
Quantization-Free Autoregressive Action Transformer

Ziyad Sheebaelhamd
University of Tübingen

Michael Tschannen
Google DeepMind

Michael Muehlebach
Max Planck Institute for Intelligent Systems

Claire Vernade
University of Tübingen

Abstract

Current transformer-based imitation learning approaches introduce discrete action representations and train an autoregressive transformer decoder on the resulting latent code. However, the initial quantization breaks the continuous structure of the action space thereby limiting the capabilities of the generative model. We propose a quantization-free method instead that leverages Generative Infinite-Vocabulary Transformers (GIVT) as a direct, continuous policy parametrization for autoregressive transformers. This simplifies the imitation learning pipeline while achieving state-of-the-art performance on a variety of popular simulated robotics tasks. We enhance our policy roll-outs by carefully studying sampling algorithms, further improving the results.

1 Introduction

Generative modeling lies at the heart of modern machine learning, enabling the production of outputs that adhere to user-specified objectives and constraints by navigating high-dimensional spaces of potential outcomes. Fundamentally, this process can be understood as a form of search where naive or exhaustive strategies are computationally intractable due to the size of the space. We consider imitation learning for robotic systems as an extreme example, where the search space has uncountable cardinality and a complex structure arising from nonlinear dynamics, interactions with the environment and the fundamental multimodality in human or robot behavior. In many real-world robotics applications, the control signals involved are continuous in nature [33, 59, 28]. Current state-of-the-art methods for modeling policies in continuous control tasks broadly fall into two categories: autoregressive transformer models [26, 45] and diffusion models [7, 54, 40], which will be discussed in the next two paragraphs.

Existing autoregressive policies, on the one hand, ignore the challenge of learning in a continuous domain by discretizing actions [26, 45]. This discretization can introduce several drawbacks: It discards the inherent structure of the continuous space, increases complexity by adding a separate quantization step, and may limit expressiveness or accuracy when fine-grained control is required. Furthermore, the popular approach of learning a VQ-VAE [52] of the action space involves nondifferentiable operations and therefore requires advanced tricks during representation learning to sidestep optimization difficulties and instabilities [24, 20, 18, 31]. Hence, our goal is to design autoregressive transformers that preserve the continuous nature of the action space, leveraging the smooth and potentially multimodal behavior patterns that experts exhibit.

Diffusion policies, on the other hand, are adept at capturing continuous action distributions, but face two critical challenges in control tasks. First, real-time applications like robotics demand fast, efficient inference, yet diffusion models rely on iterative sampling, requiring dozens of forward passes

Correspondence: Ziyad Sheebaelhamd <ziyad.sheebaelhamd@uni-tuebingen.de>

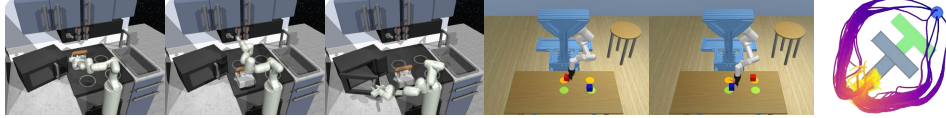
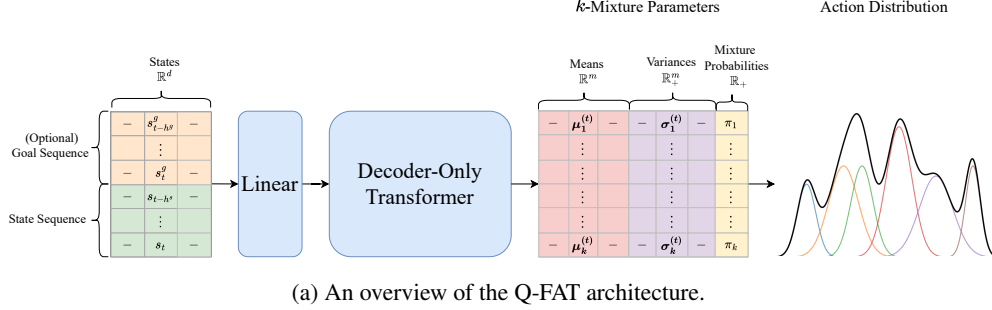


Figure 1: Figure (a) shows the overview of QFAT. A sequence of h^s previous states and h^g goal states are projected into the Transformer’s embedding dimension using a linear layer. The Transformer then predicts the action distribution by predicting the GMM means, variances and mixture probabilities. Figure (b) shows a sample of generated trajectories from 3 representative environments, demonstrating the captured multi-modality in the sequence of solving tasks (Kitchen and UR3 Block Push) or the direction from which an object is approached (PushT).

to generate actions. This makes them orders-of-magnitude slower than autoregressive or feedforward policies [26]. Second, the downstream task of imitation learning is often reinforcement learning [43, 39, 58, 17], which typically requires exact action likelihoods for tasks like exploration and off-policy corrections [49, 1, 27, 56]. Diffusion models, however, generate samples by integrating (stochastic) differential equations [48, 47], making likelihood estimation computationally expensive.

To address these limitations, we propose the use of a Generative Infinite Vocabulary Transformer (GIVT) [50] that models the policy as a Gaussian Mixture Model (GMM) on top of a decoder-only transformer architecture [53, 41]. This approach enables modeling continuous actions as “tokens” by directly parameterizing the components of the GMM in each time step, avoiding the need for discretization and thus preserving the inherent geometry of the action space. This choice allows us to exploit the benefits of large-scale sequence modeling, such as autoregressive policy generation and access to explicit likelihood estimates, while avoiding the pitfalls of artificially segmenting continuous control signals into discrete bins. In addition, we discuss two different sampling strategies that reduce variability in the generated trajectories.

In summary, our work:

- Eliminates the need for action quantization in autoregressive policy parameterization, simplifying behavioral cloning pipelines while retaining the ability to directly evaluate action likelihoods.
- Proposes two sampling algorithms to reduce the variance of the generated trajectories, including a global mode-finding algorithm.
- Demonstrates state-of-the-art performance on standard benchmarks, both for conditional and unconditional policy generation using proprioceptive and image-based states, confirming the effectiveness of our method for multimodal policy modeling.

The implementation is available at <https://github.com/ziyadsheeba/qfat>.

2 Sequence to Action Prediction

We consider the following standard setting for imitation learning of simulated robotics tasks. An agent equipped with continuous controls $\mathcal{A}$ evolves in a continuous state space $\mathcal{S}$ in discrete time. In

behavioral cloning, we assume that a so-called *behavior policy* π_b collected a dataset $D = \{\tau_i\}_{i=0}^{N-1}$ of N trajectories in the environment, where each trajectory $\tau_i = \{s_0^i, a_0^i, \dots, a_{n^i-1}^i, s_{n^i}^i\}$ is a sequence of states $s_t^i \in \mathcal{S}$ and actions $a_t^i \in \mathcal{A}$, with n^i denoting the length of trajectory i . Note that we do not impose a Markovian assumption on the behavior policy. In general a_t^i might depend on a history of states $\{s_j^i\}_{j=t-h}^t$, where h is a fixed window.

Our objective is to learn a *sequence-to-action* predictor, or *imitating policy* π_θ . We treat the objective as a supervised learning problem, where the parameter θ of our policy is optimized to minimize the negative log-likelihood with respect to π_b , $\mathbb{E}_{\tau \sim \pi_b}[-\log(\pi_\theta)(\tau)]$. To do so, we minimize:

$$\mathcal{L}(\theta) = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{t=0}^{n^i-1} \log \pi_\theta(a_t^i | s_{t-h:t}^i). \quad (1)$$

Note that a standard result in variational inference [15] shows that this objective is equivalent to minimizing the Kullback-Leibler (KL) divergence between the behavioral policy π_b and the learned policy π_θ , implying that the learned policy π_θ should match, or clone, the behavior of π_b . However, the choice of parametrization is crucial to guarantee a good approximation of the behavior policy.

