QueST: Self-Supervised Skill Abstractions for Learning Continuous Control

Atharva Mete **Haotian Xue** **Albert Wilcox** **Yongxin Chen** **Animesh Garg**

Georgia Institute of Technology

Abstract

Generalization capabilities, or rather a lack thereof, is one of the most important unsolved problems in the field of robot learning, and while several large scale efforts have set out to tackle this problem, unsolved it remains. In this paper, we hypothesize that learning temporal action abstractions using latent variable models (LVMs), which learn to map data to a compressed latent space and back, is a promising direction towards low-level skills that can readily be used for new tasks. Although several works have attempted to show this, they have generally been limited by architectures that do not faithfully capture sharable representations. To address this we present Quantized Skill Transformer (QueST), which learns a larger and more flexible latent encoding that is more capable of modeling the breadth of low-level skills necessary for a variety of tasks. To make use of this extra flexibility, QueST imparts causal inductive bias from the action sequence data into the latent space, leading to more semantically useful and transferable representations. We compare to state-of-the-art imitation learning and LVM baselines and see that QueST's architecture leads to strong performance on several multitask and few-shot learning benchmarks. Further results and videos are available at https://quest-model.github.io.

1 Introduction

One of the grand goals of robotic learning is a general-purpose model that can learn from complex multitask demonstration data and generalize to new tasks in a zero-shot or few-shot manner. While such general-purpose models have become ubiquitous in natural language (NLP) [65, 71, 51, 52, 62] and computer vision (CV) [76, 32, 7], they have eluded robotics researchers. Whereas CV and NLP can achieve positive transfer by scaling up models trained on internet scale datasets [32, 71, 55, 61, 38], even large scale robot data collection efforts [10, 11, 46, 20, 30] have been insufficient for this approach. To that end, we posit that in order to achieve positive transfer in robotics, it is important to design architectures that specifically lend themselves to efficient cross-task transfer.

There has recently been a surge of work towards the goal of learning generalist policies from large, diverse datasets. Several papers have used techniques such as action discretization [56, 16, 34, 10, 11, 46, 54] and implicit models [23, 15, 25] to model multimodal action distributions. In particular, the behavior transformer line of work shows that a carefully discretized action space combined with a GPT-style transformer leads to impressive capabilities modeling multimodal behavior distributions [56, 16, 34]. In another vein, several works have attempted to scale demonstration data and achieve positive transfer of low-level skills between high-level tasks [10, 11, 46, 45, 18, 12, 54, 3]. While these works have shown some transfer, for example applying policies for known tasks to unfamiliar objects, they have generally failed to achieve transfer of low-level skills to novel tasks [4]. We

[1]Correspondence to: amete7@gatech.edu

Preprint. Under review.

hypothesize that in the relatively low-data regime of robot learning, it is promising to explicitly force the model to learn sharable representations. To that end we study latent variable models (LVMs), which learn to map data to a compressed latent space and back, introducing an information bottleneck which encourages the model to learn shared representations across the training data. Specifically we consider the application of LVMs to learn low-dimensional representations of action sequences. Such representations are termed temporal action abstractions or motion primitives – in this paper we refer to these abstractions as 'skills'.

A wide body of work has considered the application of LVMs to robotics. One line of work learns temporal action abstractions (skills) in continuous latent space with a Gaussian prior [40, 49, 58]. While this line of work showed some initial promise of learning latent plans, it has failed to scale to difficult multitask settings due to the loose nature of the latent structure and posterior collapse issues that inhibit the learning of shared representations. On the other hand, recent work in CV and NLP has shown that vector-quantized discrete latent spaces are capable of learning semantically meaningful representations from data like phonetics in speech [9, 6] or melody in music [17, 2]. This insight, along with prior work showing that discretized action spaces can help to address the multimodality problem in when learning from large datasets [56, 16, 34, 14], motivates methods learning temporal action abstractions with discrete latent spaces. Several recent works have set out to do this [34, 73, 75, 29], showing some degree of positive transfer between tasks in multi-task and few-shot settings. However, they are generally limited by architectures that do not faithfully capture transferable representations [34, 73], or depend on state prediction and state-based objective functions which are impractical for many real robot tasks [75, 29].

In this paper, we present **Qu**antize**d** **S**kill **T**ransformer (QueST), a simple yet novel architecture for learning generalizable low-level skills within a discrete latent space. The key insight behind QueST is its ability to flexibly capture variable length motion primitives by representing them with a sequence of discrete codebook entries. We achieve this through a unique encoder-decoder architecture primarily designed to impart causal inductive bias in action sequence data into the latent space. Such formulation enables us to employ powerful sequence modeling approaches to plan and composably reason within the space of low-level skills. Through our experiments, we show that autoregressive modeling of these latent skills with a GPT-like transformer outperforms state-of-the-art baselines on challenging robotic manipulation benchmarks, where QueST shows an 8% improvement in multitask and 14% improvement in few-shot imitation learning over the next best baselines. We also conduct a detailed ablation and sensitivity study to validate our key architectural design decisions.

2 Related Works

The proposed framework in this paper introduces a methodology for self-supervised skill abstraction, followed by decision-making within this skill space. Several related works have explored similar sub-directions such as decision-making in the latent space and decision making with a transformer:

2.1 Learning from Offline Data

Behavior cloning (BC) [50] aims to learn a policy by directly mapping observations to actions, and is typically trained end-to-end using pre-collected pairs of observation and behavior data. While this on its surface is a simple supervised learning problem, there are several properties of robot demonstration data that should be considered when building BC systems. First, large BC datasets collected from a variety of human demonstrators tend to contain data sampled from multimodal distributions. To address this, some works opt to sample actions from Gaussian Mixture Models (GMM) [42], while others explore implicit models including those derived from energy-based models [23, 28] or diffusion models [15, 70, 14, 67, 27]. The Behavior Transformer (BeT) line of work [56, 16, 34] shows that transformer-based categorical policies in carefully discretized action spaces do a good job handling multimodal demonstrator distributions and QueST builds upon this by contributing a more capable discrete latent skill model.

Another key property of robot demonstration data is that sequential actions are often highly correlated with one another, and exploiting this can lead to stronger performance while ignoring it can lead to policies which are susceptible to temporally correlated confounders [60]. Recently several works have set out to handle this by predicting action chunks. For example, the Action Chunking Transformer (ACT) line of work [72, 24] shows that a transformer trained as a CVAE [59] to output chunks of

actions performs well for a wide variety of manipulation tasks, and diffusion policy [15] shows across the board improvements when predicting action chunks. As discussed in detail in Section 2.3, QueST builds on a long line of work which handles sequential correlations through temporally-extended action abstractions [40, 49, 58, 34, 73, 75, 29].

2.2 Multi-task and Few-shot Imitation Learning

In the past, robot learning researchers have approached multi-task decision making settings using a wide variety of methods such as supervised pre-training and fine-tuning [20, 41], meta-learning [22, 19] and action retrieval [47, 43]. There has recently been a large focus on multi-task language-conditioned imitation learning for robotics with several papers attempting to address the problem by training large models on large demonstration datasets [10, 11, 46, 45, 18, 12, 1]. While these papers achieve impressive multitask results, they mostly rely on sufficient data coverage and fail to generalize beyond their training distribution [4]. Thus, they lack abstractions that can readily be applied to learn new tasks, especially in a low-data regime. On the other hand LVMs like QueST are designed to learn sharable representations that can be applied to new tasks.

2.3 Decision Making in Learned Latent Spaces

LVMs, modeled by a paired encoder-decoder, have found extensive applications in computer vision [64, 31, 8] and generative models [55, 21, 13]. Recent studies also demonstrate the utility of latent space representations in robot decision-making, spanning offline RL [57, 49, 39], imitation learning [14, 66, 36, 34], and temporal action abstraction [56, 73, 74]. Most similar to our work are those that learn temporally abstracted discrete latent skill spaces. PRISE [73] learns single-step state-action abstractions within a discrete space and then does temporal abstraction by applying BPE tokenization. While this method shows promise in learning multi-task and few-shot policies, BPE is well suited for text and is known to suffer in domains with highly dynamic vocabularies, in robotics its equivalent to varying action distributions across unseen tasks. TAP [75] and H-GAP [29] utilize a self-supervised auto-encoder to learn skill codes, but their functionality relies on Model Predictive Control (MPC) using state prediction with ground-truth state-conditioned objective functions, which make it difficult to apply to real-world manipulation tasks. VQ-BeT [34] bears the strongest resemblance to QueST, also using a pre-trained discrete latent skill space to discretize the action space for a transformer-based policy prior. However, their quantization approach does not leverage the inherent structure in action sequences, limiting the representational capabilities of the latent space. Thus it's shown to heavily rely on an continuous offset predictor for best performance. Unlike these works, QueST's learned latent space is highly flexible yet structured and expressive, allowing it to effectively model action distribution across many distinct tasks in a meaningful shared representation.

