Learning Factorized Diffusion Policies for Conditional Action Diffusion

Omkar Patil[1], Prabin Kumar Rath[2], Kartikay Pangaonkar[3], Eric Rosen , and Nakul Gopalan[5]

[1,2,3,5]School of Computing and Augmented Intelligence, Arizona State University

***Abstract*—Diffusion models have emerged as a promising choice for learning robot skills from demonstrations. However, diffusion models are neither robust to visual distribution shifts nor sample-efficient for policy learning. In this work, we present 'Factorized Diffusion Policies' abbreviated as FDP, a novel theoretical framework to learn action diffusion models without the need to jointly condition on all observational modalities such as proprioception and vision. Using our factored approach leads to 10% absolute performance improvement for ten RLBench and four Adroit tasks when compared to a standard diffusion policy which jointly conditions on all modalities. Moreover, FDP results in 25% higher absolute performance across five RL Bench tasks with distribution shifts such as visual changes or distractors, where existing diffusion policies fail catastrophically. Our real-world experiments show that FDP is safe and relatively robust to deploy against visual distractors and appearance changes when compared to standard diffusion policies. Videos are available at https://fdp-policy.github.io/fdp-policy/.**

I. Introduction

Diffusion models have emerged as a promising choice for learning robot skills from demonstrations [5]. Following various diffusion models, several generative models originally proposed in the vision literature have been used for robot learning, exploiting properties such as one-step inference [48, 26] and multimodal priors [3]. However, unlike computer vision, conditioning is critical in robotics due to the numerous observational modalities that influence the robot's action choices. Humans prioritize different sensory modalities according to the specific requirements of the task [40]. Humans have also been shown to prioritize the more reliable modality between vision and haptics [11]. Naturally, based on the task, robot skills should also depend more strongly on certain observational modes than others. For instance, repetitive motions like dance are more likely to depend on the robot's proprioception, while search and rescue is conditioned strongly on its vision.

However, the current method of training diffusion policies jointly conditions the action diffusion process on all the observational modalities for every task [5]. This is a monolithic joint conditioning approach – "when all you have is a hammer, everything looks like a nail". Learning the full conditional action distribution

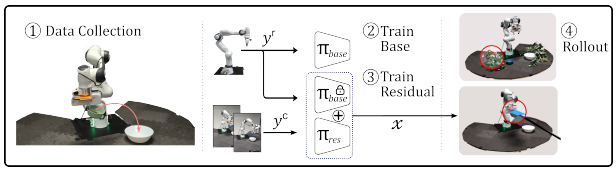

Fig. 1: Training and inference for learning visuomotor policies using FDP with vision as a residual over proprioception. FDP is robust to deployment with distractors and camera occlusions.

makes Diffusion Policies sensitive to distribution shifts in any of the modalities. We show that learning the full conditional results in low sample efficiency, brittleness to distribution shifts. In this work, we propose a novel *theoretical framework* 'Factorized Diffusion Policies' FDP for learning action diffusion models that decouples observational modalities for prioritization. At its core, FDP learns a *residual model* using some input modalities that have been omitted while training a base model with *prioritized inputs*. The base and residual model outputs are then composed to obtain samples from the full conditional action distribution. In addition, we present an architecture that enables efficient learning of the residual model in the FDP framework. We demonstrate that prioritization of modalities may yield significant gains in sample efficiency and naturally improves policy robustness to distribution shifts in the residual observations. Our contributions are as follows.

1) We introduce Factorized Diffusion Policies (FDP), a novel theoretical framework for training diffusion models on robot demonstration data that decouples observation modalities for prioritization. We derive a novel loss function for learning a residual model on top of a policy trained with prioritized modalities, and propose an efficient architectural implementation to ease its learning.

2) Our experiments show that prioritization of observational modalities produces significant sample efficiency gains in several RLBench [18] and Adroit hand manipulation [12] environments. We show through several distractor experiments on RLBench

that learning a visual residual model using FDP results in policies that are 25% more performant over standard diffusion policies.

3) We collect demonstrations across several task environments on a real robot and evaluate both FDP and standard diffusion policy in the original environments as well as in modified versions with visual distractors and appearance changes. In our real-world experiments, FDP outperforms diffusion policies by over 40% in the presence of distractors, occlusions and appearance changes.

II. Background and Related Work

Diffusion Models. Gaussian diffusion models [32] learn the reverse diffusion kernel $p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t)$ for a fixed forward kernel that adds Gaussian noise at each step $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathcal{I})$, such that $q(\boldsymbol{x}_T) \approx \mathcal{N}(0, \mathcal{I})$. Here, $t <= T$ is the diffusion time step and α_t is the noise schedule. For training the model, maximization of the evidence lower bound on the log-likelihood of the data distribution $\log q(\boldsymbol{x}_0)$ yields the commonly used loss function in Equation 1 [16, 22].

$$\mathcal{L}_t(\boldsymbol{\theta}) = \mathbb{E}_{\boldsymbol{x}_0 \sim q(\boldsymbol{x}_0),\, \boldsymbol{\epsilon}_0 \sim \mathcal{N}(0,\mathcal{I})} \left[\lambda_t [||\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)||_2^2]\right] \quad (1)$$

Here, λ_t, a function of α_t is the weighting parameter for different time steps, usually taken as 1 [16]. The model is trained to predict the noise $\boldsymbol{\epsilon}_0$ added to the data sample $\boldsymbol{x}_0$ to generate the noisy sample $\boldsymbol{x}_t$ taken as input to the network.

Connection to Score-based Models. Song et al. [36] presented a unified framework showing that both diffusion models [32, 16] and score-based models [34] can be interpreted as discretizations of different forward stochastic differential equations (SDEs). Denoising score matching (DSM) [38] is used to learn the score $\nabla_{\tilde{\boldsymbol{x}}} \log q_\sigma(\tilde{\boldsymbol{x}})$ at different noise scales σ required for sampling from the data distribution via the corresponding reverse-time SDEs [1]. Explicit Score Matching (ESM) [17, 38] was proposed to estimate this score by minimizing the Fisher divergence with the Gaussian-smoothed data distribution $q_\sigma(\tilde{\boldsymbol{x}}) = \int q(\boldsymbol{x})\mathcal{N}(\tilde{\boldsymbol{x}}; \boldsymbol{x}, \sigma^2 I) d\boldsymbol{x}$. DSM alleviates the computational difficulties of ESM [38, 35, 2], and is shown in Equation 2, where $s_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}})$ represents the learned score model.

$$\begin{aligned}\mathcal{J}_{\sigma_t}(\boldsymbol{\theta}) &\overset{\text{ESM}}{=} \mathbb{E}_{q_{\sigma_t}(\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|\nabla_{\tilde{\boldsymbol{x}}} \log q_{\sigma_t}(\tilde{\boldsymbol{x}}) - s_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}})\right\|_2^2\right] \\ &\overset{\text{DSM}}{=} \mathbb{E}_{q_{\sigma_t}(\boldsymbol{x},\tilde{\boldsymbol{x}})}\left[\frac{1}{2}\left\|\nabla_{\tilde{\boldsymbol{x}}} \log q_{\sigma_t}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) - s_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}})\right\|_2^2\right] + C\end{aligned}$$

Diffusion models use a forward transition kernel $q(\boldsymbol{x}_t|\boldsymbol{x}_0) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\bar{\alpha}_t}\boldsymbol{x}_0, (1-\bar{\alpha}_t)I)$ with discrete time and $\bar{\alpha}_i = \prod_{j=1}^{i} \alpha_j$, yielding the loss shown in Equation 1, while score-based model typically use $\mathcal{N}(\tilde{\boldsymbol{x}}; \boldsymbol{x}, \sigma_t^2 I)$, where α_t and σ_t are respective noise scales. Based on the equivalence of Equations 1 and 2, an optimal diffusion model learned using Equation 1, is related to the score of the diffused data distribution by $\boldsymbol{\epsilon}^*_{\boldsymbol{\theta}}(\boldsymbol{x}_t, t)/\sqrt{1-\bar{\alpha}_t} = \nabla_{\boldsymbol{x}_t} \log q(\boldsymbol{x}_t)$ [36, 22]. Typically, diffusion models generate samples via progressive denoising through the reverse diffusion process [16], while score-matching models sample from the data distribution using Langevin dynamics [30, 29].

Relevant work in Robotics. Sample efficiency and generalization are of primary importance in robotics, as scaling the collection of multimodal data is difficult and the number of variations of tasks is unbounded. While generative model families such as diffusion [5], score-based models [28], stochastic interpolants [3], and flows [48] have been applied in robotics, they do not address these limitations. Prior compositional works have tried to address these problems by composing learned constraints to generalize to new task combinations in manipulation [20] and planning [44], or composing distributions across heterogeneous modalities for tool use [42]. However, all the previous works compose learned or analytical distributions, limiting their application to combinations of existing solutions. Instead, in our FDP framework, we learn a residual over a base policy that, when composed with the base policy, provides samples corresponding to the data distribution. Recent augmentation-based methods [45, 4] improve generalization but add a substantial computational overhead and remain vulnerable to visual failures like temporary camera occlusions or dynamic scene changes. In contrast, FDP is an algorithmic improvement that achieves robustness to such perturbations without data augmentations as demonstrated in our real-world experiments.

III. Methodology

Assume that we have robot demonstrations $D = \{(\boldsymbol{x}, \boldsymbol{y})_i\}$ where $i = 1..N$, consisting of actions $\boldsymbol{x}$ and different observational modalities $\boldsymbol{y}^{1:M}$, such as images or point clouds from different cameras and proprioception data. We are interested in learning $p(\boldsymbol{x}|\boldsymbol{y})$ from the data such that given a task description, current camera images, state of the robot, and other observations, we can sample an action $\boldsymbol{x}$ with a high likelihood in the data distribution. Most treatments of diffusion models have been studied primarily in the context of single-modality distributions, such as those over image pixels [16, 33, 34]. This formulation has been directly adopted by the robotics community [5, 28, 21], leading to the optimization objective shown in Equation 3.

$$\mathcal{L}_t(\boldsymbol{\theta}) = \mathbb{E}_{(\boldsymbol{x}_0,\boldsymbol{y})\sim q(\boldsymbol{x}_0,\boldsymbol{y}),\, \boldsymbol{\epsilon}_0 \sim \mathcal{N}(0,\mathcal{I})} \left[||\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{y}, t)||_2^2\right] \quad (3)$$

Here, the network $\boldsymbol{\epsilon}_\theta$ is conditioned on the observation $\boldsymbol{y}$, and is trained to predict the noise added to the action sample $\boldsymbol{x}$. Although prior work in robotics adopts this conditional formulation [5], it is assumed without formal justification that the trained network maximizes the log-likelihood of the conditional distribution $p(\boldsymbol{x}|\boldsymbol{y})$. Hence, we present our first result as follows.

Lemma III.1. *The diffusion loss function $\mathcal{L}_t(\boldsymbol{\theta})$ as defined in Equation 3, in expectation over the time-steps $1 \leq t \leq T$, maximizes the variational lower bound on the log-likelihood of the conditional data distribution $\log q(\boldsymbol{x}|\boldsymbol{y})$, under a Markovian noising process $\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ and the conditional reverse transition kernel as $\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})$.*

The proof for Lemma III.1 is presented in Appendix C-B. In Equation 3, $\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, \boldsymbol{y}, t)$ arises from the reparametrization of the reverse transition kernel $q_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y}^{1:M})$, and from a score-based perspective, it learns the score of the full action conditional $\nabla_{\boldsymbol{x}_t} \log q(\boldsymbol{x}_t|\boldsymbol{y}^{1:M})$ times a constant. We argue that learning the full conditional directly is restrictive in several aspects of robot learning. Firstly, it necessitates the joint collection of the robot action and all observational modalities. Secondly, the model is vulnerable to even small distribution shifts in *any* modality. These shifts require a prohibitively large amount of data to address when the observation modalities are high-dimensional. Finally, among the multiple observation modalities it is hard to pinpoint the level of each mode's task dependent influence with limited data. Hence, we present FDP, a method to add structure and decouple observational modalities in the score of the full action conditional $\nabla_{\boldsymbol{x}_t} \log q(\boldsymbol{x}_t|\boldsymbol{y}^{1:M})$. By factorizing modalities using Bayes' theorem and learning residuals for subsequent terms, FDP effectively encodes task requirements and learns policies robust to distribution shifts in residual modalities.

A. Factorized Diffusion Policies

Let $\boldsymbol{y}^{1:k}$ be the prioritized observational modalities of the M total modalities, where $\boldsymbol{y}^{1:k} \equiv \boldsymbol{y}^1, .., \boldsymbol{y}^k$ and $1 \leq k < M$. To decouple the observational modalities, we utilize Bayes' theorem on the score of the full action conditional to obtain the following.

$$\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{y}^{1:M}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{y}^{1:k}; \boldsymbol{\theta}) + \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}^{k+1:M}|\boldsymbol{x}_t, \boldsymbol{y}^{1:k}; \boldsymbol{\phi})$$

To prioritize modalities $y^{1:k}$, we propose to first learn a diffusion policy π_{base}: $\boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, \boldsymbol{y}^{1:k}, t)$ that corresponds to the first score term on the right-hand side of Equation 4, times a constant. To learn the second score term, explicitly training a classifier $p(\boldsymbol{y}^{k+1:M}|\boldsymbol{x}_t, \boldsymbol{y}^{1:k})$ [6] is impractical due to the high dimensionality and continuity of observational modalities $\boldsymbol{y}^{1:M}$, such as images. Hence, we employ explicit score matching [17, 38] as shown in Equation 5.

$$D_F^t = \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:M})} \left[\frac{1}{2} \left\| \begin{array}{l} \nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}) \\ -\nabla_{\boldsymbol{x}_t} \log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}) \end{array} \right\|_2^2 \right]$$

Here, observational modalities $\boldsymbol{y}^{1:M}$ can be noised with a Gaussian kernel $\mathcal{N}(\tilde{\boldsymbol{y}}; \boldsymbol{y}, \tau^2 I)$ of variance τ^2 that is small enough such that $p_\tau(\tilde{\boldsymbol{y}}^i) \approx p(\boldsymbol{y}^i)$. Chao et al. [2] show that the empirical score is difficult to estimate for large datasets and derive the denoising likelihood score matching (DLSM) objective for conditional distributions, which forms the basis for our next result.

