

PlayFusion: Skill Acquisition via Diffusion from Language-Annotated Play

Anonymous Author(s)

Affiliation

Address

email

1 **Abstract:** Learning from unstructured and uncurated data has become the dominant
2 paradigm for generative approaches in language and vision. Such unstructured
3 and unguided behavior data, commonly known as *play*, is also easier to collect in
4 robotics but much more difficult to learn from due to its inherently multimodal,
5 noisy, and suboptimal nature. In this paper, we study this problem of learning
6 goal-directed skill policies from unstructured play data which is labeled with
7 language in hindsight. Specifically, we leverage advances in diffusion models to
8 learn a multi-task diffusion model to extract robotic skills from play data. Using
9 a conditional denoising diffusion process in the space of states and actions, we
10 can gracefully handle the complexity and multimodality of play data and generate
11 diverse and interesting robot behaviors. To make diffusion models more useful
12 for skill learning, we encourage robotic agents to acquire a vocabulary of skills by
13 introducing discrete bottlenecks into the conditional behavior generation process.
14 In our experiments, we demonstrate the effectiveness of our approach across a
15 wide variety of environments in both simulation and the real world. Video results
16 available at <https://play-fusion.github.io>.

17 **Keywords:** Diffusion Models, Learning from Play, Language-Driven Robotics

18 1 Introduction

19 Humans reuse past experience via a broad repertoire of skills learned through experience that allows
20 us to quickly solve new tasks and adapt across environments. For example, if one knows how to
21 operate and load a dishwasher, many of the skills (e.g., opening the articulated door, adjusting the
22 rack, putting objects in) will transfer seamlessly. How to learn such skills for robots and from what
23 kind is a long-standing research question. Robotic skill abstraction has been studied as a way to
24 transfer knowledge between environments and tasks [1, 2, 3]. It has been common to use primitives
25 as actions in the options framework [4, 5], which are often hand-engineered [6, 7, 8, 9, 10, 11] or
26 learned via imitation [12, 13, 14]. These allow for much more sample-efficient control but require
27 knowledge of the task and need to be tuned for new settings. On the other hand, there have been
28 efforts to *automatically* discover skills using latent variable models [15, 16, 17, 18, 19, 20, 21, 22].
29 While they can work in any setting, such models are often extremely data-hungry and have difficulty
30 scaling to the real world due to the data quality at hand.

31 As a result, real-world paradigms are based on imitation or offline reinforcement learning (RL) but
32 both these require several assumptions about the datasets. In imitation learning, human teleoperators
33 must perform tasks near-perfectly, reset the robot to some initial state, perform the task near-perfectly
34 again, and repeat several times. In offline RL, data is assumed to contain reward labels, which is
35 impractical in many real-world setups where reward engineering is cumbersome. In contrast, it is
36 much easier to collect uncurated data from human teleoperators if they are instructed only to explore,
37 resulting in *play data* [21, 22, 23]. Learning from play (LFP) has emerged as a viable alternative to
38 traditional data collection methods for behavior generation. It offers several advantages: (1) it is

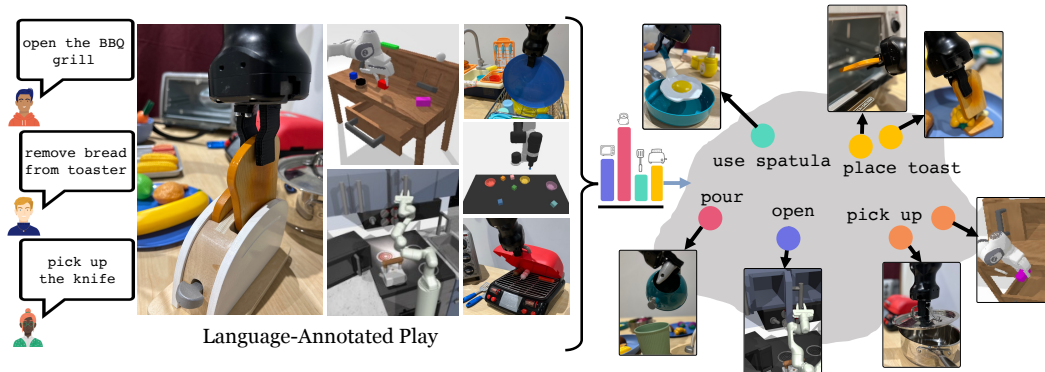


Figure 1: Across multiple real-world and simulated robotic settings, we show that our model can extract semantically meaningful skills from language-annotated play data. Such data is highly multimodal and offers no optimality guarantees. Video results of PlayFusion are available at <https://play-fusion.github.io>.

39 efficient because large datasets of play can be collected without the need for setting up and executing
 40 perfect demonstrations, and (2) the data collected is rich and diverse because it contains a broad
 41 distribution of behavior ranging from completions of complex tasks to random meandering around
 42 the environment. An important quality of such data is that it is grounded with some semantic goal that
 43 the "player" is aiming to achieve. We believe a simple abstraction for this is *language* instructions,
 44 which can describe almost any play trajectory.

45 A major challenge in learning from play is that the data is highly multimodal, i.e., there are many
 46 different ways to achieve a specific goal, and given a sample from the play data, there are many
 47 different goals that could have generated it. One popular way to handle highly multimodal data is
 48 by modeling the full distribution via generative models. In recent years, there has been remarkable
 49 progress in large generative models [24, 25, 26, 27], especially in the class of diffusion models [28, 29],
 50 which have been shown to generate high-resolution images – a property well suited for vision-based
 51 robotic control. In fact, diffusion models have shown to be effective in capturing complex, continuous
 52 actions [30, 31, 29, 32, 33] in the context of robotics. However, these diffusion model-based
 53 approaches have not been empirically shown yet to work on unstructured data. We argue that the
 54 ability of diffusion models to fully capture complex data paired with their potential for text-driven
 55 generation can make them good candidates to learn from language-annotated play data.

56 One additional consideration is that in reality, humans only deal with a few skills. Almost every task
 57 manipulation task involves some grasping and some post-grasp movement. We believe that learning
 58 *discrete skills* will not only make the whole process more efficient but will also allow interpolation
 59 between skills and generalizations to new tasks. To address this, we propose **PlayFusion**, a diffusion
 60 model which can learn from language-annotated play data via discrete bottlenecks. We maintain the
 61 multimodal properties of our current system while allowing for a more discrete representation of
 62 skills. Empirically, we show that our method outperforms state-of-the-art approaches on six different
 63 environments: three challenging real-world manipulation settings as well as the CALVIN [34], Franka
 64 Kitchen [22], and Ravens [35, 36] simulation benchmarks.

65 2 Related Work

66 **Goal and Language Conditioned Skill Learning** One method of specifying the task is via goal-
 67 conditioned learning, often by using the actual achieved last state as the goal [37, 38, 39, 40]. There is
 68 also recent work on using rewards to condition robot behavior [41], but this requires a reward-labeled
 69 dataset, which makes stronger assumptions than play data. Furthermore, there is a large body of
 70 work on language-conditioned learning [42, 36, 43, 44, 45, 46, 47], which specifies the task through
 71 language instructions. Instead of conditioning the policy on fully labeled and curated data, we take
 72 advantage of unstructured play data which is annotated with language in hindsight.

