IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

Motion Manifold Flow Primitives for Task-Conditioned Trajectory Generation under Complex Task-Motion Dependencies

Yonghyeon Lee[1], Byeongho Lee[2], Seungyeon Kim[3], and Frank C. Park[3]

***Abstract*—Effective movement primitives should be capable of encoding and generating a rich repertoire of trajectories conditioned on task-defining parameters such as vision or language inputs. While recent methods based on the motion manifold hypothesis, which assumes that a set of trajectories lies on a lower-dimensional nonlinear subspace, address challenges such as limited dataset size and the high dimensionality of trajectory data, they often struggle to capture complex task-motion dependencies, i.e., when motion distributions shift drastically with task variations. To address this, we introduce Motion Manifold Flow Primitives (MMFP), a framework that decouples the training of the motion manifold from task-conditioned distributions. Specifically, we employ flow matching models, state-of-the-art conditional deep generative models, to learn task-conditioned distributions in the latent coordinate space of the learned motion manifold. Experiments are conducted on language-guided trajectory generation tasks, where many-to-many text-motion correspondences introduce complex task-motion dependencies, highlighting MMFP's superiority over existing methods.**

***Index Terms*—Imitation Learning, Learning from Demonstration, Representation Learning, Movement Primitives**

I. Introduction

MOVEMENT primitives are mathematical models that encode trajectories *offline* and enable rapid *online* generation, typically fitted to data from human demonstrations [1], [2], [3], [4], [5], [6]. They avoid the need for time-consuming online motion planning that reduces the system's reactivity, allowing fast adaptation to changes in environments.

In this paper, we adopt the view that effective primitive models should satisfy two key properties. First, they should encode a diverse set of trajectories for a given task to enhance adaptability, ensuring that if some trajectories become unexecutable, such as being blocked by unforeseen obstacles, feasible alternatives remain available. Second, they should be capable of generating motions conditioned on task-defining parameters, such as vision data describing the environment or language commands reflecting user intent, ensuring applicability across a wide range of tasks.

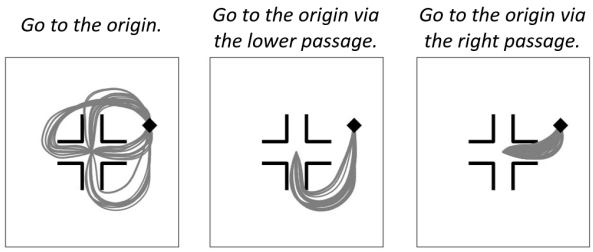

Fig. 1. An illustrative example of a language-guided navigation scenario with complex task-motion dependencies: The motion distribution shifts dramatically – for instance, in the number of modalities – when the task parameter (in this case, a language command) changes.

Learning these primitives poses two main challenges. First, obtaining a sufficiently large dataset of human demonstrations is challenging, often leading to limited data availability. This difficulty is amplified by the need to collect data for each specific task parameter, further increasing data demands. Second, trajectory data is inherently high-dimensional, particularly for long-horizon motions. When expressed as a sequence of configurations at discrete time steps, its dimensionality increases linearly with the number of time steps.

Leveraging a low-dimensional manifold structure can be an effective solution. Assuming the set of demonstration trajectories lies on a lower-dimensional manifold embedded in the trajectory space (the manifold hypothesis), recent works have focused on identifying this manifold to reduce data dimensionality and address both challenges [4], [5]. Task-conditioned variational autoencoders (TCVAE) [4] and motion manifold primitives (MMP) [5] employ conditional autoencoder architectures, demonstrating promising results for task-conditioned, diverse, high-dimensional trajectory generation.

However, existing manifold-based models fail to capture the *complex* task dependencies of motion distributions, where "complex dependencies" refer to cases in which the motion distribution undergoes dramatic shifts (e.g., changes in the number of modalities or the volume of support) as the task parameter varies. For example, consider the navigation sce-

Manuscript received: January 7, 2025; Revised: April 5, 2025; Accepted: May 20, 2025.

This paper was recommended for publication by Editor Aleksandra Faust upon evaluation of the Associate Editor and Reviewers' comments. This work was supported in part by IITP-MSIT grant RS-2021-II212068 (SNU AI Innovation Hub), IITP-MSIT grant 2022-220480, RS-2022-II220480 (Training and Inference Methods for Goal Oriented AI Agents), IITP-MSIT grant RS-2024-00436680 (Collaborative Research Projects with Microsoft Research) under the Global Research Support Program in the Digital Field program, KIAT grant P0020536 (HRD Program for Industrial Innovation), SRRC NRF grant RS-2023-00208052, SNU-AIIS, SNU-IPAI, SNU-IAMD, SNU BK21+ Program in Mechanical Engineering, SNU Institute for Engineering Research, Hyundai Motor Company and Kia, and Microsoft Research Asia.

[1]Y. Lee is with the Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA 02139 USA yhl@mit.edu

[2]B. Lee is with the Center for Humanoid Research, Korea Institute of Science and Technology, Seoul 02792, South Korea bhlee194@gmail.com

[3]S. Kim and F. C. Park are with Robotics Laboratory, Seoul National University, Seoul 08826, South Korea ksy@robotics.snu.ac.kr, fcp@snu.ac.kr

Digital Object Identifier (DOI): see top of this page.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

nario shown in Fig. 1, where the task parameter is specified by a language command. If the command is "Go to the origin", the robot can select any of the four available passages. However, if the command changes to "Go to the origin via the lower passage", the robot must exclusively take the lower passage. This change drastically alters the motion distribution, reducing the number of modalities. As we demonstrate later in our experiments, existing models fail in these scenarios. The fundamental reason for these failures is their reliance on conditional autoencoder architectures with a shared latent prior distribution, as we detail in Section IV-A.

In this paper, we propose Motion Manifold Flow Primitives (MMFP), a framework that decouples the training of the motion manifold and task-conditioned distributions, effectively capturing complex task-motion dependencies. Our approach leverages flow matching models [7], [8], state-of-the-art conditional deep generative models capable of effectively capturing complex conditional distributions, in the latent coordinate space of the learned motion manifold. As a result, MMFP simultaneously addresses the challenges of limited datasets, high dimensionality, and complex task-motion dependencies, and significantly outperforms existing manifold-based models.