Theorem III.2. *Explicit score matching for $\nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k})$, as expressed in Equation 5 with $\boldsymbol{x}$ noised with the diffusion kernel $\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)\mathcal{I})$, is equivalent to the following loss:*

$$\mathcal{L}_{res}^t(\boldsymbol{\phi}) = \mathbb{E}_{\substack{\boldsymbol{x}_0, \boldsymbol{y}^{1:M}, \tilde{\boldsymbol{y}}^{1:M} \sim p_\tau \\ \boldsymbol{\epsilon}_0 \sim \mathcal{N}(0, \mathcal{I})}} \left[\frac{1}{2} \left\| \boldsymbol{\epsilon}_0 - \begin{array}{l} \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}, t) \\ -\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:M}, t) \end{array} \right\|_2^2 \right]$$

Here $\epsilon_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}, t)$ is fixed and optimal such that $\epsilon_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}, t) = \mathbb{E}[\epsilon_0|\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}]$ and estimates $\sqrt{1-\bar{\alpha_t}} * \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k})$. The parameterized model $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:M}, t)$ is learned to estimate $\sqrt{1-\bar{\alpha_t}} * \nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k})$.

The proof for Theorem III.2 is presented in Appendix C-C. Equation 6 allows us to learn the score of the classifier π_{res}: $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:M}, t)$ times a constant as a residual over frozen π_{base} to predict noise ϵ_0 added to the action x_0. Learning π_{res} as a residual of π_{base} ensures that the model does not overfit modalities $y_{k+1:M}$, but only learns correlations to bridge the gap between the expected score and the predicted score of the model π_{base} trained on the prioritized modalities $y^{1:k}$. Hence, policies learned in this factorized way are naturally robust to distribution shifts in the residual modalities. Moreover, explicit prioritization of $y^{1:k}$ by training π_{base} prior to learning the residual leads to sample efficiency, as the model learns correlations with the stronger modality without having to attend to other modalities. Since diffusion models are trained on discrete time steps, the residual is learned on the same time discretization as used for π_{base}. Once trained, actions can be sampled from the conditional distribution $p(\boldsymbol{x}|\boldsymbol{y}^{1:M})$ using reverse diffusion [16] on the composition [10] of π_{base} and π_{res}:

$$\begin{aligned} \boldsymbol{x}_{t-1} &\sim \mathcal{N}\left(\boldsymbol{x}_t; \tfrac{1}{\sqrt{\alpha_t}}\left(\boldsymbol{x}_t - \tfrac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}(\boldsymbol{x}_t, \boldsymbol{y}^{1:M}, t)\right), \sqrt{1-\alpha_t}\mathcal{I}\right) \\ \boldsymbol{\epsilon}(\boldsymbol{x}_t, \boldsymbol{y}^{1:M}, t) &= \boldsymbol{\epsilon_\theta}(\boldsymbol{x}_t, \boldsymbol{y}^{1:k}, t) + \boldsymbol{\epsilon_\phi}(\boldsymbol{x}_t, \boldsymbol{y}^{1:M}, t) \end{aligned} \quad (7)$$

The specific instantiations of FDP for combinations of modalities are presented in Appendix C-A. Note that the base mode π_{base}: $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{y}^{1:k}, t)$ can be further decomposed with respect to observational modalities. In this work, the residual model π_{res} is learned in expectation over data collected jointly for all modalities $y^{1:M}$, of which $y^{1:k}$ are used for training π_{base}. However, we emphasize that an important feature of our learning formulation is that it enables decoupled data collection for additional conditionals that could then be used to learn the residual. This potentially may alleviate some difficulties encountered for scaling coupled data in robotics and is left for future work.

B. Architectural Implementations of FDP

The base and residual models in FDP, denoted by π_{base} and π_{res}, can be instantiated using standard architectures such as UNet [31] or DiT [24]. FDP involves the additional step of learning π_{res} as a residual over a frozen π_{base}. During inference we compose the outputs of these as shown in Figure 2 [b]. However, we find this late-stage residual learning to be inefficient in practice and propose a more integrated way to compose π_{base} and π_{res}, as shown in Figure 2. This architecture enables a simplified training objective for the residual model, equivalent to Equation 3. Instead of learning a residual for the final score output, π_{res} learns the blockwise residual over the intermediate outputs of the frozen π_{base}. Specifically, let $\mathcal{F}^i_{\text{base}}$ and $\mathcal{F}^i_{\text{res}}$ denote the i-th DiT block outputs of the base and residual models, respectively. Then the composed output at level i can be written as $\mathcal{F}^i_{base}(\boldsymbol{x}', \boldsymbol{y}'^{1:k}) + \mathcal{Z}(\mathcal{F}^i_{res}(\boldsymbol{x}', \boldsymbol{y}'^{1:M}))$, where x' and $y'^{1:M}$ are layer inputs. Similar to Zhang et al. [47], $\mathcal{Z}$ is a zero-initialized layer to avoid harmful updates at the start of the training and to ensure that gradient updates to the residual model improve the predictions of the composed model over π_{base}. Crucially, we find that preserving the diversity of π_{base} is essential: overfitting the base model leaves little residual signal to learn, reducing generalization. Our experiments show that selecting the π_{base} checkpoint with the lowest validation loss provides a good foundation for residual learning. Our residual model is structured following the Vision Transformers architecture [9]. In π_{res}, all observational modalities are passed through self-attention layers after encoding. Our visual residual model encodes camera images into a single patch to reduce computational overhead. Complete implementation details and architectural ablations are provided in Appendix D and H respectively.

IV. Simulation Experiments

We train and evaluate FDP and related baselines in ten tasks of RLBench [18] and four tasks of Adroit [27] and Robomimic [23] each. More details in the Appendix E and our webpage https://fdp-policy.github.io/fdp-policy/.

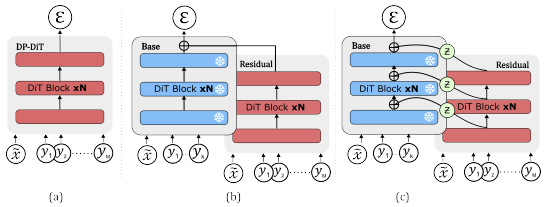

Fig. 2: Architectural representations for [a] diffusion policy that jointly conditions on all observational modalities, [b] simple FDP architecture that composes the score outputs from π_{base} and π_{res} and [c] FDP architecture with block-wise composition with a layer $\mathcal{Z}$ applied on π_{res}.

Baselines. For evaluation of sample efficiency in visuomotor tasks, we compare against several approaches that differ in the way in which they probabilistically model generative policy learning. However, for all approaches, we choose DiT-small ($\sim$90M) [24] as our model architecture. We implement Diffusion Policy [5] using DiT, referred to as DP-DiT in our results. For comparison, we also include UNet [31] implemented by Chi et al. [5] in our baselines as DP-UNet. We reformulate POCO [42] to compose the modalities of proprioception y^r and vision y^c. We train the motion π_{base} and vision π_{res} models independently, prior to sampling from the composed distribution [10] using $\boldsymbol{\epsilon}(\boldsymbol{x}_t, \boldsymbol{y}^r, \boldsymbol{y}^c, t) = \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t, \boldsymbol{y}^r, \boldsymbol{y}^c, t) + \lambda * \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{y}^r, t)$. Here, $\lambda = 0.1$ based on POCO's ablations [42]. We also report results for classifier-free guidance [15] as CFG, where we train a single model and switch out the vision modality with a probability of 0.2. We then sample using $\boldsymbol{\epsilon}(\boldsymbol{x}_t, \boldsymbol{y}^r, \boldsymbol{y}^c, t) = \lambda_1 * \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{y}^r, \boldsymbol{y}^c, t) + \lambda_2 * \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{y}^r, \boldsymbol{\phi}, t)$, where we set $\lambda_1 = 1.1$ and $\lambda_2 = 0.1$, as suggested by [15]. For real-world and distractor experiments in simulation, we compare against DP-DiT.

Research Question 1: Can task-specific prioritization of modalities using FDP lead to sample efficiency gains in learning visuomotor tasks? Prioritization of proprioception using the FDP framework outperforms all baselines in four tasks of RLBench across different number of demonstrations and three Adroit tasks as shown in Figure 3. In RLBench, FDP achieves 20% higher performance on average with 10 demonstrations and 10% higher performance on average with 100 demonstrations over the strongest baseline. FDP results in sample-efficient policies, especially with low number of demonstrations as the model is able to attend strongly to proprioception, only learning a residual for the visual observations. Note that a fully trained π_{base} motion model on 100 demonstrations fails in these tasks in isolation, implying that vision is required to complete these tasks

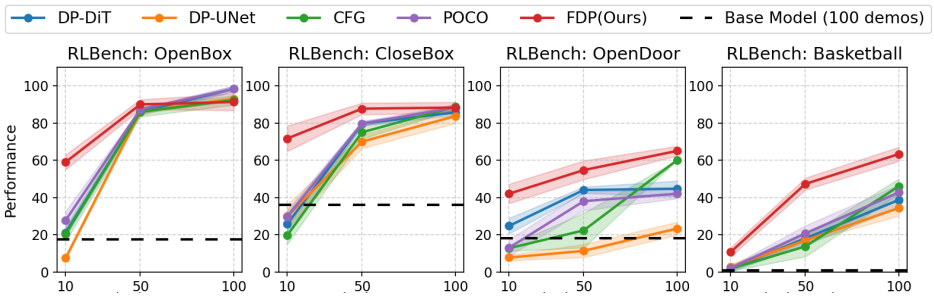

Fig. 3: Evaluation for FDP and baselines on four RLBench and four Adroit tasks. FDP results in sample-efficient policies at low number of demonstrations. More plots in Appendix F.

successfully. In the Adroit environments results shown in Appendix F, proprioception prioritized FDP performs better by $\sim$10% on average over DP-DiT for Door, Pen and Hammer tasks. For the Relocate task, which involves grasping a ball placed randomly on the table and relocating it to a random goal location, the robot action is strongly influenced by the environment state specifying the ball and the goal location. Hence, learning a motion π_{base} does not work in favor of improving policy performance, as the effect of the state of the environment is learned as a residual over π_{base}. Prioritization of proprioception will lead to sample-efficiency in repetitive tasks. Tasks that correlate heavily with robot proprioception are not uncommon as the robot is solving them in the first person view, and can move close to the object if required. More results on these and six other RLBench tasks in Appendix F.

Research Question 2: Does learning the visual modality as a residual in the FDP framework result in robustness to distractors and appearance changes? We present the results of the policy evaluations in the distractor environments in Table I.1. Both DP-DiT and FDP are trained on 100 demonstrations collected in the original environment and evaluated in three settings: the original environment, an environment augmented with distractors, and an environment with visual modifications to the manipulated objects. FDP significantly outperforms DP-DiT in both distractor-augmented and visually modified environments by more than 40%. Additionally, we further collect five demonstrations in each modified environment to investigate the benefits of few-shot adaptation to out-of-distribution data. Notably, FDP responds more effectively to additional demonstrations in the modified settings, improving it's performance by 15% on average over DP's 10%. In particular, FDP updates only the residual model π_{res} with new demonstrations, adjusting the conditional distribution on visual modalities $p(y^c|x, y^r)$ without modifying the full conditional action distribution $p(x|y^r, y^c)$. We extend this setting to point clouds and learn a visual residual on DP3 [46], as compared to DP3 with RGB inputs. Point clouds are sample-efficient for policy learning as they effectively encode the geometric structure of the scene in a single modality [46, 49, 19]. However, our distractor experiments show that FDP with a visual residual learned over DP3 is $\sim$ 20% more performant than DP3 with RGB inputs. Further experimental details are in Appendix F.

Research Question 3: How sensitive is the visuomotor task performance to prioritization of proprioception? In visuomotor tasks, prioritizing proprioception over learning the full conditional distribution can be advantageous when the object placement diversity is low or when robustness to visual distribution shifts is critical. To demonstrate this, we construct three RLBench environments for a block-picking task, each featuring an increasingly larger object placement area. The results are presented in Table I.2. As expected, FDP is sample-efficient and outperforms DP-DiT across all variation scales with only 10 demonstrations. However, with increasing number of demonstrations, DP-DiT eventually surpasses FDP at larger variation scales. Notably, DP-DiT still fails when visual distractors are introduced, whereas FDP remains robust and outperforms DP-DiT even at higher scales of task-variation in distractor-augmented environments. We also evaluated the performance of DP-DiT and FDP for fine-manipulation tasks on the Robomimic dataset. Although FDP is more sample efficient for the tasks of `Lift` and `Can`, it achieves a lower success rate than DP-DiT for `Square` and `Toolhang`, as shown in Table I.3. This is to be expected, as fine-manipulation tasks present a bottleneck in the joint state-action distribution, and FDP factorizes the distribution into components where some modalities are learned as residuals over the others.

