *cube blue*, *hang low*, and *hang high*. The *cube-X* tasks require picking up a colored cube from the table and placing it in a target bowl. The *hang-X* tasks involve grasping a mug and precisely hanging it on the corresponding branch of a mug stand. Task setups and illustrations are shown in Fig. 3.

Algorithm 1 FDP Training

```
Require: Dataset $\mathcal{D}$, Denoisers $\{\varepsilon_{\theta_i}\}$, $\textsc{Router}_\psi$
1: while not converged do
2:     Sample $(\mathbf{a}, \mathbf{o}) \sim \mathcal{D}$ and noise $\boldsymbol{\epsilon}^k$
3:     $\{w_i\} \leftarrow \textsc{Router}_\psi(\mathbf{o})$
4:     $\mathcal{L} \leftarrow \left\|\boldsymbol{\epsilon}^k - \sum_i w_i\, \varepsilon_{\theta_i}(\mathbf{a} + \boldsymbol{\epsilon}^k, \mathbf{o}, k)\right\|_2^2$
5:     $\forall i, \theta_i \leftarrow \theta_i + \nabla_{\theta_i}\mathcal{L}$
6:     $\psi \leftarrow \psi + \nabla_\psi \mathcal{L}$
7: end while
8: return $\{\varepsilon_{\theta_i}\}$
```

Algorithm 2 FDP Inference

```
Require: Denoisers $\{\varepsilon_i\}$, $\textsc{Router}$, Observation $\mathbf{o}_t$
1: $\{w_{t,i}\} \leftarrow \textsc{Router}(\mathbf{o}_t)$
2: $\mathbf{a}_t^K \leftarrow \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: for $k \leftarrow K, K-1, ..., 1$ do
4:     $\nabla \mathbf{a}^k \leftarrow \sum_i w_{t,i}\, \varepsilon_i(\mathbf{a}_t^k, \mathbf{o}_t, k)$
5:     $\mathbf{a}_t^{k-1} \leftarrow \mathbf{a}_t^k - \gamma_k \nabla \mathbf{a}^k + \mathcal{N}(\mathbf{0}, \sigma_k \mathbf{I})$
6: end for
7: $\mathbf{a}_t \leftarrow \mathbf{a}_t^0$
8: return $\mathbf{a}_t$
```

E. Implementation Details

Overview. All policies take RGB images and joint angles as input and predict trajectories of absolute joint angles. A history window of size 2 is used, and each model predicts a 16-step trajectory, from which the first 8 actions are executed. We adopt the DDPM framework [8] with 100 diffusion steps during both training and inference. All models are trained for the same number of gradient steps. No task identity is provided—there is no task-conditioning or task-specific routing in any policy.

Baselines. We compare against three existing approaches. DP [1] is a monolithic U-Net-based policy without modular structure. SDP [6] introduces a Mixture-of-Experts (MoE) architecture with expert selection based on observations. MoDE [7] also uses an MoE design but routes experts based on the diffusion noise level. For fair comparison, we remove task-specific routers from SDP and task-goal conditioning from MoDE, ensuring that all baselines rely solely on observations. We follow original architecture configurations: four experts per layer with two selected per forward pass, and proportionally reduce MoDE model size to match the others. For additional architectural and training details, we refer readers to the respective papers.

FDP. Our method, FDP, explicitly factorizes the policy into four U-Net-based diffusion components. These modules are trained jointly and composed through score aggregation using mixture weights generated by a two-layer MLP router conditioned on observations.

Adaptation. For the + *New Module* strategy, we adopt the upcycling approach [9], where new components are initialized by copying weights from randomly selected existing modules. In SDP and MoDE, adaptation involves adding two new upcycled expert blocks per MoE layer, FDP adds one new U-Net diffusion component.

F. Additional Results on Multitask Learning

We provide detailed results for multitask learning across all MetaWorld and RLBench tasks evaluated in our experiments. Table V shows per-task success rates and overall averages over 40 evaluation rollouts per task. While most baselines perform reasonably well on simpler tasks such as *door open* and *drawer open*, they tend to underperform on tasks that require more precise coordination or complex spatial reasoning (e.g., *peg insert*, *take umbrella out*). Our method, FDP, consistently achieves strong performance across all tasks, benefiting from its modular structure and better sub-skill decomposition.