3 Preliminary

3.1 Problem Setting

We consider a dataset $D = \{\{(O_1, a_1), \ldots, (O_{T_i}, a_{T_i})\}_{i=0}^{N_k}, L_k\}_{k=0}^{M}$ where a_t is a continuous-valued action and O_t is a tuple consisting of a high-dimensional sensory observation. The data is collected via either human teleoperation or scripted policies for M different task each with a label L_k. In our setting, O_t consist of RGB image observations from the front camera and gripper camera (if available) along with proprioceptive state of the agent. L_k is a natural language description of the k^{th} task but can also be a one-hot encoding.

3.2 Finite Scalar Quantization

We build on Finite Scalar Quantization (FSQ) [44] as a discrete bottleneck in our model. It's a drop-in replacement for Vector Quantization (VQ) layers in VAEs with a simple scalar quantization scheme. Here the input representation e is projected to very few dimensions (typically 3 to 5) which are then bounded and rounded, creating an implicit codebook.

$$z = \text{round\_ste}(f(e)), \text{ where } f \text{ is the bounding function} \tag{1}$$

Given a feature vector $e \in \mathbb{R}^d$ to quantize, instead of learning a parameterized codebook [64] and quantizing e by matching the nearest neighbor in the codebook, FSQ quantizes e by first bounding

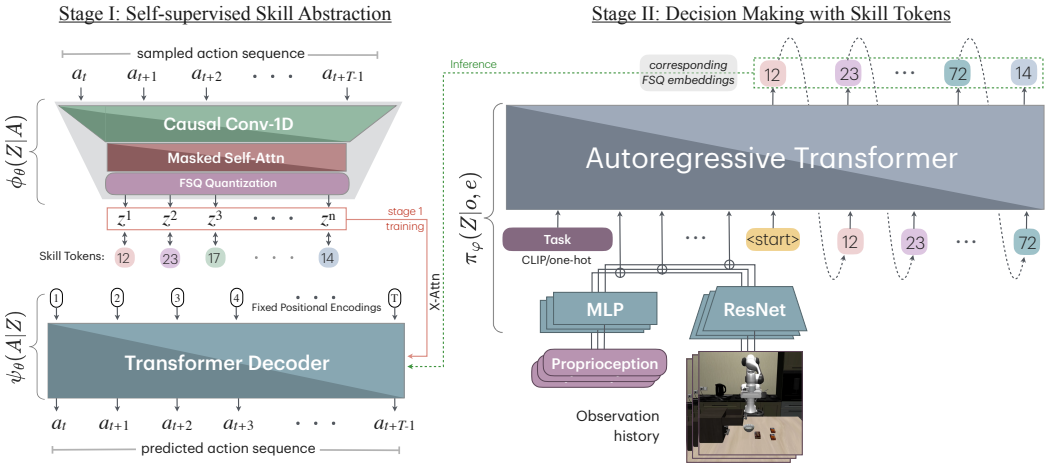


Figure 1: **Overview of Quantized Skill Transformer**: we factorize the policy that outputs action based on task descriptions e and observations o encoding into two parts: $\pi(A|o,e) = \psi_\theta(A|Z)\pi_\varphi(Z|o,e)$, where Z is a sequence of skill tokens for the action sequence A. In Stage I, we learn skill abstraction in a self-supervised way with a quantized autoencoder. In Stage II, we learn skill-based policy in the style of next-token prediction using a multi-modal transformer.

it into certain range with f (e.g. $f = \lfloor \alpha/2 \rfloor \odot \tanh(e)$) and then rounding each dimension into integer numbers directly. $\alpha \in \mathbb{Z}^d$ defines the width of the codebook for each dimension (e.g. $\alpha = [8,5,5,5], d = 4$). Finally, it is easy to see that the size of the quantization space is $\Pi_{i=1}^{d} \alpha_i$. An MLP can be used to transform z into continuous space of required dimension further.

A common problem with vector-quantized codebooks (VQ) [64] is the under-utilization of the codebook. Recent works have attempted to address by heuristics like reinitializing the codebook, stochastic formulations, or some regularization [33, 68]. In contrast, FSQ achieves much better codebook utilization for large codebook sizes with much fewer parameters and simplified training without any auxiliary losses or aforementioned tricks. Due to its simplicity and proven benefits, we use FSQ in our main experiments, but since many prior works in this space use VQ we also perform an ablation with it (see section 5.5).

4 Method

In this section, we describe the key ideas behind Quantized Skill Transformer. In Section 4.1 we present our encoder-decoder architecture, which is designed to provide the flexibility to learn a wide range of skills with inductive biases to ensure that the learned skills are useful. In Section 4.2, we detail our skill prior, which we train to autoregressively predict codebook skills. Our full pipeline is shown in Figure **??**.

4.1 Stage I: Learning the Skill Codebook

As a motivating example, consider the task of lifting a pot and placing it on a stove beside. This consist of primitives like reaching the pot, grasping it, lifting it to a certain height, reaching the stove and finally placing it on the stove. Each of these primitives are of variable lengths, and to properly model these skills it is important to learn a latent skill space with the flexibility to model all of them. At the same time, it is important that the learned skills are semantically meaningful so that they can be reused for new tasks, for example reusing the reaching skill for an object lifting task. In order to address these desiderata, we introduce the novel autoencoder architecture shown in Figure 1 consisting of an encoder ϕ_θ and decoder ψ_θ.

The input to the encoder ϕ_θ is an action sequence $a_{t:t+T-1}$ sampled from the dataset, which we pass through several 1D causal strided-convolution layers [63]. This step reduces the sequence length to achieve the desired temporal abstraction depending on the stride lengths and the number of layers. We follow the convolutional layers with masked self-attention layers for sequence modeling. With

a downsampling factor of F, the encoder outputs in total $n = T/F$ embeddings. The embeddings are then quantized using FSQ as per the equation 1 into n discrete latent codes $\{z^i\}$ termed as skill tokens:

$$(z^1, \ldots, z^n) = \text{FSQ}(\phi_\theta(a_t, \ldots, a_{t+T-1})). \tag{2}$$

Having an input sequence of actions mapped to multiple skill tokens gives this architecture more flexibility to model complex sequences of actions. At the same time, each component of the encoder is causal, meaning that an output representation at a position t cannot depend on input from any future timesteps. We found this inductive bias to encourage the model to learn semantically useful action representations by modeling the inherent causality in the action data. We validate this design choice in the ablations. (see section 5.5)

Typical autoencoder decoders are simply mirrored versions of the encoders, but this would prevent the decoder from attending to all quantized codes. This is important because individual codes do not represent anything meaningful but a sequence of codes represents a particular meaningful motion [44]. In order to maintain causality while attending to all codes, the decoder ψ_θ cross attends between fixed sinusoidal positional embedding inputs and the skill tokens, similarly with [72]. The architecture is a transformer decoder block consisting of alternate masked self-attention and cross-attention layers, after which the output embeddings are projected back to the original action dimension using an MLP layer. Thus, given a sequence Z of skill codes, ψ_θ reconstructs the original action

$$(\hat{a}_t, \ldots, \hat{a}_{t+T-1}) = \psi_\theta(z^1, \ldots, z^n) \tag{3}$$

As in [34], the autoencoder is trained by minimizing the ℓ_1 reconstruction loss:

$$\mathcal{L}_{\text{recon}}(\theta) = \|\psi_\theta(\text{FSQ}(\phi_\theta(a_{t:t+T-1}))) - a_{t:t+T-1}\|_1. \tag{4}$$

Unlike prior work which often conditions on the state as well as the actions [29, 5, 73, 36], we choose to learn state-independent abstractions that solely capture motion primitives irrespective of the current scene or task. Through our experiments we show that our model learns generalizable abstractions that are shared and can be transferred across tasks.