3 The Q-FAT Algorithm

Current transformer-based policy parametrization approaches [26, 45, 32] consist of first discretizing the action space, e.g. using a VQ-VAE [52, 57], to comply with the standard transformer architecture requiring a discrete vocabulary. Inspired by the success of GIVT [50] in the image generation domain, our model *Quantization-Free Autoregressive Action Transformer* (Q-FAT) adapts GIVT to the continuous action behavioral cloning setting, modeling the action distribution at each step by *predicting the parameters of a multivariate Gaussian mixture model (GMM)*. Namely, Q-FAT models the probability of an action $a_t \in \mathbb{R}^m$ at step t as

$$\pi_\theta(a_t | s_{t-1}, \dots, s_{t-h^s}) = \sum_{i=1}^k \pi_i^t \mathcal{N}(a_t | \mu_i^t, \sigma_i^t), \quad (2)$$

where π_i^t , μ_i^t , and σ_i^t are produced by the transformer decoder [53, 41] at step t , conditioned on a history of previous states $\{s_i\}_{i=t-h^s}^{t-1}$ with $s_i \in \mathbb{R}^d$. Using a state history of length h^s instead of only using the most recent state accounts for the potential non-Markovian structure in the demonstration dataset. Rather than predicting the full covariance Σ_i^t per mixture component, we assume a diagonal Gaussian distribution and only predict the diagonal entries of the covariance matrix σ_i^t . While this simplifying assumption enables efficient likelihood computations, it does not impose independence across action dimensions in general. Q-FAT is trained using standard teacher forcing with a causal attention mask. We highlight that we only use state feedback and do not condition on the predicted actions. We discuss the important architecture decisions in Appendix D.

3.1 Goal Conditioning

For some tasks such as tracking, a near-future goal can be appended to the input to *condition* the resulting policy. The conditioning is formalized by defining a set of *goal states* $\mathcal{G} \subseteq \mathcal{S}$. We then learn a goal-conditioned policy $\pi_\theta(a_t | s_{t-1}, \dots, s_{t-h^s}, g)$ where $g \subseteq \mathcal{G}$. In practice, we define g as a sequence of future states to be reached, i.e., $g = \{s_i\}_{i=t}^{t+h^g}$, where the goal states are typically generated from a high-level planner that plans instrumental goal states as done in Gupta et al. [11]. In our experiments, we simulate goal states from the demonstration dataset.

3.2 Action Sampling from a k -GMM

QFAT samples the next action from the conditional distribution, $a_t \sim \pi_\theta(\cdot | s_{t-1}, \dots, s_{t-h^s}, g)$, which is a GMM with k components. In practice, the model can be trained to output a short sequence of actions that are then executed in an open-loop fashion. This is done to ensure action consistency and avoid over-fitting to idle actions [7]. For simplicity, we describe only the next-action generation setting.

One can directly sample from the GMM that parametrizes the policy. For the *vanilla GMM sampling*, the procedure is:

1. Sample a mixture index k from the categorical distribution defined by the probabilities $\{\pi_i\}_{i=1}^k$.
2. Sample an action $\mathbf{a}$ from the corresponding Gaussian component $\mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\sigma}_i^2)$.

While this approach directly samples from the learned policy distribution, it can degrade performance in practice when the dataset is noisy. In that case, the GMM may overfit the data distribution and produce *noisy* actions, resulting in jitter or unstable trajectories. We discuss below how Q-FAT can be easily equipped with stabilizing sampling techniques that directly exploit the fitted GMM distribution.

3.3 Stabilizing Output Trajectories

When controlling real-world robotics systems, it is important to generate action sequences without jitter, as high frequency inputs may damage actuators and excite undesirable structural modes [2]. Thanks to the richness of the GMM representation of the policy, we can directly design stabilization patches.

Down-scaling the Mixture Variances A common heuristic to mitigate noisy samples is to adjust the sampling temperature, which in our GMM parametrization is equivalent to *down-scaling the variance* in each Gaussian component [29, 16, 50, 51]. Concretely, one replaces each variance $\boldsymbol{\sigma}_i^2$ by $\alpha \boldsymbol{\sigma}_i^2$, for some small $\alpha \in (0, 1)$ —sometimes by several orders of magnitude ($\alpha \ll 1$).

This compresses each component, forcing sampled actions to concentrate more tightly around each component’s mean. However, merely shrinking the variances does not address the large-scale spread when mixture components are far apart. The inter-component variance remains, and noisy samples may still arise from small but non-negligible transitions among disparate components.

This phenomenon is exacerbated when the choice of the number of components k is overspecified (see Figure 2 and further discussion in Appendix A). Furthermore, it has been shown by Carreira-Perpinan [5] that GMMs in dimension $d \geq 2$ may have a mode that does not directly align with the mean of any component. In such a situation, the down-scaling technique is likely to result in an insidious loss of the extra mode (see Figure 2).

Mode-tracking via the Mean-Shift Algorithm We explored a principled way to reduce the variance of sampling from multimodal distributions by *explicitly identifying the major modes* and then *sampling from those modes* with an appropriate probability while controlling variance. An effective way to find the modes in a GMM is via the mean-shift algorithm [5], which is an efficient fixed-point iteration algorithm (see Appendix B for further details). We propose a multinomial distribution over the modes¹ $\{m_1, \dots, m_J\}$ that respects the shape of the density:

$$\Pr(m_j) = \frac{w_j}{\sum_{j'} w_{j'}}, \quad (3)$$

$$\text{where } w_j = p_{\text{GMM}}(m_j) \cdot |-H_j|^{-\frac{1}{2}} \quad (4)$$

$$\text{and } H_j = \nabla^2 \log p_{\text{GMM}}(m_j). \quad (5)$$

Note that each H_j admits a closed-form expression and can be computed in $O(km^2)$ operations. One can then further inject noise to the sampled mode using a Gaussian noise with a fixed variance or use the Hessian around the mode to account for the curvature of the distribution to determine the variance (see more details on this process in Appendix B). We emphasize that this mode-tracking algorithm can be efficiently implemented on GPUs using common deep learning libraries and introduces a minor overhead compared to the transformer forward pass.

4 Experiments

We evaluate the performance of Q-FAT through extensive experiments across 9 different tasks using five representative simulated robotics environments: PushT [7], Kitchen [11], UR3 BlockPush,

¹The number of modes J is hard to control theoretically [5].

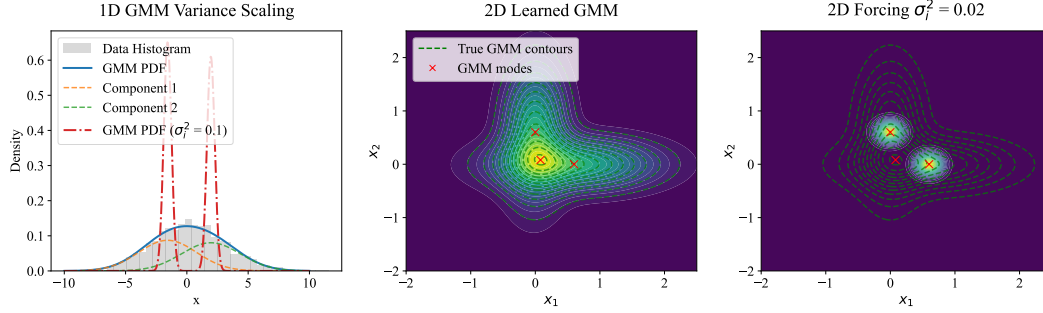


Figure 2: The figures demonstrate the potential detrimental effects of down-scaling the component variances in a learned Gaussian Mixture Model (GMM). The left figure illustrates how down-scaling variance introduces irreducible variance proportional to the distance between component means, when the number of components (k) is misspecified (e.g., approximating a unimodal distribution with two components). The two figures on the right show how reducing variance can lead to a loss of multimodality, specifically causing the central mode to disappear in a 2D two-mixture Gaussian that presents three modes.