It is instructive to highlight the differences between recent learning-from-demonstration methods based on diffusion and flow matching models – such as diffusion policies [9], [10] and flow matching policies [11], [12] – and our proposed MMFP. The key distinction lies in whether the low-dimensional manifold structure is explicitly utilized. These existing methods focus on generating relatively low-dimensional *local* trajectories for policy learning, whereas our aim is to produce *global*, high-dimensional trajectories for motion planning, requiring dimensionality reduction. As we show in our experiments, simply training diffusion or flow matching models without manifold learning fails to generate valid global trajectories.

We provide case studies on language-guided trajectory generation, where each demonstration trajectory is paired with multiple hierarchical text descriptions. This setup introduces many-to-many mappings between text and motion, leading to complex text dependencies in the motion distributions. MMFP demonstrates excellent performance, while diffusion and flow-based models trained directly in the high-dimensional trajectory space [13], [7], [8] and existing motion manifold-based methods [4], [5] fail to generate successful trajectories.

II. Related Works

A. Movement primitives

Movement primitives have been widely studied, including dynamical systems-based approaches like DMPs [14], [15] and stable dynamical systems [3], [16], [17], which ensure closed-loop stability. Probabilistic and manifold-based methods have also been proposed to capture diverse motions [18], [2], [19], [4], [5], [6], [20].

Of particular relevance to our work are *manifold-based models* and *task-conditioned movement primitives*. Recent studies have explored autoencoder-based manifold learning algorithms for learning movement primitives [4], [20], [5], [6]. [20] focuses on learning a manifold of *configurations*, followed by solving geodesic equations for motion planning. In contrast, [4], [5], [6] learn a manifold of *trajectories*, enabling direct motion sampling. Our approach aligns with the latter, as it also focuses on learning a manifold of trajectories.

Task-Parametrized Gaussian Mixture Model (TP-GMM) can produce new movements for unseen task parameters; however, it primarily addresses parameters defined as coordinate frames, making it less suitable for more general types of inputs such as vision or language [18]. Among the manifold-based models mentioned above, [4], [5] – except [6] which does not address task-conditioned motion generation – adopt conditional autoencoder architectures capable of handling a broader range of task parameters while encoding and generating diverse trajectories, closely aligning with our research goals. However, these methods fall short of capturing the *complex* task-motion dependencies within the motion distribution, a limitation we specifically address in this paper.

B. Diffusion and flow matching for imitation learning

Diffusion and flow matching models have recently shown strong performance in conditional generation tasks [21], [7], [8], [22]. In robotics, they have gained traction in imitation learning for training stochastic policies – models that generate actions conditioned on observations. Notable examples include Diffusion Policy [9], 3D Diffusion Policy [10], and Flow Matching Policy [12], which address various observation modalities such as 2D images and 3D point clouds.

The key difference between recent diffusion or flow matching policies and our method lies in the research objective. While these approaches focus on learning a *local policy* that generates low-dimensional *local* actions – both temporally and spatially – often limited to a few steps of desired configurations, our goal is to develop a *global motion planner* capable of producing high-dimensional *global* trajectories. This increased dimensionality makes a naive application of diffusion or flow matching methods prone to failure, necessitating motion manifold learning as proposed in this paper.

Moreover, while local policies can generate actions with high-frequency feedback, the produced actions often lack temporal consistency with previous ones. When actions are defined at the trajectory level, the final state of one action may not smoothly connect with the initial state of the next, leading to non-smooth behavior. For tasks where frequent trajectory regeneration is unnecessary, a global planner is generally better suited for producing long-horizon, smooth motions.

III. Preliminaries

A. Autoencoder-based manifold learning

In this section, we briefly introduce an *autoencoder* and its manifold learning perspective [23], [24], [25], [26], [27], [28], [29], [30]. Consider a high-dimensional data space $\mathcal{X}$ and a set of data points $\mathcal{D} = \{x^i \in \mathcal{X}\}_{i=1}^N$, and assume the data points $\{x^i\}$ lie approximately on some lower-dimensional manifold $\mathcal{M}$ in $\mathcal{X}$ (the manifold hypothesis). Suppose $\mathcal{M}$ is an m-dimensional manifold and let $\mathcal{Z}$ be a latent space $\mathbb{R}^m$.

An autoencoder consists of an encoder, a mapping $g : \mathcal{X} \to \mathcal{Z}$, and a decoder, a mapping $f : \mathcal{Z} \to \mathcal{X}$. These

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

are often approximated with deep neural networks and trained to minimize the reconstruction loss $\frac{1}{N}\sum_{i=1}^{N} d^2(f \circ g(x^i), x^i)$ given a distance metric $d(\cdot,\cdot)$ on $\mathcal{X}$. We note that, given a sufficiently low reconstruction error, all the data points $\{x^i\}$ should lie on the image of the decoder f. Under some mild conditions – that are (i) m is lower than the dimension of the data space $\mathcal{X}$ and (ii) f is smooth and its Jacobian $\frac{\partial f}{\partial z}(z) \in \mathbb{R}^{dim(\mathcal{X})\times m}$ is full rank everywhere –, the image of f is an m-dimensional differentiable manifold embedded in $\mathcal{X}$. In other words, the decoder produces a lower-dimensional manifold where the data points approximately lie.

B. Flow matching models

In this section, we give a brief overview of *flow matching models* [7], [31], a state-of-the-art class of deep generative models. Let $\{x^i \in \mathcal{X}\}_{i=1}^N$ be a set of data points sampled from the underlying probability density $q(x)$. Consider a non-autonomous vector field $v : [0,1] \times \mathcal{X} \to T\mathcal{X}$ that leads to a flow $\phi : [0,1] \times \mathcal{X} \to \mathcal{X}$ via the following ordinary differential equation (ODE): $\frac{d}{ds}\phi_s(x) = v_s(\phi_s(x))$, where $\phi_0(x) = x$. Given a prior density at $s = 0$ denoted by p_0, the flow ϕ_s leads to a probability density path for $s \in [0,1]$ [7]: $p_s(x) = p_0(\phi_s^{-1}(x)) \det\big(\frac{\partial \phi_s^{-1}}{\partial x}(x)\big)$. Our objective is to learn a neural network model of $v_s(x)$ so that the flow of $v_s(x)$ transforms a simple prior density p_0 (e.g., Gaussian) to the target data distribution $p_1 \approx q$. Then we can sample new data points by solving the ODE, $x' = v_s(x)$, from $s = 0$ to $s = 1$ with initial points sampled from p_0.