4.2 Stage II: Learning the Skill Prior

After training the encoder ϕ_θ and decoder ψ_θ, we train a skill prior $\pi_\varphi(Z|e, o)$ to predict skills $Z = z^{1:n}$ corresponding to the demonstrator action distribution conditioned on a task embedding e and a length h sequence of image observations and proprioception inputs, $o = (i_{t-h}, p_{t-h}), \ldots, (i_t, p_t)$. We encode image observations with a separate learned vision encoder for each camera view and encode proprioception using an MLP encoder, all of which are trained end-to-end with the rest of the skill prior. The observation token $\mathcal{T}^o_t$ for a timestep t is obtained by concatenating outputs from all the aforementioned encoders. Task embeddings are designed specifically for each environment suite, as discussed in more detail in Section 5. See Appendix B for more details about the encoders.

Because skill tokens are highly dependent on one another according to the complex nonlinear representations learned by the autoencoder, it is important that the skill prior has the modeling capacity to reason about these dependencies. To achieve this, we employ a decoder-only transformer to model the distribution of skill tokens $\pi_\varphi(Z|\mathcal{T}^o_{t-h:t}, e)$ autoregressively as:

$$\pi_\varphi(Z|\mathcal{T}^o_{t-h:t}, e) = \prod_{i=1}^{n} \pi_\varphi(z^i|\texttt{<s>}, z^{1:i-1}, \mathcal{T}^o_{t-h:t}, e) \tag{5}$$

where `<s>` is a learnable start token that marks the start of skill tokens. We add sinusoidal positional embeddings only to the skill tokens. To optimize the skill prior, we sample a sequence of demonstrator actions $a_{t:t+T-1}$ and use the trained encoder ϕ_θ to extract a latent skill vector $Z_t = z^{1:n}$ according to Equation 2. Then, we optimize π_φ using the following negative log-likelihood loss:

$$\mathcal{L}_{\text{task}}(\varphi) = -\log \pi_\varphi(Z_t|\mathcal{T}^o_{t-h:t}, e). \tag{6}$$

The full skill prior pipeline is shown in Figure 1.

Few-Shot Finetuning: For few-shot finetuning on new tasks, we use a model pre-trained on large set of tasks and finetune it on a small number of demonstrations (5 in our experiments) from the held-out task. Although finetuning only stage-2 is enough, we empirically found that finetuning the

decoder on the predicted skill tokens gives a boost in the performance. Specifically, we finetune the decoder using following decoder loss:

$$\mathcal{L}_{\text{decoder}}(\theta) = \|\psi_\theta(\text{sg}(\hat{Z}_t)) - a_{t:t+T-1}\|_1 \tag{7}$$

where sg is the stop gradient operator. We present the results with and without decoder finetuning both. Additionally, we note that the encoder is still frozen in this setting.

4.3 Inference with Quantized Skill Transformer

At inference time, QueST uses the skill prior π_φ alongside the decoder ψ_θ to sample actions. Conditioned on the encoded observation history $\mathcal{T}^o_{t-h:t}$ and task embedding e, we use top-k sampling with a temperature of τ to autoregressively sample a skill vector $\hat{Z} \sim \pi_\varphi(\cdot|\mathcal{T}^o_{t-h:t}, e)$ from the skill prior. In practice, we find $k = 5$ and $\tau = 1$ to work well across all environments. Then, we use the decoder to map the skill vector back to the action space, producing a sequence of predicted actions $\hat{a}_{t:t+T-1} = \psi_\theta(\hat{Z})$. In a receding horizon fashion, we execute the first $T_a \leq T$ actions before replanning.

5 Experiments

We design the experiments to empirically evaluate the performance of Quantized Skill Transformer in three practical settings: (1) Multitask IL, (2) Few-shot adaptation, and (3) Long-horizon IL. Lastly, we perform some ablations to empirically justify our model design choices.

5.1 Benchmarks and Baselines

We use the following benchmark suites to evaluate in the settings discussed above:

LIBERO [37] is a lifelong learning benchmark featuring several task suites consisting of a variety of language-labeled rigid- and articulated-body manipulation tasks. Specifically, we evaluate on the LIBERO-90 suite, which consists of 90 manipulation tasks, and the LIBERO-LONG suite, which consists of 10 long-horizon tasks composed of two tasks from the LIBERO-90 suite. As described in more detail below, we use the LIBERO benchmark to study the multitask IL, few-shot adaptation and long-horizon IL settings. Because tasks from this benchmark are language-annotated, we use the output of a frozen CLIP [53] encoder for the task conditioning input e.

MetaWorld [69] features a wide range of manipulation tasks designed to test few-shot learning algorithms. We use the Meta-Learning 45 (ML45) suite which consists of 45 training tasks and 5 difficult held-out tasks which are structurally similar to the training tasks. We use this benchmark to test multi-task and few-shot learning. Because this benchmark does not include language labels, we use learned task embeddings e for task conditioning.

Baselines: We compare to the following baselines, which include similar discrete LVM pipelines as well as state of the art imitation learning algorithms:

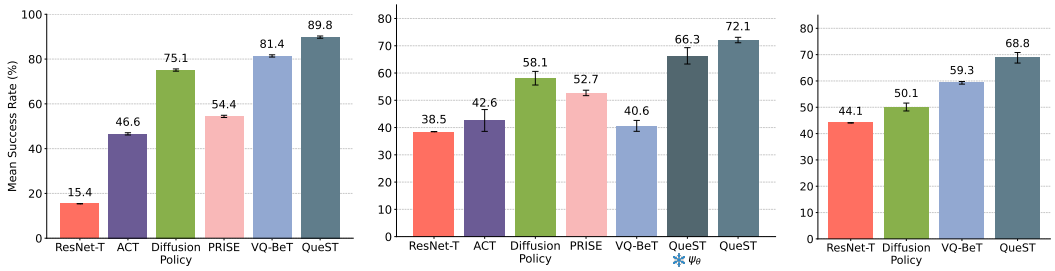
1. The **ResNet-T** model from [37], which encodes observation and task instructions using ResNet-18 with FiLM [48], applies a transformer sequence model, and uses a GMM head to predict actions.

2. The UNet-based **diffusion policy** from [15], which uses a 1D convolutional UNet to map samples from a Gaussian prior to action samples from the demonstrator distribution according to a learned denoising process.

3. **ACT** [72], which trains a transformer as a CVAE [59] to predict action chunks.

4. **VQ-BeT** [34], which learns a discrete latent space using a VQ-VAE [64] and uses a transformer to predict discrete latent codes.

5. **PRISE** [73], which first quantizes observation-action pairs and performs temporal abstraction using byte pair encoding (BPE) to learn a skill token vocabulary, which it uses as an action space for few-shot learning.



(a) Multitask on LIBERO-90. (b) Few-shot on LIBERO-LONG. (c) Multitask on LIBERO-LONG

Figure 2: Multitask performance on LIBERO-90 (a) and LIBERO-LONG (c), and few shot performance on LIBERO-LONG (b). For (a) and (c) we train on the datasets described in Sections 5.2 and 5.4. For (b) we finetune the model from (a) on a condensed dataset as described in Section 5.3. Results show the mean and error bar represents standard deviation across four random seeds. Results for ACT and PRISE are taken from Zheng et al. [73], and the others we reimplemented and ran ourselves.

5.2 Performance on Multitask BC

We evaluate the goal-conditioned multi-task imitation learning capabilities of QueST and the baselines using the LIBERO-90 and ML45 benchmark suites. For LIBERO-90, the learner receives 50 expert demonstrations per task from the author-provided dataset. For ML45, we use the provided scripted policies to collect 100 demonstrations per task. We evaluate the model at the end of training and for each task run 40 evaluation rollouts (50 for MetaWorld) starting from the initial states selected sequentially from a predefined set. We report the aggregated results across 4 seeds (5 seeds for MetaWorld).

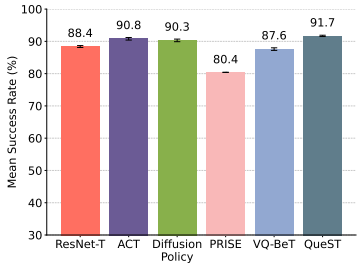
In Figure 2a, we present the average success rate across 90 tasks in LIBERO-90 against the aforementioned baselines. Quantized Skill Transformer achieves state-of-the-art results on LIBERO-90 benchmark, outperforming the best-performing baseline (VQ-BeT) by a margin of 8%. We attribute its performance to its learned latent space that enables effective knowledge sharing across tasks. While VQ-BeT also shows strong performance, we see that QueST's architecture lends itself better to sharing representations across tasks. Overall all LVM baselines outperform ResNet-T, a simple BC baseline, suggesting that it's a more effective modeling approach especially in a low-data regime like robotics. Figure 3a shows the average success rate across 45 tasks in ML45 benchmark. Being a simpler benchmark, all methods perform almost similar in Multitask-IL setting which is consistent with the trend observed in [73].