In particular, on RLBench tasks, where action distributions are more multimodal and contact-rich, FDP outperforms others by a significant margin. This suggests improved capacity for modeling diverse behaviors and generalizing across complex task structures.

To further illustrate the advantages of our approach, Fig. 4 presents qualitative comparisons of real-world rollouts. Successful trials are shown in the top rows, while failure cases (bottom rows) highlight common errors made by baseline methods, such as imprecise end-effector poses and grasp failures. FDP consistently produces robust and accurate execution across task variants.

G. Additional Results on Task Adaptation

We report detailed task-wise results for adaptation experiments in both simulation and real-world settings.

Tables VI and VII present full adaptation results for both simulation and real-world settings. In simulation, FDP achieves the highest average success rate across both Meta-World and RLBench when using the + *New Module* adaptation strategy, indicating its superior ability to incorporate new task knowledge without disrupting prior components.

Notably, FDP achieves strong performance even when only the router and observation encoder are updated, outperforming SDP and MoDE by a significant margin. This implies that the component distributions in FDP are more reusable and interpretable than those in other modular methods, which often rely on entangled expert combinations.

In real-world experiments, we observe similar trends: FDP consistently performs better or matches the best baseline across various adaptation settings. These results highlight FDP 's practical advantages for continual learning and real-world deployment, where quick adaptation to novel tasks with limited supervision is critical.

H. Diffusion Components Analysis

We provide enlarged visualizations of rollout trajectories for each diffusion component in FDP to complement the compressed figure shown in the main paper (Fig. 5). As

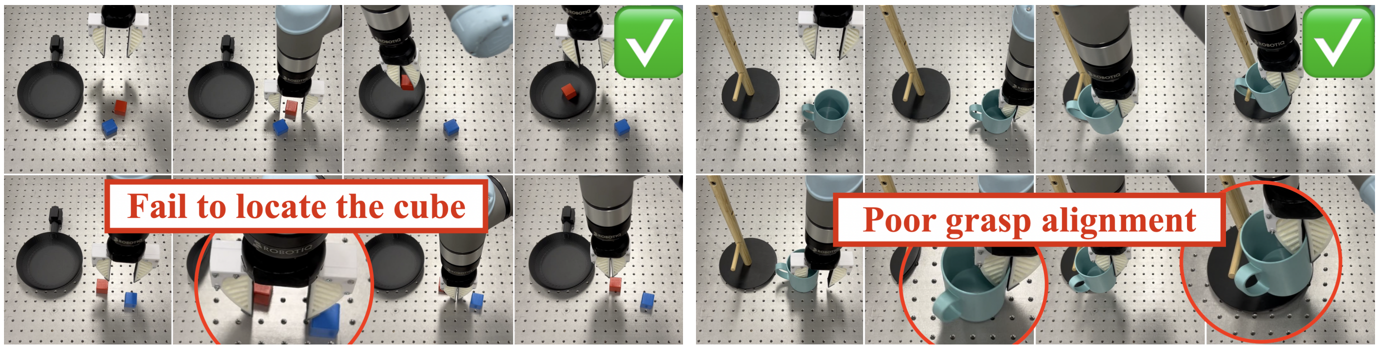


Fig. 7: **Qualitative real-world rollouts.** Each row shows execution for *cube-X* (left) and *hang-X* (right) tasks. Successes are shown in the top row; failure cases (bottom) illustrate challenges faced by baselines.

	MetaWorld						
Policy	Door Open	Drawer Open	Assembly	Window Close	Peg Insert	Hammer	Avg.
DP	0.900	1.000	1.000	0.950	0.250	0.250	0.725
SDP	0.900	1.000	1.000	1.000	0.200	0.200	0.717
MoDE	1.000	1.000	0.950	1.000	0.200	0.250	0.733
`FDP`	1.000	1.000	1.000	1.000	0.250	0.250	0.750

	RLBench				
Policy	Toilet Seat Up	Open Box	Open Drawer	Take Umbrella Out	Avg.
DP	0.500	0.825	0.800	0.100	0.569
SDP	0.425	0.775	0.025	0.050	0.319
MoDE	0.450	0.500	0.475	0.150	0.394
`FDP`	0.500	0.900	0.800	0.350	0.638

TABLE V: **Multitask learning evaluation on MetaWorld and RLBench**. We report average success rate over 40 samples.