V. Real-world Experiments

We evaluate FDP and the DP-DiT baseline across four real-world domains and report their task success rates. The domains are – *Close Drawer* as a simple task where the robot has to push the drawer; *Put Block in Bowl* that assesses the policy's ability to perform precise pick-and-place actions; *Pour in Bowl* to evaluate the

Table 1.1: Robustness to Visual Distractors (100 demos)
(FDP significantly improves generalization to visual changes)

Task	Environment	DP-DiT	FDP
OpenBox	Original	**98.3** ± 1.5	91.3 ± 4.5
	Zero-shot color	43.3 ± 2.5	**46.7**± 1.5
	5 demos color	34.7 ± 3.5	**76.7** ± 0.6
	Zero-shot distractors	1.7 ± 2.1	**16.7**± 2.1
	5 demos distractors	42.3 ± 4.0	**53.3** ± 2.3
Basketball in Hoop	Original	38.7 ± 4.2	**63.3** ± 3.8
	Zero-shot color	29.0 ± 8.7	**63.0** ± 1.0
	5 demos color	13.0 ± 0.0	**45.0** ± 2.6
	Zero-shot distractors	0.7 ± 1.2	**56.3** ± 3.2
	5 demos distractors	2.7 ± 1.2	**39.7** ± 4.9
Open Door	Original	44.7 ± 4.2	**65.0** ± 2.6
	Zero-shot color1	0.0 ± 0.0	**14.3** ± 3.1
	5 demos color1	17.7 ± 3.1	**52.0** ± 7.2
	Zero-shot color2	0.3 ± 0.6	**30.7** ± 2.5
	5 demos color2	20.0 ± 5.2	**53.7** ± 3.8

Table 1.2: Block Pick Success Rates
(FDP performs better in tasks with less variation.)

Variations	Model	10 demos	50 demos	Distractors
Small	FDP	**73.7** ± 3.8	**98.7** ± 1.5	**99.3** ± 0.6
	DP-DiT	29.7 ± 3.1	95.3 ± 3.2	0.0 ± 0.0
Medium	FDP	**21.3** ± 3.5	55.0 ± 2.6	**58.3** ± 3.1
	DP-DiT	12.0 ± 1.0	**69.0** ± 7.0	2.0 ± 1.0
Large	FDP	**6.3** ± 3.1	20.3 ± 3.5	0.7 ± 1.2
	DP-DiT	3.3 ± 0.6	**45.7** ± 7.1	0.0 ± 0.0

Table 1.3: Robomimic Lowdim Task Success Rates (100 demos)
(Evaluating FDP at precise and long-horizon manipulation.)

Task	DP-DiT	CFG	POCO	FDP
Lift	99.0 ± 1.7	98.7 ± 0.6	98.7 ± 1.5	99.7 ± 0.6
Can	99.0 ± 1.0	98.7 ± 1.5	98.7 ± 1.5	99.7 ± 0.6
Square	**80.3** ± 4.6	**80.0** ± 3.0	76.3 ± 6.0	58.0 ± 6.6
Toolhang	**60.0** ± 7.5	**60.7** ± 3.5	55.7 ± 7.5	45.7 ± 3.8

TABLE I: Tests for robustness and the effects of factorization across domains.

policy's dexterity in operating near joint limits and *Fold Towel* to assess effectiveness in manipulating deformable objects.

We collect 50 demonstrations per domain on a Franka FR3 robot using a 6D space mouse, recording both proprioceptive and visual observations from two cameras—one mounted on the gripper and a static camera covering the workspace. The trained policies are evaluated on four task variations in each domain: (a) `default`: an in-distribution setup matching the conditions used during demonstration collection; (b) `color`: the object's color is altered to test robustness to visual appearance changes; (c) `distractor`: novel, unseen objects such as vegetation props and soft toys are added to the scene to introduce clutter; and (d) `occlusion`: visual input is intermittently blocked during policy rollout to simulate partial observability. Figure 8 shows different task domains and their variations used in our experiments. More details on the robot system setup can be found in Appendix G. We use 10 rollouts in each experiment and report the task success rate as shown in Table II.

Task Domain	default		color		dist.		occl.	
	DP	**FDP**	**DP**	**FDP**	**DP**	**FDP**	**DP**	**FDP**
Close Drawer	**90**	**90**	**90**	**90**	10	**80**	0	**80**
Put Block in Bowl	**80**	**80**	0	**60**	0	**60**	10	**60**
Pour in Bowl	70	**80**	40	**80**	20	**60**	10	**50**
Fold Towel	40	**60**	40	**70**	30	**70**	10	**50**

TABLE II: Success rates (%) of Diffusion Policies (DP) and Factorized Diffusion Policies (FDP) across real-world tasks with 10 rollouts per condition.

Result Analysis. We find that FDP is robust to distribution shifts in the environment. DP regularly produces unachievable robot actions under `distractor` and `occlusion` settings, often triggering safety stops, resulting in task failure. In contrast, FDP guided by its motion prior, consistently generates stable actions even under severe occlusions and cluttered scenes, yielding an average absolute performance improvement of 40% over DP. In the `default` experiment we observe that the FDP policy outperforms DP in the pouring and towel-folding tasks, which require precise object manipulation. With just 50 demonstrations, DP overfits in these fine-grained tasks due to limited motion diversity, whereas FDP, leverages its residual guidance and effectively learns robust policies.

VI. Conclusion

We present Factorized Diffusion Policies (FDP), a novel theoretical framework for prioritization of observation modalities in policy learning. We provide probabilistic grounding for diffusion policy learning and reveal the pitfalls of learning a full conditional on all the observational modalities. FDP decouples the modalities and proposes a framework for their selective prioritization. We derive a novel loss function to realize the decoupling of modalities and support it with a novel architecture for efficient training. Through extensive experiments, we demonstrate several benefits of modality prioritization, including improved sample efficiency and increased robustness to visual distractors and camera occlusions when learning a residual for vision. FDP opens new avenues for future research, such as scalable integration of diversely sourced observational modalities for robot policy learning. Finally, our real-world experiments highlight that FDP maintains strong performance even under significant visual disruptions, outperforming diffusion policies by over 40%.

VII. Limitations

FDP is a theoretical framework to decouple observational modalities for robot policy learning. Predominantly, we present results for visuomotor tasks, but our method is generic and can be extended to other modalities. We see the following issues with our FDP framework –

A. Prioritizing Modalities

We focus on the benefits and pitfalls of prioritizing proprioception and alternatively learning a residual for vision in our experiments. However, for applications in a broader scope, the choice of modalities to be prioritized will need to be studied and is not answered in this work. Understanding which modality to prioritize for a particular task can be a challenging question for diffusion models in general and might be severly task dependent. This might indicate that there might be an inference time choice of composing modality that an agent might have to make.

B. Factorizing modalities is not a general solution for all tasks

As we show in our experiments, decoupling modalities may not be the right choice for every task or skill. This is because some tasks require the full joint distribution of observations. We also want to point out here that it is challenging to know whether a task requires the full joint distribution or the factored distribution in a specific prioritized order. Future approaches could learn to automatically attend on the right modality or the joint distribution, much like humans do.

C. Architectural Choices

Moreover, the framework can also benefit from further architectural improvements that realize a better trade-off between the strength of the guidance imparted by the residual model π_{res} and its robustness to perturbations in its inputs. The current training setup requires a two-step process for learning π_{base} and π_{res} that presents a time and computational overhead over training standard diffusion policies which might be cumbersome at deployment.

D. Baselines outside of Diffusion based policy models

In this work we only compare to diffusion based policy models as we are attempting to improve their robustness and extend their capabilities of factorization. However, a large scale comparison against different type of policy models is left to be done. For now we do not think comparisons against non-diffusion policy types is critical but it is desirable to understand when to use which type of policy for a robot.

E. Large Vision Action Models

There are large scale vision action models that can perform tasks specified by language in visual environments sometimes even zero shot. Here we are studying how to learn individual skills using diffusion based behavior cloning approaches. These skills might then be used in a larger stack of a vision-action model. The question of sample efficiency and robustness to distractors will always be important independent of the scale of the models themselves.

REFERENCES

[1] Brian DO Anderson. Reverse-time diffusion equation models. Stochastic Processes and their Applications, 12(3):313–326, 1982.

[2] Chen-Hao Chao, Wei-Fang Sun, Bo-Wun Cheng, Yi-Chen Lo, Chia-Che Chang, Yu-Lun Liu, Yu-Lin Chang, Chia-Ping Chen, and Chun-Yi Lee. Denoising likelihood score matching for conditional score-based data generation, 2022. URL https://arxiv.org/abs/2203.14206.

[3] Kaiqi Chen, Eugene Lim, Kelvin Lin, Yiyang Chen, and Harold Soh. Don't start from scratch: Behavioral refinement via interpolant-based policy diffusion. arXiv preprint arXiv:2402.16075, 2024.

[4] Zoey Chen, Sho Kiami, Abhishek Gupta, and Vikash Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. arXiv preprint arXiv:2302.06671, 2023.

[5] 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, page 02783649241273668, 2023.

[6] Prafulla Dhariwal and Alex Nichol. Diffusion models beat gans on image synthesis, 2021. URL https://arxiv.org/abs/2105.05233.

[7] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. Advances in neural information processing systems, 34:8780–8794, 2021.

[8] Carl Doersch. Tutorial on variational autoencoders. arXiv preprint arXiv:1606.05908, 2016.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929, 2020.

[10] 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, 2023.

[11] Marc O Ernst and Martin S Banks. Humans integrate visual and haptic information in a statistically optimal fashion. Nature, 415(6870):429–433, 2002.

[12] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

[13] Siddhant Haldar and Lerrel Pinto. Point policy: Unifying observations and actions with key points for robot manipulation. arXiv preprint arXiv:2502.20391, 2025.

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

[15] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

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

[17] Aapo Hyvärinen and Peter Dayan. Estimation of non-normalized statistical models by score matching. Journal of Machine Learning Research, 6(4), 2005.

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

[19] Tsung-Wei Ke, Nikolaos Gkanatsios, and Katerina Fragkiadaki. 3d diffuser actor: Policy diffusion with 3d scene representations. arXiv preprint arXiv:2402.10885, 2024.

[20] Weiyu Liu, Jiayuan Mao, Joy Hsu, Tucker Hermans, Animesh Garg, and Jiajun Wu. Composable part-based manipulation. arXiv preprint arXiv:2405.05876, 2024.

[21] Xiaokang Liu, Kevin Yuchen Ma, Chen Gao, and Mike Zheng Shou. Diffusion models in robotics: A survey. 2025.

[22] Calvin Luo. Understanding diffusion models: A unified perspective, 2022. URL https://arxiv.org/abs/2208.11970.

[23] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. arXiv preprint arXiv:2108.03298, 2021.

[24] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

[25] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In AAAI, 2018.

[26] Aaditya Prasad, Kevin Lin, Jimmy Wu, Linqi Zhou, and Jeannette Bohg. Consistency policy: Accelerated visuomotor policies via consistency distillation. arXiv preprint arXiv:2405.07503, 2024.

[27] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. arXiv preprint arXiv:1709.10087, 2017.

[28] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. arXiv preprint arXiv:2304.02532, 2023.

[29] Gareth O Roberts and Jeffrey S Rosenthal. Optimal scaling of discrete approximations to langevin diffusions. Journal of the Royal Statistical Society: Series B (Statistical Methodology), 60(1):255–268, 1998.

[30] Gareth O Roberts and Richard L Tweedie. Exponential convergence of langevin distributions and their discrete approximations. 1996.

[31] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Medical image computing and computer-assisted intervention–MICCAI 2015: 18th international conference, Munich, Germany, October 5-9, 2015, proceedings, part III 18, pages 234–241. Springer, 2015.

[32] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015.

[33] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models, 2022. URL https://arxiv.org/abs/2010.02502.

[34] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution, 2020.

[35] Yang Song and Diederik P. Kingma. How to train your energy-based models, 2021.

[36] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. arXiv preprint arXiv:2011.13456, 2020.

[37] Marcel Torne, Anthony Simeonov, Zechu Li, April Chan, Tao Chen, Abhishek Gupta, and Pulkit Agrawal. Reconciling reality through simulation: A real-to-sim-to-real approach for robust manipulation. arXiv preprint arXiv:2403.03949, 2024.

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

[39] Patrick von Platen, Suraj Patil, Anton Lozhkov, Pedro Cuenca, Nathan Lambert, Kashif Rasul, Mishig Davaadorj, Dhruv Nair, Sayak Paul, William Berman, Yiyi Xu, Steven Liu, and Thomas Wolf. Diffusers: State-of-the-art diffusion models. https://github.com/huggingface/diffusers, 2022.

[40] Basil Wahn and Peter König. Is attentional resource allocation across sensory modalities task-dependent? Advances in cognitive psychology, 13 (1):83, 2017.

[41] Dian Wang, Stephen Hart, David Surovik, Tarik Kelestemur, Haojie Huang, Haibo Zhao, Mark Yeatman, Jiuguang Wang, Robin Walters, and Robert Platt. Equivariant diffusion policy. arXiv preprint arXiv:2407.01812, 2024.

[42] Lirui Wang, Jialiang Zhao, Yilun Du, Edward H Adelson, and Russ Tedrake. Poco: Policy composition from and for heterogeneous robot learning. arXiv preprint arXiv:2402.02511, 2024.

[43] Jingyun Yang, Zi-ang Cao, Congyue Deng, Rika Antonova, Shuran Song, and Jeannette Bohg. Equibot: Sim (3)-equivariant diffusion policy for generalizable and data efficient learning. arXiv preprint arXiv:2407.01479, 2024.