A key innovation in [7], [31] is the *flow matching loss*, which enables training $v_s(x)$ efficiently *without* simulating the ODE:

$$\mathbb{E}_{x_0\sim p_0(x),\, x_1\sim q(x),\, s\sim\mathcal{U}[0,1]}\Big[\|v_s(x_s) - (x_1 - x_0)\|^2\Big], \quad (1)$$

where $x_s = (1-s)x_0 + sx_1$. Here, $\mathcal{U}[0,1]$ is the uniform distribution over $[0,1]$. Minimizing (1) gives a vector field $v_s(x)$ that transforms p_0 into a final density p_1, closely approximating the data distribution q.

We are particularly interested in a *conditional* density function $p(x|c)$, where c is a conditioning variable. To address this, a neural network vector field $v_s(x,c)$ takes both x and c as inputs, and is trained using a similar objective:

$$\mathbb{E}_{x_0\sim p(x_0),\, (x_1,c)\sim q(x,c),\, s\sim\mathcal{U}[0,1]}\Big[\|v_s(x_s,c) - (x_1 - x_0)\|^2\Big], \quad (2)$$

where $x_s = (1-s)x_0 + sx_1$, and $q(x,c)$ is the underlying joint distribution of data points and their conditioning variables. By incorporating c into the vector field, the flow can adapt to various context inputs, enabling conditional sample generation.

IV. Motion Manifold Flow Primitives

We begin this section by introducing the notations and assumptions used throughout the paper. Let $\mathcal{Q}$ be a configuration space, and denote a sequence of configurations by $x = (q_1, \ldots, q_T)$, referred to as a trajectory, where $q_i \in \mathcal{Q}$ for all i and T is a fixed positive integer representing the length of trajectory. The trajectory space is then denoted by $\mathcal{X} = \mathcal{Q}^T$, which is typically very high-dimensional. We denote the task parameter by $c \in \mathcal{C}$ and assume access to a trajectory-task pair dataset $\mathcal{D} = \{(x^i, c^i)\}_{i=1}^N$, where many-to-many correspondences exist between x and c (e.g., multiple values of c may correspond to the same x and vice versa).

In the subsequent sections, we first explain the limitations of existing task-conditioned motion manifold primitives, introduce our Motion Manifold Flow Primitives (MMFP), and describe its specific application when the task parameter is a free-form text input.

A. Limitations of existing motion manifold primitives

Existing motion manifold-based models, TCVAE [4] and MMP [5], adopt conditional autoencoder architectures for task-conditioned trajectory generation. Both assume a lower-dimensional latent space $\mathcal{Z}$ and train an encoder $g : \mathcal{X} \to \mathcal{Z}$ and a conditional decoder $f : \mathcal{Z} \times \mathcal{C} \to \mathcal{X}$ (or their stochastic variants) that map a trajectory x to a latent representation z and generate a trajectory given (z, c). The latent prior $p_{\text{prior}}(z)$ is typically modeled as a Gaussian or a Gaussian mixture model. Motion sampling involves drawing z from the prior and mapping it through the decoder f.

We note that the latent prior is *shared* across all task parameters, requiring the conditional decoder to sufficiently distort the same prior into different motion distributions based on the task parameter. However, when motion distributions shift dramatically (complex task-motion dependencies), the decoder often struggles to capture the complex nonlinear transformations required. In contrast, the flow matching models introduced in Section III-B continuously transform the prior via an ODE, where smooth, incremental changes accumulate into significant variations, making them better suited for modeling such complex dependencies.

B. Decoupling manifold learning and conditional densities

Our Motion Manifold Flow Primitives (MMFP) framework consists of two modules: (i) motion manifold and (ii) latent flow. The motion manifold model consists of two neural networks, an encoder $g : \mathcal{X} \to \mathcal{Z}$ and a decoder $f : \mathcal{Z} \to \mathcal{X}$, which are trained as follows:

$$\min_{f,g} \frac{1}{N}\sum_{i=1}^{N} d^2(x^i, f(g(x^i))) + \eta\|g(x^i)\|^2 + \delta\mathcal{E}(f,g), \quad (3)$$

where η, δ are some positive scalars. The second term penalizes the norm of latent values to prevent them from diverging excessively far from the origin. The third term is added to ensure their smoothness, which is defined as follows: $\mathcal{E}(f,g) := \mathbb{E}_z\big[\sum_{t=1}^{T-1}\|f^{t+1}(z) - f^t(z)\|^2\big]$, where $f(z) = (f^1(z), \ldots, f^T(z)) \in \mathcal{Q}^T$. The latent value z in this expectation is sampled with augmentation as $z = \alpha g(x^i) + (1-\alpha)g(x^j)$, where $\alpha \sim \mathcal{U}[-0.4, 1.4]$ and x^i, x^j are sampled from the dataset, to extend the regularization effect to regions where data is not available, adopting [26].