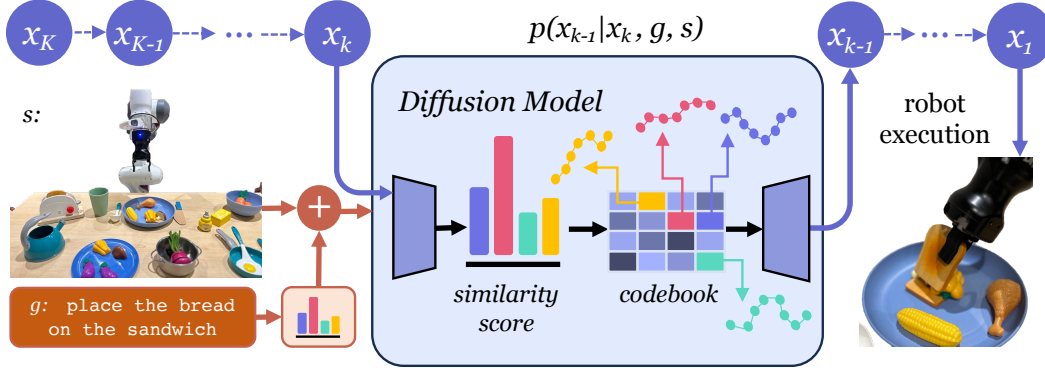


Figure 2: Overview of how PlayFusion extracts useful skills from language-annotated play by leveraging discrete bottlenecks in both the language embedding and diffusion model U-Net. We generate robot trajectories via an iterative denoising process conditioned on language and current state.

73 **Learning from Play** Unlike demonstrations, play data is not assumed to be optimal for any specific
 74 task as it is collected by human teleoperators who are instructed only to explore. Play-LMP and
 75 MCIL [21, 48] generate behaviors by learning motor primitives from play data using a VAE [49, 50].
 76 RIL [22] is a hierarchical imitation learning approach and C-BeT [23] generates behaviors using
 77 a transformer-based policy and leverages action discretization to handle multimodality. LAD [51]
 78 incorporates diffusion for learning from play, but keeps several components of VAE-based approaches
 79 for encoding latent plans; we forgo those elements completely.

80 **Behavior Modeling with Generative Models** A promising architecture for behavior modeling
 81 with generative models is the diffusion probabilistic model [28, 52, 53, 54]. Diffuser [30], Decision
 82 Diffuser [31], Diffusion-QL [29] and IDQL [32] apply diffusion models to the offline reinforcement
 83 learning (RL) problem. In real-world robotic applications, Diffusion Policy [33] demonstrated strong
 84 results in visuomotor policy learning from demonstrations. Different from these works, we learn
 85 from play data containing semantic labels instead of offline RL datasets or expert demonstrations.
 86 Some approaches [55, 56] incorporate diffusion in robotics but not for generating low-level actions.

87 **Discrete control** A key challenge in robot learning is the exponentially large, continuous action
 88 space. Option or skill-based learning is appealing as it can circumvent this problem and allow the agent
 89 to learn in a structured, countable action space [57, 58, 59, 60]. Learned action discretization [52, 23]
 90 has allowed approaches to scale to complex tasks. C-BeT [23] applies real-world robotic control
 91 with transformers [41, 61, 62] to the goal-conditioned setting; [63] train a dynamics model over
 92 discrete latent states. We leverage the discrete properties of VQ-VAEs and their natural connection to
 93 language-labeled skills.

94 3 Background

95 **Denoising Diffusion Probabilistic Models (DDPMs)** DDPMs [28] model the output generation
 96 process as a denoising process, which is often referred to as Stochastic Langevin Dynamics. To
 97 generate the output, the DDPM starts by sampling x^K from a Gaussian noise distribution. It then
 98 performs a series of denoising iterations, totaling K iterations, to generate a sequence of intermediate
 99 outputs, x^k, x^{k-1}, \dots, x^0 . This iterative process continues until a noise-free output x^0 is produced.
 100 The denoising process is governed by the following equation:

$$x^{k-1} = \alpha(x^k - \gamma \epsilon_\theta(x^k, k) + N(0, \sigma^2 I)) \quad (1)$$

101 Here, ϵ_θ represents the noise prediction network with a learnable parameter θ , and $N(0, \sigma^2 I)$ denotes
 102 the Gaussian noise added at each iteration. This equation is used to generate intermediate outputs
 103 with gradually decreasing noise levels until a noise-free output is obtained. To train the DDPM,
 104 the process begins by randomly selecting x^0 from the training dataset. For each selected sample, a

105 denoising iteration k is randomly chosen, and a noise ϵ^k is sampled with the appropriate variance for
 106 the selected iteration. The noise prediction network is then trained to predict the noise by minimizing:

$$\mathcal{L} = \|\epsilon^k - \epsilon_\theta(x^0 + \epsilon^k, k)\|^2 \quad (2)$$

107 **Discrete Representations** We utilize VQ-VAE [64] inspired models in PlayFusion as they can
 108 provide a way to discretize the skill space. Given an input x , a VQ-VAE trains an encoder E to
 109 predict latent $E(x) = z$ and maintains a codebook of discrete latent codes e . The VQ layer selects j
 110 as $\arg \min_i \|z - e_i\|$, finding the closest code to the embedding, which is used to reconstruct x . The
 111 training loss is

$$\mathcal{L}_{\text{VQVAE}} = \mathcal{L}_{\text{recon}}(x, D(e_j)) + \|z - \text{sg}(e_j)\|_2 + \|\text{sg}(z) - e_j\|_2 \quad (3)$$

112 where D is the VQ-VAE decoder. The reconstruction loss is augmented with a quantization loss,
 113 bringing chosen codebook embedding vectors e_j toward the encoder outputs in order to train the
 114 codebook, as well a loss to encourage the encoder to "commit" to one of the embeddings.

115 **Learning from Play Data (LfP)** In the LfP setting, we are given a dataset $\{(s, a)\} \in S \times A$. There
 116 are no assumptions about tasks performed in these sequences or the optimality of the data collection
 117 method. Similar to the formulation of [23], the goal is to learn a policy $\pi = S \times S \rightarrow A$ where the
 118 input is the current state s_t and goal $g = s_T$. In some cases, (including ours), the goals are instead
 119 described via language annotations.

120 4 PlayFusion: Discrete Diffusion for Language-Annotated Play

121 Humans do not think about low-level control when performing everyday tasks. Our understanding
 122 of skills like door opening or picking up objects has already been grounded in countless prior
 123 experiences, and we can comfortably perform these in new settings. Skills are acquired through our
 124 prior experiences – successes, failures, and everything in between. PlayFusion focuses on learning
 125 these skills through *language-annotated* play data.

126 However, learning from play data is still difficult as continuous control skills are not easy to identify
 127 due to several challenges: (1) data can come from multiple modalities as there are many actions that
 128 the robot could have taken at any point, (2) we want the model to acquire a vocabulary of meaningful
 129 skills and (3) we want to generalize *beyond* the training data and have the model transfer skills to
 130 new settings. To address the challenges, we leverage recent advances in **diffusion-model** large-scale
 131 text-to-image generation. Such models [33, 30, 29] can inherently model multimodality via their
 132 iterative denoising process. To effectively transfer skills to new settings, we propose a modified
 133 diffusion model with the ability to **discretize** learned behavior from language-annotated data. Figure
 134 2 shows an overview of our method.