		MetaWorld					**RLBench**		
Method	Policy	Door Close	Drawer Close	Disassemble	Window Open	Avg.	Toilet Seat Down	Close Box	Avg.
Full Parameter	DP	1.000	1.000	0.600	1.000	0.900	0.900	0.700	0.800
	SDP	0.933	1.000	0.633	1.000	0.892	0.950	0.575	0.763
	MoDE	1.000	1.000	0.667	1.000	0.917	0.925	0.700	0.813
	`FDP`	1.000	1.000	0.667	1.000	0.917	0.800	0.850	0.825
Router	SDP	0.733	0.000	0.000	0.067	0.200	0.200	0.025	0.113
	MoDE	0.000	0.000	0.000	0.267	0.067	0.100	0.000	0.050
	`FDP`	0.867	0.667	0.000	0.067	0.400	0.150	0.000	0.075
+ Observation Encoder	SDP	1.000	1.000	0.333	1.000	0.833	0.525	0.150	0.338
	MoDE	0.933	1.000	0.400	1.000	0.833	0.675	0.200	0.438
	`FDP`	1.000	0.933	0.500	1.000	0.858	0.650	0.275	0.463
+ New Module	SDP	1.000	1.000	0.600	1.000	0.900	0.675	0.225	0.450
	MoDE	1.000	1.000	0.633	1.000	0.908	0.825	0.375	0.600
	`FDP`	1.000	1.000	0.700	1.000	0.925	0.925	0.775	0.850

TABLE VI: **Adaptation evaluation on MetaWorld and RLBench**. Pretrained on tasks shown in Table V. We report average success rate on 60 MetaWorld samples and 40 RLBench samples.

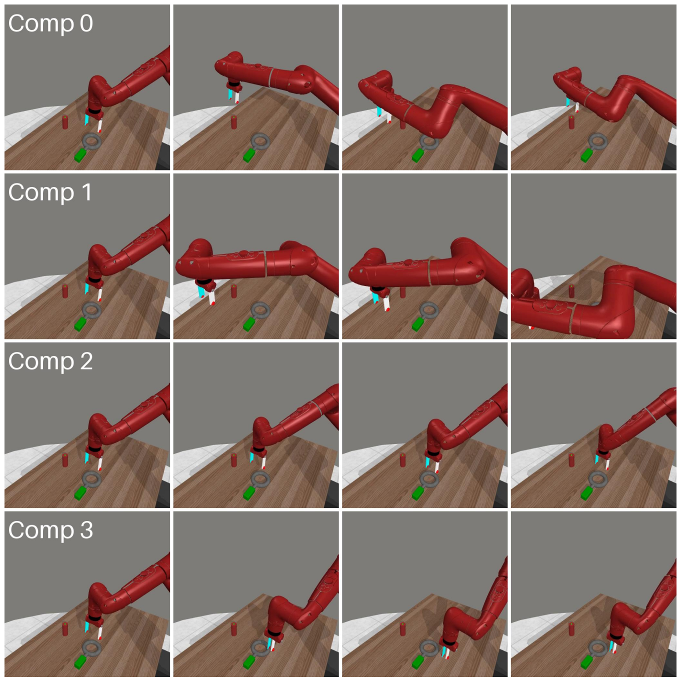
depicted in Fig. 8, each component learns to specialize in a distinct sub-skill, validating the modular structure of our approach.

In the *assembly* task, components 0 and 1 guide the robot to align with the assembly stand, component 2 adjusts the end-effector position to align with the ring, and component 3 performs the grasping motion. In the *hammer* task, components 0 and 1 coordinate to align and approach the pin, component 2 transitions the arm toward the hammer, and component 3 executes the grasp. Notably, each component displays temporally consistent behavior, and their combined outputs lead to successful task execution. This clear division of responsibility reflects meaningful behavioral decomposition, with each component contributing a targeted motion primitive.