Given a pre-trained encoder g, we generate a set of latent-task pairs $\{(z^i, c^i)\}_{i=1}^N$ where $z^i = g(x^i)$. A neural network vector field $v_s(z,c)$ is then trained using the flow matching

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

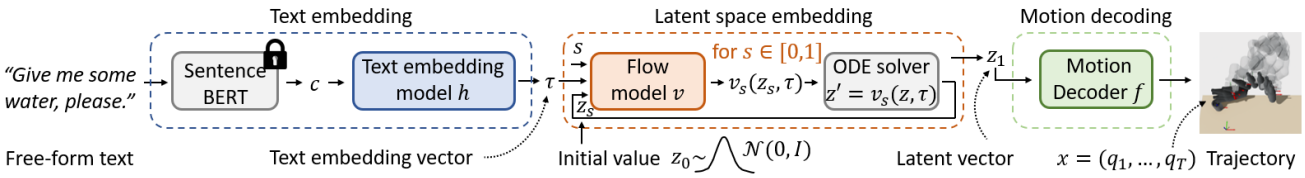

Fig. 2. The procedure of motion generation in MMFP: (i) the Sentence-BERT encodes a free-form text into a vector c, (ii) the text embedding model h maps c to a text embedding vector τ, (iii) we solve the ODE $z' = v_s(z, \tau)$ from $s = 0$ to $s = 1$ with an initial value z_0 sampled from Gaussian $\mathcal{N}(z|0, I)$ and obtain $z_1 \in \mathcal{Z}$, and (iv) the motion decoder f maps z_1 to a trajectory $x = (q_1, \ldots, q_T)$.

loss in (2). With this process, we learn a distinct latent prior for each conditional vector c, i.e., we learn $p(z|c)$, overcoming the shortcomings of the shared prior $p_{\text{prior}}(z)$ used in previous works. Finally, given a task parameter c, motion sampling is performed by integrating the ODE $dz/ds = v_s(z, c)$ from $s = 0$ to $s = 1$ to obtain latent samples z_1 (with the subscript indicating the value at $s = 1$), followed by decoding them using $f : z_1 \mapsto x$.

A reasonable question may arise: Why not use diffusion models [21], which are modeled with stochastic differential equations (SDEs), instead of flow matching models based on ordinary differential equations (ODEs), in the latent space? One answer is that while both approaches can capture complex conditional dependencies, flow matching models enable faster generation [7]. Moreover, given a limited dataset size, flow matching tends to achieve better interpolation and generalization compared to diffusion models, as optimal transport paths used in flow matching are smoother than the diffusion paths [7]. Our empirical studies further support this, showing that latent flow models outperform latent diffusion models; see Section V-A.

C. Language as a task parameter

We use a pre-trained text encoder, Sentence-BERT [32], without fine-tuning, to encode free-form texts into 768-dimensional vectors. These vectors serve as task parameters, denoted by c. However, we empirically observed that directly inputting the 768-dimensional task parameter into the velocity model $v_s(z, c)$ does not produce satisfactory results. To address this limitation, we introduce a text embedding model, a neural network h, which encodes c into a lower-dimensional vector $\tau \in \mathcal{T}$.

We train the text embedding model $h : c \mapsto \tau$ and velocity field $v : (s, z, \tau) \mapsto dz/ds$ simultaneously with the following *regularized* flow matching loss:

$$\sum_{i,k} \mathbb{E}_{s,z_0}\left[\|v_s(z_s^i, h(c^i)) - (z^i - z_0)\|^2\right] + \gamma \|h(c^i) - h(\tilde{c}^{ik})\|^2, \quad (4)$$

where $z_s^i = (1 - s)z_0 + sz^i$, and s, z_0 are sampled from $\mathcal{U}[0, 1]$ and a Gaussian prior, respectively. The second term, scaled by a positive weight γ, encourages robustness against diverse text variations. Here, $\tilde{c}^{ik}$ for $k = 1, \ldots, K$ are Sentence-BERT encoding vectors with meanings similar to those of c^i, generated using the Large Language Model *GPT-4* [33]. It is worth noting that GPT-4 may have some biases and may miss certain variations. As LLM models continue to advance, we can leverage the most recent versions for improved performance. We refer to Fig. 2 for an overview of the motion sampling procedure using MMFP given a free-form text input.

V. Experiments

In this section, we evaluate our method, MMFP, primarily compared to (i) Denoising Diffusion Probabilistic Models (DDPM) [13], (ii) Flow Matching (FM) [7], (iii) Task-Conditional Variational Autoencoder (TCVAE) with a Gaussian prior [4], and (iv) Motion Manifold Primitives with a Gaussian mixture prior (MMP) [5]. DDPM and FM are directly trained in the trajectory space $\mathcal{X}$. When $\mathcal{X} = \text{SE}(3)^T$, a Riemannian manifold, we train a DDPM in local coordinates (using exponential coordinates for the $\text{SO}(3)$ component) and use Riemannian Flow Matching (RFM) [8]. MMFP trained without the robustness regularization term (the second term in (4)) will be denoted as MMFP w/o reg.

Throughout, a key hypothesis we are testing is that while DDPM and FM (or RFM) trained directly in the trajectory space fail to learn reasonable motions due to high dimensionality, and existing TCVAE and MMP struggle to capture complex task-motion dependencies, MMFP can effectively address these challenges – especially with enhanced robustness to task variations through a robustness regularization term.

For MMFP, we choose the latent space dimension $m = 3$ for $\mathcal{Z} = \mathbb{R}^m$ and text embedding dimension $p = 3$ for $\mathcal{T} = \mathbb{R}^p$ (unless otherwise specified). For TCVAE and MMP, with thorough tuning, we set the latent space dimension to be 3 or 4 and text embedding dimension to be 64. Throughout, we employ fully connected neural networks with ELU activation functions across all network components.

Evaluation metrics: The primary objective is to generate correct motions corresponding to the given text inputs. To evaluate whether the model generates accurate motions that perform the task described in the given text, we train a trajectory classifier and use it to report the motion accuracy; the higher, the better. Additionally, the model should be able to generate not just a single trajectory but diverse trajectories, imitated from the given demonstration dataset, depending on the text input. For example, see Fig. 6. We use the Maximum Mean Discrepancy (MMD) [34] – which measures the distance between two probability distributions, computed from their samples – to measure the similarity between the set of generated trajectories given a text input and the set of demonstration trajectories annotated with that text; the lower, the better.

Texts are annotated to each trajectory at multiple different levels (e.g., see Fig. 5 *Left*), with higher levels specifying more

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

detailed requirements. Multiple trajectories can correspond to the same text (e.g., all trajectories being assigned the same level 1 text), resulting in many-to-many text-motion correspondences. The MMD metrics are measured separately for each level text input, then averaged across text descriptions at the same level. Accuracy and MMD metrics are measured using the seen texts. Then, to evaluate the robustness to text variations, we also report robust MMD metrics, which are computed with unseen text inputs generated by ChatGPT.