135 4.1 Language Conditioned Play Data

136 Our setup consists of language conditioned play data [21] $D_{\text{play}} = \{(s_t^{(i)}, a_t^{(i)})\}_{i=1}^N$: long sequences
 137 of robot behavior data containing many kinds of behaviors, collected by human operators instructed
 138 to perform interesting tasks. In this setting, we assume that there is some optimality to the data,
 139 i.e. $a_t \sim \mathcal{F}(s_t, z_g)$, where z_g is a latent variable that models the intention of the operator. We
 140 thus leverage *language* labels to estimate z_g . Given a sequence $\tau = \{s_i, a_i\}_{t=k}^H$, label τ with an
 141 instruction l which is passed into a language model [65], g_{lang} , referring to it as z_l throughout the
 142 paper. One can also use goal images, but we might not have access to these at test time. While our
 143 method can use any z_g as conditioning, assume that the play data has access to language annotations l .
 144 Our policy $\pi(a_t | s_t, z_l)$ contains a few simple components. We use a ResNet[66]-based visual encoder
 145 ϕ_v to encode s_t (a sequence of images) and an MLP based language encoder ϕ_l to downproject the
 146 language embedding z_l . The policy uses $g = [\phi_l(z_l), \phi_v(s_t)]$ as conditioning to the action decoder
 147 f_{act} . Previous approaches [21, 34] use latent variable models to deal with multimodality. We find that
 148 modelling f_{act} as a diffusion process enables us to circumvent this.

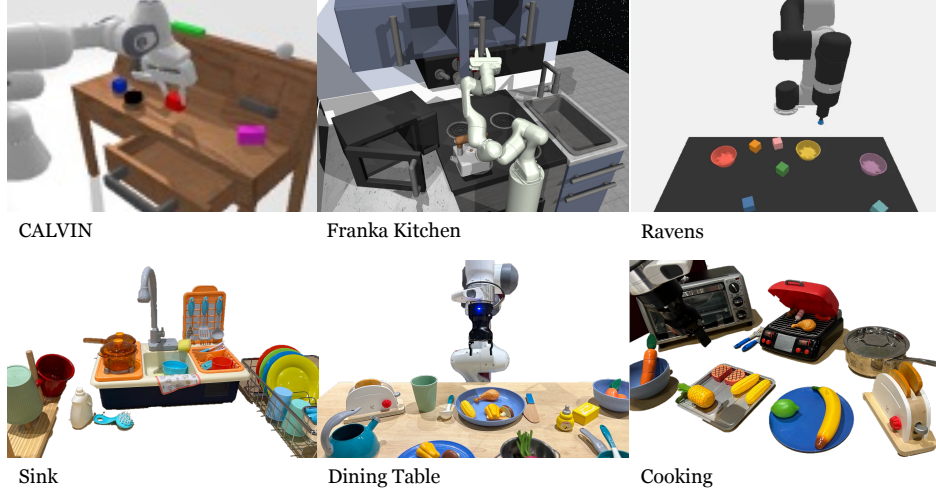


Figure 3: Simulated (top row) and real-world (bottom row) environments used for our evaluations. In each real-world setup, the robot is tasked with picking up one of the objects (e.g., plate, cup, carrot, bread, corn) and relocating it to a specified location (e.g., drying rack, plate, toaster, grill, pot).

149 4.2 Multi-modal Behavior Generation via Diffusion

150 With f_{act} , we aim to predict robot actions given the current state, using a DDPM to approximate the
 151 conditional distribution $P(a_t|s_t)$. In our setting, we additionally condition on the goal g . Formally,
 152 we train the model to generate robot actions a_t conditioned on goal g and current state s_t , so we
 153 modify Equations 1 and 6 to obtain:

$$a_t^{k-1} = \alpha(a_t^k - \gamma \epsilon_\theta(g, s_t, a_t^k, k) + N(0, \sigma^2 I)) \quad (4)$$

$$154 \quad \mathcal{L} = \|\epsilon^k - \epsilon_\theta(g, s_t, a_t^0 + \epsilon^k, k)\|_2 \quad (5)$$

155 We use the notation above for simplicity, but in practice, we predict a sequence of T_a future actions
 156 a_t, \dots, a_{t+T_a} instead of only the most immediate action, a technique known as *action chunking*.
 157 This is done in some recent works [33, 67] and is shown to improve temporal consistency.

158 4.3 Discrete Diffusion for Control

159 Moreover, humans often break down tasks into smaller skills, which are often repeatable. In fact,
 160 most tasks can be achieved with a relatively small set. On the other hand, both the latent goals that we
 161 learn as well as the action diffusion process are continuous. Making sure learnt skills are discrete can
 162 not only allow for better performance but also better generalization to new settings. However, naively
 163 enforcing discretization can lead to suboptimal behavior. We want to ensure that conditioned on a
 164 latent goal, g , action predictions from f_{act} are both multimodal and yet only represent a few modes.
 165 Thus, we propose a discrete *bottleneck* instead.

166 For the action generation process to represent a useful skill space, we want to enforce discreteness
 167 where the actions interact with latent goal. PlayFusion adds a vector quantization bottleneck in
 168 the diffusion process, specifically in the network $\epsilon_\theta(x) = \epsilon_\theta(g, s_t, a_t^0 + \epsilon^k, k)$. ϵ_θ is U-Net which
 169 fuses the language conditioning into the action denoising. We modify the U-Net architecture with
 170 a codebook of discrete latent codes e_u , a discrete bottleneck for the diffusion model. Given an
 171 input x the U-Net encoder produces a latent $\psi_\epsilon(x)$, which is passed into the decoder to produce
 172 $\epsilon_\theta(x) = \gamma_\epsilon(\psi_\epsilon(x))$. This bottleneck layer selects j as $\arg \min_i \|\psi_\epsilon(x) - e_i\|$, finding the closest
 173 code to the embedding, which is used to reconstruct x . To account for this, we augment the training
 174 procedure with the quantization and commitment losses, similar to VQ-VAE.

175 **Generalization via discrete language conditioning** Consider an agent that has learnt skills formed
 176 from the atomic units A, B, C and D, of the form A + B, B + C and C + D. To truly extend its

177 capabilities beyond the initial training data, the agent must learn to interpolate and extrapolate from
 178 these existing skills, being able to perform tasks like A + D that it hasn’t explicitly been trained on.
 179 Given that the action generation in the diffusion process is already quantized, our hypothesis is that
 180 a discrete goal space will be synergistic and allow the policy to compose skills better. Thus, we
 181 maintain a codebook of discrete latent codes e_l for the language embeddings output by the language
 182 goal network $\phi_l(z_l)$, selecting $e_{l,j}$ which is closest to $\phi_l(z_l)$. The full loss function that we use to
 183 train PlayFusion is as follows:

$$\begin{aligned}
 \mathcal{L}_{PlayFusion} = & \|\epsilon^k - \epsilon_\theta(x^0 + \epsilon^k, k)\|_2 + \beta_1 \underbrace{\|sg(\psi_\epsilon(x) - e_{u,j})\|_2}_{\text{U-Net quantization loss}} + \beta_1 \underbrace{\|\psi_\epsilon(x) - sg(e_{u,j})\|_2}_{\text{U-Net commitment loss}} \\
 & + \beta_2 \underbrace{\|sg(\phi_l(z_l)) - e_{l,j}\|_2}_{\text{lang. quantization loss}} + \beta_2 \underbrace{\|\phi_l(z_l) - sg(e_{l,j})\|_2}_{\text{lang. commitment loss}}
 \end{aligned}
 \tag{6}$$

184 where β_1 and β_2 are coefficients to determine the tradeoff between covering a diversity of possible
 185 behaviors and encouraging behaviors belonging to similar skills to be brought close to each other.

186 **Sampling from PlayFusion** Given a novel language instruction at test time z' , we obtain the
 187 quantized encoding $\phi_l(z')$, combining it with the visual encoding to get conditioning g' . We sample
 188 a set of actions $a_{t:t+k} \sim \mathcal{N}(0, 1)$, pass them through the discrete denoising process in Equation 4.