BlockPush [10] and Multimodal Ant [26]. These environments cover the most common control signals in practice, namely position and velocity of robot’s end effector and robot joint angles. A detailed description of each environment is provided in Appendix C. We further ran preliminary experiments on an autonomous driving dataset (nuScene [4]) and we add the experimental results in Appendix E. Our experiments are designed to address the following key research questions:

1. Is Q-FAT competitive with state-of-the-art methods in both conditional and unconditional behavior imitation?
2. Can Q-FAT effectively capture the diversity of demonstration datasets while avoiding mode collapse?
3. What is the effect of different sampling techniques on the performance of Q-FAT?
4. How sensitive is Q-FAT’s performance to the choice of the mixture components k in the learned policy?
5. How does the inference speed of Q-FAT compare to VQ-BeT?

We find that Q-FAT achieves state-of-the-art performance across the majority of conditional and unconditional tasks, achieving state-of-the-art balance between the quality and diversity of the generated actions. We report the results of Q-FAT using the variance reduced sampling (Section 3.2) with a factor of 10^{-6} and we discuss the implications of this choice in Section 4.3. Further training details are provided in Appendix D. A summary of our results is presented in Table 1.

4.1 Baselines and Performance Metrics

Baselines We compare Q-FAT against a range of state-of-the-art baselines. We restrict our focus to comparing against diffusion and transformer-based policy learning methods. For unconditional tasks, the baselines include: (1) BeT [45], which performs action discretization using k-means clustering and models the action distribution with a transformer decoder; (2) VQ-BeT [26], an extension of BeT that leverages vector quantization via a hierarchical Variational Autoencoder (VQ-VAE) [57] for action discretization; and (3) two variants of Diffusion Policy [7], namely a convolutional-based implementation and a transformer-based implementation. For goal-conditional tasks, we extend the comparison to include conditional versions of BeT and VQ-BeT, as well as two additional baselines: BESO [42], a denoising diffusion-based method with a conditional variant (C-BESO) and a classifier-free guided variant (CFG-BESO).

4.2 Quality and Diversity of Q-FAT’s Actions

Evaluation Metrics We assess performance using *task success rate* and *induced behavioral entropy*. Task success is measured differently per environment: in Kitchen, Multimodal Ant and UR3 Block-

Table 1: Performance comparison between unconditional and conditional behaviors on the task success rates. We report the performance of Q-FAT with variance down-scaling with a factor of 10^{-6} .

— Unconditional Tasks —						
Environment	Metric	BeT	DiffPolicy-C	DiffPolicy-T	VQ-BeT	Q-FAT
PushT	Final IoU ($\cdot/1$)	0.39	0.73	0.74	0.78	0.80
Image PushT		0.01	0.66	0.45	0.68	0.69
Kitchen	Goals ($\cdot/4$)	3.07	2.62	3.44	3.66	3.84
Image Kitchen		2.48	3.11	3.01	2.98	3.15
Multimodal Ant		2.73	3.12	2.90	3.22	3.30
UR3 BlockPush	Goals ($\cdot/2$)	1.59	1.83	1.82	1.84	1.99
BlockPush		1.67	0.47	1.93	1.79	1.77

— Conditional Tasks —						
Environment	Metric	C-BeT	C-BESO	CFG-BESO	VQ-BeT	Q-FAT
Cond. Kitchen	Goals ($\cdot/4$)	0.15	3.75	3.47	3.78	3.78
Cond. UR3 BlockPush	Goals ($\cdot/2$)	0.00	1.14	0.92	1.94	1.96

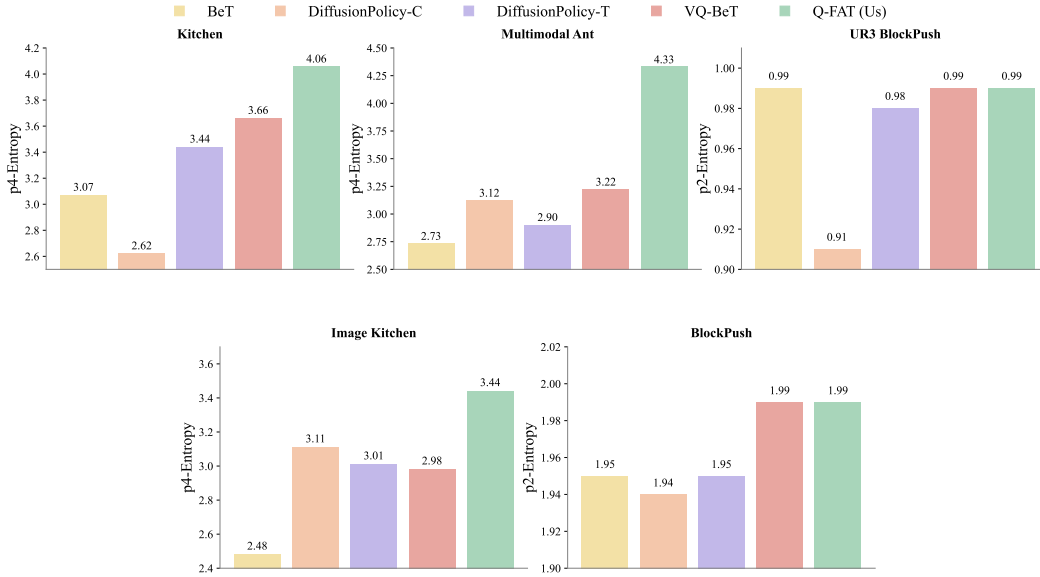


Figure 3: Overview of Q-FAT’s behavioural entropy on unconditional behavior generation compared to baselines.

Push, the task success rate corresponds to the number of goals achieved per roll-out, while in PushT, task success is quantified using the *final* Intersection over Union (IoU) between the object’s position and the target area. Behavioral diversity is measured by computing task completion sequence entropy (length 4 for Kitchen and Multimodal Ant, length 2 for UR3 BlockPush), reflecting variability in task completion. Baseline performance follows metrics from Lee et al. [26], and results are reported as the average success rate of the best checkpoint over 1000 episodes, including behavioral entropy where applicable.