We begin this section by comparing latent diffusion and flow matching models on text-based 2D motion generation tasks. We then present experiments involving higher-dimensional configuration spaces, including text-based SE(3) pouring motion generation and 7-DoF multi-task robot arm motion generation.

A. Latent diffusion vs latent flow matching

In this section, using the simple 2D trajectory generation task shown in Fig. 3, we compare our motion manifold flow primitives trained with latent flow matching models to those trained with latent diffusion models [21], denoted as MMP + Diffusion, while using the same motion manifold (an autoencoder). The dataset consists of 20 demonstration trajectories, all starting from the same point and ending at a common goal (the origin). Each trajectory has a length of $T = 201$. For each trajectory, two text annotations are provided (see Fig. 3). Path and task accuracy refer to whether the generated trajectories select the correct passage direction and successfully reach the goal point without collision, respectively.

Diffusion models involve several design choices [35]: the diffusion paths and noise schedules. In this experiment, latent diffusion models are trained using various Gaussian probability paths, including the variance-exploding path, denoted as Diffusion-VE, and two variance-preserving paths with different noise schedules, denoted as Diffusion-VP-1 and Diffusion-VP-2 (linear noise scheduling with higher and lower coefficients, respectively), which are standard choices because of their well-understood analytic properties.

As shown in Table I and Fig. 4, MMFP achieves strong performance across both levels of text inputs simultaneously. In contrast, MMP combined with diffusion models yields less satisfactory results, performing well on either the level 1 or level 2 task but failing to achieve strong performance on both simultaneously. These results can be attributed to the smoother optimal transport paths [7] used in flow matching compared to diffusion paths. These smoother paths result in a smoother ground-truth vector field that the neural network $v_s(z, c)$ needs to fit, making the fitting problem easier – which can have a significant impact with sparse data – improving accuracy and generalization, and reducing the number of function evaluations required during sampling.

B. SE(3) pouring motion generation

In this section, we train text-based pouring motion generation models, where the dataset is obtained from the human demonstration videos. The demonstrator is instructed to pour water or wine in five different pouring directions (i.e., from the very left, left, center, right, and very right side). When pouring wine, the demonstrator is instructed to turn the wrist clockwise at the end. From ten videos, we extract SE(3) trajectories of the bottles, and the trajectory lengths are pre-processed so that $T = 480$. For each trajectory, we give three text annotations, as shown in Fig. 5 (*Left*). Notably, multiple trajectories can be associated with a single text label, leading to many-to-many correspondences between text and motion. For example, all 10 trajectories are labeled with the level 1 text, "Give me a drink, anything please".

Table II shows MMD, robust MMD, and Accuracy; our MMFP only shows good scores in all metrics. We note that the regularization in MMFP significantly improves the level 3 robust MMD metric. The DDPM and RFM overall produce very poor MMD results. As shown in Fig. 5 (*Right*), the trajectory generated by DDPM does not even converge near the cup and that of RFM is very jerky. This occurs because the dimensionality of the trajectory is too high relative to the limited amount of available data, indicating that appropriate dimensionality reduction is essential. TCVAE and MMP, both adopting the motion manifold hypothesis, produce motions of reasonable quality. Nevertheless, they exhibit a limitation in understanding level 3 text descriptions, particularly regarding pouring directions. Examples illustrating this limitation can

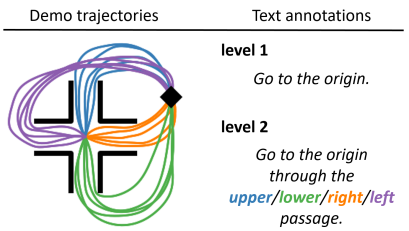

Fig. 3. Demonstration trajectories with multiple text annotations. Each trajectory is assigned with two text labels, the level 1 and level 2 texts.

TABLE I
THE MMD METRIC AND ACCURACY (%) FOR 2D MOTION GENERATION.

	MMD (↓)		Accuracy (↑)	
	level 1	level 2	path	task
MMP + Diffusion-VE	0.075	**0.007**	100	94.6
MMP + Diffusion-VP-1	0.073	**0.003**	100	95.0
MMP + Diffusion-VP-2	**0.030**	0.055	94.3	81.0
MMFP	**0.025**	**0.004**	**100**	**99.8**

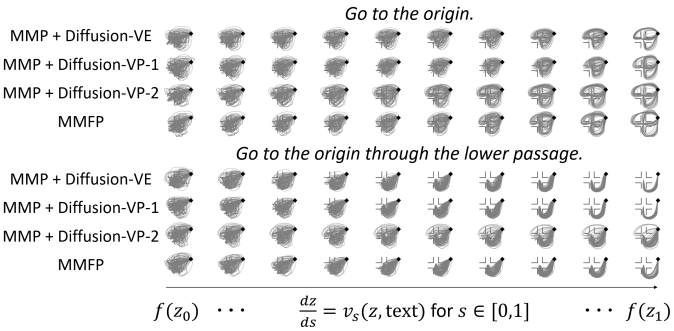

Fig. 4. The evolution of generated trajectories (from left to right) follows each of the latent models, either diffusion or flow, with the trajectories on the far right representing the final output samples.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

TABLE II
THE MMD AND ROBUST MMD METRICS AND THE ACCURACY (%) FOR POURING MOTION GENERATION.