189 5 Experiments

190 In this section, we investigate PlayFusion and its ability to scale to complex tasks, as well as
 191 generalization to new settings. We ask the following questions: (1) Can PlayFusion allow for learning
 192 complex manipulation tasks from language annotated play data? (2) Can our method perform
 193 efficiently in the real-world setup beyond the simulated environment? (3) How well can PlayFusion
 194 generalize to out of distribution settings? (4) Can PlayFusion in fact learn discrete skills? (5) How do
 195 various design choices, such as quantization, language conditioning, etc., affect PlayFusion? We aim
 196 to answer these through experiments in three different simulation and real world settings.

197 **Environmental Setup** We test our approach across a wide variety of environments in both simula-
 198 tions as well as the real world. For simulation, we evaluate three benchmarks: (a) CALVIN [34], (b)
 199 Franka Kitchen [22], and (c) Language-Conditioned Ravens [35, 36]. For the real-world setup, we
 200 create three different environments: `cooking`, `dining table` and `sink`, shown in Figure 3. More
 201 details of the environment setup are in the supplementary.

202 **Baselines** We handle task conditioning in the same way for our method as well as all baselines,
 203 using the same visual and language encoders. We compare our method with the following baselines:
 204 (a) *Learning Motor Primitives from Play (Play-LMP)*: Play-LMP [21] generates behaviors by learning
 205 motor primitives from play data using a VAE, which encodes action sequences into latents and then
 206 decodes them into actions. (b) *Conditional Behavior Transformer (C-BeT)*: C-BeT [23] generates
 207 behaviors using a transformer-based policy and leverages action discretization to handle multimodality.
 208 (c) *Goal-Conditioned Behavior Cloning (GCBC)*: GCBC [21, 68] is conditional behavior cloning.

209 5.1 Results in Simulation and Real World

210 **PlayFusion in simulation** Table 1 shows success rates for PlayFusion, Play-LMP, C-BeT, and
 211 GCBC on the simulation benchmarks. On both CALVIN setups, we outperform the baselines by a
 212 wide margin, which demonstrates the effectiveness of our method in large-scale language-conditioned
 213 policy learning from complex, multimodal play data. The baselines perform comparatively better
 214 on the Franka Kitchen environments, where the training datasets are smaller and the data covers a
 215 more narrow behavior distribution and the benefit of handling multimodality is smaller; however,

	Simulation					Real World		
	CALVIN A	CALVIN B	Kitchen A	Kitchen B	Ravens	Dining Table	Cooking	Sink
C-BeT [23]	26.3 ± 0.8	23.4 ± 0.9	45.6 ± 2.3	24.4 ± 2.3	13.4	20.0	0.0	10.0
Play-LMP [21]	19.9 ± 1.0	22.0 ± 0.4	1.9 ± 1.5	0.0 ± 0.0	0.2	0.0	0.0	0.0
GCBC [21]	23.2 ± 2.0	30.4 ± 1.4	38.0 ± 3.3	15.5 ± 4.5	1.6	5.0	0.0	5.0
Ours	45.2 ± 1.2	58.7 ± 0.7	47.5 ± 2.0	27.7 ± 0.9	35.8	45.0	30.0	20.0

Table 1: Success rates for PlayFusion and the baselines on simulation and real-world settings. PlayFusion consistently outperforms all of the baselines.

216 PlayFusion still outperforms or matches all baselines. PlayFusion also achieves significantly higher
 217 success rate than the baselines on Ravens (see appendix for per-task results), which is not as large-
 218 scale as CALVIN but covers a large portion of the state space due to the diversity of instructions.

219 **Long horizon tasks** Using the Long Horizon
 220 CALVIN evaluation suite, we test the ability
 221 of agents to stitch together different tasks, with
 222 transitioning between tasks being particularly
 223 difficult. One such long horizon chain might be
 224 "turn on the led" → "open drawer" → "push
 225 the blue block" → "pick up the blue block"
 226 → "place in slider". We rollout 128 different
 227 long horizon chains containing five instructions
 228 each and record the number of instructions suc-
 229 cessfully completed. As shown in Table 2, we
 230 find that PlayFusion significantly outperforms
 231 the baselines in both CALVIN A and CALVIN
 232 B. The diffusion process gracefully handles the
 233 multimodality of not only each individual task
 234 in the chain but also of the highly varied data
 235 the agent has seen of transitions between tasks.

	Av. Seq Len	No. of Instructions				
		1	2	3	4	5
<i>CALVIN A :</i>						
C-BeT	0.262	25.2	1.0	0.0	0.0	0.0
Play-LMP	0.175	16.5	1.0	0.0	0.0	0.0
GCBC	0.194	19.4	0.0	0.0	0.0	0.0
<i>CALVIN B :</i>						
C-BeT	0.272	27.2	0.0	0.0	0.0	0.0
Play-LMP	0.117	11.7	0.0	0.0	0.0	0.0
GCBC	0.291	27.2	1.9	0.0	0.0	0.0
Ours (A)	0.417	37.1	2.9	1.0	0.0	0.0
Ours (B)	0.611	54.4	6.0	0.0	0.0	0.0

Table 2: Average sequence length on Long Horizon CALVIN and success rate for the n -th instructions.

236 **Generalization in the real world** Table 1 shows results for PlayFusion and the baselines in our
 237 real world evaluation setups. These setups are particularly challenging for two reasons: (1) inherent
 238 challenges with real-world robotics such as noisier data and constantly changing environment
 239 conditions such as lighting, and (2) they are designed to test skill-level compositional generalization.
 240 Specifically, the agents are required to compose skills A + B and C + D into A + D; for example,
 241 they might be trained on "pick up the carrot and place it in the pan" and "pick up the bread and put
 242 it in the toaster" and must generalize to "pick up the carrot and put it in the toaster". Our method
 243 significantly outperforms the baselines in these settings, showcasing the ability of the diffusion model
 244 in modeling complex distributions and the emergence of learned skills via the discrete bottleneck.
 245 Video results are at <https://play-fusion.github.io>.

246 5.2 Analysis of Discrete Representations

247 **Learning discrete skills** Table 3 studies the impact of
 248 our discrete bottlenecks (for Ravens results, see the ap-
 249 pendix). The success rate is, on average, worsened with the
 250 removal of either the U-Net discretization and the language
 251 embedding discretization. We also qualitatively study
 252 whether semantically similar skills are actually mapped to
 253 similar areas of the latent space and should therefore be
 254 brought together by the discrete bottleneck. In Figure 4,
 255 we show that skills involving similar locations (e.g., pan)
 256 or objects (e.g., carrot) are encoded into similar embeddings. In Figure 4, we show the embeddings



Figure 4: Visualization of the codebook embeddings for various real-world skills.

257 of different trajectories. The top two rows share the first skill (which is to remove the lid from the
 258 pan) and place an object in the pan. The bottom two rows share the second skill (grasping the carrot).
 259 Embeddings that contain the same skill have a similar pattern, which further indicates that the latent
 260 skill space being learned is somewhat discretized.

261 **Balancing the discrete bottlenecks** In Table 4, we study
 262 the effects of different β_1 and β_2 values on CALVIN A
 263 performance, i.e., the relative weightings for the additional
 264 terms in the loss function corresponding to the U-Net
 265 discretization and language embedding discretization. We
 266 find that $\beta_1 = \beta_2 = 0.5$ results in the best performance.
 267 In general, equally weighing the four additional losses
 268 (two for U-Net and two for language) leads to improved performance over imbalanced weightings.
 269 $\beta_1 = \beta_2 = 0.5$ is also better than $\beta_1 = \beta_2 = 1$, indicating that over-incentivizing discretization can
 270 be detrimental to diffusion model learning. Further analyses can be found in the appendix.