[44] Zhutian Yang, Jiayuan Mao, Yilun Du, Jiajun Wu, Joshua B Tenenbaum, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Compositional diffusion-based continuous constraint solvers. arXiv preprint arXiv:2309.00966, 2023.

[45] Tianhe Yu, Ted Xiao, Austin Stone, Jonathan Tompson, Anthony Brohan, Su Wang, Jaspiar Singh, Clayton Tan, Jodilyn Peralta, Brian Ichter, et al. Scaling robot learning with semantically imagined experience. arXiv preprint arXiv:2302.11550, 2023.

[46] Yanjie Ze, Gu Zhang, Kangning Zhang, Chenyuan Hu, Muhan Wang, and Huazhe Xu. 3d diffusion policy: Generalizable visuomotor policy learning via simple 3d representations. arXiv preprint arXiv:2403.03954, 2024.

[47] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. Adding conditional control to text-to-image diffusion models, 2023. URL https://arxiv.org/abs/2302.05543.

[48] Qinglun Zhang, Zhen Liu, Haoqiang Fan, Guanghui Liu, Bing Zeng, and Shuaicheng Liu. Flowpolicy: Enabling fast and robust 3d flow-based policy via consistency flow matching for robot manipulation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 39, pages 14754–14762, 2025.

[49] Haoyi Zhu, Yating Wang, Di Huang, Weicai Ye, Wanli Ouyang, and Tong He. Point cloud matters: Rethinking the impact of different observation spaces on robot learning. Advances in Neural Information Processing Systems, 37:77799–77830, 2024.

[50] Yifeng Zhu, Abhishek Joshi, Peter Stone, and Yuke Zhu. Viola: Imitation learning for vision-based manipulation with object proposal priors. In Conference on Robot Learning, pages 1199–1210. PMLR, 2023.

APPENDIX A
OUTLINE OF THE APPENDIX

Contents

APPENDIX B
EXTENDED BACKGROUND AND RELATED WORKS

A. Classifier Guided Diffusion

Due to the relative abundance of unlabeled data such as images, diffusion or score-based models are trained to learn single-modality distributions such as those on image pixels [16, 33, 34]. Classifier guided diffusion [6] received considerable attention due to its ability to condition pre-trained generative models on class labels to sample images belonging specific categories. Using Bayes' theorem we can sample from a class $\boldsymbol{y}$ by decomposing the conditional score at time step t into the classifier gradient and the unconditional score $\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{y}; \boldsymbol{\theta}, \boldsymbol{\phi}) = \nabla_{x_t} \log p(\boldsymbol{x}_t; \boldsymbol{\theta}) + \nabla_{x_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t; \boldsymbol{\phi})$. However, classifier guided diffusion needs an explicit classifier trained on noisy samples to estimate the gradients $\nabla_{x_t} \log p(\boldsymbol{y}|\boldsymbol{x}_t; \boldsymbol{\phi})$.

B. Sample Efficiency In Robot Learning

Sample efficiency is of primary importance in robotics, as scaling multimodal data is difficult and the number of variation of tasks is unbounded. While generative model families such as diffusion [5], score-based models [28], stochastic interpolants [3], and flows [48] have been applied, they do not directly address this limitation. Although FDP factorizes observational modalities for diffusion policies, it is not limited to the choice of generative modeling family. Several works have specifically improved the sample efficiency of diffusion models. Ze et al. [46] use point clouds to show generalization in 3D space with fewer data, but rely on high-quality depth information. Several works exploit task-space symmetries by incorporating SO(2) [41] or SIM(3) [43] equivariance into diffusion models to boost sample efficiency; however, are limited in terms of task selection.

C. Compositional Robot Learning

Composition of diffusion models [10] has emerged as a promising framework for solving novel tasks by combining existing solutions. Prior work composes learned constraints to generalize to new task combinations in manipulation [20] and planning [44], or composes distributions across heterogeneous modalities for tool use [42]. However, all the previous works compose learned or analytical distributions, limiting their application to combinations of existing solutions. In the FDP framework, we instead learn a residual over an existing policy that, when composed together, results in samples corresponding to the data distribution.

D. Robustness to Visual Distractors

Existing behavioral cloning approaches [5] lack robustness to visual distractors and artifacts such as camera

viewpoint changes. Recent augmentation-based methods improve generalization by generating semantically modified images [45] or retargeting behaviors to novel situations [4], but they require substantial computational resources and remain vulnerable to visual failures like temporary camera blackouts or dynamic scene changes. In contrast, FDP with prioritized proprioception achieves robustness to such perturbations without requiring data augmentation, as demonstrated in our real-world experiments. Several recent works make policy learning robust to distractors by imposing constraints such as object and robot point tracking [13], building digital twins [37], and generating object proposals [50]. However, FDP is a novel reformulation of learning conditional action diffusion models, while being naturally robust, which may further benefit from these works.

APPENDIX C
THEORETICAL PROOFS

We prove the FDP loss for learning a residual model, presented as Equation 6. Our aim is to decouple the different observational modalities $\boldsymbol{y}^k$, $1 \leq k \leq M$ for task-based prioritization. Let π_{base} be a diffusion policy trained on k prioritized modalities over N demonstrations using the diffusion loss of Equation 3. Instead of learning a full action conditional, we propose to learn a residual over π_{base} for the subsequent modalities. The score of the conditional action distribution including the $k{+}1{:}M$ modalities $p(\boldsymbol{x}|\boldsymbol{y}^{1:M})$, where $\boldsymbol{y}^{1:k} \equiv \boldsymbol{y}^1, .., \boldsymbol{y}^k$, can be written using Bayes's theorem at diffusion timestep t as follows:

$$\begin{aligned}\nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{y}^{1:M};\boldsymbol{\theta},\boldsymbol{\phi}) = \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{y}^{k+1:M}|\boldsymbol{x}_t,\boldsymbol{y}^{1:k};\boldsymbol{\phi}) \\ + \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\boldsymbol{y}^{1:k};\boldsymbol{\theta})\end{aligned}$$

Here $\nabla_{x_t} \log p(\boldsymbol{x}_t|\boldsymbol{y}^{1:k};\boldsymbol{\theta})$ is the score of the model trained on the prioritized k observational modalities, while $\nabla_{x_t} \log p(\boldsymbol{y}^{k+1:M}|\boldsymbol{x}_t,\boldsymbol{y}^{1:k};\boldsymbol{\phi})$ corresponds to the score of the classifier for the modalities $\boldsymbol{y}^{k+1:M}$. The method proposed by classifier guided-diffusion [6] to explicitly train a classifier $p(\boldsymbol{y}^{k+1:M}|\boldsymbol{x}_t,\boldsymbol{y}^{1:k})$ on the noisy samples of $\boldsymbol{x}_t$ and $\boldsymbol{y}^{1:k}$ is impractical in our setting due to the high dimensionality and continuity of observational modalities such as images, which differ significantly from discrete class labels.

The central idea of this work is to instead directly parametrize the gradient of the classifier $\nabla_{x_t} \log p(\boldsymbol{y}^{k+1:M}|\boldsymbol{x}_t,\boldsymbol{y}^{1:k};\boldsymbol{\phi})$ using a neural network, which we refer to as the residual model π_{res}, rather than to learn a classifier and then obtain its gradients. To learn the score of the residual model, we employ score matching [35].

$$\begin{aligned}\mathcal{D}_F = \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\Bigg[\frac{1}{2}\Big\|\nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}) \\ - \nabla_{\boldsymbol{x}_t} \log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})\Big\|_2^2\Bigg] \qquad (9)\end{aligned}$$

Here, observational modalities $\boldsymbol{y}^{1:M}$ can be noised with a Gaussian kernel $\mathcal{N}(\tilde{\boldsymbol{y}};\boldsymbol{y},\tau^2 I)$ of variance τ^2 that is small enough such that $p_\tau(\tilde{\boldsymbol{y}}^i) \approx p(\boldsymbol{y}^i)$. This smoothing ensures that the resulting distribution has a continuous and differentiable density that satisfies the regularity conditions required for score matching [38]. However, Chao et al. [2] show that the true score is difficult to estimate for large datasets and derive the denoising likelihood score matching (DLSM) objective for conditional distributions, which forms the basis of theorem III.2, proved in Appendix C-C.

A. Instantiating FDP for Different Modalities

We now provide concrete implementations of FDP for visual and point-cloud tasks. In practice we observe that noising the modalities does not affect the performance, and drop the notation going forward. Existing diffusion models trained as visuomotor polices learn the score for the conditional distribution $p(\boldsymbol{x}|\boldsymbol{y}^r, \boldsymbol{y}^c)$, where $\boldsymbol{y}^r$ corresponds to proprioceptive observations and $\boldsymbol{y}^c$ correspond to visual observations from cameras. For proprioception-prioritized FDP, we decouple these observational modalities and instead learn the scores for a motion π_{base}: $p(\boldsymbol{x}|\boldsymbol{y}^r)$ and a vision π_{res}: $p(\boldsymbol{y}^c|\boldsymbol{x}, \boldsymbol{y}^r)$. The equations of diffusion loss for π_{base} and π_{res} can be written as follows.

$$\mathcal{L}^t_{base}(\boldsymbol{\theta}) = \mathbb{E}_{q(\boldsymbol{x}_0,\boldsymbol{y}^r)\mathcal{N}(\boldsymbol{\epsilon}_0;0,\mathcal{I})}\left[\|\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{y}^r, t)\|_2^2\right] \tag{10}$$

$$\mathcal{L}^t_{res}(\boldsymbol{\phi}) = \mathbb{E}_{\substack{\boldsymbol{x}_0,\boldsymbol{y}^r,\boldsymbol{y}^c\sim p_\tau \\ \boldsymbol{\epsilon}_0\sim\mathcal{N}(0,\mathcal{I})}}\left[\frac{1}{2}\left\| \begin{array}{l} \boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\boldsymbol{y}^c, \boldsymbol{x}_t, \boldsymbol{y}^r, t) \\ -\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, \boldsymbol{y}^r, t) \end{array} \right\|_2^2\right] \tag{11}$$

Similarly, we can learn a vision π_{res} for a point-cloud π_{base} as shown in Equation 13. Here, $\boldsymbol{y}^p$ denotes the point cloud modality. Note that we define a modality in terms of how it is encoded into the model. The images from different cameras may then correspond to different modalities, but are represented in Equations 11 and 13 as a single entity.

$$\mathcal{L}^t_{base}(\boldsymbol{\theta}) = \mathbb{E}_{\substack{\boldsymbol{x}_0,\boldsymbol{y}^r,\boldsymbol{y}^c\sim q \\ \boldsymbol{\epsilon}_0\sim\mathcal{N}(0,\mathcal{I})}}\left[\|\boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \boldsymbol{y}^r, \boldsymbol{y}^c, t)\|_2^2\right] \tag{12}$$

$$\mathcal{L}^t_{res}(\boldsymbol{\phi}) = \mathbb{E}_{\substack{\boldsymbol{x}_0,\boldsymbol{y}^r,\boldsymbol{y}^c,\boldsymbol{y}^p\sim p_\tau \\ \boldsymbol{\epsilon}_0\sim\mathcal{N}(0,\mathcal{I})}}\left[\frac{1}{2}\left\| \begin{array}{l} \boldsymbol{\epsilon}_0 - \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\boldsymbol{y}^p, \boldsymbol{x}_t, \boldsymbol{y}^r, \boldsymbol{y}^c, t) \\ -\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, \boldsymbol{y}^r, \boldsymbol{y}^c, t) \end{array} \right\|_2^2\right] \tag{13}$$

B. Proof of Diffusion Loss for Full Conditional Action Distribution

We show that a conditional diffusion process as defined by Dhariwal and Nichol [6] results in the loss of Equation 3 [5] being a maximizer of the variational lower bound on the log-likelihood of the conditional data distribution $\log q(\boldsymbol{x}|\boldsymbol{y})$.