	MMD (↓)			robust MMD (↓)			Accuracy (↑)		
	level 1	level 2	level 3	level 1	level 2	level 3	pouring style	pouring direction	both
DDPM [13]	0.398	0.484	1.306	0.400	0.486	1.304	50.5	20.0	10.2
RFM [8]	0.411	0.431	0.778	0.413	0.425	1.127	87.0	36.4	30.8
TCVAE [4]	0.117	0.211	0.824	0.131	0.191	1.041	52.0	25.4	15.3
MMP [5]	**0.045**	**0.115**	0.950	**0.055**	**0.112**	0.970	47.5	19.6	9.3
MMFP w/o reg (ours)	**0.055**	**0.114**	**0.009**	**0.056**	**0.097**	0.096	**98.5**	**93.2**	**99.9**
MMFP (ours)	**0.042**	**0.093**	**0.007**	**0.052**	**0.094**	**0.016**	**99.0**	**92.6**	**99.9**

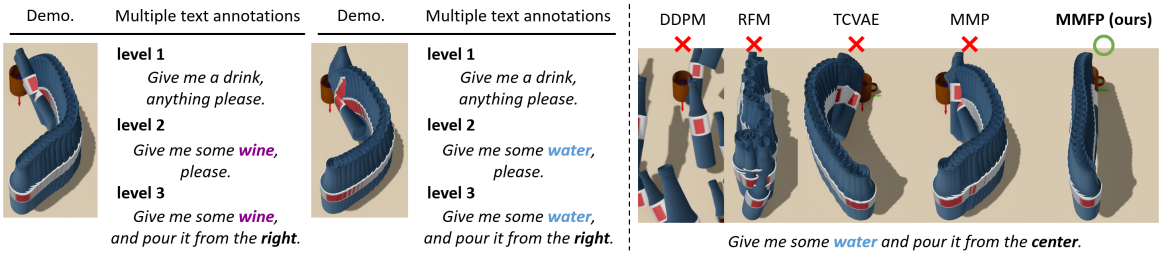

Fig. 5. *Left*: Example demonstration trajectories, each of which is annotated with three different level texts. *Right*: DDPM and RFM produce jerky motions, TCVAE and MMP yield trajectories in incorrect directions, and only MMFP generates a successful trajectory

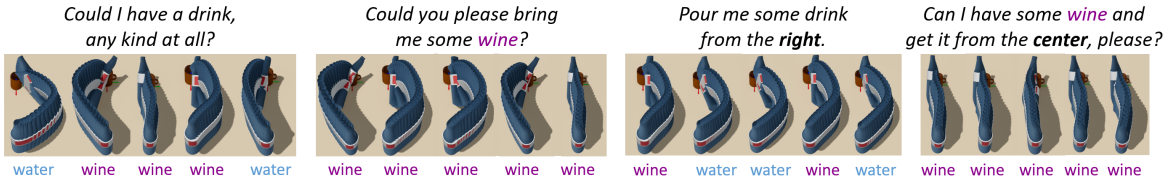

Fig. 6. A variety of accurate trajectories generated by MMFP given unbiased text inputs.

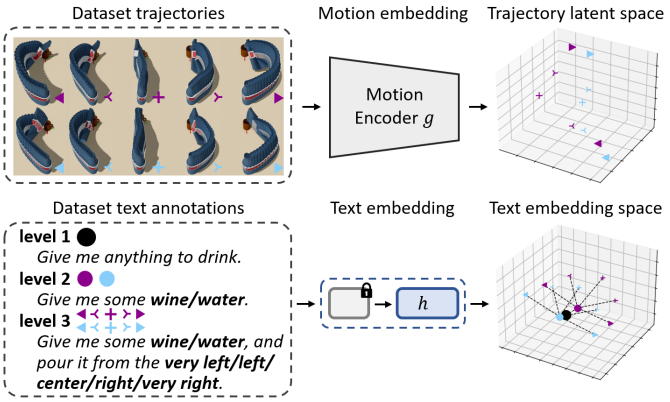

Fig. 7. *Upper*: SE(3) pouring trajectory data encoded in the three-dimensional latent space by our motion encoder. *Lower*: Texts used in training encoded in the three-dimensional text embedding space by our text embedding model.

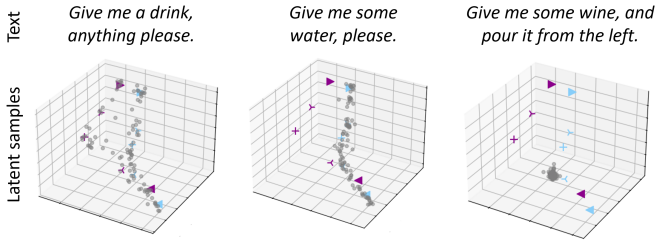

Fig. 8. Latent points sampled by our trained latent flow model in MMFP, conditioned on level 1, 2, and 3 texts, reveal dramatic shifts in the motion distribution arising from minor changes in the input text.

Fig. 9. Pouring motions performed by a 7-DoF Franka Panda robot, where inverse kinematics (IK) are solved to obtain joint trajectories from the generated SE(3) pouring trajectories.

be observed in Fig. 5 (*Right*), where they fail in pouring from accurate directions. Lastly, Fig. 6 shows a variety of accurately generated motions by MMFP given unbiased user text inputs not seen during training.

Furthermore, we present a visual analysis of our MMFP, wherein the three-dimensional nature of both the latent space $\mathcal{Z}$ and the text embedding space $\mathcal{T}$ facilitates straightforward visualization. Fig. 7 (*Upper*) shows the latent values of the trajectory data, i.e., $z^i = g(x^i) \in \mathbb{R}^3$, where g is a trained encoder. Fig. 7 (*Lower*) displays the embedding vectors of the text inputs, i.e., $\tau^i = h(c^i) \in \mathbb{R}^3$, where h is a trained text embedding module. Semantically similar trajectories and texts are positioned closely together, exhibiting smooth transitions as their meanings shift, indicating successful model training.

IEEE Robotics and Automation Letters (RA-L) paper, presented at ICRA 2026, Vienna, Austria. Cite as RA-L paper.

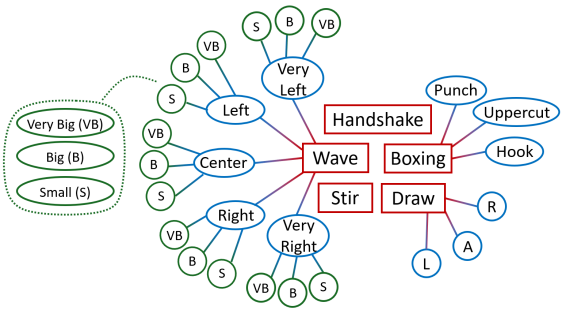

Fig. 10. Multi-task motion dataset with five motion types: waving, handshake, stirring, drawing letters, and boxing. Waving has three hierarchies (direction, gesture size); drawing and boxing have two (letter type; movement style). Handshake and stirring have no hierarchy.