Methods	CALVIN A	CALVIN B
Ours	45.2 ± 1.2	58.7 ± 0.7
No U-Net discretiz.	45.3 ± 2.1	55.1 ± 1.4
No lang discretiz.	40.3 ± 1.6	54.1 ± 1.2

Table 3: Effect of discrete bottlenecks.

271 5.3 Ablations of Design Choices

272 **Effect of language model** Although our method is orthogonal to
 273 the language model used, we test its sensitivity to this. As shown in
 274 Table 4, we find that common models such as MiniLM [65], Distil-
 275 roberta [69], MPNet [70], and BERT [71] have similar performance,
 276 showing that PlayFusion is mostly robust to this design choice. We
 277 hypothesize that the discrete bottleneck applied to the language em-
 278 beddings helps to achieve this robustness. CLIP [72] embeddings
 279 result in much lower success rates, most likely due the fact that
 280 Internet images may not contain similar "play data" instructions.

	Success Rate
<i>Effect of conditioning:</i>	
Global	54.1
Conditional Noise	40.2
Visual Pre-training	38.1
<i>Effect of language model:</i>	
all-MiniLM-L6-v2	47.1
all-distilroberta-v1	48.4
all-mpnet-base-v2	48.8
BERT	48.8
CLIP (ResNet50)	35.2
CLIP (ViTB32)	43.9
<i>Loss weights (U-Net & Language) :</i>	
0.5 & 0.5	47.1
1 & 1	45.1
0.1 & 1	45.5
1 & 0.1	43.4
0.25 & 0.75	37.7
0.75 & 0.25	43.4

281 **Effect of conditioning** Table 4 studies various different possibil-
 282 ities for conditioning the diffusion model generations on language
 283 and vision in CALVIN A. When working with diffusion models there
 284 are multiple different ways we can approach how to feed it goals,
 285 images of the scene etc. We found that PlayFusion is mostly robust
 286 to this, with global conditioning providing benefits for smaller mod-
 287 els (such as those in the real world). We also attempted to condition
 288 the diffusion model noise on the goal but found that this negatively
 289 impacted performance. For the visual conditioning, we studied the
 290 effect of initializing the image encoder with large-scale pre-trained models [73]), finding that it does
 291 not help, and PlayFusion can learn the visual encoder end-to-end from scratch.

Table 4: Effects of conditioning, language model, and loss weights.

292 For data scaling curves and more analyses on design choices, see the appendix.

293 6 Limitations and Discussion

294 In this paper, we introduced a novel approach for learning a multi-task robotic control policy using
 295 a denoising diffusion process on trajectories, conditioned on language instructions. Our method
 296 exploits the effectiveness of diffusion models in handling multimodality and introduces two discrete
 297 bottlenecks in the diffusion model in order to incentivize the model to learn semantically meaningful
 298 skills. PlayFusion does require the collection of teleoperated play data paired with after-the-fact
 299 language annotations, which still require human effort despite being already less expensive and
 300 time-consuming to collect than demonstrations. It would be interesting to label the play data with a
 301 captioning model or other autonomous method. Furthermore, there is room for improvement in our
 302 performance on our real-world setups. Additionally, our real-world experiments could be expanded
 303 to even more complex household settings such as study rooms, bed rooms, and living rooms. Overall,
 304 our approach can significantly enhance the ability of robots to operate autonomously in complex and
 305 dynamic environments, making them more useful in a wide range of applications.

306 **References**

- 307 [1] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal
308 abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2):181–211, 1999.
- 309 [2] S. Thrun, C. Faloutsos, T. Mitchell, and L. Wasserman. Automated learning and discovery
310 state-of-the-art and research topics in a rapidly growing field. *Ai Magazine*, 20(3):78–78, 1999.
- 311 [3] M. Pickett and A. G. Barto. Policyblocks: An algorithm for creating useful macro-actions in
312 reinforcement learning. In *ICML*, volume 19, pages 506–513, 2002.
- 313 [4] P.-L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *AAAI*, 2017.
- 314 [5] R. S. Sutton, D. Precup, and S. Singh. Between mdps and semi-mdps: A framework for temporal
315 abstraction in reinforcement learning. *Artificial Intelligence*, 1999.
- 316 [6] C. Daniel, G. Neumann, O. Kroemer, and J. Peters. Hierarchical relative entropy policy search.
317 *Journal of Machine Learning Research*, 2016.
- 318 [7] F. Stulp, E. A. Theodorou, and S. Schaal. Reinforcement learning with sequences of motion
319 primitives for robust manipulation. *Transactions on Robotics*, 2012.
- 320 [8] J. Kober and J. Peters. Learning motor primitives for robotics. In *ICRA*, 2009.
- 321 [9] P. Pastor, M. Kalakrishnan, S. Chitta, E. Theodorou, and S. Schaal. Skill learning and task
322 outcome prediction for manipulation. In *ICRA*, 2011.
- 323 [10] M. Dalal, D. Pathak, and R. R. Salakhutdinov. Accelerating robotic reinforcement learning via
324 parameterized action primitives. *NeurIPS*, 2021.
- 325 [11] S. Nasiriany, H. Liu, and Y. Zhu. Augmenting reinforcement learning with behavior primitives
326 for diverse manipulation tasks. In *ICRA*, 2022.
- 327 [12] K. Pertsch, Y. Lee, and J. Lim. Accelerating reinforcement learning with learned skill priors. In
328 *Conference on robot learning*, pages 188–204. PMLR, 2021.
- 329 [13] S. Bahl, A. Gupta, and D. Pathak. Hierarchical neural dynamic policies. *RSS*, 2021.
- 330 [14] K. Pertsch, R. Desai, V. Kumar, F. Meier, J. J. Lim, D. Batra, and A. Rai. Cross-domain transfer
331 via semantic skill imitation. *arXiv preprint arXiv:2212.07407*, 2022.
- 332 [15] B. Eysenbach, A. Gupta, J. Ibarz, and S. Levine. Diversity is all you need: learning skills
333 without a reward function. *arXiv preprint arXiv:1802.06070*, 2018.
- 334 [16] A. Sharma, S. Gu, S. Levine, V. Kumar, and K. Hausman. Dynamics-aware unsupervised
335 discovery of skills. *arXiv preprint arXiv:1907.01657*, 2019.
- 336 [17] J. Merel, S. Tulyasuvunakool, A. Ahuja, Y. Tassa, L. Hasenclever, V. Pham, T. Erez, G. Wayne,
337 and N. Heess. Catch & carry: reusable neural controllers for vision-guided whole-body tasks.
338 *ACM Transactions on Graphics (TOG)*, 39(4):39–1, 2020.
- 339 [18] T. Shankar and A. Gupta. Learning robot skills with temporal variational inference. In
340 *International Conference on Machine Learning*, pages 8624–8633. PMLR, 2020.
- 341 [19] T. Kipf, Y. Li, H. Dai, V. Zambaldi, A. Sanchez-Gonzalez, E. Grefenstette, P. Kohli, and
342 P. Battaglia. Compile: Compositional imitation learning and execution. In *International
343 Conference on Machine Learning*, pages 3418–3428. PMLR, 2019.
- 344 [20] W. Whitney, R. Agarwal, K. Cho, and A. Gupta. Dynamics-aware embeddings. *arXiv preprint
345 arXiv:1908.09357*, 2019.