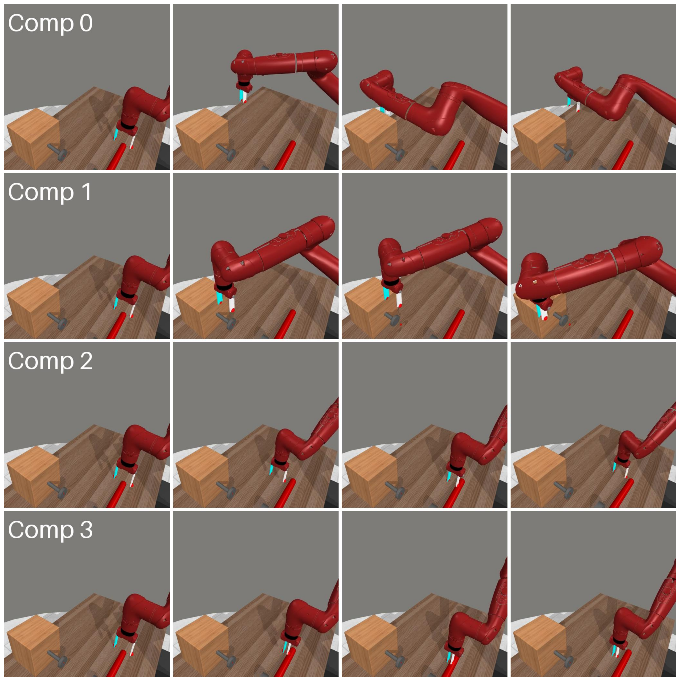
This visualization confirms that the learned components are functionally interpretable and behaviorally distinct. Unlike MoE-based baselines, where experts are reused across skills in entangled ways, our method enables disentangled, reusable skill modules.

I. Scaling of Number of Diffusion Components

We study how the number of diffusion components in `FDP` affects multitask performance. Experiments are conducted on selected tasks from MetaWorld (*door close*, *drawer close*,



(a) Assembly



(b) Hammer

Fig. 8: **Rollout trajectories of individual diffusion components** in FDP. **(a)** In *assembly*, components 0 and 1 align the robot with the stand, component 2 aligns with the ring, and component 3 executes the grasp. **(b)** In *hammer*, components 0 and 1 align and approach the pin, component 2 approaches the hammer, and component 3 performs the grasp. Checkout videos on factorized-diffusion-policy.github.io.

Method	Policy	Cube Blue	Hang High	Avg.
Full Parameter	DP	0.750	0.850	0.800
	SDP	0.700	0.800	0.750
	MoDE	0.750	0.800	0.775
	FDP	0.850	0.750	0.800
Router	SDP	0.000	0.100	0.050
	MoDE	0.100	0.050	0.075
	FDP	0.050	0.100	0.075
+ Observation Encoder	SDP	0.500	0.450	0.475
	MoDE	0.500	0.550	0.525
	FDP	0.550	0.550	0.550
+ New Module	SDP	0.650	0.550	0.600
	MoDE	0.700	0.650	0.675
	FDP	0.850	0.850	0.850

TABLE VII: **Adaptation evaluation in real-world**. Pretrained on tasks shown in Table II. We report average success rate on 20 samples.

# Comp	MetaWorld	RLBench
2	0.867	0.544
3	0.900	0.588
4	0.913	0.638
5	0.913	0.644
6	0.917	0.650
7	0.919	0.656

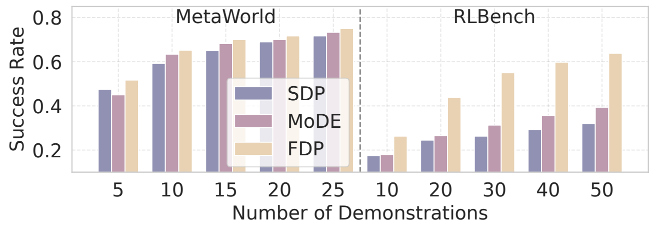
TABLE VIII: **Multitask performance of FDP with different numbers of components** on MetaWorld and RLBench. Performance improves up to 4 components and plateaus thereafter.