Comparison to Baselines Q-FAT achieves state-of-the-art performance on the success metrics, as shown in Table 1. The performance gap observed with BeT can be attributed to the limitations of k-means clustering in high-dimensional spaces, which may hinder its ability to effectively capture complex action distributions. This is particularly exacerbated due to BeT attempting to discretize the whole unconditional action space, and not the conditional one. In the case of VQ-BeT, the sequential nature of first predicting a discrete action token followed by an action correction step introduces two potential sources of error: inaccuracies in token prediction and errors in the subsequent correction

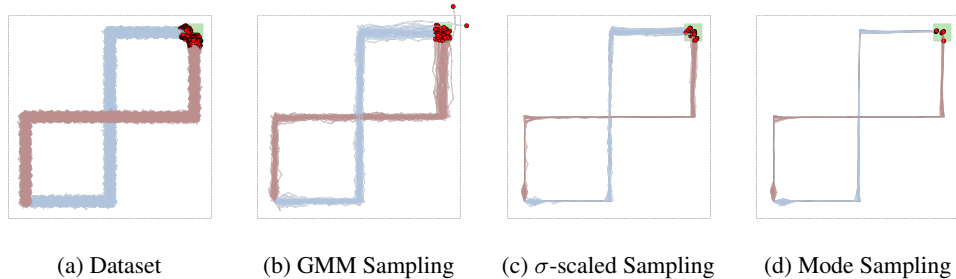


Figure 4: Sampling techniques from an 8-mixture Q-FAT policy on a multiroute environment [45]. (a) Raw dataset with two pairs of equally likely paths (blue and red) from start to target (green). (b) Direct GMM sampling yields noisy samples due to the captured dataset noise. (c) Variance scaling (10^{-8}) reduces per-component variance but not the variance from inter-component distances. (d) Mode sampling largely suppresses noise while preserving dataset multimodality.

mechanism. This places significant reliance on the vector quantization network, which is challenging to optimize due to the need to discretize the entire action space and the required two-stage training process — first for the quantization network and then for the policy. For diffusion-based policies, the absence of an explicit likelihood function makes it difficult to control the variance of sampled actions, particularly in the presence of noisy data. This can be particularly problematic in fine-grained control tasks, where precise action execution is crucial for success. Consequently, while these approaches demonstrate strong performance in certain scenarios, they may struggle in tasks requiring high accuracy and reliability under noisy data. Q-FAT largely outperforms or matches the behavioral entropy of all baselines (Figure 3), which assesses diversity across the generated motion.

4.3 Effects of Sampling Techniques

We assessed the effect of the different sampling techniques discussed in Section 3.2. We found that down-scaling the component variances had a 7% and a 6.5% increase in the success performance metrics in the Kitchen environment and PushT, respectively, compared to the GMM sampling, while only seeing a marginal 2% drop in the diversity of the generated behaviors in the Kitchen environment.

The modes sampling matched the performance of the variance down-scaling on the evaluation metrics. This could be attributed to the fact that the learned mixture components are sufficiently distant in the environments we explored, thus the effects discussed in Section 3.2 are not pronounced.

Despite the quantitative metrics not showing significant differences between the variance down-scaling method and the mode sampling, we visualize the trajectories produced by both sampling techniques on a toy multi-route environment introduced by Shafuallah et al. [45] and see the effects of the irreducible variance in Figure 4. We further display a similar effect in the PushT environment in Figure 5.

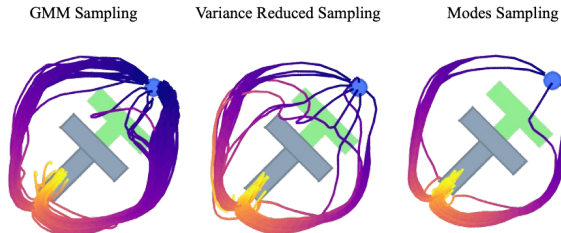


Figure 5: We visualize 400 trajectories generated from Q-FAT with 16 mixtures on the PushT environment with different sampling methods. One can see that mode sampling reduces the variance and does not produce trajectory artifacts.

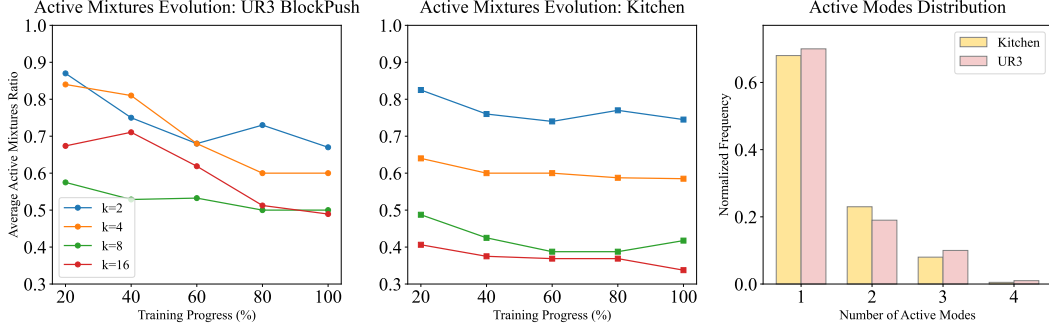


Figure 6: The left two plots display the evolution of the number of active mixtures during inference as training progresses, showing a clear pruning effect. On the rightmost plot, we display the distribution of the number of active modes at each timestep during inference for a converged model. One can see that trajectories remain largely unimodal.

4.4 Effect of the Number of Mixtures

We systematically evaluated the effect of varying the number of mixture components k in Q-FAT, where k controls the number of modes in the conditional action distribution for a given state context.

Specifically, we considered $k \in \{2, 4, 8, 16\}$ in two benchmark domains, the Kitchen and UR3 BlockPush environments. Our results indicate that Q-FAT’s success performance is highly robust across these values of k , with negligible degradation in performance. In the Kitchen environment, we observed an 18% increase in behavioral entropy when k was increased from 2 to 4, after which entropy remained approximately constant as k continued to increase. In contrast, in the UR3 BlockPush environment, the behavioral entropy remained stable regardless of k . The key to achieving this robustness is careful initialization.

In particular, we found it important to initialize each mixture component’s mean to be distant from the others. Concretely, we placed these means on a hyper-cube spanning the range $[-1, 1]$ per dimension, maximizing their mutual separation, and assigned each component a unit variance. The range $[-1, 1]$ is a reasonable choice, as actions are typically bounded in practice and normalized in our experiments.

To investigate how the mixture components were utilized during inference, we tracked the average number of *active* components over the course of training (see Figure 6, left). We observed that the model effectively prunes redundant mixture components as training progresses, with the pruning effect becoming more pronounced when more components are initially employed. Interestingly, in the Kitchen environment, which has a nine dimensional action space, less pruning occurs as k increases compared to the UR3 environment, whose action space is only two dimensional. A plausible explanation for this discrepancy lies in the dimensionality of the action space.

In higher dimensions, each Gaussian component covers a proportionally smaller volume of the action space. Consequently, the model requires more components to adequately represent the conditional action distribution, and therefore prunes fewer mixture components. In lower-dimensional environments, by contrast, each component can cover a larger fraction of the data, which makes aggressive pruning more viable.

We further plotted the distribution over the number of active modes for the Kitchen and the UR3 BlockPush environments. We observed that approximately 70% of the course of a trajectory is uni-modal (Figure 6, right). This shows that multi-modality is required only a minority of the time during a rollout. This observation could be useful for future work to exploit this uni-modality computationally.

4.5 Inference Time Analysis

To compare computational efficiency, we evaluated the inference times of Q-FAT and VQ-BeT on a 16 GB MacBook Pro CPU. The analysis utilized models trained in the Kitchen environment, with hyperparameters for Q-FAT specified in Table 3 and for VQ-BeT as described in [26]. Q-FAT

is approximately 2x faster on the tested hardware due to its efficient action decoding head. The breakdown of inference time per model component is detailed in Table 2.

Table 2: Inference time breakdown for Q-FAT and VQ-BeT on the models used for the Kitchen environment. Q-FAT has a lighter action-decoding head, resulting in a significant speedup.

Component	VQ-BeT (ms)	Q-FAT (ms)
Shared Backbone (minGPT)	~1.0	~1.0
Action Decoding Head	~1.2	~0.1
Total Inference Time	~2.2	~1.1

The primary performance difference arises from the action decoding head. While both models use an identical backbone architecture, VQ-BeT’s reliance on two separate 2-layer MLPs for action decoding creates a computational bottleneck. In contrast, Q-FAT’s use of a single, efficient linear action decoding layer results in a significant speedup.