- 346 [21] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning
347 latent plans from play. *arXiv preprint arXiv:1903.01973*, 2019.
- 348 [22] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving
349 long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*,
350 2019.
- 351 [23] Z. J. Cui, Y. Wang, N. Muhammad, L. Pinto, et al. From play to policy: Conditional behavior
352 generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.
- 353 [24] A. Ramesh, M. Pavlov, G. Goh, S. Gray, C. Voss, A. Radford, M. Chen, and I. Sutskever.
354 Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages
355 8821–8831. PMLR, 2021.
- 356 [25] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image
357 generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022.
- 358 [26] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis
359 with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision
360 and Pattern Recognition*, pages 10684–10695, 2022.
- 361 [27] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam,
362 G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural
363 information processing systems*, 33:1877–1901, 2020.
- 364 [28] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. *Advances in Neural
365 Information Processing Systems*, 33:6840–6851, 2020.
- 366 [29] Z. Wang, J. J. Hunt, and M. Zhou. Diffusion policies as an expressive policy class for offline
367 reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.
- 368 [30] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior
369 synthesis. *arXiv preprint arXiv:2205.09991*, 2022.
- 370 [31] A. Ajay, Y. Du, A. Gupta, J. Tenenbaum, T. Jaakkola, and P. Agrawal. Is conditional generative
371 modeling all you need for decision-making? *arXiv preprint arXiv:2211.15657*, 2022.
- 372 [32] P. Hansen-Estruch, I. Kostrikov, M. Janner, J. G. Kuba, and S. Levine. Idql: Implicit q-learning
373 as an actor-critic method with diffusion policies. *arXiv preprint arXiv:2304.10573*, 2023.
- 374 [33] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy:
375 Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.
- 376 [34] O. Mees, L. Hermann, E. Rosete-Beas, and W. Burgard. Calvin: A benchmark for language-
377 conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and
378 Automation Letters*, 7(3):7327–7334, 2022.
- 379 [35] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin,
380 D. Duong, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for
381 robotic manipulation. *CoRL*, 2020.
- 382 [36] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipula-
383 tion. In *Conference on Robot Learning*, pages 894–906. PMLR, 2022.
- 384 [37] L. P. Kaelbling. Learning to achieve goals. In *IJCAI*, volume 2, pages 1094–8. Citeseer, 1993.
- 385 [38] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin,
386 O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. *Advances in neural information
387 processing systems*, 30, 2017.

- 388 [39] D. Ghosh, A. Gupta, A. Reddy, J. Fu, C. Devin, B. Eysenbach, and S. Levine. Learning to reach
389 goals via iterated supervised learning. *arXiv preprint arXiv:1912.06088*, 2019.
- 390 [40] A. Goyal, A. Friesen, A. Banino, T. Weber, N. R. Ke, A. P. Badia, A. Guez, M. Mirza, P. C.
391 Humphreys, K. Konyushova, et al. Retrieval-augmented reinforcement learning. In *International*
392 *Conference on Machine Learning*, pages 7740–7765. PMLR, 2022.
- 393 [41] L. Chen, K. Lu, A. Rajeswaran, K. Lee, A. Grover, M. Laskin, P. Abbeel, A. Srinivas, and
394 I. Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances*
395 *in neural information processing systems*, 34:15084–15097, 2021.
- 396 [42] C. Lynch and P. Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 40:
397 105, 2020.
- 398 [43] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. Bc-z:
399 Zero-shot task generalization with robotic imitation learning. In *Conference on Robot Learning*,
400 pages 991–1002. PMLR, 2022.
- 401 [44] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan,
402 K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic
403 affordances. *arXiv preprint arXiv:2204.01691*, 2022.
- 404 [45] S. Nair, E. Mitchell, K. Chen, S. Savarese, C. Finn, et al. Learning language-conditioned robot
405 behavior from offline data and crowd-sourced annotation. In *Conference on Robot Learning*,
406 pages 1303–1315. PMLR, 2022.
- 407 [46] O. Mees, L. Hermann, and W. Burgard. What matters in language conditioned robotic imitation
408 learning over unstructured data. *IEEE Robotics and Automation Letters*, 7(4):11205–11212,
409 2022.
- 410 [47] M. Shridhar, L. Manuelli, and D. Fox. Perceiver-actor: A multi-task transformer for robotic
411 manipulation. In *Conference on Robot Learning*, pages 785–799. PMLR, 2023.
- 412 [48] C. Lynch and P. Sermanet. Language conditioned imitation learning over unstructured data.
413 *arXiv preprint arXiv:2005.07648*, 2020.
- 414 [49] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*,
415 2013.
- 416 [50] D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate
417 inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.
- 418 [51] E. Zhang, Y. Lu, W. Wang, and A. Zhang. Lad: Language augmented diffusion for reinforcement
419 learning. *arXiv preprint arXiv:2210.15629*, 2022.
- 420 [52] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k
421 modes with one stone. *Advances in neural information processing systems*, 35:22955–22968,
422 2022.
- 423 [53] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent
424 dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.
- 425 [54] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent
426 imagination. *arXiv preprint arXiv:1912.01603*, 2019.
- 427 [55] Y. Dai, M. Yang, B. Dai, H. Dai, O. Nachum, J. Tenenbaum, D. Schuurmans, and P. Abbeel.
428 Learning universal policies via text-guided video generation. *arXiv preprint arXiv:2302.00111*,
429 2023.

- 430 [56] W. Liu, Y. Du, T. Hermans, S. Chernova, and C. Paxton. Structdiffusion: Language-guided 304
431 creation of physically-valid structures using unseen objects. *arXiv preprint arXiv:2211.04604*,
432 305:2, 2022.
- 433 [57] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker,
434 M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and
435 D. Hassabis. Mastering the game of go without human knowledge. *Nature*, 2017.
- 436 [58] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser,
437 I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural
438 networks and tree search. *nature*, 529(7587):484, 2016.
- 439 [59] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves,
440 M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou,
441 H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. Human-level control through
442 deep reinforcement learning. *Nature*, 518(7540):529–533, Feb. 2015.
- 443 [60] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An
444 evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279,
445 jun 2013.
- 446 [61] M. Janner, Q. Li, and S. Levine. Offline reinforcement learning as one big sequence modeling
447 problem. *Advances in neural information processing systems*, 34:1273–1286, 2021.
- 448 [62] P. Wu, A. Majumdar, K. Stone, Y. Lin, I. Mordatch, P. Abbeel, and A. Rajeswaran. Masked
449 trajectory models for prediction, representation, and control. *arXiv preprint arXiv:2305.02968*,
450 2023.
- 451 [63] S. Ozair, Y. Li, A. Razavi, I. Antonoglou, A. Van Den Oord, and O. Vinyals. Vector quantized
452 models for planning. In *International Conference on Machine Learning*, pages 8302–8313.
453 PMLR, 2021.
- 454 [64] A. van den Oord, O. Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, pages
455 6309–6318, 2017.
- 456 [65] N. Reimers and I. Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks.
457 *arXiv preprint arXiv:1908.10084*, 2019.
- 458 [66] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *CoRR*,
459 abs/1512.03385, 2015. URL <http://arxiv.org/abs/1512.03385>.
- 460 [67] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation
461 with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.
- 462 [68] Y. Ding, C. Florensa, M. Phielipp, and P. Abbeel. Goal-conditioned imitation learning. *arXiv*
463 *preprint arXiv:1906.05838*, 2019.
- 464 [69] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer,
465 and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint*
466 *arXiv:1907.11692*, 2019.
- 467 [70] K. Song, X. Tan, T. Qin, J. Lu, and T.-Y. Liu. Mpnet: Masked and permuted pre-training for
468 language understanding. *Advances in Neural Information Processing Systems*, 33:16857–16867,
469 2020.
- 470 [71] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional
471 transformers for language understanding. 2018.