disassemble, *window open*) and RLBench (*toilet seat up*, *open box*, *open drawer*, *take umbrella out*). As shown in Table VIII, increasing the number of components from 2 to 4 consistently improves performance, indicating greater expressiveness and better sub-skill specialization. Beyond 4 components, performance plateaus, suggesting diminishing returns. Overall, 4 components provide a good trade-off between model complexity and performance for the tasks considered. We recommend using 4 components as a reasonable starting point, though some hyperparameter tuning may improve performance.

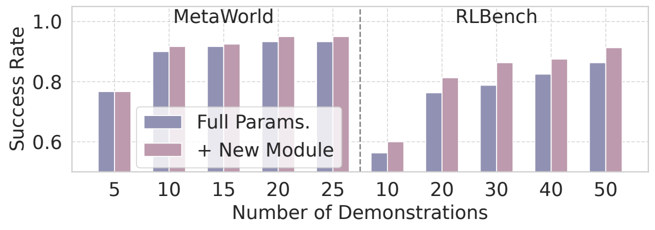
J. Scaling of Number of Demonstrations

Multitask Learning. We evaluate how FDP benefits from increasing amounts of demonstration data. As shown in Fig. 9a, performance improves steadily with more demonstrations. FDP consistently outperforms baselines, with particularly large gains on RLBench where complex, contact-rich interactions make effective decomposition especially valuable.

Task Adaptation. We analyze how adaptation performance scales with the number of demonstrations, and compare the proposed + *New Module* strategy with full-parameter fine-tuning. As shown in Fig. 9b, both strategies benefit from more data, but + *New Module* achieves comparable or better performance with even fewer demonstrations (on RLBench). This highlights the strength of our modular design in enabling data-efficient adaptation while avoiding the cost and potential catastrophic forgetting associated with updating all model parameters.



(a) Multitask Learning



(b) Task Adaptation

Fig. 9: **Performance scaling with number of demonstrations. (a)** Multitask learning success rate of FDP across tasks listed in Table I. **(b)** Task adaptation performance on tasks from Table III.

K. Training Convergence

We compare the training efficiency of FDP against MoDE and SDP by analyzing convergence curves on validation trajectories. Specifically, we track the mean squared error (MSE) loss used during diffusion training, measured over validation episodes across training epochs. Results are shown in Fig. 10 for both MetaWorld and RLBench tasks.

FDP consistently achieves lower validation MSE in fewer epochs, indicating faster convergence. MoDE converges more slowly, while SDP shows higher variance and slower reduction in loss, likely due to instability in expert selection and poor load balancing during training. These results support our claim that continuous score composition in FDP improves optimization stability compared to discrete Mixture-of-Expert (MoE) methods.

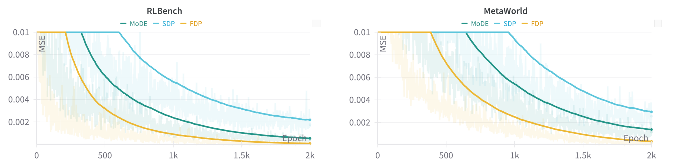


Fig. 10: **Training convergence curves**. Mean squared error (MSE) loss over training epochs for RLBench and MetaWorld tasks. FDP consistently converges faster and more stably than MoDE and SDP, indicating improved training efficiency and optimization stability.

L. Rollout Visualizations

We present additional qualitative results to illustrate the behavior of our policy FDP across diverse manipulation tasks in both simulation and real-world. Fig. 11 shows rollout sequences captured from different stages of task execution. These filmstrips visualize the robot's actions from the initial observation to successful task completion. Tasks involve reaching, aligning, grasping, and manipulating articulated or occluded objects. The qualitative rollouts reinforce the quantitative findings and further validate that compositional policy factorization leads to coherent, interpretable, and reusable behavior modules.