5 Related Work

Offline Learning for Decision Making Learning to act from offline data has received significant attention over the years due to its vast potential in practical applications. This line of research can be divided into two categories: offline reinforcement learning [25, 27, 22, 6] and imitation learning [34, 37, 38, 13]. In offline reinforcement learning, on the one hand, the goal is to learn a policy from demonstrations with suboptimal behaviors, which necessitates the access to rewards in the demonstration dataset. Imitation learning, on the other hand, assumes that the trajectories are collected from near optimal behaviors, avoiding access to reward signals in the collected datasets. This increases the attractiveness of imitation learning in practical setups, since reward signals are hard to define [9, 23]. Q-FAT falls under the framework of behavioral cloning, which is a form of imitation learning where the policy learning is treated as a supervised learning problem.

Generative Models for Cloning Behavior Complex generative models have been employed to capture the full data diversity due to the complex and multimodal behavior in human and robot datasets. Earlier work has explored the use of Gaussian processes [46], energy-based models [10] and generative adversarial networks [21] as policy parametrizations. More recently, more attention has been directed towards more powerful models, such as transformers [45, 26, 32, 55] and diffusion models [7, 54, 40, 36]. Unlike other transformer-based policies, Q-FAT uses a continuous parametrization of the output by predicting a GMM. Note that this idea has been explored before using LSTM backbones [29, 45], however, it has been shown to produce poor performance [45, 26], which we attribute to the instability of training LSTM networks and their limited capacity to model long sequences [14].

6 Discussion and Future Work

In this work, we introduced Q-FAT, a quantization-free action transformer that achieves state-of-the-art performance across various simulated robotics environments, opening new research directions in both behavioral modeling and reinforcement learning. Compared to prior work [26], Q-FAT overcomes the challenging non-differentiable step in action discretization while maintaining or improving performance.

By integrating Q-FAT into end-to-end policy learning pipelines, future research could evaluate its influence on sample efficiency and final task performance. Another promising area involves incorporating Bayesian priors into Q-FAT’s action distribution estimates, which could facilitate active exploration in complex, high-dimensional settings. For fine manipulation tasks, researchers could further investigate coarse-to-fine sampling strategies based on Gaussian mixture model representations, potentially improving both exploration breadth and control. Finally, extending Q-FAT to non-Euclidean action spaces, such as those with Riemannian geometry, may enable more accurate and natural representations for tasks like legged locomotion or dexterous manipulation.

7 Limitations

While we have demonstrated that Q-FAT is effective in learning complex action distributions, the underlying GMM loss function used to train the policy assumes a Euclidean action space. In some environments (e.g., humanoid robots), the action space has a Riemannian geometry, which could potentially result in difficulty during learning. This can be counteracted by mapping actions into a latent Euclidean space using an action autoencoder [19] and perform the GMM loss in the latent space. However, unlike previous work that uses discrete action encoders [26, 45], Q-FAT allows the flexibility of using continuous latent spaces, allowing end-to-end differentiability of the joint encoder and policy models. While this is an exciting research direction, we leave this for future work.

8 Impact Statement

Our work tackles simulated continuous control tasks whose downstream consequences are important for robotics. The probabilistic nature of our method may impact the safety of the generated behavior and should be further studied and tested. However, unlike prior work, our approach enables uncertainty estimates directly in the action space, which have the potential to improve safety by providing uncertainty quantification.

9 Acknowledgment

We would like to thank Youssef Saied for the valuable discussions throughout the project and Nicolas Nguyen for reviewing the manuscript. C.V. and Z.S. are funded by the German Research Foundation (DFG) under both the project 468806714 of the Emmy Noether Programme and under Germany’s Excellence Strategy – EXC number 2064/1 – Project number 390727645. M.M. is funded by the German Research Foundation (DFG) under the project 456587626 of the Emmy Noether Programme. We also thank the international Max Planck Research School for Intelligent Systems (IMPRS-IS) for the support.

References

- [1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.
- [2] Karl J Åström. PID controllers: theory, design, and tuning. *The International Society of Measurement and Control*, 1995.
- [3] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems*, 33:1877–1901, 2020.
- [4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuscenes: A multimodal dataset for autonomous driving. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11621–11631, 2020.
- [5] Miguel A. Carreira-Perpinan. Mode-finding for mixtures of Gaussian distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1318–1323, 2000.
- [6] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in Neural Information Processing Systems*, 34:15084–15097, 2021.
- [7] Cheng Chi, Zhenjia Xu, Siyuan Feng, Eric Cousineau, Yilun Du, Benjamin Burchfiel, Russ Tedrake, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *The International Journal of Robotics Research*, 2023.

- [8] Pim De Haan, Dinesh Jayaraman, and Sergey Levine. Causal confusion in imitation learning. *Advances in Neural Information Processing Systems*, 32, 2019.
- [9] Rati Devidze, Goran Radanovic, Parameswaran Kamalaruban, and Adish Singla. Explicable reward design for reinforcement learning agents. *Advances in Neural Information Processing Systems*, 34:20118–20131, 2021.
- [10] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168, 2022.
- [11] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.
- [12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.
- [13] Jonathan Ho and Stefano Ermon. Generative adversarial imitation learning. *Advances in Neural Information Processing Systems*, 29, 2016.
- [14] S Hochreiter. Long short-term memory. *Neural Computation*, MIT-Press, 1997.
- [15] Matthew D Hoffman, David M Blei, Chong Wang, and John Paisley. Stochastic variational inference. *Journal of Machine Learning Research*, 2013.
- [16] Matthew W Hoffman, Bobak Shahriari, John Aslanides, Gabriel Barth-Maron, Nikola Momchev, Danila Sinopalnikov, Piotr Stańczyk, Sabela Ramos, Anton Raichuk, Damien Vincent, et al. Acme: A research framework for distributed reinforcement learning. *arXiv preprint arXiv:2006.00979*, 2020.
- [17] Jiaheng Hu, Rose Hendrix, Ali Farhadi, Aniruddha Kembhavi, Roberto Martín-Martín, Peter Stone, Kuo-Hao Zeng, and Kiana Ehsani. Flare: Achieving masterful and adaptive robot policies with large-scale reinforcement learning fine-tuning. *arXiv preprint arXiv:2409.16578*, 2024.
- [18] Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *International Conference on Machine Learning*, pages 14096–14113, 2023.
- [19] Diederik P Kingma, Max Welling, et al. Auto-encoding variational bayes, 2013.
- [20] Alexander Kolesnikov, André Susano Pinto, Lucas Beyer, Xiaohua Zhai, Jeremiah Harmsen, and Neil Houlsby. Uvim: A unified modeling approach for vision with learned guiding codes. *Advances in Neural Information Processing Systems*, 35:26295–26308, 2022.
- [21] Alex Kuefler, Jeremy Morton, Tim Wheeler, and Mykel Kochenderfer. Imitating driver behavior with generative adversarial networks. In *IEEE Intelligent Vehicles Symposium*, pages 204–211, 2017.
- [22] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. *Advances in Neural Information Processing Systems*, 33: 1179–1191, 2020.
- [23] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning? *arXiv preprint arXiv:2204.05618*, 2022.
- [24] Adrian Łańcucki, Jan Chorowski, Guillaume Sanchez, Ricard Marxer, Nanxin Chen, Hans JGA Dolfing, Sameer Khurana, Tanel Alumäe, and Antoine Laurent. Robust training of vector quantized bottleneck models. In *International Joint Conference on Neural Networks*, pages 1–7, 2020.
- [25] Sascha Lange, Thomas Gabel, and Martin Riedmiller. Batch reinforcement learning. In *Reinforcement learning: State-of-the-art*, pages 45–73. Springer, 2012.