We attribute the reasonably good performance of the diffusion policy to its nature as a latent variable model, which employs a continuous latent variable with the same dimensionality as the actions. However, the consistent outperformance of QueST over the diffusion policy provides compelling evidence for the benefits of using a bottlenecked latent variable. This bottleneck encourages the model to learn shared representations, resulting in enhanced performance. While both VQ-BeT and PRISE employ a latent bottleneck, VQ-BeT's architecture neglects the inherent inductive biases in the action data, which we believe results in a less well-structured latent space. PRISE incorporates this using a latent forward transition model, but is bottlenecked by the use of BPE which we posit is not suitable for such a dynamic latent space.

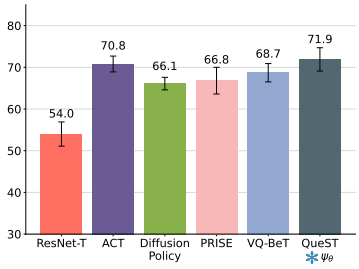
5.3 Few-shot Adaptation to Unseen Tasks

In this setting we take the pretrained model from section 5.2 and test its 5-shot performance on unseen tasks from LIBERO-LONG and held-out set in ML45. We sample only five demonstrations for each task, generate the skill tokens using pretrained encoder and use them to finetune the skill prior and the decoder as described in Section 4.2. We also present the results without finetuning the decoder (frozen ψ_θ in figure 2b & 3b) to validate its generalization to unseen skill tokens sequences.

Figure. 2b shows the average success rate for 5-shot IL across 8 unseen tasks in LIBERO-LONG. QueST achieves SOTA performance, surpassing all other baselines by an absolute margin of 14%. Though we see a drop of 6% without decoder finetuning, it still outperforms all the baselines. These results highlight the superiority of QueST in learning transferable representations of action abstractions and effectively leveraging them for downstream decision making. For a fair comparison,



(a) Multitask on ML45.



(b) Few-shot on ML45.

Figure 3: Multitask and few-shot success rate on the Metaworld ML45 task suite. In (a) we train on the dataset described in Section 5.2, and in (b) we finetune the model from (a) using 5 demonstrations each from a set of held out tasks. Results show the mean and error bar represents standard deviation across five random seeds. Results for PRISE are taken from Zheng et al. [73], and the others we reimplemented and ran ourselves.

	VQ	Obs. Cond.	Mirror Dec.	Ours
LIBERO-90	81.2 ± 0.6	81.9 ± 1.1	86.3 ± 0.9	$\mathbf{89.8 \pm 0.4}$
Few Shot	62.5 ± 2.0	61.3 ± 2.2	45.4 ± 2.0	$\mathbf{68.8 \pm 1.7}$

Table 1: Success rates after ablating design details of QueST. We present the mean across four random seeds and error tolerances show the standard deviation.

we also tried fine-tuning the decoder of VQ-BeT but did not observe any gains from it. VQ-BeT struggles in this setting as it heavily relies on offset head to output continuous action corrections which requires more data-samples for sufficient coverage in the continuous action space. Figure 3b shows the average success rate for 5-shot IL across 5 unseen tasks in MetaWorld. Similar to multitask results, all methods perform comparably, with QueST showing a slight improvement over the others. QueST leverages its learned skill tokens to compositionally model thier distribution for an unseen task in just 5 demonstration examples. For few-shot evaluation protocal please refer Appendix E.1.

5.4 Long-horizon BC

In this setting, we aim to purely study and compare the performance of our model on long-horizon tasks. We train the model (both stages) solely on LIBERO-LONG complete dataset (50 demonstrations per task) and evaluate with the same scheme as described earlier.

Figure 2c shows the average success rate across 10 LIBERO-LONG tasks. Again, QueST outperforms all other baselines by a large margin demonstrating its superior long-horizon modeling capabilities. We attribute this to the use of transformer in the prior along with temporal abstraction introduced by skill tokens.

Overall, we see that our model outperforms baselines like VQ-BeT in multitask settings, showing stronger modelling capacity. At the same time, it has the correct latent structure to outperform baselines like diffusion in few shot settings, especially even with frozen decoder, indicating strong generalization capabilities of learned skill-space.

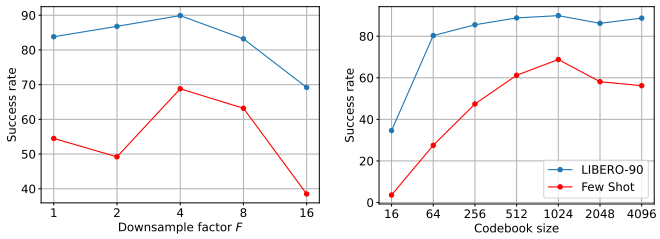
5.5 Ablations

We validate the proposed architecture by ablating some of its key design decisions. All the ablations are performed on LIBERO studying their effects on both multitask and fewshot IL settings.

1. **Vector Quantization**: We replace the FSQ layer with a Vector Quantization layer of nearly the same codebook size.

	Non Causal ϕ_θ	Non Causal ψ_θ	Fully Non Causal	Ours
LIBERO-90	82.0 ± 1.6	85.1 ± 1.8	78.5 ± 0.5	$\mathbf{89.8 \pm 0.4}$
Few Shot	58.8 ± 3.0	61.6 ± 2.5	56.1 ± 1.8	$\mathbf{68.8 \pm 1.7}$

Table 2: Success rates after ablating the causality in QueST. We present the mean across four random seeds and error tolerances show the standard deviation.



(a) Downsampling factor sensitivity. (b) Codebook size sensitivity.

Figure 4: We conduct a sensitivity experiment across downsampling factors (a) and codebook sizes (b) on the LIBERO benchmark. For (a) we fix a sequence length of $T = 32$. Overall, we see that the few-shot version is more sensitive to hyperparameters and that $F = 4$ with 1024 codebook vectors are good choices.

2. **Observation Conditioned Decoder:** Many prior condition the action decoder with current observation [75, 29]. We experiment with this by appending observation tokens to the skill tokens and allowing the transformer decoder to jointly cross-attend to both.

3. **Mirrored Decoder:** Following a typical autoencoder design, we use a decoder that mirrors the encoder, using transposed convolutions instead of strided convolutions, and with the strides in reverse order as in the encoder. This decoder directly takes skill-token embeddings as input and outputs the continuous actions.

4. **Causality:** We ablate the use of causal layers in various parts of our network.

We report the ablation results in both multitask and few-shot settings on LIBERO in Table 1 and Table 2. We see that QueST outperforms all ablations, validating our design choices.

We also perform a sensitivity experiment over several hyperparameters including downsampling factor and codebook size in Figure 4. Across the board we see that the hyperparameters are more important in the difficult few-shot learning setting. In Figure 4a we see that both algorithms have the best performance with a modest downsampling factor of $F = 4$, and in Figure 4b we see that QueST does well with a 1024 codebook vectors. For more discussion on ablations please refer Appendix C

6 Conclusion

We present Quantized Skill Transformer, a novel LVM architecture for learning sharable skills in a discrete latent space. The key idea behind QueST is to represent action sequences as a series of codebook vectors, and we demonstrate that using causal convolutions and masked transformers provides an inductive bias that encourages the model to learn useful shared representations. We evaluate QueST across 145 robot manipulation tasks, and show that it outperforms several state-of-the-art baselines in multitask and few-shot learning settings. Our results highlight the usefulness of QueST's encoder (decoder) as semantically-sound, task-agnostic tokenizer (detokenizer) for continuous actions, and its potential to leverage Large Multi-modal Language Models in stage-2.

Limitations While the benchmarks we consider encompass a wide variety of tasks, the held-out tasks are still structurally similar to the pretraining set, which makes few-shot transfer feasible. In scenarios with a more diverse task, current model may struggle to solve new tasks solely within the learned skill space. A promising direction is to train stage-1 on larger datasets, such as Open X-Embodiment [46], with an expanded codebook that could capture more diverse motion primitives. Additionally, our current architecture only accounts for causality. Future work should explore other inductive biases, like geometric invariance and dynamic consistency, to enhance abstraction learning.