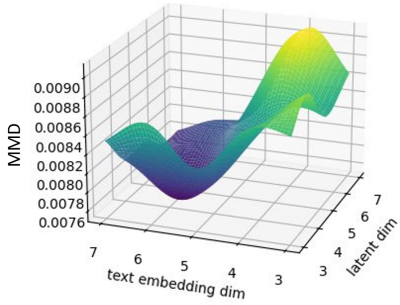

Fig. 11. Mean MMD metrics as functions of latent and text embedding dimensions (lower is better). Best performance is achieved at intermediate dimensions – not too small or too large.

Fig. 8 highlights the multi-modality of the text-conditioned latent distribution and, more importantly, the complex text-motion dependencies, as evidenced by the substantial shifts in the distribution with varying text inputs. In the figure, moving from left to right corresponds to scenarios where level 1 to level 3 texts are given as inputs. When a level 1 text is provided, the distribution must fit two modalities, and the volume of the distribution is the largest. In contrast, when level 2 or level 3 texts are provided, there is only one modality. Note that the support of the distribution varies significantly from level 1 to level 3, becoming much smaller at level 3. This implies that a latent text-conditioned distribution should capture such dramatic changes, fitting a complex, non-smooth function. Existing manifold-based models relying on conditional autoencoders struggle to capture such a complex function, whereas our MMFP effectively models it.

Lastly, we conduct real robot experiments (see Fig. 9). Given the generated SE(3) trajectories of the bottle, we compute joint space trajectories by solving the inverse kinematics. The generated SE(3) trajectory is smooth and densely discretized enough that, despite the relatively high speed of the bottle, the robot can accurately track it.

C. Multi-task 7-DoF robot arm motion generation

In this section, we demonstrate that our model is applicable to multi-task scenarios involving even more complex task-motion dependencies. Fig. 10 illustrates our dataset, which consists of five motion types, each represented by a trajectory of size 720×7. Some motions – such as waving, boxing, and drawing – have multiple, hierarchical text labels, while others – handshake and stirring – have a single label. The dataset contains a total of 54 trajectories (2 for each waving motion and 3 for each of the others). In addition to complex task-motion relationships, there is also overlap between trajectories of different tasks – for example, both handshake and punch motions involve extending the arm forward at the beginning. This overlap makes it more challenging to learn accurate, conditional motion distributions.

In the previous section, we showed that a latent and text embedding dimension of 3 was sufficient for the 6-DoF pouring example. However, with a larger and more complex dataset, it is reasonable to expect that higher-dimensional latent and text embedding space dimensions are needed. Fig. 11 shows the mean MMD values – averaged over both robust and vanilla MMD metrics at each level – for different dimension choices. The best performance is achieved with a latent dimension of 4 and a text embedding dimension of 6, while further increases lead to a decline in performance. This suggests that a moderate increase in dimensionality is sufficient to capture the complexity of larger datasets, whereas overly high dimensions may complicate the problem.

Fig. 12 shows diverse motions generated by MMFP, all of which align well with their corresponding text inputs. The generated motions span long-horizon trajectories; for instance, in waving or handshake movements, the hand repeatedly moves back and forth. When given less specific texts, the model generates diverse motions as expected, while more detailed instructions yield less diverse, unimodal, and consistent outputs. These results demonstrate that MMFP effectively handles long-horizon, high-dimensional trajectory data and captures complex task-motion dependencies, even in challenging multi-task settings.

VI. Conclusion

The Motion Manifold Flow Primitive (MMFP) framework combines motion manifolds and latent flows, learning task-conditioned motion distributions with complex task-motion dependencies, all while relying on a small number of trajectory data. The performance has been validated through extensive experiments involving the language-guided generation of SE(3) and 7-DoF joint trajectories. We believe the MMFP framework can be further improved in several ways. The current implementation relies solely on text inputs, and extending it to incorporate visual inputs and support closed-loop feedback control could make the primitives significantly more versatile. Additionally, the framework currently requires fixed-length discrete-time trajectories, which could be addressed by adopting sequence models, such as transformers or recurrent neural networks, or continuous-time representations with parametric curve models, as proposed in [6].

References

[1] D.-H. Park, H. Hoffmann, P. Pastor, and S. Schaal, "Movement reproduction and obstacle avoidance with dynamic movement primitives and potential fields," in *Humanoids 2008-8th IEEE-RAS International Conference on Humanoid Robots*. IEEE, 2008, pp. 91–98.

[2] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, "Probabilistic movement primitives," *Advances in neural information processing systems*, vol. 26, 2013.

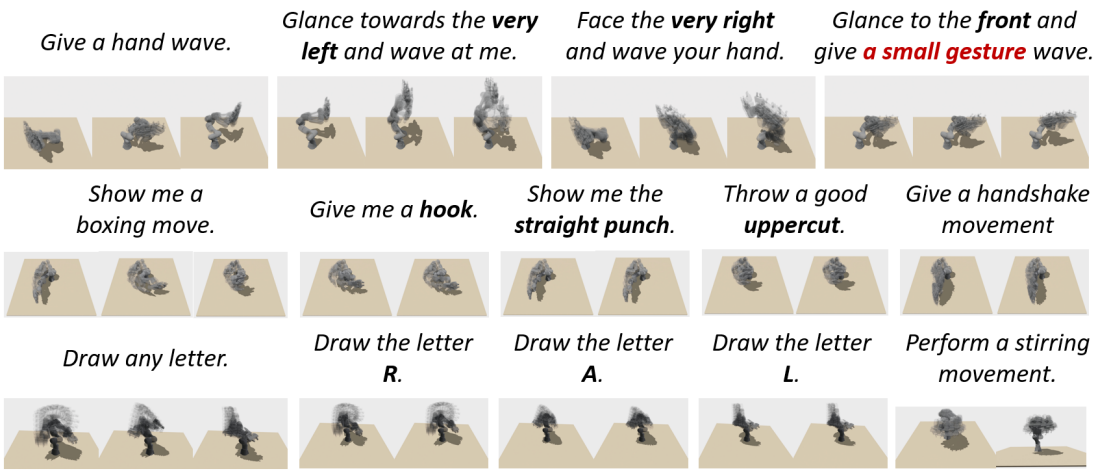

Fig. 12. Correct 7-DoF robot arm motions generated by MMFP from text inputs. Diverse motions are produced from less specific commands (e.g., "Give a hand wave", "Show me a boxing move", "Draw any letter"), while more detailed texts yield accurate, specified motions. MMFP effectively captures complex task-motion dependencies, even in multi-task scenarios. Example videos: Waving, Boxing and Handshake, and Drawing and Stirring.