- [26] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.
- [27] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.
- [28] Hao Ma, Dieter Büchler, Bernhard Schölkopf, and Michael Muehlebach. Reinforcement learning with model-based feedforward inputs for robotic table tennis. *Autonomous Robots*, 47(8):1387–1403, 2023.
- [29] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. *arXiv preprint arXiv:2108.03298*, 2021.
- [30] Jiageng Mao, Yuxi Qian, Junjie Ye, Hang Zhao, and Yue Wang. Gpt-driver: Learning to drive with gpt. *arXiv preprint arXiv:2310.01415*, 2023.
- [31] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations*, 2024.
- [32] Atharva Mete, Haotian Xue, Albert Wilcox, Yongxin Chen, and Animesh Garg. Quest: Self-supervised skill abstractions for learning continuous control. *arXiv preprint arXiv:2407.15840*, 2024.
- [33] Michael Muehlebach and Raffaello D’Andrea. Nonlinear analysis and control of a reaction-wheel-based 3-d inverted pendulum. *IEEE Transactions on Control Systems Technology*, 25(1): 235–246, 2017.
- [34] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.
- [35] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *International Conference on Machine Learning*, pages 17359–17371, 2022.
- [36] Tim Pearce, Tabish Rashid, Anssi Kanervisto, Dave Bignell, Mingfei Sun, Raluca Georgescu, Sergio Valcarcel Macua, Shan Zheng Tan, Ida Momennejad, Katja Hofmann, et al. Imitating human behaviour with diffusion models. *arXiv preprint arXiv:2301.10677*, 2023.
- [37] Xue Bin Peng, Pieter Abbeel, Sergey Levine, and Michiel Van de Panne. Deepmimic: Example-guided deep reinforcement learning of physics-based character skills. *ACM Transactions On Graphics*, 37(4):1–14, 2018.
- [38] Xue Bin Peng, Ze Ma, Pieter Abbeel, Sergey Levine, and Angjoo Kanazawa. Amp: Adversarial motion priors for stylized physics-based character control. *ACM Transactions on Graphics*, 40(4):1–20, 2021.
- [39] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682–697, 2008.
- [40] Michael Psenka, Alejandro Escontrela, Pieter Abbeel, and Yi Ma. Learning a diffusion model policy from rewards via q-score matching. *arXiv preprint arXiv:2312.11752*, 2023.
- [41] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI*, 2018. URL https://cdn.openai.com/research-covers/language-unsupervised/language_understanding_paper.pdf.
- [42] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

- [43] Stefan Schaal. Learning from demonstration. *Advances in Neural Information Processing Systems*, 9, 1996.
- [44] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.
- [45] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in Neural Information Processing Systems*, 35:22955–22968, 2022.
- [46] A.P. Shon, K. Grochow, and R.P.N. Rao. Robotic imitation from human motion capture using Gaussian processes. In *IEEE-RAS International Conference on Humanoid Robots*, pages 129–134, 2005.
- [47] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.
- [48] Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. *arXiv preprint arXiv:2303.01469*, 2023.
- [49] Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12, 1999.
- [50] Michael Tschannen, Cian Eastwood, and Fabian Mentzer. Givt: Generative infinite-vocabulary transformers. In *European Conference on Computer Vision*, pages 292–309. Springer, 2024.
- [51] Michael Tschannen, André Susano Pinto, and Alexander Kolesnikov. Jetformer: An autoregressive generative model of raw images and text. In *International Conference on Learning Representations*, 2025.
- [52] Aaron Van Den Oord and Oriol Vinyals. Neural discrete representation learning. *Advances in Neural Information Processing Systems*, 30, 2017.
- [53] A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.
- [54] Zhendong Wang, Jonathan J Hunt, and Mingyuan Zhou. Diffusion policies as an expressive policy class for offline reinforcement learning. *arXiv preprint arXiv:2208.06193*, 2022.
- [55] Hongtao Wu, Ya Jing, Chilam Cheang, Guangzeng Chen, Jiafeng Xu, Xinghang Li, Minghuan Liu, Hang Li, and Tao Kong. Unleashing large-scale video generative pre-training for visual robot manipulation. *arXiv preprint arXiv:2312.13139*, 2023.
- [56] Yifan Wu, George Tucker, and Ofir Nachum. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*, 2019.
- [57] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 30:495–507, 2021.
- [58] Yuke Zhu, Ziyu Wang, Josh Merel, Andrei Rusu, Tom Erez, Serkan Cabi, Saran Tunyasuvunakool, János Kramár, Raia Hadsell, Nando de Freitas, et al. Reinforcement and imitation learning for diverse visuomotor skills. *arXiv preprint arXiv:1802.09564*, 2018.
- [59] Jasan Zughaibi, Bradley J Nelson, and Michael Muehlebach. Dynamic electromagnetic navigation. *Robotics and Automation Letters*, 10(6):6095–6102, 2025.

A GMM Properties

GMM Moments

Let $\mathbf{X}$ be a random variable from a k -mixture GMM distribution in $\mathbb{R}^d$. The probability density function can then be written as:

$$p(\mathbf{x}) = \sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad (6)$$

where $\mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ is the multivariate Gaussian distribution for the i -th component. The mean and the covariance of $\mathbf{X}$ are given by:

$$\mathbb{E}[\mathbf{X}] = \sum_{i=1}^k \pi_i \boldsymbol{\mu}_i, \quad (7)$$

$$\text{Cov}[\mathbf{X}] = \sum_{i=1}^k \pi_i (\boldsymbol{\Sigma}_i + (\boldsymbol{\mu}_i - \mathbb{E}[\mathbf{X}])(\boldsymbol{\mu}_i - \mathbb{E}[\mathbf{X}])^\top), \quad (8)$$

where π_i are the mixing coefficients, and $\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i$ are the mean and covariance parameters for the i -th Gaussian component. Note that while the mean of the GMM is a weighted sum of the component means, the covariance is not. The total covariance reflects both the inherent variability within each component and the variability introduced by the differences in component means.

B GMM Mode Finding and Sampling

Mode Finding

To find the modes of the GMM, we compute the gradient of the probability density function with respect to $\mathbf{x}$ and equate it to zero:

$$\nabla p(\mathbf{x}) = \sum_{i=1}^k \pi_i \nabla \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \mathbf{0}, \quad (9)$$

where the gradient of the multivariate Gaussian distribution is given by

$$\nabla \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\mu}_i - \mathbf{x}). \quad (10)$$

We obtain therefore:

$$\sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\mu}_i - \mathbf{x}) = \mathbf{0}. \quad (11)$$

We identify the modes of the GMM by computing all critical points $\mathbf{x}^*$ that satisfy (11). Moreover, to ensure that a critical point $\mathbf{x}^*$ is indeed a mode, the Hessian of the density function at that point must be negative definite. The Hessian of the GMM density is given by

$$\nabla^2 p(\mathbf{x}) = \sum_{i=1}^k \pi_i \nabla^2 \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i), \quad (12)$$

where the Hessian of the multivariate Gaussian distribution is:

$$\nabla^2 \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) = \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) [\boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\mu}_i - \mathbf{x})(\boldsymbol{\mu}_i - \mathbf{x})^\top \boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_i^{-1}]. \quad (13)$$

Therefore, the Hessian of the GMM density becomes

$$\nabla^2 p(\mathbf{x}) = \sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{x} \mid \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) [\boldsymbol{\Sigma}_i^{-1} (\boldsymbol{\mu}_i - \mathbf{x})(\boldsymbol{\mu}_i - \mathbf{x})^\top \boldsymbol{\Sigma}_i^{-1} - \boldsymbol{\Sigma}_i^{-1}], \quad (14)$$

and a critical point $\mathbf{x}^*$ is a mode if: $\nabla p(\mathbf{x}^*) = \mathbf{0}$ and $\nabla^2 p(\mathbf{x}^*)$ is negative definite.