A statement on societal impact. This paper works towards the broader goal of automating a wide range of manipulation tasks. While this can have positive impacts, such as helping people with mobility impairments or performing dirty tasks humans would rather not do, it can also have negative impacts such as automating peoples' jobs away and further concentrating wealth in the hands of a handful of companies. It is important that we in the machine learning community advocate for equitable use of the technology we develop.

References

[1] Q-transformer: Scalable offline reinforcement learning via autoregressive q-functions. In *7th Annual Conference on Robot Learning*, 2023.

[2] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, Q. Huang, A. Jansen, A. Roberts, M. Tagliasacchi, M. Sharifi, N. Zeghidour, and C. Frank. Musiclm: Generating music from text, 2023.

[3] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, M. Yan, and A. Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022.

[4] M. Ahn, D. Dwibedi, C. Finn, M. G. Arenas, K. Gopalakrishnan, K. Hausman, B. Ichter, A. Irpan, N. Joshi, R. Julian, S. Kirmani, I. Leal, E. Lee, S. Levine, Y. Lu, I. Leal, S. Maddineni, K. Rao, D. Sadigh, P. Sanketi, P. Sermanet, Q. Vuong, S. Welker, F. Xia, T. Xiao, P. Xu, S. Xu, and Z. Xu. Autort: Embodied foundation models for large scale orchestration of robotic agents, 2024.

[5] A. Ajay, A. Kumar, P. Agrawal, S. Levine, and O. Nachum. Opal: Offline primitive discovery for accelerating offline reinforcement learning, 2021.

[6] A. Baevski, H. Zhou, A. Mohamed, and M. Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *CoRR*, abs/2006.11477, 2020. URL https://arxiv.org/abs/2006.11477.

[7] Y. Bai, X. Geng, K. Mangalam, A. Bar, A. Yuille, T. Darrell, J. Malik, and A. A. Efros. Sequential modeling enables scalable learning for large vision models. *arXiv preprint arXiv:2312.00785*, 2023.

[8] H. Bao, L. Dong, S. Piao, and F. Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[9] Z. Borsos, R. Marinier, D. Vincent, E. Kharitonov, O. Pietquin, M. Sharifi, D. Roblek, O. Teboul, D. Grangier, M. Tagliasacchi, and N. Zeghidour. Audiolm: a language modeling approach to audio generation, 2023.

[10] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[11] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[12] A. Bucker, L. Figueredo, S. Haddadin, A. Kapoor, S. Ma, S. Vemprala, and R. Bonatti. Latte: Language trajectory transformer. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 7287–7294. IEEE, 2023.

[13] H. Chang, H. Zhang, L. Jiang, C. Liu, and W. T. Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11315–11325, 2022.

[14] L. Chen, S. Bahl, and D. Pathak. Playfusion: Skill acquisition via diffusion from language-annotated play. In *Conference on Robot Learning*, pages 2012–2029. PMLR, 2023.

[15] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[16] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[17] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever. Jukebox: A generative model for music, 2020.

[18] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

[19] Y. Duan, M. Andrychowicz, B. C. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba. One-shot imitation learning, 2017.

[20] F. Ebert, Y. Yang, K. Schmeckpeper, B. Bucher, G. Georgakis, K. Daniilidis, C. Finn, and S. Levine. Bridge data: Boosting generalization of robotic skills with cross-domain datasets, 2021.

[21] P. Esser, R. Rombach, and B. Ommer. Taming transformers for high-resolution image synthesis. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12873–12883, 2021.

[22] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine. One-shot visual imitation learning via meta-learning, 2017.

[23] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[24] Z. Fu, T. Z. Zhao, and C. Finn. Mobile aloha: Learning bimanual mobile manipulation with low-cost whole-body teleoperation. *arXiv preprint arXiv:2401.02117*, 2024.

[25] H. Ha, P. Florence, and S. Song. Scaling up and distilling down: Language-guided robot skill acquisition. In *Proceedings of the 2023 Conference on Robot Learning*, 2023.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] M. Janner, Y. Du, J. B. Tenenbaum, and S. Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[28] D. Jarrett, I. Bica, and M. van der Schaar. Strictly batch imitation learning by energy-based distribution matching. *Advances in Neural Information Processing Systems*, 33:7354–7365, 2020.

[29] Z. Jiang, Y. Xu, N. Wagener, Y. Luo, M. Janner, E. Grefenstette, T. Rocktäschel, and Y. Tian. H-gap: Humanoid control with a generalist planner. *arXiv preprint arXiv:2312.02682*, 2023.

[30] A. Khazatsky, K. Pertsch, S. Nair, A. Balakrishna, S. Dasari, S. Karamcheti, S. Nasiriany, M. K. Srirama, L. Y. Chen, K. Ellis, P. D. Fagan, J. Hejna, M. Itkina, M. Lepert, Y. J. Ma, P. T. Miller, J. Wu, S. Belkhale, S. Dass, H. Ha, A. Jain, A. Lee, Y. Lee, M. Memmel, S. Park, I. Radosavovic, K. Wang, A. Zhan, K. Black, C. Chi, K. B. Hatch, S. Lin, J. Lu, J. Mercat, A. Rehman, P. R. Sanketi, A. Sharma, C. Simpson, Q. Vuong, H. R. Walke, B. Wulfe, T. Xiao, J. H. Yang, A. Yavary, T. Z. Zhao, C. Agia, R. Baijal, M. G. Castro, D. Chen, Q. Chen, T. Chung, J. Drake, E. P. Foster, J. Gao, D. A. Herrera, M. Heo, K. Hsu, J. Hu, D. Jackson, C. Le, Y. Li, K. Lin, R. Lin, Z. Ma, A. Maddukuri, S. Mirchandani, D. Morton, T. Nguyen, A. O'Neill, R. Scalise, D. Seale, V. Son, S. Tian, E. Tran, A. E. Wang, Y. Wu, A. Xie, J. Yang, P. Yin, Y. Zhang, O. Bastani, G. Berseth, J. Bohg, K. Goldberg, A. Gupta, A. Gupta, D. Jayaraman, J. J. Lim, J. Malik, R. Martín-Martín, S. Ramamoorthy, D. Sadigh, S. Song, J. Wu, M. C. Yip, Y. Zhu, T. Kollar, S. Levine, and C. Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024.

[31] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[32] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[33] D. Lee, C. Kim, S. Kim, M. Cho, and W.-S. Han. Autoregressive image generation using residual quantization, 2022.

[34] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[35] S. Levine, C. Finn, T. Darrell, and P. Abbeel. End-to-end training of deep visuomotor policies, 2016.

[36] Z. Liang, Y. Mu, H. Ma, M. Tomizuka, M. Ding, and P. Luo. Skilldiffuser: Interpretable hierarchical planning via skill abstractions in diffusion-based task execution. *arXiv preprint arXiv:2312.11598*, 2023.

[37] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[38] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning, 2023.

[39] J. Luo, P. Dong, J. Wu, A. Kumar, X. Geng, and S. Levine. Action-quantized offline reinforcement learning for robotic skill learning. In *Conference on Robot Learning*, pages 1348–1361. PMLR, 2023.

[40] C. Lynch, M. Khansari, T. Xiao, V. Kumar, J. Tompson, S. Levine, and P. Sermanet. Learning latent plans from play. *Conference on Robot Learning (CoRL)*, 2019. URL https://arxiv.org/abs/1903.01973.

[41] Z. Mandi, F. Liu, K. Lee, and P. Abbeel. Towards more generalizable one-shot visual imitation learning. In *2022 International Conference on Robotics and Automation (ICRA)*, pages 2434–2444. IEEE, 2022.

[42] A. Mandlekar, D. Xu, J. Wong, S. Nasiriany, C. Wang, R. Kulkarni, L. Fei-Fei, S. Savarese, Y. Zhu, and R. Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.

[43] E. Mansimov and K. Cho. Simple nearest neighbor policy method for continuous control tasks, 2018. URL https://openreview.net/forum?id=ByL48G-AW.

[44] F. Mentzer, D. Minnen, E. Agustsson, and M. Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.

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

[46] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[47] N. D. Palo and E. Johns. Dinobot: Robot manipulation via retrieval and alignment with vision foundation models. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2024.