Lemma C.1. *The diffusion loss function $\mathcal{L}_t(\boldsymbol{\theta})$ as defined in Equation 3, in expectation over the time-steps $1 \leq t \leq T$, maximizes the variational lower bound on the log-likelihood of the conditional data distribution $\log q(\boldsymbol{x}|\boldsymbol{y})$, under a Markovian noising process $\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1})$ and the conditional reverse transition kernel as $\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t, \boldsymbol{y})$.*

Here, we derive the diffusion loss function for the conditional distribution $p(\boldsymbol{x}|\boldsymbol{y})$ instead of only $p(\boldsymbol{x})$. A parallel derivation for conditional variational auto-encoders can be found in Doersch [8]. Following Dhariwal and Nichol [7], we start with a conditional Markovian noising forward process $\hat{q}$ similar to $q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) = \mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_{t-1}, (1-\alpha_t)\mathcal{I})$, and define the following:

$$\hat{q}(\boldsymbol{x}_0) := q(\boldsymbol{x}_0) \tag{14}$$

$$\hat{q}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{y}) := q(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t) \tag{15}$$

$$\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0, \boldsymbol{y}) := \prod_{t=1}^{T} \hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}) \tag{16}$$

We now reproduce some results that will be used later in the derivation of diffusion loss for conditional distributions. Dhariwal and Nichol [7] also show that

$$\hat{q}(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{x}_{t+1}) = \hat{q}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t, \boldsymbol{y})\frac{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_t)}{\hat{q}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)} \tag{17}$$

$$= \hat{q}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)\frac{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_t)}{\hat{q}(\boldsymbol{x}_{t+1}|\boldsymbol{x}_t)} \tag{18}$$

$$= \hat{q}(\boldsymbol{y}|\boldsymbol{x}_t) \tag{19}$$

Moreover, the unconditional reverse transition kernels can be shown to be equal using Bayes theorem, given Equations 14 and 15: $\hat{\boldsymbol{q}}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}) = \boldsymbol{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})$. Dhariwal and Nichol [7] use the result from Equation 19 to show the following for conditional reverse transition kernels.

$$\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1}, \boldsymbol{y}) = \frac{\hat{q}(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \boldsymbol{y})}{\hat{q}(\boldsymbol{x}_{t+1}, \boldsymbol{y})} \tag{20}$$

$$= \frac{\hat{q}(\boldsymbol{x}_t, \boldsymbol{x}_{t+1}, \boldsymbol{y})}{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_{t+1})\hat{q}(\boldsymbol{x}_{t+1})} \tag{21}$$

$$= \frac{\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})\hat{q}(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{x}_{t+1})\hat{q}(\boldsymbol{x}_{t+1})}{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_{t+1})\hat{q}(\boldsymbol{x}_{t+1})} \tag{22}$$

$$= \frac{\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})\hat{q}(\boldsymbol{y}|\boldsymbol{x}_t, \boldsymbol{x}_{t+1})}{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_{t+1})} \tag{23}$$

$$= \frac{q(\boldsymbol{x}_t|\boldsymbol{x}_{t+1})\hat{q}(\boldsymbol{y}|\boldsymbol{x}_t)}{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_{t+1})} \tag{24}$$

Further, we can show the following using Equations 15 and 16 and the Markovian noising process. It states that the joint distribution of the noised samples conditioned on $\boldsymbol{y}$ and $\boldsymbol{x}_0$ are the same for both $\hat{q}$ and q.

$$\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0, \boldsymbol{y}) = \prod_{t=1}^{T} \hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{y}) \tag{25}$$

$$= \prod_{t=1}^{T} q(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}) \tag{26}$$

$$= q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0) \tag{27}$$

We adapt the derivation of diffusion loss from Luo [22] to work with conditional distributions by maximizing the log-likelihood of the conditional data distribution $\log p(\boldsymbol{x}|\boldsymbol{y})$ leading to evidence lower bound (ELBO).

$$\log p(\boldsymbol{x}|\boldsymbol{y}) = \log \int p(\boldsymbol{x}_{0:T}|\boldsymbol{y}) d\boldsymbol{x}_{1:T} \tag{28}$$

$$= \log \int \frac{p(\boldsymbol{x}_{0:T}|\boldsymbol{y})\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0, \boldsymbol{y})}{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0, \boldsymbol{y})} d\boldsymbol{x}_{1:T} \tag{29}$$

$$= \log \mathbb{E}_{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\left[\frac{p(\boldsymbol{x}_{0:T}|\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\right] \tag{30}$$

$$\geq \mathbb{E}_{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{p(\boldsymbol{x}_{0:T}|\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\right] \tag{31}$$

The ELBO can be further simplified as follows

$$\begin{aligned}
\log p(\boldsymbol{x}|\boldsymbol{y}) &\geq \mathbb{E}_{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{p(\boldsymbol{x}_{0:T}|\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\right] \\
&= \mathbb{E}_{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{p(\boldsymbol{x}_T|\boldsymbol{y})\prod_{t=1}^{T} p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y})}{\prod_{t=1}^{T}\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{y})}\right] \\
&= \mathbb{E}_{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{p(\boldsymbol{x}_T|\boldsymbol{y})p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_1|\boldsymbol{x}_0,\boldsymbol{y})}\right. \\
&\quad \left. + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_{t-1},\boldsymbol{x}_0,\boldsymbol{y})}\right] \\
&= \mathbb{E}_{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{p(\boldsymbol{x}_T|\boldsymbol{y})p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_1|\boldsymbol{x}_0,\boldsymbol{y})}\right. \\
&\quad \left. + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y})}{\frac{\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y})\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_0,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_0,\boldsymbol{y})}}\right] \\
&= \mathbb{E}_{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{p(\boldsymbol{x}_T|\boldsymbol{y})p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_1|\boldsymbol{x}_0,\boldsymbol{y})}\right. \\
&\quad \left. + \log \frac{\hat{q}(\boldsymbol{x}_1|\boldsymbol{x}_0,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_T|\boldsymbol{x}_0,\boldsymbol{y})} + \log \prod_{t=2}^{T} \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y})}\right] \\
&= \mathbb{E}_{\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{p(\boldsymbol{x}_T|\boldsymbol{y})p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_T|\boldsymbol{x}_0,\boldsymbol{y})}\right. \\
&\quad \left. + \sum_{t=2}^{T} \log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y})}\right] \\
&= \mathbb{E}_{\hat{q}(\boldsymbol{x}_1|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1,\boldsymbol{y})\right] \\
&\quad + \mathbb{E}_{\hat{q}(\boldsymbol{x}_T|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{p(\boldsymbol{x}_T|\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_T|\boldsymbol{x}_0,\boldsymbol{y})}\right] \\
&\quad + \sum_{t=2}^{T} \mathbb{E}_{\hat{q}(\boldsymbol{x}_t,\boldsymbol{x}_{t-1}|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y})}{\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y})}\right] \\
&= \underbrace{\mathbb{E}_{\hat{q}(\boldsymbol{x}_1|\boldsymbol{x}_0,\boldsymbol{y})}\left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_0|\boldsymbol{x}_1,\boldsymbol{y})\right]}_{\text{reconstruction term}} \\
&\quad - \underbrace{D_{\mathrm{KL}}(\hat{q}(\boldsymbol{x}_T|\boldsymbol{x}_0,\boldsymbol{y}) \parallel p(\boldsymbol{x}_T|\boldsymbol{y}))}_{\text{prior matching term}} \\
&\quad - \sum_{t=2}^{T} \underbrace{\mathbb{E}_{\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_0,\boldsymbol{y})}[}_{\text{denoising matching term}} \\
&\qquad \underbrace{D_{\mathrm{KL}}(\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y}))]}
\end{aligned} \tag{32}$$

The reconstruction term is ignored for training [16, 22], and the prior matching term does not have any trainable parameters. We further simplify the denoising matching term using Equation 24 further conditioned on $\boldsymbol{x}_0$.

$$\begin{aligned}
&- \sum_{t=2}^{T} \mathbb{E}_{\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_0,\boldsymbol{y})}\left[D_{\mathrm{KL}}(\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y}) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y}))\right] \\
&= - \sum_{t=2}^{T} \mathbb{E}_{\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_0,\boldsymbol{y})}\left[\mathbb{E}_{\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y})\right.\right. \\
&\quad \left.\left. - \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y})\right]\right] \\
&= - \sum_{t=2}^{T} \mathbb{E}_{\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_0,\boldsymbol{y})}\left[\mathbb{E}_{\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0)\right.\right. \\
&\quad \left.\left. + \log \frac{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_{t-1},\boldsymbol{x}_0)}{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_t,\boldsymbol{x}_0)} - \log p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y})\right]\right] \\
&= - \sum_{t=2}^{T} \mathbb{E}_{\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_0,\boldsymbol{y})}\left[D_{\mathrm{KL}}(\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0) \parallel p_{\boldsymbol{\theta}}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{y}))\right] \\
&\quad - \sum_{t=2}^{T} \mathbb{E}_{\hat{q}(\boldsymbol{x}_t|\boldsymbol{x}_0,\boldsymbol{y})}\left[\mathbb{E}_{\hat{q}(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t,\boldsymbol{x}_0,\boldsymbol{y})}\left[\log \frac{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_{t-1},\boldsymbol{x}_0)}{\hat{q}(\boldsymbol{y}|\boldsymbol{x}_t,\boldsymbol{x}_0)}\right]\right]
\end{aligned} \tag{33}$$

Note that the expectation is taken over a distribution independent of $\boldsymbol{y}$, since $\hat{q}(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0,\boldsymbol{y}) = q(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0)$, as shown in Equation 27. It is easy to see that the first term in the resulting expression is the KL divergence between the model parameterized with the condition $\boldsymbol{y}$ and the unconditional reverse transition kernel, leading to the popularly used diffusion loss of Equation 3. However, an additional term is introduced for the conditional diffusion process. This minimizes the difference in the likelihood of the labels between consecutive denoising steps. However, since it does not have trainable parameters, we will ignore it.

C. Proof for FDP Loss

Theorem C.2. *Explicit score matching for $\nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k})$, as expressed in Equation 5 with $\boldsymbol{x}$ noised with the diffusion kernel $\mathcal{N}(\boldsymbol{x}_t; \sqrt{\alpha_t}\boldsymbol{x}_0, (1-\alpha_t)\mathcal{I})$, is equivalent to the following loss:*

$$\mathcal{L}^t_{res}(\boldsymbol{\phi}) = \mathbb{E}_{\substack{\boldsymbol{x}_0,\boldsymbol{y}^{1:M},\tilde{\boldsymbol{y}}^{1:M}\sim p_\tau \\ \boldsymbol{\epsilon}_0\sim\mathcal{N}(0,\mathcal{I})}}\left[\frac{1}{2}\left\| \begin{array}{l} \boldsymbol{\epsilon}_0 - \boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}, t) \\ +\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:M}, t) \end{array} \right\|^2_2\right] \tag{34}$$

Here $\epsilon_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}, t)$ is fixed and optimal such that $\epsilon_{\boldsymbol{\theta}}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}, t)$=$\mathbb{E}[\epsilon_0|\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}]$ and estimates $\sqrt{1-\bar{\alpha}_t} * \nabla_{\boldsymbol{x}_t} \log p(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k})$. The parameterized model $\hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:M}, t)$ is learned to estimate $\sqrt{1-\bar{\alpha}_t} * \nabla_{\boldsymbol{x}_t} \log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k})$.

Chao et al. [2] in their insightful work for score-based models, show that the following two losses differ only by a constant.

$$\mathcal{D}_F(p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{x}}) \| p_{\alpha,\tau}(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{x}})) = \mathbb{E}_{p_{\alpha,\tau}(\tilde{\boldsymbol{x}},\tilde{\boldsymbol{y}})}\left[\frac{1}{2}\left\| \begin{array}{l} \nabla_{\tilde{\boldsymbol{x}}} \log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{x}}) \\ -\nabla_{\tilde{\boldsymbol{x}}} \log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{x}}) \end{array} \right\|^2_2\right] \tag{35}$$

$$\mathcal{L}_{DLSM}(\boldsymbol{\phi}) = \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x},\tilde{\boldsymbol{x}},\boldsymbol{y},\tilde{\boldsymbol{y}})}\left[\frac{1}{2}\left\| \begin{array}{l} \nabla_{\tilde{\boldsymbol{x}}} \log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}|\tilde{\boldsymbol{x}}) + \nabla_{\tilde{\boldsymbol{x}}} \log p_{\boldsymbol{\theta}}(\tilde{\boldsymbol{x}}) \\ -\nabla_{\tilde{\boldsymbol{x}}} \log p_{\alpha}(\tilde{\boldsymbol{x}}|\boldsymbol{x}) \end{array} \right\|^2_2\right] \tag{36}$$

We extend their proof for diffusion models and multiple conditionals below. Explicit Score Matching loss between the residual model and the true score of the classifier can be further expanded as:

$$D_F^t(p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})||p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})) \tag{37}$$

$$= \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\Big[\frac{1}{2}||\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}) - \nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})||_2^2\Big] \tag{38}$$

$$= \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\Big[\frac{1}{2}||\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})||_2^2\Big] + \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\Big[\frac{1}{2}||\nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})||_2^2\Big] - \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\big[\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})\rangle\big] \tag{39}$$

$$= \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\Big[\frac{1}{2}||\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})||_2^2\Big] + \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\Big[\frac{1}{2}||\nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})||_2^2\Big] - \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\big[\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k},\tilde{\boldsymbol{y}}^{k+1:M})\rangle - \nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k})\rangle\big] \tag{40}$$

$$= \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\Big[\frac{1}{2}||\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})||_2^2\Big] + \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\Big[\frac{1}{2}||\nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})||_2^2\Big] + \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\big[\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k})\rangle\big] \underbrace{-\mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\big[\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}),}_{\text{Term 1}} \nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k},\tilde{\boldsymbol{y}}^{k+1:M})\rangle\big] \tag{41}$$