While GMMs are often parameterized by a finite number of components, the resulting distribution can exhibit a richer structure, particularly in higher-dimensional spaces. Specifically, **in dimensions greater than two** ($d > 2$), a GMM can possess **more modes (local maxima) than the number of its constituent components** [5]. This phenomenon arises due to the interplay between the covariance structures and the relative positions of the Gaussian components. In higher dimensions, the overlapping regions of different components can create multiple peaks in the probability density function that cannot be attributed to individual components.

Mean-Shift

We start by solving (11) for $\mathbf{x}$ as follows:

$$\mathbf{x} = \left(\sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \boldsymbol{\Sigma}_i^{-1} \right)^{-1} \sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i. \quad (15)$$

To obtain the mean-shift update in practice, one can treat the solution to (15) as a *fixed-point iteration*. More precisely, we define the operator $T : \mathbb{R}^d \rightarrow \mathbb{R}^d$ by

$$T(\mathbf{x}) = \left(\sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \boldsymbol{\Sigma}_i^{-1} \right)^{-1} \sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \boldsymbol{\Sigma}_i^{-1} \boldsymbol{\mu}_i. \quad (16)$$

Then, starting from an initial guess $\mathbf{x}^{(0)}$, the mean-shift procedure updates the estimate of $\mathbf{x}$ via

$$\mathbf{x}^{(t+1)} = T(\mathbf{x}^{(t)}).$$

The fixed-point iteration has the following interpretation: each iteration *shifts* the current iterate after t iterations closer to the modes (high-density regions) of the underlying density defined by the mixture of Gaussians. In practice, it has been shown that initializing $\mathbf{x}^{(0)}$ with the component means $\boldsymbol{\mu}_i$ results in fast convergence [5], which we also observe in our experiments. To account for the fact that the number of modes could exceed the number mixture components, we add extra initialization points from the convex-hull of the component centroids capturing the most important modes.

Mean-Shift with Diagonal-Covariance Components

An important special case arises when the covariance matrices $\boldsymbol{\Sigma}_i$ are *diagonal*. In that scenario, let

$$\boldsymbol{\Sigma}_i = \text{diag}(\sigma_{i,1}^2, \dots, \sigma_{i,d}^2).$$

Since each $\boldsymbol{\Sigma}_i^{-1}$ is also diagonal (with entries $1/\sigma_{i,j}^2$ along the diagonal), the vector and matrix operations inside $T(\mathbf{x})$ decouple across dimensions. Hence the inverse of $\sum_{i=1}^k \pi_i \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \boldsymbol{\Sigma}_i^{-1}$ reduces to a diagonal matrix whose components can be computed efficiently. As a result, each coordinate of the updated point can be determined independently, making the mean-shift iteration particularly fast on modern hardware.

Modes Sampling

To sample the modes proportional to their coverage of the data support, we approximate the integral of the GMM density in the local neighborhood of each mode using a *Laplace approximation* and use the value of the integral around each mode to compute the relative weight of the mode. We summarize the full mode-finding and sampling procedure in Algorithm 1.

C Environment Details

In our experiments, we evaluated Q-FAT in three distinct environments: Kitchen, PushT, and UR3 BlockPush. In the following, we provide a brief description of each environment and its variants.

- **Franka Kitchen:** This environment involves a Franka Panda robotic arm with a nine dimensional action space, designed for multi-task manipulation in a simulated kitchen

Algorithm 1 Mode Extraction and Sampling

```
1: Inputs: GMM parameters  $\{\pi_i, \mu_i, \Sigma_i\}_{i=1}^k$ ; tolerance  $\epsilon$ ; rejection threshold  $\theta$ ; max iterations  
   max_it; number of initializations n_init;  
2: Initialize Mode Set:  $\mathcal{M} \leftarrow \emptyset$   
3: Initialize Seed Set:  
4:  $\mathcal{X} \leftarrow \text{Uniform\_Sample}(\text{Convex\_Hull}(\{\mu_i\}), n\_init)$   
5: for  $x \in \mathcal{X}$  do  
6:   Iteration counter  $i \leftarrow 0$   
7:   Define the operator  $T(\cdot)$  as in Equation 16  
8:   repeat  
9:     Update  $\mathbf{x}$  via fixed-point iteration:  $\mathbf{x} \leftarrow T(\mathbf{x})$   
10:    Increment  $i$   
11:   until  $i \geq \text{max\_it}$  or  $\|\mathbf{x} - \mathbf{x}_{\text{old}}\| < \epsilon$   
12:   Compute  $\mathbf{H} = \nabla^2 p(\mathbf{x})$   
13:   if  $\max(\text{eigenvalues}(\mathbf{H})) < 0$  then  
14:     Merge nearby modes within a distance threshold, keeping the highest-probability mode  
15:     Add mode to  $\mathcal{M}$   
16:   end if  
17: end for  
18:  
19: Compute Mode Weights:  
20:  $w_j = p(\mathbf{x}_j) |\nabla^2 \log p(\mathbf{x}_j)|^{-1/2}$ ,  $j = 0, \dots, |\mathcal{M}|$   
21: Sample mode: Sample  $j \sim \text{Categorical}(\{w_j\}_{j=0}^{|\mathcal{M}|})$   
22: Return:  $\mathbf{x}_j$ 
```

setting [11]. The dataset consists of 566 human-collected demonstrations, where each trajectory completes a subset of four out of seven possible tasks in varying orders. For the image-based version of the environment, we rendered the collected trajectories into 112x112 images.

- **PushT:** The PushT environment focuses on pushing a T-shaped block to a target position on a table using two dimensional end-effector velocity control [7]. The dataset includes 206 human-collected demonstrations.
- **UR3 BlockPush:** In this task, a UR3 robotic arm is tasked with moving two blocks to designated goal circles on a table [10]. The dataset exhibits multimodality, as the blocks can be moved in either order. In the unconditional setting, we evaluate whether both blocks reach their respective goals. In the conditional setting, the model is provided with the target positions of the blocks, and performance is assessed based on the order in which the blocks reach their goals.
- **BlockPush:** The goal of this task is moving two (red and blue) blocks two targets [10]. The blocks can be moved in either order and can be put into either of the targets. While this environment is similar to UR3 BlockPush, it exhibits more stochasticity due to the random initialization of the block positions, and the targets being randomly translated and rotated. The dataset contained 1000 demonstrations.
- **Multimodal Ant:** The Multimodal Ant environment [26] is a modification of the Ant environment introduced by Schulman et al. [44] where the goal of the ant is to visit four distinct locations placed on vertices of a square. The multimodality in the dataset arises from the different order the goals are visited. The dataset contains 600 human-collected trajectories

D Training Details

For our experiments, we used the minGPT [3] backbone as the decoder-only transformer implementation. The hyperparameters used for each of the environments are detailed in Table 3.