[48] E. Perez, F. Strub, H. De Vries, V. Dumoulin, and A. Courville. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[49] K. Pertsch, Y. Lee, and J. Lim. Accelerating reinforcement learning with learned skill priors. In *Conference on robot learning*, pages 188–204. PMLR, 2021.

[50] D. A. Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988.

[51] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[52] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever. Language models are unsupervised multitask learners. 2019.

[53] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[54] S. Reed, K. Zolna, E. Parisotto, S. G. Colmenarejo, A. Novikov, G. Barth-Maron, M. Gimenez, Y. Sulsky, J. Kay, J. T. Springenberg, T. Eccles, J. Bruce, A. Razavi, A. Edwards, N. Heess, Y. Chen, R. Hadsell, O. Vinyals, M. Bordbar, and N. de Freitas. A generalist agent, 2022.

[55] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[56] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[57] L. X. Shi, J. J. Lim, and Y. Lee. Skill-based model-based reinforcement learning. *arXiv preprint arXiv:2207.07560*, 2022.

[58] A. Singh, H. Liu, G. Zhou, A. Yu, N. Rhinehart, and S. Levine. Parrot: Data-driven behavioral priors for reinforcement learning. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=Ysuv-WOFeKR.

[59] K. Sohn, H. Lee, and X. Yan. Learning structured output representation using deep conditional generative models. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL https://proceedings.neurips.cc/paper_files/paper/2015/file/8d55a249e6baa5c06772297520da2051-Paper.pdf.

[60] G. Swamy, S. Choudhury, J. A. Bagnell, and Z. S. Wu. Causal imitation learning under temporally correlated noise, 2022.

[61] G. Team. Gemini: A family of highly capable multimodal models, 2024.

[62] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[63] A. van den Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. In *Arxiv*, 2016. URL https://arxiv.org/abs/1609.03499.

[64] A. Van Den Oord, O. Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[65] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[66] W. Wan, Y. Zhu, R. Shah, and Y. Zhu. Lotus: Continual imitation learning for robot manipulation through unsupervised skill discovery. *arXiv preprint arXiv:2311.02058*, 2023.

[67] Z. Xian, N. Gkanatsios, T. Gervet, T.-W. Ke, and K. Fragkiadaki. Chaineddiffuser: Unifying trajectory diffusion and keypose prediction for robotic manipulation. In *7th Annual Conference on Robot Learning*, 2023.

[68] J. Yu, X. Li, J. Y. Koh, H. Zhang, R. Pang, J. Qin, A. Ku, Y. Xu, J. Baldridge, and Y. Wu. Vector-quantized image modeling with improved VQGAN. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=pfNyExj7z2.

[69] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[70] Y. Ze, G. Zhang, K. Zhang, C. Hu, M. Wang, and H. Xu. 3d diffusion policy. *arXiv preprint arXiv:2403.03954*, 2024.

[71] S. Zhang, L. Dong, X. Li, S. Zhang, X. Sun, S. Wang, J. Li, R. Hu, T. Zhang, F. Wu, et al. Instruction tuning for large language models: A survey. *arXiv preprint arXiv:2308.10792*, 2023.

[72] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware, 2023.

[73] R. Zheng, C.-A. Cheng, H. Daumé III, F. Huang, and A. Kolobov. Prise: Learning temporal action abstractions as a sequence compression problem. *arXiv preprint arXiv:2402.10450*, 2024.

[74] R. Zheng, X. Wang, Y. Sun, S. Ma, J. Zhao, H. Xu, H. Daumé III, and F. Huang. Taco: Temporal latent action-driven contrastive loss for visual reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[75] zhengyao jiang, T. Zhang, M. Janner, Y. Li, T. Rocktäschel, E. Grefenstette, and Y. Tian. Efficient planning in a compact latent action space. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022. URL https://openreview.net/forum?id=pVBETTS2av.

[76] X. Zou, J. Yang, H. Zhang, F. Li, L. Li, J. Wang, L. Wang, J. Gao, and Y. J. Lee. Segment everything everywhere all at once. *Advances in Neural Information Processing Systems*, 36, 2024.

Appendix

A Website

For further results and videos please see our website. https://quest-model.github.io

B Experiment Details

B.1 Hyperparameters:

We present hyperparameters in the following tables

Table 3: Stage 1 Parameters

Parameter	Value
encoder dim	256
decoder dim	256
sequence length (T)	32
encoder heads	4
encoder layers	2
decoder heads	4
decoder layers	4
attention dropout	0.1
fsq level	[8, 5, 5, 5]
conv layers	3
kernel sizes	[5, 3, 3]
strides	[2, 2, 1]

Table 4: Stage 2 Parameters

Parameter	Value
start token	1000
vocab size	1000
block size (n)	8
number of layers	6
number of heads	6
embedding dimension	384
attention dropout	0.1
embedding dropout	0.1
beam size	5
temperature	1.0
decoder loss scale	10
execution horizon (T_a)	8
observation history	1

B.2 Architecture Implementation:

For vision encoder we used a shallow Convolutional Neural Network (CNN), consisting of the first four layers of ResNet18 [26] followed by a spatial softmax [35]. In encoder, we use causal convolution layers from [29]. For transformer blocks, we used the transformers library from hugging face https://huggingface.co/docs/transformers/ with appropriate masking for ensuring causality.

B.3 Baseline Implementation:

To ensure fair comparison of different model architectures, we use same input modalities and same observation & task encoders for all baselines. VQ-BeT needs a goal image, we instead give it task embedding as goal. Same as QueST, we concatenate observation embeddings for all modalities at any timestep and project them to respective model's hidden dimension.

Depending on the dataset, we also tune some key hyperparameters for the baselines and present the results for best performing ones.

1. **ResNet-T:** Transformer trunk's hidden dimension and number of layers determines the model capacity. Original implementation [37] uses the hidden dimension of 64 with 4 layers. We observed improved performance for the hidden dimension of 256 with 6 layers and hence report all results for that. As per original implementation we use an observation history of 10 timesteps.
2. **Diffusion Policy:** The model capacity is determined by hidden dimension of U-Net layers. Most widely used implementations use $[256, 512, 1024]$, we ablate a larger model with $[256, 256, 512, 512, 1024]$ but did not observe any performance gains. We also ablate prediction (T) and execution horizon (T_a) with $16, 32$ and $8, 16$ respectively and observed

best performance for $T = 32, T_a = 16$ on LIBERO and $T = 16, T_a = 8$ for MetaWorld. As per original paper ablations [15] an observation history of 1 was used.

3. **VQ-BeT:** Since LIBERO and MetaWorld are larger datasets as compared to the benchmarks in original VQ-BeT paper, we ablate some parameters to increase the model capacity. Specifically, the stage 1 encoder by default is a single MLP layer of dimension 128. We ablate this with $2, 4$ layers and with $256, 512$ dimensions but observed worse reconstruction loss with increase in capacity. We use residual-VQ configuration of $32/2 \approx 1024$ sized codebook which is close to the codebook size of 1000 for QueST. We use an observation window size of 10 and ablate the action window size (T) with $1, 5, 32$. On LIBERO, the performance was lowest for $T = 1$, and highest for $T = 5$. VQ-BeT maps the whole input sequence to just one embedding leading to extreme compression for larger sequence length and thus performs worse with $T = 32$.

B.4 Compute:

The models are implemented in PyTorch. For all our experiments we use a server consisting of 8 Nvidia RTX 1080Ti 10GB memory each. And all our models easily fit on one GPU for training.

C Discussion on Ablations

For aiding this discussion we present the ablation results again in table 5 and table 6 below.

	VQ	Obs. Cond.	Mirror Dec.	Ours
LIBERO-90	81.2 ± 0.6	81.9 ± 1.1	86.3 ± 0.9	$\mathbf{89.8 \pm 0.4}$
Few Shot	62.5 ± 2.0	61.3 ± 2.2	45.4 ± 2.0	$\mathbf{68.8 \pm 1.7}$

Table 5: Success rates after ablating design details of QueST.