Simplifying the Term 1 further:

$$\begin{aligned}
&-\mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\big[\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:M})\rangle\big] \\
&= -\int_{\boldsymbol{x}_t}\int_{\tilde{\boldsymbol{y}}^{1:M}} p_\tau(\tilde{\boldsymbol{y}}^{1:M})p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:M}) \langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \frac{\nabla_{\boldsymbol{x}_t}p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:M})}{p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:M})}\rangle d\tilde{\boldsymbol{y}}^{1:M}d\boldsymbol{x}_t \\
&= -\int_{\boldsymbol{x}_t}\int_{\tilde{\boldsymbol{y}}^{1:M}} p_\tau(\tilde{\boldsymbol{y}}^{1:M})\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\int_{\boldsymbol{x}_0} p_{0,\tau}(\boldsymbol{x}_0|\tilde{\boldsymbol{y}}^{1:M})p_{\alpha,\tau}(\boldsymbol{x}_t|\boldsymbol{x}_0,\tilde{\boldsymbol{y}}^{1:M})d\boldsymbol{x}_0\rangle d\tilde{\boldsymbol{y}}^{1:M}d\boldsymbol{x}_t \\
&= -\int_{\boldsymbol{x}_t}\int_{\tilde{\boldsymbol{y}}^{1:M}} p_\tau(\tilde{\boldsymbol{y}}^{1:M})\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\int_{\boldsymbol{x}_0}\int_{\boldsymbol{y}^{1:M}} p_{0,\tau}(\boldsymbol{x}_0|\tilde{\boldsymbol{y}}^{1:M})p_{\alpha,\tau}(\boldsymbol{x}_t|\boldsymbol{x}_0,\tilde{\boldsymbol{y}}^{1:M},\boldsymbol{y}^{1:M}) \cdot p_(\boldsymbol{y}^{1:M}|\boldsymbol{x}_0,\tilde{\boldsymbol{y}}^{1:M})d\boldsymbol{y}^{1:M}d\boldsymbol{x}_0\rangle d\tilde{\boldsymbol{y}}^{1:M}d\boldsymbol{x}_t \\
&= -\int_{\boldsymbol{x}_t}\int_{\tilde{\boldsymbol{y}}^{1:M}}\int_{\boldsymbol{x}_0}\int_{\boldsymbol{y}^{1:M}} p_\tau(\boldsymbol{x}_0,\boldsymbol{x}_t,\boldsymbol{y}^{1:M},\tilde{\boldsymbol{y}}^{1:M}) \langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}),
\end{aligned}$$

$$\begin{aligned}
&\nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\boldsymbol{x}_0,\tilde{\boldsymbol{y}}^{1:M},\boldsymbol{y}^{1:M})\rangle d\boldsymbol{y}^{1:M}d\boldsymbol{x}_0 d\tilde{\boldsymbol{y}}^{1:M}d\boldsymbol{x}_t \\
&= -\mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_0,\boldsymbol{x}_t,\boldsymbol{y}^{1:M},\tilde{\boldsymbol{y}}^{1:M})}\big[\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\log p_\alpha(\boldsymbol{x}_t|\boldsymbol{x}_0)\rangle\big]
\end{aligned}$$

Plugging this back into Equation 41, we get-

$$\begin{aligned}
&\mathcal{D}_F\left(p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})\middle\|p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})\right) \\
&= \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})\right\|_2^2\right] + \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})\right\|_2^2\right] \\
&\quad + \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\left[\left\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k})\right\rangle\right] \\
&\quad - \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_0,\boldsymbol{x}_t,\boldsymbol{y}^{1:M},\tilde{\boldsymbol{y}}^{1:M})}\left[\left\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\log p_\alpha(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\rangle\right]
\end{aligned} \tag{42}$$

Here, $\mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\left[\frac{1}{2}||\nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})||_2^2\right]$ is a constant. Further, adding the constant $\mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})}\left[\frac{1}{2}||\nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k}) - \nabla_{\boldsymbol{x}_t}\log p_\alpha(\boldsymbol{x}_t|\boldsymbol{x}_0)||_2^2\right]$ to Equation 42, we get:

$$\begin{aligned}
&\mathcal{D}_F\left(p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})\,\|\,p_{\alpha,\tau}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})\right) \\
&= \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})\right\|_2^2\right] + \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:M})}\left[\left\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k})\right\rangle\right] \\
&\quad - \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_0,\boldsymbol{x}_t,\boldsymbol{y}^{1:M},\tilde{\boldsymbol{y}}^{1:M})}\left[\left\langle\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}), \nabla_{\boldsymbol{x}_t}\log p_\alpha(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\rangle\right] \\
&\quad + \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k})}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t}\log p_{\alpha,\tau}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k}) - \nabla_{\boldsymbol{x}_t}\log p_\alpha(\boldsymbol{x}_t|\boldsymbol{x}_0)\right\|_2^2 + C\right]
\end{aligned} \tag{43}$$

$$= \mathbb{E}_{p_{\alpha,\tau}(\boldsymbol{x},\boldsymbol{x}_t,\boldsymbol{y}^{1:M},\tilde{\boldsymbol{y}}^{1:M})}\left[\frac{1}{2}\left\|\nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}|\boldsymbol{x}_t,\tilde{\boldsymbol{y}}^{1:k}) + \nabla_{\boldsymbol{x}_t}\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_t|\tilde{\boldsymbol{y}}^{1:k}) - \nabla_{\boldsymbol{x}_t}\log p_\alpha(\boldsymbol{x}_t|\boldsymbol{x})\right\|_2^2\right] + C \tag{44}$$

Simplifying $\nabla_{\boldsymbol{x}_t}\log p_\alpha(\boldsymbol{x}_t|\boldsymbol{x})$ to $-\boldsymbol{\epsilon}_0/\sqrt{1-\bar{\alpha}_t}$, where $\boldsymbol{\epsilon}_0 \sim \mathcal{N}(0,\mathcal{I})$, and replacing the scores multiplied to $\sqrt{1-\bar{\alpha}_t}$ with their parametrized models we obtain:

$$\mathcal{L}_{res}^t(\boldsymbol{\phi}) = \mathbb{E}_{p_\tau(\boldsymbol{x},\boldsymbol{y}^{1:M},\tilde{\boldsymbol{y}}^{1:M})}\mathbb{E}_{\boldsymbol{\epsilon}_0\sim\mathcal{N}(0,\mathcal{I})}\left[\frac{1}{2(1-\bar{\alpha}_t)}\middle\|\boldsymbol{\epsilon}_0\right.$$

$$-\boldsymbol{\epsilon}_\theta(\boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}, t) + \hat{\boldsymbol{\epsilon}}_{\boldsymbol{\phi}}(\tilde{\boldsymbol{y}}^{k+1:M}, \boldsymbol{x}_t, \tilde{\boldsymbol{y}}^{1:k}, t)\Big\|_2^2\Big] + C \tag{45}$$

Here $1/(1-\bar{\alpha_t})$ is a constant and does not affect the optimization objective at time t. Hence, we show that the Explicit Score Matching loss in Equation 5 is equivalent to minimizing the loss $\mathcal{L}^t_{res}$ in Theorem III.2, differing up to a multiplicative and additive constant.

APPENDIX D
ARCHITECTURE AND IMPLEMENTATION DETAILS

All transformer-based models are trained over 2000 epochs for visual tasks and 3000 epochs for low-dimensional tasks. Unet [31] is trained over 3000 epochs for visual tasks and 5000 epochs for low-dimensional tasks. We train models on visual tasks with a batch size of 64, and low-dimensional tasks with a batch size of 256. All models are trained on NVIDIA A5000 or A40 GPUs, with training times ranging from 6 to 12 hours depending on model size and the number of camera inputs. Our current implementations support an action prediction latency of $\sim$50ms for DP-DiT, $\sim$100ms for UNet [5] and output composition of models as shown in Figure 2 $[b]$ and $\sim$150ms for FDP model shown in $[c]$. The codebase will be publicly released upon acceptance.

DP-DiT. We use DiT-S ($\sim$33M parameters)[24] as the base architecture, with 12 layers, 6 heads and a hidden dimension of 6. Peebles and Xie [24] specifically show that the conditioning using AdaLn-Zero outperforms other forms of conditioning such as in-context and cross-attention for image generation. However, we observe slightly stronger performance when the weights for the final layer of AdaLn are initialized with a Gaussian. We use different untrained ResNet-18 ($\sim$12M parameters)[14] encoders for each camera and also encode proprioception using a separate encoder. All encoded conditionals across the observation horizon are concatenated before using AdaLn. The model size conditioned on the input images from 2 cameras is $\sim$56 M parameters. All DiT models are trained using a learning rate of $1e{-}4$ and a weight decay of $1e{-}3$. We also perform exponential moving average (EMA) to reduce the variance in training. The same DiT backbone is used for low-dimensional, visual, and point-cloud tasks. We use the 100-step DDPM [16] noise scheduler suggested by Chi et al. [5] implemented using HuggingFace Diffusers [39]. Sampling is performed using 8-step DDIM [33].

DP-UNet. We use the 1D-UNet [31] implementation from Chi et al. [5]. UNet is trained using a learning rate of $1e{-}4$ and a weight decay of $1e{-}6$. Although the parameter count of the DiT model does not vary significantly with increasing context length due to self-attention, that is not the case with UNet. We use a relatively smaller UNet for low-dimensional tasks and a UNet with a larger channel width for visual tasks. For an observation horizon of 3 and an action horizon of 15 for visuomotor tasks, the parameter count of UNet increases to ($\sim 336M$), not including the ResNet weights. UNet uses FiLM [25] layers for conditioning on a single embedding, which is built for different visual inputs and proprioception similar to DP-DiT.

FDP. We experiment with several implementations of FDP in this paper, as shown in Figures 2 $[b]$ and $[c]$. The simplest implementation shown in $[b]$ simply adds the output of the base and the residual model, with the base kept frozen. The architectures of these models are exactly identical to DP-DiT, except that they are conditioned on different modalities. For Figure 2 $[c]$ used to present the results in the paper, the architecture of the base model is the same as that of DP-DiT. However, the residual model is designed similarly to the ViT [9] architecture. Since we do not denoise the inputs, we encode and pass all the inputs across the observation horizon through self-attention. We condition them on noisy actions using AdaLn. The images are encoded using patch embedding, where we keep the patch size equal to the size of the image to reduce the number of parameters. Crucially, we apply a zero layer on the block outputs of the residual model that are added to the corresponding blocks of the base model. We implement two variants of the zero-layer: a zero-initialized convolutional layer and a zero-initialized linear layer. For the convolutional layer, π_{base} learned on proprioception is of $\sim 30M$ parameters and the residual model π_{res} with two camera image inputs is of $\sim 55M$ parameters. However, the linear zero layer bloats the residual model's size to $\sim 290M$. Other hyperparameters such as the learning rate, weight decay, and the noise schedule are the same as DP-DiT.

APPENDIX E
EXPERIMENTAL SETUP

Environments. We evaluate our approach across more than 10 tasks from RLBench [18] and all four tasks from the Adroit dexterous manipulation suite [12], as illustrated in Figures 4 and 5. RLBench offers a diverse set of tasks and includes a built-in planner for demonstration collection. We train visuomotor policies in RLBench using a multi-camera setup that records 96×96 RGB images, with joint positions as the action modality. To assess robustness under visual distribution shift, we introduce appearance modifications to the environment, shown in Figure 6. Our experiments on RLBench use five camera views (wrist, front, overhead, right-shoulder, and left-shoulder), an observation horizon of 2, and an action horizon of 16.

The Adroit benchmark comprises high-dimensional hand manipulation tasks performed using a 24-DoF anthropomorphic hand (see Figure 4). It includes four tasks—*Door*, *Hammer*, *Pen*, and *Relocate*—that demand fine motor control and complex object interaction. We modify the success condition of the *Hammer* task, requiring the nail to be within a distance of 0.2 (instead of 0.1) from the board. Each Adroit task is represented by a task-specific low-dimensional state vector. We use an observation horizon of 3 and an action horizon of 15 across all Adroit experiments.

We also evaluate on the Robomimic dataset [23], which provides low-dimensional state observations and uses an action space defined as the change in end-effector position and orientation (axis-angle). The benchmark includes four tasks: *Lift*, *Can*, *Square*, and *Toolhang*, with the latter two requiring higher precision. Following Chi et al. [5], we use an observation horizon of 1 and an action horizon of 10 for all Robomimic experiments.

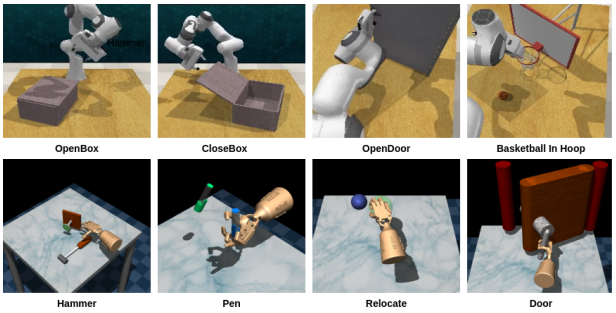

Fig. 4: RLBench and Adroit Tasks considered in the main paper.

Evaluation Methodology. Our sample efficiency results are reported as the mean and standard deviation of the success rates over 100 rollouts for 3 seeds each (total of 300 rollouts). Our results for distractor experiments in simulation and our ablations are averaged over 150 rollouts across 3 seeds for each variation of the environment. For the real-world experiments, we report the task success rate over 10 rollouts for the original and modified environments with distractors and appearance changes. We provide experimental configurations for each environment in Table III.

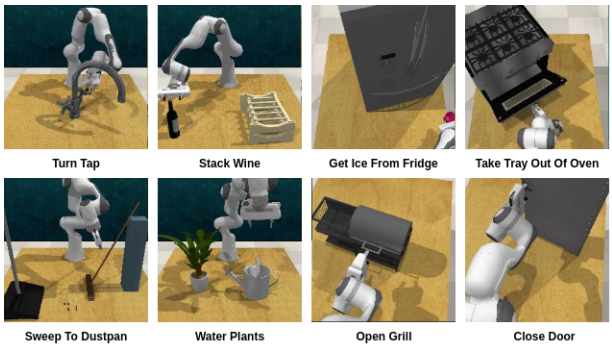

Fig. 5: Additional RLBench tasks considered in our experiments

Suite	Task	Env. Obs. Dim.	Rob. Obs. Dim.	Action Dim.	Max. Len.	# Train Demos	Obs/Act Horizon
Robomimic	Lift	10	9	7	400	10/50/100	o1h10
	Can	14	9	7	400	10/50/100	o1h10
	Square	14	9	7	400	10/50/100	o1h10
	Toolhang	44	9	7	700	10/50/100	o1h10
Adroit	Door	12	27	28	475	22	o2h16
	Hammer	13	33	26	475	22	o2h16
	Pen	21	24	24	475	22	o2h16
	Relocate	9	30	30	475	22	o2h16
RLBench	Various	–	8	8	300/600	10/50/100	o3h15
Real-World	Various	–	9	9	400	50	o3h15

TABLE III: Environment specifications including observation, robot and action dimensions, max trajectory lengths, and number of training demonstrations. The final column (Obs/Act Horizon) denotes the number of observation frames (`oN`) and action steps (`hM`) used in training. For RLBench and Real-world settings, the environment observation dimension depends on the number of cameras (1/3/5) and embedding size (512).