D.1 Architectural Design Choices

The policy was trained using teacher forcing with a causal attention mask, as this provided a stronger learning signal compared to computing the loss only on the last action given a context of states. We found that not using teacher forcing and employing a full attention mask significantly degraded performance. We further experimented with feeding back the sampled actions into the model’s input by concatenating states and actions into a single feature vector. This approach degraded policy performance in simulated environments, likely due to the model overfitting to the highly correlated previous actions in the context while ignoring state information. Injecting Gaussian noise into the previous actions was necessary to achieve reasonable performance in this setting; however, this provided no discernible benefit compared to a policy without action feedback in our experiments.

D.2 History Masking

During training, we observed instances where both validation and training losses decreased, yet the environment rewards declined. We attribute this phenomenon to causal confusion [8], where the model overfits the recent sequence of states. Instead of learning true causal relationships, the model relies on temporally correlated but non-causal patterns to predict future actions, leading to suboptimal performance. To mitigate this, we introduced history masking, randomly masking the context (excluding the current state) with a certain probability during training. Applying this technique in the Kitchen, Multimodal Ant, and BlockPush environments (with masking probabilities of 0.7, 0.3, and 0.3, respectively) led to an improved correlation between environment rewards and validation loss.

D.3 Image-based Environments

For Image PushT and Image Kitchen, we fine-tuned a ResNet18 encoder [12], extracting feature maps from the first four layers to construct low-level features [35]. These features were then spatially average-pooled and fed into the transformer with a dimensionality of 1024. For Image PushT, we applied data augmentation (color jitter, random cropping, and random gray-scaling with probability 0.5), which proved crucial for preserving the invariance of the pre-trained ResNet and preventing overfitting to our dataset. Conversely, for Image Kitchen, we found data augmentation deteriorated performance.

D.4 Goal Conditioning

In goal-conditional tasks, we conditioned the model on a sequence of future states equal in length to the state history. During inference, we randomly sampled a reference trajectory from the dataset. The agent’s reward was computed only if it achieved the goal sequence in the same order as the reference trajectory.

D.5 Evaluation

All training datasets were normalized using min-max scaling to ensure state and action features lie within the range $[-1, 1]$. Following Lee et al. [26], Shafiullah et al. [45], we split the data into 95% for training and 5% for validation. The policy was evaluated periodically during training using a small number of environment roll-outs (20 to 50). The models achieving the lowest validation loss and highest success metrics were selected for reporting.

D.6 Model Size

For a fair comparison with VQ-BeT [26], we ensured our models have a comparable (or smaller) number of trainable parameters, on the order of $10^5 - 10^6$. We slightly deviated from their transformer encoder hyperparameters to account for the additional capacity utilized by their hierarchical vector

Hyperparameter	PushT	Image PushT	Kitchen	UR3	Multimodal Ant	BlockPush	nuScenes
Layers	6	6	6	6	6	4	6
Attention heads	8	8	8	8	8	4	8
Embedding dimension	128	128	128	128	128	72	128
Dropout probability	0.1	0.1	0.1	0.1	0.1	0.1	-
State history size	5	5	10	10	10	3	-
Action horizon	5	5	1	1	1	1	6
Training epochs	1000	1000	1200	400	1000	1200	1000
Batch size	256	256	128	2048	128	256	128
Number of mixtures k	4	4	4	4	4	2	2
Maximum learning rate	1e-3	1e-3	1e-3	1e-4	1e-3	1e-4	1e-4
Minimum learning rate	-	-	1e-6	1e-6	1e-6	1e-7	-
Learning Rate Schedule	-	-	Cosine	Cosine	Cosine	Cosine	-
Optimizer	Adam	Adam	Adam	Adam	Adam	Adam	Adam

Table 3: Environment hyperparameters.

quantization autoencoder for action discretization. Training hyperparameters, such as batch size and learning rates, were adjusted to accommodate the differences in loss functions between the methods. Further details are provided in Table 3 and Table 13 in Lee et al. [26].

D.7 Hardware

Experiments were conducted on a heterogeneous cluster, making precise hardware control infeasible. However, all experiments were run on a single desktop-grade GPU with at most 32 GB of memory. Training a single model typically took 4-8 hours, depending on dataset size and the frequency of environment evaluations for validation.

E nuScenes Experiment

To evaluate our model’s applicability beyond robotic manipulation, we use the nuScenes [4] dataset, a large-scale benchmark for autonomous driving. We utilize the object-centric, preprocessed version of the dataset from Mao et al. [30], and follow the exact preprocessing and tokenization steps for driving data as detailed in the VQ-BeT paper [26]. The model’s input is a set of tokens representing the driving mission (e.g., turn left), the ego-vehicle’s current state (velocity, acceleration), its recent trajectory history, and the state of surrounding objects (position, class). The task is to predict the ego-vehicle’s trajectory over the next six timesteps. Success is evaluated using two primary metrics: the average L2 distance between the predicted and ground-truth trajectories to measure accuracy, and the collision rate to assess the safety of the generated path.

For this sequential prediction task, we found it necessary to adapt our model’s architecture. A naive approach of predicting a single high-dimensional vector concatenating all future waypoints resulted in degraded performance. This is likely because the model is forced to learn a direct mapping to a highly complex and multimodal joint distribution of future states. Such a method struggles to enforce temporal consistency, often producing kinematically implausible trajectories.

Table 4: Preliminary results from our initial experiments on the nuScenes autonomous driving benchmark. Lower values are better.

Method	L2 Error (m)	Collision Rate (%)
Diffusion	0.96	0.44
VQ-BeT	0.73	0.29
Q-FAT (ours)	0.75	0.37

Consequently, we adopted an autoregressive prediction scheme, where the model forecasts one waypoint at a time, conditioned on its previous predictions. This approach simplifies the learning problem by factorizing the joint distribution into a sequence of more manageable conditional distributions. By doing so, the model implicitly learns the transition dynamics of the environment, ensuring that the generated trajectory is temporally coherent and physically plausible. Furthermore, instead of predicting the absolute coordinates of each waypoint, our model predicts the *delta*, or displacement,

from the previous waypoint. This encourages the model to learn a policy that is invariant to the absolute frame of reference, which has been shown to improve generalization.

The results from these initial experiments are presented in Table 4. These findings show that Q-FAT achieves performance competitive with the VQ-BeT baseline, demonstrating that our method is effective in this challenging domain. The training hyperparameters can be found in Table 3.

NeurIPS Paper Checklist

1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

Answer: [\[Yes\]](#)

Justification: We introduce Q-FAT and evaluate it extensively on common bench marks, demonstrating SOTA performance.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: [\[Yes\]](#)

Justification: See Section 7.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. Theory assumptions and proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [\[NA\]](#)

Justification: We do not present any theoretical claims.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. Experimental result reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [\[Yes\]](#)

Justification: See Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
 - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
 - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
 - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
 - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: Our code and datasets will be made publicly available.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. Experimental setting/details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: See Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. Experiment statistical significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: It was computationally expensive to train multiple models for each environment to assess the modeling uncertainty. As for the aleatoric uncertainty in the environments, it was negligible since we ran evaluations on 1000 episodes and averaged the results.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. Experiments compute resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: See Appendix D.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. Code of ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: We will open source our code with public licenses. We further clearly stated the societal impact in Section 8.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. Broader impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: See Section 8.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We train specialized models for simulated robotics tasks, which does not require safeguards.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We ensured that all the data and code that was used was made publicly available for research use.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, paperswithcode.com/datasets has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [\[Yes\]](#)

Justification: We will open source our code.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [\[NA\]](#)

Justification: Our work does not include human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [\[NA\]](#)

Justification: Our work does not include human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. Declaration of LLM usage

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: We did not use LLMs to conduct this research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (<https://neurips.cc/Conferences/2025/LLM>) for what should or should not be described.