- Replacing FSQ with VQ still outperforms VQ-BeT in few-shot setting suggesting that QueST's superior performance is not only due to a better quantization scheme but also due to it architecture that flexibly maps an input sequence to multiple embeddings and allows for efficient transfer.
- It's tempting to ground the mapping between z-tokens and actions with observation tokens with an intuition that z-tokens will define a coarse set of actions and observation tokens will aid finer action decoding. But we observe worse performance with this. We hypothesize that the reconstruction objective forces encoder and decoder for most optimal quantization at the bottleneck layer but with extra observation information the decoder might focus more on observation tokens in turn hurting the quantization. This observation goes hand-in-hand with a closely related prior work SPiRL[57] that tried same ablation and found that state conditioned decoder hurts downstream RL.
- We observe a poorer performance in both multitask and few-shot settings with a conventional stage 1 autoencoder. This validates the QueST's cross-attention architecture that allows for attending to all z-tokens and maintaining causality at the same time.

	Non Causal ϕ_θ	Non Causal ψ_θ	Fully Non Causal	Ours
LIBERO-90	82.0 ± 1.6	85.1 ± 1.8	78.5 ± 0.5	$\mathbf{89.8 \pm 0.4}$
Few Shot	58.8 ± 3.0	61.6 ± 2.5	56.1 ± 1.8	$\mathbf{68.8 \pm 1.7}$

Table 6: Success rates after ablating the causality in QueST.

- We observe that a fully-causal stage-1 is most optimal and a non-causal decoder does not hurt as much as a non-causal encoder does. This can be explained with a simplistic setting where the input to stage-1 are 2D trajectories of a point agent. Consider an anti-clockwise circular trajectory and an S-shaped one where the first half of the later overlaps with the first half (semi-circle) of the former. When both of these trajectory sequences are inputted to the stage-1, a non-causal encoder will assign distinct sequences of z-tokens for both trajectories. But a causal encoder will assign same sequence of z-tokens for the first half

of both trajectories and distinct to later parts. This allows the model to re-use the z-tokens corresponding to a semi-circle for creating other shaped-trajectories that has semi-circle in them for example C-shaped or infinity-shaped trajectories.

	Frozen ψ_θ	Finetuned ψ_θ	
		loss scale 10	loss scale 100
Few Shot	66.0 ± 3.6	$\mathbf{70.2 \pm 2.6}$	66.0 ± 1.0

Table 7: Success rates for decoder finetuning settings in few-shot IL.

- Table 7 illustrates the impact of decoder finetuning in LIBERO-LONG fewshot IL setting. QueST outperforms all baselines even without finetuning the decoder. Finetuning decoder should not be necessary in this setting, as LIBERO-LONG tasks are combination of two tasks from LIBERO-90 (pretraining set). This highlights QueST's effectiveness in stitching trajectories using its learned skill-space. We report the finetuning results in the main paper, as they exhibit better performance.

D Skill-space visualization

We present a t-SNE visualization (Figure 5) illustrating the learned skill-space across multiple set of similar tasks. We consider four different combinations of similar tasks to effectively examine the z-embeddings corresponding to their trajectories. Each data point in the plot represents a vector of n z-embeddings at a specific timestep throughout the entire episode, with decreasing transparency indicating temporal progression. We show that the QueST encoder learns a semantically meaningful skill-space that encodes shared representations of similar motion primitives across different tasks. Notably, the skill-space learning happens in the first stage training which does not make use of any task labels.

E Additional Results

E.1 Fewshot IL

Fewshot Evaluation Protocol: In finetuning phase, we finetune ResNet-T, VQ-BeT & QueST for 100 epochs and ACT & Diffusion Policy for 200 epochs. For each task in MetaWorld, we evaluate each method across 10 evenly spaced checkpoints for 5 seeds on 50 distinct initial states and report the results corresponding to the best performing checkpoint. For Libero, we found the final checkpoint to perform best for all methods and hence report results corresponding to it across 4 seeds.

Table 8: LIBERO 5-shot IL success rates across unseen 10 tasks. Results across 4 random seeds.

Task ID	ResNet-T	ACT	Diffusion Policy	PRISE	VQ-BeT	QueST
1	6.7 ± 2.3	20.0 ± 6.0	33.3 ± 20.9	26.7 ± 6.4	23.3 ± 2.3	$\mathbf{66.6 \pm 2.3}$
2	48.3 ± 10.3	33.3 ± 13.1	78.3 ± 17.1	48.3 ± 9.4	43.3 ± 2.3	$\mathbf{88.3 \pm 4.7}$
3	60.0 ± 4.1	67.7 ± 6.2	$\mathbf{80.0 \pm 7.0}$	70.0 ± 0.0	68.3 ± 6.2	$\mathbf{78.3 \pm 13.1}$
4	66.7 ± 8.4	70.3 ± 6.2	$\mathbf{100.0 \pm 0.0}$	78.3 ± 8.8	41.6 ± 6.2	93.3 ± 6.2
5	26.7 ± 3.1	35.0 ± 4.1	$\mathbf{48.3 \pm 4.7}$	45.0 ± 10.8	33.3 ± 6.2	35.0 ± 7.0
6	46.7 ± 13.1	68.3 ± 6.5	30.0 ± 21.2	$\mathbf{90.0 \pm 4.1}$	48.3 ± 6.2	$\mathbf{86.6 \pm 6.2}$
7	21.7 ± 2.4	15.0 ± 0.0	26.6 ± 2.3	25.0 ± 4.1	41.6 ± 14.3	$\mathbf{51.6 \pm 6.2}$
8	35.0 ± 7.1	26.7 ± 7.0	13.3 ± 9.4	45.0 ± 8.1	25.0 ± 4.0	$\mathbf{61.6 \pm 8.5}$
9	-	-	$\mathbf{55.0 \pm 4.0}$	-	15.0 ± 7.0	46.6 ± 6.2
10	-	-	$\mathbf{68.3 \pm 6.2}$	-	25.0 ± 0.0	$\mathbf{65.0 \pm 12.2}$

Figure 5: t-SNE visualization of skill-token embeddings. Here, the transparency decreases as the episode progresses. The overall patterns clearly shows how similar motion primitives like approaching, picking and placing from different tasks are aligned with one another. This analysis includes the first 11 tasks from LIBERO-90. For better comprehension, we encourage readers to review the corresponding rollouts on the website.

Table 9: MetaWorld 5-shot IL success rates across 5 unseen tasks. Results across 5 random seeds.

Task ID	ResNet-T	ACT	Diffusion Policy	PRISE	VQ-BeT	QueST
box-close-v2	63.2 ± 5.2	67.2 ± 5.2	68.0 ± 1.6	60.8 ± 6.6	75.3 ± 9.6	$\mathbf{84.0 \pm 7.3}$
disassemble-v2	68.8 ± 2.0	83.2 ± 3.2	81.3 ± 3.8	74.1 ± 7.3	92.7 ± 1.9	$\mathbf{76.4 \pm 26.0}$
hand-insert-v2	37.2 ± 4.1	53.2 ± 3.7	39.3 ± 1.9	$\mathbf{60.0 \pm 5.0}$	48.0 ± 6.5	49.6 ± 6.4
pick-place-wall-v2	42.8 ± 3.7	74.4 ± 6.9	70.7 ± 5.2	71.7 ± 5.7	65.3 ± 1.9	$\mathbf{76.8 \pm 11.4}$
stick-pull-v2	58.0 ± 8.8	76.0 ± 3.6	71.3 ± 1.9	67.5 ± 5.6	62.0 ± 11.4	$\mathbf{72.8 \pm 11.1}$

E.2 Multitask IL

Table 10: LIBERO-90 multitask IL success rates across 90 tasks. Results across 4 random seeds.