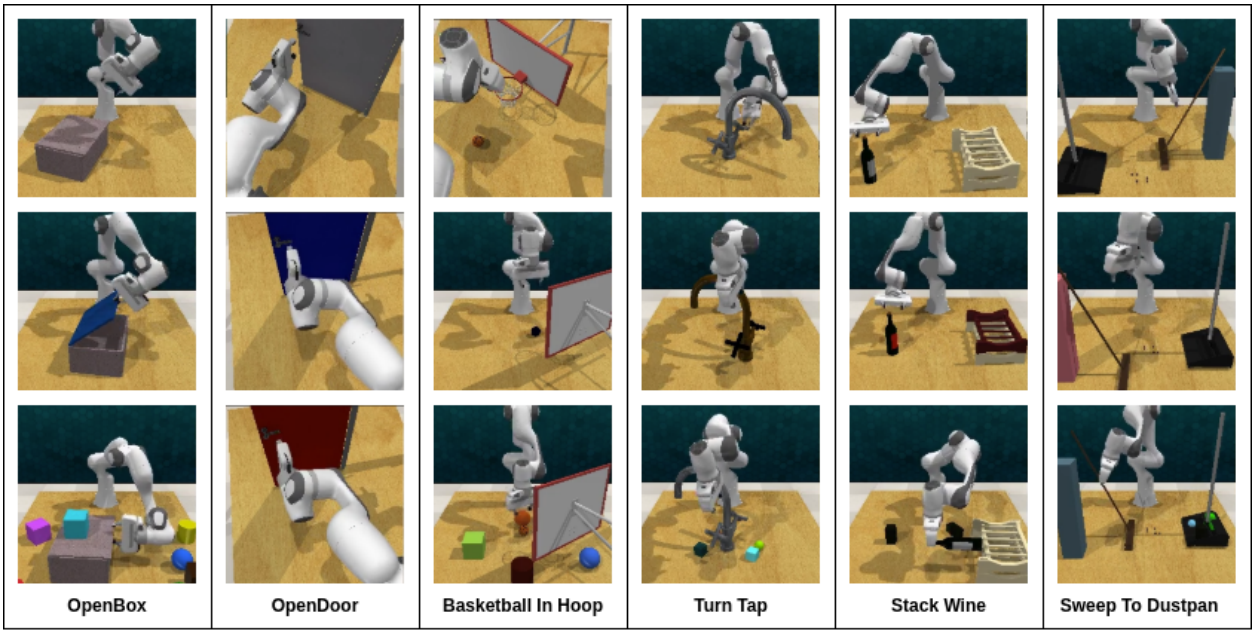

Fig. 6: RLBench tasks with appearance changes and distractors

APPENDIX F
EXTENDED SIMULATION RESULTS

Research Question 1: Can task-specific prioritization of modalities using FDP lead to sample efficiency gains in learning visuomotor tasks?

In this section, FDP refers to the prioritization of proprioceptive inputs, with vision modeled as a residual—unless explicitly stated otherwise. The results for RLBench, corresponding to Figure 3, are presented in Table IV. Additional results for the two-camera setup (wrist and front views) are provided in Table V. FDP with prioritized proprioception consistently outperforms DP-DiT across a range of tasks, particularly in low-data regimes. For reference, we include the performance of a motion-only model (trained solely on proprioception with 100 demonstrations) beneath each task name. The poor performance of these motion models highlights the importance of visual information for successful task execution. Although FDP learns a residual for vision, it effectively extracts visual correlations to guide the motion model, leading to superior performance compared to learning the full conditional distribution (DP-DiT) directly.

We show the results in the Adroit environments in Table VI. The results in Table VI have a $Success$ column that indicates the number of times $Success$ token was received from the environment within the time-frame of

Task	Number of Demos	DP-DiT		DP-UNet		CFG		POCO		FDP	
		Mean	Std	Mean	Std	Mean	Std	Mean	Std	Mean	Std
OpenBox MM=17.3	10	21.0	± 2.0	7.5	± 0.7	20.3	± 4.0	27.7	± 4.9	**59.0**	± 4.0
	50	86.0	± 1.7	86.0	± 1.0	86.0	± 2.6	87.3	± 0.6	**90.0**	± 2.6
	100	**98.3**	± 1.5	93.0	± 1.0	92.3	± 2.5	**98.3**	± 1.5	91.3	± 4.5
CloseBox MM=36	10	25.7	± 5.9	30.0	± 6.2	19.7	± 4.7	29.7	± 3.5	**71.7**	± 6.8
	50	79.7	± 1.2	70.0	± 3.6	75.0	± 3.5	79.7	± 1.2	**87.7**	± 3.1
	100	85.7	± 1.5	83.7	± 3.8	**88.7**	± 1.5	**88.3**	± 1.5	**88.3**	± 2.9
OpenDoor MM=18	10	24.7	± 4.2	7.7	± 1.5	12.7	± 1.5	13.0	± 4.4	**42.0**	± 5.2
	50	44.0	± 1.7	11.3	± 3.5	22.3	± 9.3	38.0	± 5.0	**54.7**	± 5.0
	100	44.7	± 4.2	23.3	± 3.5	60.0	± 0.0	42.0	± 2.6	**65.0**	± 2.6
CloseDoor MM=0.3	10	2.0	± 1.0	3.0	± 1.0	0.0	± 0.0	4.3	± 0.6	**9.0**	± 1.7
	50	0.0	± 0.0	**7.3**	± 1.5	1.7	± 0.6	1.7	± 0.6	**6.7**	± 3.5
	100	2.7	± 1.5	**8.0**	± 0.0	5.0	± 0.0	5.0	± 2.6	**9.0**	± 1.0
Basketball in Hoop MM=0.7	10	1.7	± 1.5	2.7	± 1.5	1.7	± 1.2	2.0	± 1.0	**10.7**	± 2.5
	50	18.0	± 3.5	17.0	± 1.0	13.7	± 5.5	20.7	± 4.2	**47.3**	± 3.2
	100	38.7	± 4.2	34.3	± 4.2	46.0	± 4.0	42.7	± 4.0	**63.3**	± 3.8

TABLE IV: Performance results across different RLBench tasks for 10, 50 and 100 demonstrations. The reported means and standard deviations are computed over 100 rollouts for each of 3 random seeds.

Task	#Demos	DP-DiT Mean ± Std	FDP Mean ± Std
Turn Tap	10	21.3 ± 6.1	**28.7** ± 3.1
	50	**52.0** ± 1.0	38.3 ± 2.1
Stack Wine	10	2.0 ± 0.0	**12.0** ± 9.2
	50	41.7 ± 4.0	**50.0** ± 2.0
Get Ice from Fridge	10	0.0 ± 0.0	**16.7** ± 9.9
	50	12.0 ± 2.0	**37.3** ± 7.0
Take Tray Out of Oven	10	0.0 ± 0.0	**3.0** ± 1.4
	50	14.0 ± 5.3	**18.0** ± 2.0
Sweep to Dustpan	10	13.3 ± 3.1	**51.3** ± 4.2
	50	62.0 ± 5.3	**77.3** ± 3.1
Water Plants	10	2.7 ± 1.2	**18.0** ± 2.0
	50	6.0 ± 3.5	**55.3** ± 9.0
Open Grill	10	4.7 ± 3.1	**10.7** ± 6.1
	50	4.0 ± 2.0	**30.7** ± 8.1

TABLE V: Success rates (%) on various RLBench tasks using 10 and 50 demonstrations under a two-camera configuration. We report mean ± standard deviation over 50 rollouts for each of 3 random seeds.

the rollout. The results shown in Figure 3 correspond to $Success > 1$ condition in Table VI. However, since the nature of the tasks is dynamic, we also report the results for $Success > 25$ within the considered rollout time in Table VI. Notably, prioritization of proprioception leads to inferior results for the task of $Relocate$, as it is strongly dependent on the environment state specification of the locations of the ball and the target.

Further, we show results for Robomimic low-dimensional tasks in Table VII. We see that FDP is sample efficient for $Lift$ and Can tasks, while performing poorly on $Square$ and $Toolhang$ as they require precise manipulation. Our results are further supported by the poor performance of POCO on these tasks. Precise manipulation presents a bottleneck in the state-action distribution, implying that learning the full conditional will lead to better performance. When DP-DiT is composed with a motion model as in POCO, or the vision modality is separated out and learned as a residual of the motion model, it leads to inferior results.

Research Question 2: Does learning the visual modality as a residual in the FDP framework result in robustness to distractors and appearance changes?

We provide results for 3 more tasks trained in a two-camera setting (wrist and front) in Table VIII. Further, we provide results for learning a visual residual over DP3 [46] in Table IX. We clearly see that learning a visual residual over DP3 is robust to visual changes as compared to a DP3 model that also takes in RGB inputs. However, we see similar performance as compared to DP3 in both the default task setting and with visual appearance changes, as DP3 does not take in RGB values. This is limiting, as the model is unable to solve tasks that require differentiation in color. FDP can flexibly be extended to incorporate color either as a primary modality or a residual, even with an unequal

Task	Success	DP-DiT	DP-UNet	CFG	POCO	FDP
Door	> 1	62.7 ± 4.7	67.3 ± 4.7	42.7 ± 5.5	58.7 ± 4.6	**74.3** ± 3.1
MM=4	> 25	31.3 ± 0.6	30.7 ± 0.6	22.3 ± 7.5	29.0 ± 2.6	**45.7** ± 5.8
Pen	> 1	55.3 ± 10.4	63.3 ± 3.5	**67.7** ± 3.5	57.3 ± 0.6	62.3 ± 5.9
MM=22.7	> 25	50.3 ± 9.7	57.3 ± 2.5	**62.3** ± 0.6	53.0 ± 2.6	56.0 ± 4.4
Hammer	> 1	39.0 ± 4.0	47.3 ± 6.0	40.7 ± 2.3	40.7 ± 5.5	**51.7** ± 6.4
MM=18	> 25	38.7 ± 4.5	47.0 ± 6.0	39.0 ± 2.0	40.7 ± 5.5	**50.7** ± 6.4
Relocate	> 1	**88.7** ± 4.5	56.0 ± 7.2	1.3 ± 0.6	82.3 ± 1.2	63.0 ± 6.9
MM=2	> 25	**85.0** ± 5.2	52.0 ± 7.0	0.3 ± 0.6	79.0 ± 3.6	58.7 ± 4.0

TABLE VI: Performance comparison across Adroit tasks for different models. We show performance results for the successful completion of the task for more than 1 and 25 time steps. The mean and standard-deviation are reported over 300 rollouts split across 3 different seeds.

Task	#Demos	DP-DiT	DP-UNet	CFG	POCO	FDP
Lift	10	82.3 ± 3.1	98.0 ± 1.7	90.3 ± 1.5	83.3 ± 3.2	96.3 ± 1.5
	50	95.3 ± 1.5	100.0 ± 0.0	91.7 ± 1.5	96.3 ± 0.6	99.7 ± 0.6
	100	99.0 ± 1.7	99.7 ± 0.6	98.7 ± 0.6	98.7 ± 1.5	99.7 ± 0.6
Can	10	51.3 ± 2.3	43.7 ± 4.0	52.3 ± 3.5	58.3 ± 1.2	79.7 ± 3.5
	50	95.0 ± 1.7	93.3 ± 0.6	97.3 ± 3.8	95.3 ± 1.2	98.3 ± 1.2
	100	99.0 ± 1.0	98.3 ± 1.5	98.7 ± 1.5	98.7 ± 1.5	99.7 ± 0.6
Square	10	14.0 ± 4.4	15.0 ± 2.6	14.3 ± 6.5	15.3 ± 4.2	16.0 ± 2.6
	50	65.7 ± 2.9	65.0 ± 4.0	65.3 ± 3.5	64.7 ± 2.1	56.0 ± 1.7
	100	80.3 ± 4.6	82.3 ± 7.4	80.0 ± 3.0	76.3 ± 6.0	58.0 ± 6.6
Toolhang	10	4.3 ± 1.2	0.7 ± 1.2	4.0 ± 2.6	3.0 ± 1.0	0.3 ± 0.6
	50	43.3 ± 8.6	41.0 ± 2.6	39.3 ± 4.2	42.7 ± 1.2	26.7 ± 3.2
	100	60.0 ± 7.5	54.7 ± 3.5	60.7 ± 3.5	55.7 ± 7.5	45.7 ± 3.8

TABLE VII: Success rates (%) on low-dimensional Robomimic tasks using 10, 50, and 100 demonstrations. We report the mean ± standard deviation over 100 rollouts for each of 3 random seeds.

number of demonstrations.