- 472 [72] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell,
473 P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision.
474 In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- 475 [73] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation
476 for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.
- 477 [74] Calvin. <https://github.com/mees/calvin/>.
- 478 [75] Relay policy learning environments. [https://github.com/google-research/
479 relay-policy-learning/](https://github.com/google-research/relay-policy-learning/).
- 480 [76] Cliport. <https://github.com/cliport/cliport>.
- 481 [77] From play to policy: Conditional behavior generation from uncurated robot data. [https:
482 //github.com/jeffacce/play-to-policy](https://github.com/jeffacce/play-to-policy).

483 A Website

484 Video results are available at <https://play-fusion.github.io>.

485 B Experimental Setup

486 We evaluate our method on three simulated environments. Below, we provide their details.

487 **CALVIN [34].** The CALVIN benchmark tests a robotic agent’s ability to follow language instruc-
488 tions. CALVIN contains four manipulation environments, each of which include a desk with a
489 sliding door and a drawer that can be opened and closed, as well as a 7-DOF Franka Emika Panda
490 robot arm with a parallel gripper. The four environments differ from each other in both their spatial
491 composition (e.g., positions of drawers, doors, and objects) and visual features. The training data for
492 each environment contains around 200K trajectories, from which we sample a sequence of transitions
493 for each element of the minibatch. A portion of the dataset contains language annotations; we use
494 this subset to train our language-conditioned model. Each transition consists of the RGB image
495 observation, proprioceptive state information, and the 7-dimensional action. The agent is evaluated
496 on its success rate in completing 34 tasks, which include variations of rotation, sliding, open/close,
497 and lifting. These are specified by language instructions that are unseen during training in order
498 to test the generalization ability of the agent. We evaluate on two setups: (1) CALVIN A, where
499 the model is trained and tested on the same environment (called D→D in the benchmark) and (2)
500 CALVIN B, where the model is trained on three of the four environments and tested on the fourth
501 (called ABC→D in the benchmark).

502 **Franka Kitchen [22].** Franka Kitchen is a simulated kitchen environment with a Franka Panda
503 robot. It contains seven possible tasks: opening a sliding cabinet, opening a hinge cabinet, sliding
504 a kettle, turning on a switch, turning on the bottom burner, turning on the top burner, and opening
505 a microwave door. The dataset contains 566 VR demonstrations of humans performing four of the
506 seven tasks in sequence. Each transition consists of the RGB image observation, proprioceptive state
507 information, and the 9-dimensional action. We split each of these demonstrations into their four tasks
508 and annotate them with diverse natural language to create a language-annotated play dataset. In our
509 experiments, we evaluate agents on two setups within this environment, which we denote as Kitchen
510 A and Kitchen B. In Kitchen A, we evaluate an agent’s language generalization ability at test-time
511 by prompting it with unseen instructions asking it to perform one of the seven tasks. This requires
512 the model to identify the desired task and successfully execute it. Kitchen B is a more challenging
513 evaluation setting, where the agent must perform two of the desired seven tasks in sequence given an
514 unseen language instruction. In this setting, the agent must exhibit long-horizon reasoning capabilities
515 and perform temporally consistent actions, in addition to the language generalization required in
516 Kitchen A.

517 **Language-Conditioned Ravens [35, 36].** Ravens is a tabletop manipulation environment with a
518 Franka Panda arm. We evaluate on three tasks in the Ravens benchmark: putting blocks in bowls,
519 stacking blocks to form a pyramid, and packing blocks into boxes. The dataset consists of 1000
520 demonstrations collected by an expert policy. Although the dataset proposed in [36] contains language
521 instructions denoting which color block to move and the desired final location, they are not diverse
522 like human natural language annotations would be. In order to study our model’s performance on a
523 play-like language-annotated dataset, we instead annotate the demonstrations with diverse natural
524 language. At test-time, we prompt the agent with an unseen language instruction, similar to our other
525 setups.

526 B.1 Real World Setup

527 We create multiple play environments in the real world as well. We use a 7-DOF Franka Emika
528 Panda robot arm with a parallel gripper, operating in joint action space. We have three different

529 environments cooking, dining table and sink. All of these tasks are multi-step, i.e., in each the
 530 robot has to at least grab one object and put it in another, i.e. grab a carrot and put it inside the oven.
 531 In cooking, we test how the robot can handle articulated objects. It has to first open the oven, grill or
 532 pot, and then place an object properly inside. All of these objects have different articulations. Each
 533 of the placed objects (bread, carrot, knife, steak, spoon, etc.) have unique and different ways of being
 534 interacted with. In the sink, we test very precise manipulation skills, where the robot has to place
 535 objects in the narrow dish rack or hang objects (like mugs). In all of these settings, we test unseen
 536 goals (a combination of objects) that has never been seen before, as well as an instruction that has
 537 never been seen before. We provide more details in the Appendix.

538 B.2 Additional Analysis on Discretization Bottleneck

539 **Discretization ablation in Ravens.** Table 5 studies the impact of our discrete bottlenecks on the
 540 Ravens benchmark. The success rate is, on average, worsened with the removal of either the U-Net
 541 discretization and the language embedding discretization.

542 **Discretizing a portion of the latent.** It is possible to quantize only a portion of the U-Net latent
 543 representation. Table 6 shows results of discretizing only a portion (25% or 50%) of the latent. We
 544 find that discretizing 25% of the latent resulted in better performance. Discretizing the entire latent
 545 still works well, but discretizing a portion is a great way to balance encouraging skill learning and
 546 accurate denoising.

Table 5: Effect of discrete bottlenecks on Ravens tasks.

Methods	put-block-in-bowl	stack-block-pyramid	packing-box-pairs
Ours	63.6 ± 2.5	20.0 ± 0.0	24.0 ± 1.8
No U-Net discretization	65.5 ± 3.3	5.0 ± 2.3	18.5 ± 0.0
No lang discretization	4.1 ± 0.6	3.3 ± 2.7	7.5 ± 2.5

Table 6: Effect of discretizing different fractions of the U-Net representation.

Methods	Success Rate
Discretize 100% of latent	45.2 ± 1.2
Discretize 50% of latent	44.8 ± 0.1
Discretize 25% of latent	48.7 ± 0.8

547 B.3 Data Scaling Curves

548 Figure 5 shows data scaling curves.

549 **Effect of discrete bottlenecks.** Our method scales well with more data and performs very well even
 550 at 100K trajectories, which is half the size of the CALVIN A training dataset. The removal of the
 551 language discretization results in lower success rates across almost all dataset sizes. The removal of
 552 U-Net discretization is not as critical and can actually improve performance for very small datasets,
 553 but is on average harmful for larger datasets.

554 **Comparison to baselines.** Our method scales well with more data while C-BeT, Play-LMP, and
 555 GCBC perform poorly for all dataset sizes.

556 B.4 Dataset Details

557 **Real-world experiments.** For each environment we collected 250 episodes. This translates to around
 558 15 hours of data collection. We augmented the dataset by adding 3 or 4 variations for each language

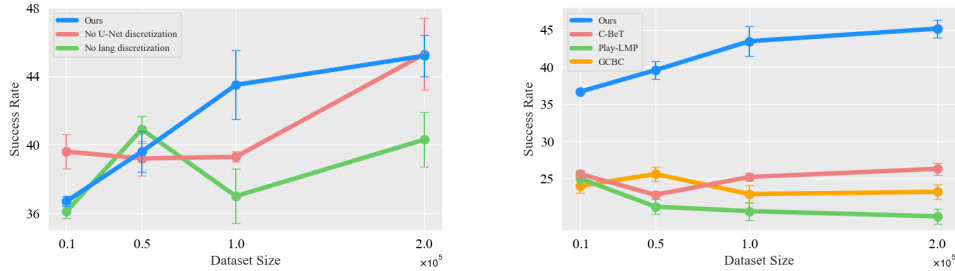


Figure 5: Data scaling curves. Left: effect of discrete bottlenecks. Right: comparison to baselines.