Task ID	ResNet-T	ACT	Diffusion Policy	PRISE	VQ-BeT	QueST
1	0.45	**1.00**	0.95	0.80	**1.00**	**1.00**
2	0.10	0.60	**1.00**	0.35	0.85	0.98
3	0.25	0.95	**1.00**	0.70	0.95	0.95
4	0.00	0.40	0.90	0.50	**1.00**	0.93
5	0.00	0.30	0.95	0.45	**1.00**	0.93

Task ID	ResNet-T	ACT	Diffusion Policy	PRISE	VQ-BeT	QueST
6	0.00	0.70	**1.00**	0.65	0.98	**1.00**
7	0.00	0.40	**1.00**	0.50	0.90	0.93
8	0.45	**1.00**	0.80	0.95	0.80	**1.00**
9	0.00	**1.00**	0.95	0.60	0.78	0.90
10	0.30	**0.95**	0.90	0.35	0.80	0.93
11	0.70	**1.00**	0.95	0.95	0.95	0.98
12	0.40	**1.00**	0.95	0.95	0.88	0.95
13	0.05	0.35	**0.90**	0.20	0.88	0.68
14	0.35	0.25	**1.00**	0.40	0.58	0.80
15	0.10	0.75	**1.00**	0.35	0.45	0.53
16	0.10	0.95	**1.00**	0.75	0.95	0.95
17	0.05	0.75	**0.85**	0.40	0.55	0.83
18	0.05	0.50	0.75	0.15	**0.88**	0.68
19	0.30	0.25	**1.00**	0.30	0.93	**1.00**
20	0.00	**1.00**	0.95	0.65	0.68	0.90
21	0.70	**1.00**	**1.00**	**1.00**	0.98	**1.00**
22	0.00	0.70	**1.00**	0.30	0.93	0.95
23	0.40	0.75	**1.00**	0.85	0.95	0.95
24	0.00	0.45	**0.90**	0.05	0.68	0.85
25	0.30	0.10	**1.00**	0.95	0.90	**1.00**
26	0.60	0.10	**1.00**	0.90	0.78	0.98
27	0.00	0.60	**0.90**	0.55	0.50	0.55
28	0.00	0.35	**0.85**	0.05	0.40	0.68
29	0.80	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
30	0.10	0.85	**1.00**	**1.00**	0.93	0.98
31	0.05	0.40	**0.90**	0.50	0.85	**0.90**
32	0.25	**1.00**	**1.00**	0.85	0.85	0.98
33	0.00	0.30	0.55	0.20	0.33	**0.68**
34	0.10	0.50	0.85	0.30	0.93	**0.98**
35	0.05	0.50	**1.00**	0.80	**1.00**	0.98
36	0.10	**1.00**	**1.00**	0.75	**1.00**	0.95
37	0.00	0.05	**0.90**	0.25	0.70	0.70
38	0.05	0.00	**0.90**	0.30	0.88	0.65
39	0.00	0.90	0.85	0.20	**0.98**	0.95
40	0.25	0.40	0.95	0.85	0.88	**1.00**
41	0.15	0.90	0.70	0.50	**0.98**	0.95
42	0.40	0.85	**1.00**	0.55	0.85	**1.00**
43	0.45	0.70	**1.00**	0.80	**1.00**	0.95
44	0.10	**0.85**	**0.85**	0.40	0.80	**0.85**
45	0.40	0.75	**1.00**	0.85	0.98	0.98
46	0.00	0.80	**1.00**	0.55	0.90	**1.00**

Task ID	ResNet-T	ACT	Diffusion Policy	PRISE	VQ-BeT	QueST
47	0.00	0.00	0.25	0.35	0.63	**0.90**
48	0.00	0.00	0.55	0.25	0.88	**1.00**
49	0.00	0.00	0.95	0.65	0.50	**1.00**
50	0.00	0.00	0.80	0.65	0.63	**1.00**
51	0.00	0.35	0.30	0.40	**0.83**	**0.83**
52	0.00	0.10	0.00	0.10	**0.93**	0.75
53	0.05	0.05	0.35	0.30	**0.80**	**0.80**
54	0.05	0.05	0.75	0.60	0.75	**0.83**
55	0.00	0.15	0.85	0.50	**0.98**	0.93
56	0.00	0.00	0.45	0.35	**0.88**	0.80
57	0.30	0.00	0.50	0.80	**1.00**	**1.00**
58	0.25	0.00	**1.00**	0.50	**1.00**	0.98
59	0.00	0.50	0.75	0.20	**1.00**	0.90
60	0.25	0.00	0.90	0.65	0.90	**0.93**
61	0.40	0.45	0.90	0.80	0.98	**1.00**
62	0.20	0.05	0.55	0.85	**1.00**	**1.00**
63	0.00	0.05	0.35	0.40	**0.80**	**0.80**
64	0.00	0.00	0.45	0.40	0.40	**0.78**
65	0.00	0.25	0.80	0.15	0.68	**0.90**
66	0.00	0.05	0.70	0.15	**0.85**	0.83
67	0.15	0.45	0.60	0.30	0.88	**0.95**
68	0.10	0.55	0.35	0.55	0.65	**0.83**
69	0.35	0.60	0.55	0.85	0.88	**0.95**
70	0.10	0.85	0.50	0.90	0.35	**0.95**
71	0.55	0.60	**0.95**	0.55	0.58	**0.95**
72	0.20	0.00	0.90	0.35	**0.95**	0.93
73	0.20	0.30	0.85	0.60	0.35	**1.00**
74	0.00	0.35	0.75	0.30	**0.90**	0.65
75	0.05	0.70	0.45	0.45	0.48	**1.00**
76	0.10	0.35	0.30	0.25	**0.88**	0.78
77	0.30	0.10	0.40	0.65	**0.93**	0.88
78	0.30	0.70	0.15	0.80	**0.98**	0.95
79	0.10	0.10	0.05	0.45	**0.95**	0.88
80	0.45	0.95	0.00	0.30	0.00	**1.00**
81	0.20	0.45	0.05	0.30	**1.00**	0.78
82	0.00	0.50	0.55	0.35	**0.85**	0.73
83	0.45	0.55	0.55	0.80	0.28	**0.88**
84	0.05	0.00	0.55	0.55	0.73	**0.85**
85	0.20	0.15	0.75	0.75	0.88	**0.95**
86	0.00	0.10	0.10	0.75	0.65	**0.95**
87	0.20	0.30	0.95	0.95	0.88	**0.98**

Task ID	ResNet-T	ACT	Diffusion Policy	PRISE	VQ-BeT	QueST
88	0.10	**1.00**	0.95	0.65	0.35	0.95
89	0.25	0.85	0.70	0.55	0.58	**1.00**
90	0.10	0.45	0.90	0.55	**0.95**	0.50

Table 11: MetaWorld multitask IL success rates across 45 tasks. Results across 5 random seeds.

Task ID	ResNet-T	ACT	Diffusion Policy	VQBeT	QueST
assembly-v2	0.73	0.97	0.88	0.82	**1.00**
basketball-v2	0.76	0.80	0.78	**0.82**	0.68
bin-picking-v2	0.89	**1.00**	0.96	0.20	0.94
button-press-topdown-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
button-press-topdown-wall-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
button-press-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
button-press-wall-v2	**1.00**	**1.00**	0.98	0.98	0.98
coffee-button-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
coffee-pull-v2	0.90	0.92	0.96	0.82	**0.98**
coffee-push-v2	0.89	**0.96**	0.86	0.94	0.90
dial-turn-v2	0.98	0.99	**1.00**	**1.00**	**1.00**
door-close-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
door-lock-v2	**1.00**	0.99	**1.00**	**1.00**	**1.00**
door-open-v2	**0.96**	0.95	**0.96**	0.94	0.94
door-unlock-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
drawer-close-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
drawer-open-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
faucet-close-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
faucet-open-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
hammer-v2	0.95	**1.00**	0.98	**1.00**	0.94
handle-press-side-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
handle-press-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
handle-pull-side-v2	0.69	0.94	0.78	0.74	**0.98**
handle-pull-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
lever-pull-v2	**0.94**	0.93	0.84	0.80	0.92
peg-insert-side-v2	0.81	**0.94**	0.90	0.76	0.86
peg-unplug-side-v2	0.88	0.91	0.88	**0.92**	0.90
pick-out-of-hole-v2	0.62	**0.89**	0.74	0.34	0.76
pick-place-v2	0.67	0.71	0.76	0.74	**0.78**
plate-slide-back-side-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
plate-slide-back-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
plate-slide-side-v2	0.98	**1.00**	0.98	0.98	**1.00**
plate-slide-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
push-back-v2	0.72	0.64	0.76	0.64	**0.80**

Task ID	ResNet-T	ACT	Diffusion Policy	VQBeT	QueST
push-v2	0.84	0.90	0.84	0.76	**0.92**
push-wall-v2	0.92	0.98	0.94	0.94	**1.00**
reach-v2	**0.39**	0.37	0.32	0.28	0.36
reach-wall-v2	0.49	0.47	**0.52**	0.36	0.42
shelf-place-v2	0.65	0.85	0.66	0.76	**0.88**
soccer-v2	0.42	0.25	0.42	0.36	**0.52**
stick-push-v2	0.75	**1.00**	0.96	0.94	0.96
sweep-into-v2	0.90	**0.92**	0.88	0.90	0.84
sweep-v2	0.98	**1.00**	0.98	**1.00**	**1.00**
window-close-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**
window-open-v2	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**