Task	**Setting**	**DiT**	**FDP**
Turn Tap	50 demos original	**52.0** ± 1.0	38.3 ± 2.1
	Zero-shot color	**45.3** ± 2.3	32.7 ± 2.3
	Zero-shot distractor	**50.7** ± 8.1	32.7 ± 10.1
Stack Wine	50 demos original	41.7 ± 4.0	**50.0** ± 2.0
	Zero-shot color	9.3 ± 3.1	**37.3** ± 12.2
	Zero-shot distractor	27.3 ± 1.2	**44.0** ± 3.5
Sweep to Dustpan	50 demos original	62.0 ± 5.3	**77.3** ± 3.1
	Zero-shot color	41.3 ± 8.1	**78.7** ± 8.1
	Zero-shot distractor	62.0 ± 5.3	**78.0** ± 9.2

TABLE VIII: Success rates (%) of DiT and FDP for three RLBench tasks in a 2-camera (front+wrist) setting, with color changes and distractors. Mean ± std over 50 rollouts (3 seeds).

Research Question 3: How sensitive is the visuomotor task performance to prioritization of proprioception?

We provide specifications for the block pick experiment with different scales of variation in Figure 7.

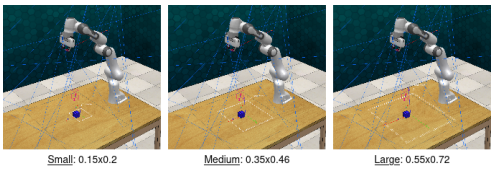

Fig. 7: Task design for block pick with different scales of variation (dim in meters).

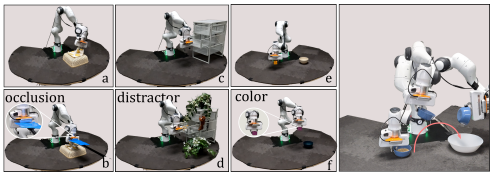

Fig. 8: Task domains and their variations. In `occlusion`, the visual input is blocked using a board; `distractor`, flower pots and toys are introduced into the scene; and in `color`, the color of the manipulated object is altered during evaluation.

APPENDIX G
REAL WORLD EXPERIMENTAL DETAILS

The task domains used in our real-world experiments as shown in Figures 9 and 8 are described below:

Task	Setting	DP3RGB-DiT	DP3-DiT	FDP
Turn Tap	50 demos original	56.7 ± 5.7	59.7 ± 4.0	57.3 ± 5.0
	zero-shot color	32.0 ± 4.0	**62.7** ± 3.1	**62.7** ± 4.6
Stack Wine	50 demos original	**90.7**± 0.6	83.7 ± 1.2	82.7 ± 7.6
	zero-shot color	70.7 ± 6.4	**82.7** ± 6.1	**81.3** ± 4.2
Sweep to Dustpan	50 demos original	84.0 ± 3.5	86.7 ± 3.8	84.7 ± 5.0
	zero-shot color	80.0 ± 2.0	**89.3** ± 4.6	**88.7** ± 1.2
Water Plants	50 demos original	62.7 ± 11.7	60.0 ± 5.3	61.3 ± 3.1
	zero-shot color	25.3 ± 6.1	**60.0** ± 5.3	**64.0** ± 2.0

TABLE IX: Mean success rates (%) and standard deviations over 150 rollouts (3 seeds) for DP3RGB-DiT, DP3-DiT, and FDP (visual residual over DP3-DiT) across four RLBench tasks under original and zero-shot color settings.

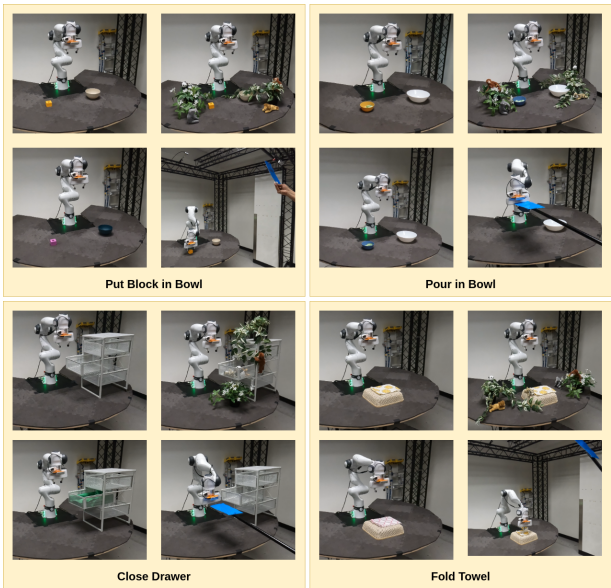

Fig. 9: Tasks considered for the real robot experiments. In clockwise direction: original task, with distractors, with camera occlusions, and with color changes.

- *Close Drawer*: Close the cabinet of an open drawer. We vary the drawer's placement angle and position relative to the robot within a range of 10° and 15 cm, respectively. This is a relatively simple task where the robot must close the drawer by pushing it with its end-effector.
- *Put Block in Bowl*: Pick up a block and place it inside a nearby bowl. The positions of both the block and the bowl are varied within a 15 cm range relative to the robot. This task assesses the policy's ability to perform precise pick-and-place actions.
- *Pour in Bowl*: Pick up a cup and pour its contents into a nearby bowl. The positions of the cup and the bowl are varied within a 15 cm range relative to the robot. This task evaluates the policy's effectiveness in operating near joint limits.
- *Fold Towel*: Fold a kitchen towel placed on a compliant surface. The towel's position is varied within a 5 cm range relative to the robot. This task evaluates the policy's capability in deformable object manipulation.

We used ROS1 Noetic for robot software development. For data collection, we used a 3D Connexion SpaceMouse Pro to set end-effector velocity targets, which were executed on the Franka robot using a differential inverse kinematics controller. Time-synchronized joint positions and camera images were recorded at 30Hz for each demonstration and later post-processed by downsampling to 10Hz for policy training. During rollout, we employed a joint position controller to sequentially execute short-horizon trajectory predictions from the policy. We allowed a trajectory length of 400 steps for each task in all our real-world robot experiments. With a horizon length of 16, this resulted in 25 policy inference steps per task.

Robot Safety Check. We implemented a safety check in our robot software to prevent potential damage to the robot during environment variation experiments. This was particularly necessary for DP, which often generated high-jerk joint targets in out-of-distribution scenarios. For each joint command, we ensured that the target was within a threshold Euclidean distance from the current joint state, i.e., $||j_{\text{target}} - j_{\text{current}}|| \leq 0.1$. If this condition was violated, policy execution was immediately halted and the rollout was considered a failure.

Policy Robustness. We observe that the FDP policy is more robust to subtle color changes than to drastic ones. For example, in the *Put Block in Bowl* task, training data was collected using an orange block. In our `color` variation evaluations, FDP achieved a higher success rate with a yellow block compared to a pink block. A similar trend was observed for DP; however, its success rate with the pink block was significantly lower than that of FDP.

We also find that FDP is fairly robust to dynamic obstacles. During our `distractor` experiments, FDP successfully completed the task even when people walked around the setup, appearing in front of the camera, or interacted with the workspace during policy execution. In similar settings, DP exhibited jerky motions leading to task failures.

Failure Modes for FDP. We observe that the FDP policy relatively struggles to generalize in tasks with high motion diversity. For example, in the *Put Block in Bowl* task, the policy often fails to grasp the block when it is positioned near the boundaries of the data distribution. We also find in the absence of distractors and occlusions, the DP's rollout was smoother than FDP. However, the motion smoothness of FDP remains consistent across task variations, unlike DP, which often fails immediately when unseen obstacles or occlusions are introduced during rollout.

Behavior of FDP is unlike DP, arising from the different modeling approaches. Since our experiments prioritize proprioception for visuomotor policies, we see that the model does not exhibit the "retrying" behavior similar to diffusion policy, where the DP attempts to recover from an out of distribution state. This behavior arises from the domination of the proprioception conditioned base model over the vision residual model. While we experimentally evaluate FDP for different classes of tasks, we leave a deeper understanding of the behavioral differences on prioritizing different modalities or even joint modalities like diffusion policies for future work.

APPENDIX H
ABLATIONS

We present several ablations of the baselines and FDP for *Door* and *Relocate* tasks from Adroit and *Open Door* task from RLBench. In Table X, we see that the Base version of DiT does not result in consistent improvements in performance. Moreover, we observe that using a learned position embedding results in higher performance for tasks that require stronger attention to certain conditional modalities over others, such as for the task *Relocate*. DiT results in strong performance across tasks for the same set of hyperparameters, unlike Transformer for Diffusion (Cross-attention) [5] which requires careful tuning [5]. We also train UNet from Chi et al. [5] with a larger action horizon, as used by Chen et al. [3], but it did not result in consistent improvements across tasks.

For FDP ablations on Adroit in Table XI, we observe that composing the score output as shown in Figure 2 [b] is not performant. Moreover, we also ablate the design of the residual model to align more closely with ControlNet [47]. This model is exactly similar in structure to DP-DiT. It passes the noisy actions through self-attention and conditions on the modalities using AdaLn. Finally, we ablate the choice of the zero-layers and see that a linear zero-layer outperforms a convolutional one, albeit with a higher parameter count. We report the linear layer results for low-dimensional tasks since the parameter count is still comparable, while we choose the convolutional layer for visual tasks with multiple cameras as the count bloats up.

We provide ablations for the *Open Door* task in Table XI. First, we ablate the compositional weight of POCO and observe that higher values perform better in tasks with limited motion diversity, where a combination of a weighted base policy π_{base} and a vision-conditioned residual π_{res} is used. In our experiments, we use a fixed weight of 0.1 across all tasks. However, searching for the optimal compositional weight is cumbersome. FDP addresses this limitation by learning the residual in a mathematically grounded manner, avoiding heuristic-based tuning of pre-trained policy combinations.

We also ablate DP-DiT with varying numbers of camera views and observe significant variability in performance. This suggests that, with limited demonstrations, the model struggles to attend to the right modality. Interestingly, the front camera does not capture the door knob, yet DP-DiT achieves high performance by learning spurious correlations with it.

For the Adroit tasks, we find that FDP with a linear zero-layer achieves the best performance. However, we use a convolutional layer in all reported experiments due to its lower parameter count. Lastly, we ablate the training duration of π_{base} and find that optimal performance typically occurs around the minimum value loss (MVL) checkpoint. Empirically, we observe that tasks with low motion diversity reach MVL later during training a proprioception-based π_{base}. In contrast, visually dependent tasks, such as *Relocate* in Adroit, overfit early on proprioception data and reach MVL earlier. Thus, MVL offers a natural stopping criterion that balances the influence and controllability of the base policy.

Model	Door		Relocate	
	reward > 1	reward > 25	reward > 1	reward > 25
DiT Base (∼130M)	51.3 ± 11.4	21.3 ± 7.6	74.0 ± 6.9	72.0 ± 3.5
DiT Small: fixed pos. embedding	58.3 ± 6.7	34.3 ± 2.5	63.0 ± 4.6	60.0 ± 3.6
DiT Small: learned pos. embedding	62.7 ± 4.7	31.3 ± 0.6	88.7 ± 4.5	85.0 ± 5.2

Model	Door		Relocate	
	reward > 1	reward > 25	reward > 1	reward > 25
Cross-attention [5]	53.3 ± 2.1	47.3 ± 2.3	40.0 ± 6.2	36.7 ± 5.7
UNet: obs horizon=4, action horizon=64	35.3 ± 6.8	16.7 ± 4.2	64.7 ± 5.1	59.0 ± 4.4
UNet: obs horizon=2, action horizon=16	67.3 ± 4.7	30.7 ± 0.6	56.0 ± 7.2	52.0 ± 7.0

TABLE X: Performance (mean ± std) on Adroit tasks, reported over 150 rollouts per model (3 seeds). Top: DiT-based models including a variant with learned embeddings. Bottom: Cross-attention and UNet baselines. Bold indicates the model selected to present results.

Model	Door		Relocate	
	r > 1	r > 25	r > 1	r > 25
FDP: [b] in Figure 2	28.3 ± 3.1	15.3 ± 2.1	1.3 ± 1.2	1.3 ± 1.2
FDP: ControlNet	60.3 ± 6.7	17.7 ± 3.1	43.3 ± 4.0	38.7 ± 4.6
FDP: Conv zero layer (∼33M)	39.7 ± 1.5	16.7 ± 4.6	57.7 ± 1.5	52.0 ± 3.6
FDP: Linear zero layer (∼145M)	74.0 ± 5.2	50.7 ± 2.1	68.7 ± 1.2	65.7 ± 1.2

TABLE XI: Performance (mean ± std) over 150 rollouts for FDP ablations on Adroit tasks- Door and Relocate. Bold indicates the model chosen to present results in the paper.

Model	Mean	Std
Transformer variants		
DiT: small (∼33M)	24.00	7.21
Cross-attention [5]	3.00	1.00
DiT: base (∼130M)	27.33	5.03
POCO: λ for π_{base} [42]		
$\lambda = 0.5$	21.33	7.02
$\lambda = 0.2$	11.33	3.06
$\boldsymbol{\lambda = 0.1}$	13.0	4.4
$\lambda = 0.01$	18.00	5.29
Camera input ablations for DP-DiT		
1 camera	42.00	6.00
2 cameras	0.67	1.15
3 cameras	8.67	4.62
5 cameras	24.67	4.16

Model	Mean	Std
π_{res} ablations (5 cameras)		
FDP: [b] in Figure 2	20.67	8.33
FDP: 16 patches	31.33	4.62
FDP: Linear zero layer	45.33	4.16
FDP: Conv, 1 patch	42.00	5.20
π_{base} training epoch (ep)		
100 ep	24.67	6.11
700 ep (MVL)	42.00	5.20
1000 ep	42.00	5.29
1500 ep	40.00	6.00
2000 ep	40.67	3.06

TABLE XII: Ablation results on the Open Door task using 10 demonstrations. Each entry shows the success rate (mean ± std) over 150 rollouts. Bold values indicate the model chosen to report results in this paper.