559 instruction (making the training dataset 750-1K episodes). The episodes were not broken into smaller
 560 annotated instructions.

561 **Simulation experiments.** We directly use the language-annotated dataset from CALVIN [34] and
 562 data generation script from CLIPort [36]. For Kitchen experiments, we used the dataset from Relay
 563 Policy Learning [22] and performed some processing and annotation to create language-annotated
 564 datasets. We provide some information in Table 7, but note that some of the numbers are estimates
 565 due to data processing procedures and refer the reader to the papers [34, 36, 22] for full details.

Table 7: Dataset details for simulation experiments.

	How was play data collected?	Hours	Eps. length	No. of lang. annotated eps.	Is a single eps. broken into smaller instructions?
CALVIN A	Teleoperators are instructed only to explore. Processing into episodes and annotating with language are done after-the-fact.	2.5	64	5K (instructions are repeated to create 200K training episodes)	No. Training trajectories are length-16 sub-episodes of the length-64 episodes. Instructions are repeated for all sub-episodes to create a total of 200K language-annotated training trajectories. (However, the length-64 window was sampled from a long stream of play data).
CALVIN B	Same as CALVIN A, but for three different environments.	7.5	64	15K (instructions are repeated to create 600K training episodes)	No. Same as CALVIN A, but for three different environments, for a total of 600K training trajectories.
Kitchen A	Teleoperators are instructed to perform 4 out of 7 possible tasks for each episode.	1.5	200	566 (split to create 2.2K training episodes)	Yes. We split each episode into the four training trajectories and label each of them with language.
Kitchen B	Same as Kitchen A.	1.5	200	566 (split to create 1.6K training episodes)	Yes. We split each episode into three training trajectories (one for each pair of consecutive tasks) and label each of them with language.
Ravens	Data is generated by rolling out an expert policy.	3	Up to 20	1000	Depends on the task. If it is sequential then the instruction changes throughout the episode and if it is single-step then there is one instruction for the episode.

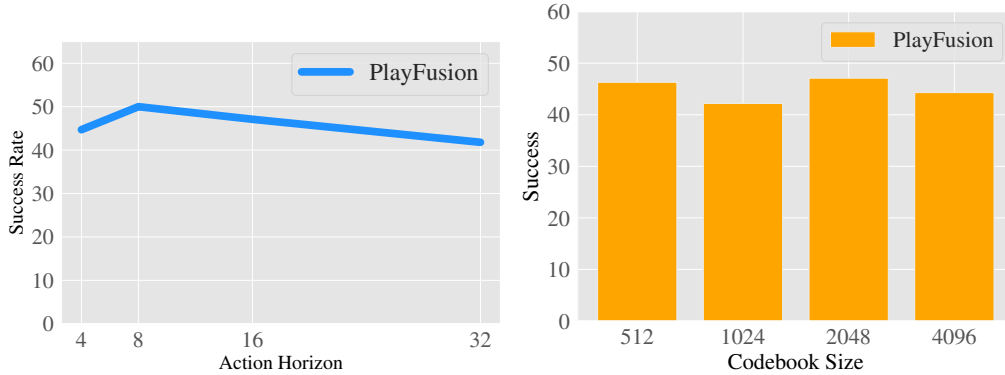


Figure 6: Effect of model design choices.

566 B.5 Model Design Choices

567 Figure 6 studies the impact of action horizon and codebook size in CALVIN A. PlayFusion is mostly
 568 robust to the action horizon T_a . We empirically found t_a of around 20% of the overall horizon worked
 569 the best. We find that PlayFusion is relatively robust to the discrete latent codebook sizes.

570 Note that asymptotically, increasing the codebook size would remove the discrete bottleneck, in
 571 principle. To study whether this happens in practice, we further increased the codebook size and show
 572 CALVIN A results in Table 8. As expected, performance drops when codebook size gets very large.

573 Table 9 shows the effect of number of diffusion timesteps in CALVIN A. We found that using 25
 574 timesteps works slightly better but our method is generally robust to this hyperparameter.

Table 8: Effect of further increasing the codebook size.

Codebook Size	Success Rate
2048	45.2 ± 1.2
8192	46.0 ± 1.2
16384	41.1 ± 0.1

Table 9: Effect of diffusion timesteps.

Timesteps	Success Rate
50	45.2 ± 1.2
100	39.9 ± 1.3
25	47.4 ± 0.8

575 B.6 Generalization to Unseen Skills

576 We performed an experiment where we removed one skill from the CALVIN training data. Specif-
 577 ically, we removed lift-red-block-slider from the training data and tested the model’s ability to
 578 interpolate between (1) lifting other blocks from the slider (e.g., lift-blue-block-slider, lift-pink-block-
 579 slider) and (2) lifting red blocks in other scenarios (e.g., lift-red-block-drawer, lift-red-block-table).
 580 We also repeated this experiment for lift-blue-block-table. We find that the removal of the discrete
 581 bottlenecks results in generally worse performance in this challenging setup (see Table 10). Al-
 582 though confidence intervals do overlap a bit, we find that our method is on average the best for both
 583 lift-red-block-slider and lift-blue-block-table.

Table 10: Performance on unseen skills.

Models	lift-red-block-slider	lift-blue-block-table
Ours	20.0 \pm 8.1	16.6 \pm 7.2
No U-Net discretization	10.0 \pm 4.6	3.3 \pm 2.7
No lang discretization	13.3 \pm 2.7	13.3 \pm 5.4

584 B.7 Ravens Experiments

Table 11 shows per-task success rates for Ravens.

Table 11: Per-task success rates for Ravens.

	put-block-in-bowl	stack-block-pyramid	packing-box-pairs
C-BeT	17.2 \pm 1.1	15.0 \pm 2.3	8.1 \pm 1.5
Play-LMP	0.0 \pm 0.0	0.0 \pm 0.0	0.8 \pm 0.2
GCBC	0.0 \pm 0.0	3.3 \pm 2.7	1.7 \pm 0.7
Ours	63.6 \pm 2.5	20.0 \pm 0.0	24.0 \pm 1.8

585

586 B.8 Implementation Details

587 Table 12 shows the main hyperparameters of our model in our simulation and real world experiments.
 588 We build off of the implementation of MCIL from CALVIN [74]. For Franka Kitchen and Ravens
 589 dataset and environment processing, we use implementations from [75] and [76], respectively. For
 590 implementations of the baselines, we modify [77] for C-BeT and [74] for Play-LMP and GCBC.
 591 Where possible, we use the same hyperparameters for PlayFusion and the baselines.

Table 12: Hyperparameters of PlayFusion in our simulation and real-world experiments.

Hyperparameter	CALVIN	Franka Kitchen	Ravens	Real World
Batch size	32	32	128	12
Codebook size	2048	2048	2048	2048
U-Net discretiz. wgt	0.5	0.5	0.5	0.5
Lang. discretiz. wgt	0.5	0.5	0.5	0.5
Action horizon T_a	16	64	2	32
Context length T_o	2	1	1	1
Language features	384	384	384	384
Learning rate	1e-4	2.5e-4	2.5e-4	2.5e-4
Diffusion timesteps	50	50	50	50
Beta scheduler	squaredcos_cap.v2	squaredcos_cap.v2	squaredcos_cap.v2	squaredcos_cap.v2
Timestep embed dim	256	256	128	256