[3] S. M. Khansari-Zadeh and A. Billard, "Learning stable nonlinear dynamical systems with gaussian mixture models," *IEEE Transactions on Robotics*, vol. 27, no. 5, pp. 943–957, 2011.

[4] M. Noseworthy, R. Paul, S. Roy, D. Park, and N. Roy, "Task-conditioned variational autoencoders for learning movement primitives," in *Conference on robot learning*. PMLR, 2020, pp. 933–944.

[5] B. Lee, Y. Lee, S. Kim, M. Son, and F. C. Park, "Equivariant motion manifold primitives," in *Conference on Robot Learning*. PMLR, 2023, pp. 1199–1221.

[6] Y. Lee, "Mmp++: Motion manifold primitives with parametric curve models," *IEEE Transactions on Robotics*, 2024.

[7] Y. Lipman, R. T. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," *arXiv preprint arXiv:2210.02747*, 2022.

[8] R. T. Chen and Y. Lipman, "Riemannian flow matching on general geometries," *arXiv preprint arXiv:2302.03660*, 2023.

[9] 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.

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

[11] M. Braun, N. Jaquier, L. Rozo, and T. Asfour, "Riemannian flow matching policy for robot motion learning," *arXiv preprint arXiv:2403.10672*, 2024.

[12] E. Chisari, N. Heppert, M. Argus, T. Welschehold, T. Brox, and A. Valada, "Learning robotic manipulation policies from point clouds with conditional flow matching," *arXiv preprint arXiv:2409.07343*, 2024.

[13] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[14] M. Saveriano, F. J. Abu-Dakka, A. Kramberger, and L. Peternel, "Dynamic movement primitives in robotics: A tutorial survey," *arXiv preprint arXiv:2102.03861*, 2021.

[15] A. J. Ijspeert, J. Nakanishi, H. Hoffmann, P. Pastor, and S. Schaal, "Dynamical movement primitives: learning attractor models for motor behaviors," *Neural computation*, vol. 25, no. 2, pp. 328–373, 2013.

[16] K. Neumann, A. Lemme, and J. J. Steil, "Neural learning of stable dynamical systems based on data-driven lyapunov candidates," in *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2013, pp. 1216–1222.

[17] S. M. Khansari-Zadeh and A. Billard, "Learning control lyapunov function to ensure stability of dynamical system-based robot reaching motions," *Robotics and Autonomous Systems*, vol. 62, no. 6, pp. 752–765, 2014.

[18] S. Calinon, "A tutorial on task-parameterized movement learning and retrieval," *Intelligent service robotics*, vol. 9, pp. 1–29, 2016.

[19] T. Daab, N. Jaquier, C. Dreher, A. Meixner, F. Krebs, and T. Asfour, "Incremental learning of full-pose via-point movement primitives on riemannian manifolds," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 2317–2323.

[20] H. Beik-Mohammadi, S. Hauberg, G. Arvanitidis, G. Neumann, and L. Rozo, "Learning riemannian manifolds for geodesic motion skills," in *Robotics: Science and Systems*, 2021.

[21] Y. Song, J. Sohl-Dickstein, D. P. Kingma, A. Kumar, S. Ermon, and B. Poole, "Score-based generative modeling through stochastic differential equations," *arXiv preprint arXiv:2011.13456*, 2020.

[22] A. Tong, N. Malkin, G. Huguet, Y. Zhang, J. Rector-Brooks, K. Fatras, G. Wolf, and Y. Bengio, "Conditional flow matching: Simulation-free dynamic optimal transport," *arXiv preprint arXiv:2302.00482*, 2023.

[23] G. Arvanitidis, L. K. Hansen, and S. Hauberg, "Latent space oddity: on the curvature of deep generative models," *arXiv preprint arXiv:1710.11379*, 2017.

[24] Y. Lee, H. Kwon, and F. Park, "Neighborhood reconstructing autoencoders," *Advances in Neural Information Processing Systems*, vol. 34, pp. 536–546, 2021.

[25] Y. Lee, S. Kim, J. Choi, and F. Park, "A statistical manifold framework for point cloud data," in *International Conference on Machine Learning*. PMLR, 2022, pp. 12 378–12 402.

[26] Y. Lee, S. Yoon, M. Son, and F. C. Park, "Regularized autoencoders for isometric representation learning," in *International Conference on Learning Representations*, 2022.

[27] C. Jang, Y. Lee, Y.-K. Noh, and F. C. Park, "Geometrically regularized autoencoders for non-euclidean data," in *The Eleventh International Conference on Learning Representations*.

[28] Y. Lee, "A geometric perspective on autoencoders," *arXiv preprint arXiv:2309.08247*, 2023.

[29] Y. Lee and F. C. Park, "On explicit curvature regularization in deep generative models," in *Topological, Algebraic and Geometric Learning Workshops 2023*. PMLR, 2023, pp. 505–518.

[30] J. Lim, J. Kim, Y. Lee, C. Jang, and F. C. Park, "Graph geometry-preserving autoencoders," in *Forty-first International Conference on Machine Learning*, 2024.

[31] Y. Lipman, M. Havasi, P. Holderrieth, N. Shaul, M. Le, B. Karrer, R. T. Chen, D. Lopez-Paz, H. Ben-Hamu, and I. Gat, "Flow matching guide and code," *arXiv preprint arXiv:2412.06264*, 2024.

[32] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019. [Online]. Available: https://arxiv.org/abs/1908.10084

[33] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[34] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. Smola, "A kernel two-sample test," *The Journal of Machine Learning Research*, vol. 13, no. 1, pp. 723–773, 2012.

[35] J. Song, C. Meng, and S. Ermon, "Denoising diffusion implicit models," *arXiv preprint arXiv:2010.02502*, 2020.