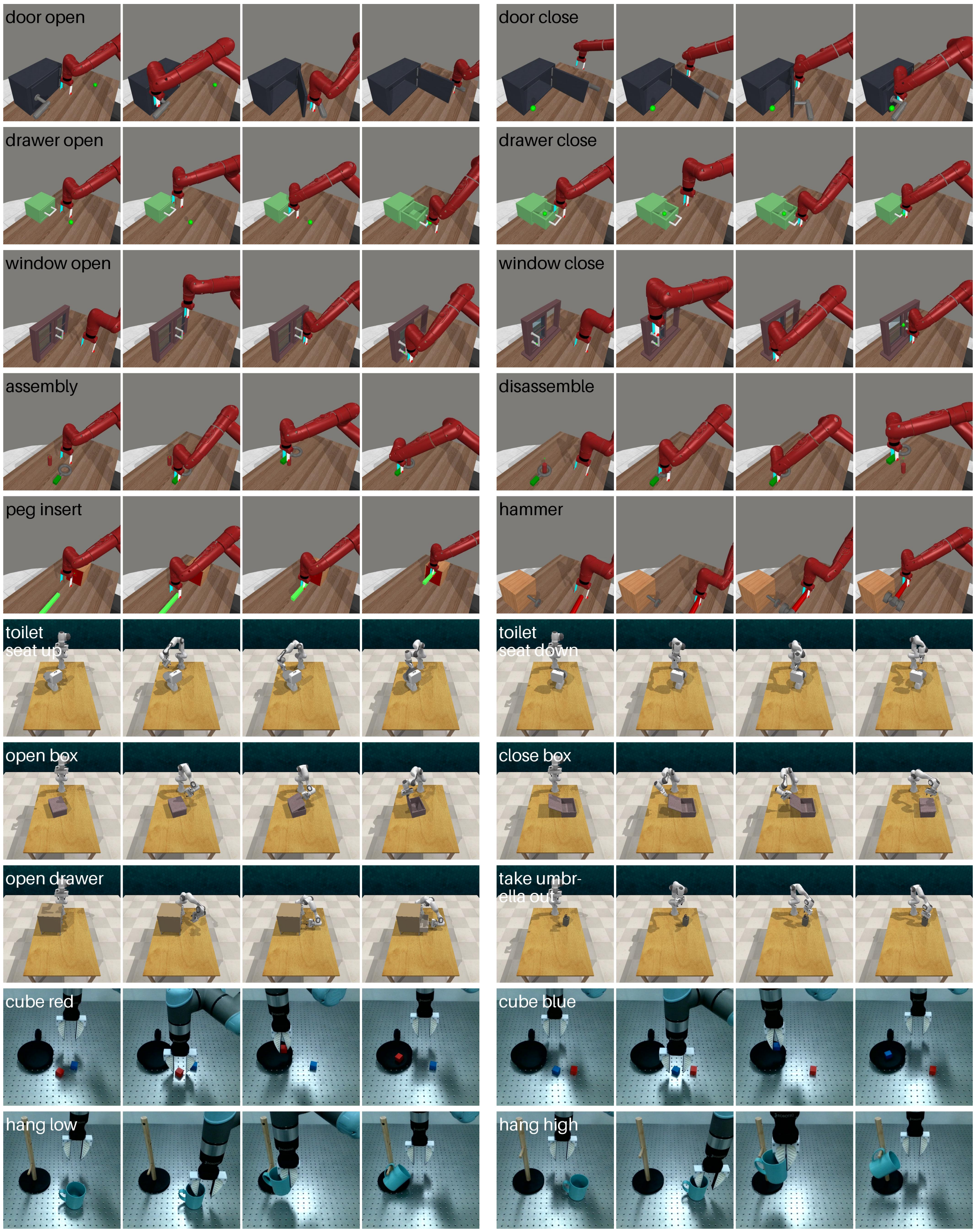


Fig. 11: Example rollout sequences for various tasks. Each filmstrip visualizes the trajectory execution of a policy on a specific manipulation task, highlighting the key interaction stages from initial observation to task completion. Checkout videos on factorized-diffusion-policy.github.io.