Trajectory Planning without Trajectory Data: A Manifold-Guided Approach

Silong Yong[1] **Anji Liu**[2] **Cunxi Dai**[1] **Carl Busart**[3]
Guanya Shi[1] **Yilun Du**[4] **Katia Sycara**[1] **Yaqi Xie**[1]

[1]Carnegie Mellon University [2]National University of Singapore
[3]DEVCOM Army Research Laboratory [4]Harvard University

Abstract

A common way for trajectory planning is to leverage generative models trained on large collections of expert trajectories. At inference time, the model generates executable trajectories by conditioning on task goal constraints. However, trajectory-based methods rely on costly supervision, scale poorly with sequence length, and often generalize poorly to unseen constraints such as novel start-goal pairs. We propose an alternative to learn the underlying state-space manifold and use the geometry of the manifold for trajectory planning. This approach requires only state observations and enables generalization to unseen constraints by constructing trajectories on the learned manifold of the state space. Experiments on Maze2D and robotic motion-planning benchmarks show that **Ariadne** constructs feasible paths from state-only supervision and generalizes to unseen start-goal combinations. On high-dimensional dual-arm planning, it remains competitive with trajectory-supervised and classical planners, while requiring no trajectory data for training. Code can be found in `https://github.com/SilongYong/Ariadne/`.

1 Introduction

Trajectory planning aims to find a time-parameterized sequence of states that satisfies environment dynamics, task objectives, and constraints. Classical approaches typically rely on search or optimization over explicit dynamics models [38, 32, 22, 23], which can become computationally expensive in high-dimensional spaces.

Recent learning-based methods instead leverage generative models, such as diffusion models [18], to learn trajectory distributions from offline data, enabling flexible plan synthesis via conditional sampling [20, 9, 5, 28]. By bridging sampling and planning, these methods model distributions over trajectory sequences. However, the trajectory space is exponentially large and highly structured, making it impractical to achieve sufficient coverage with finite data. As the sequence length increases, the number of feasible trajectories grows combinatorially, requiring prohibitively large datasets to learn a reliable model (Fig. 1, top-left). Empirically, this leads to poor generalization when conditioning on unseen start–goal pairs: the generated trajectories often fail to reach the goal or violate environment constraints, even when the constituent states have been observed during training (Fig. 1, top-right).

We aim to improve the generalizability of generative-model-based trajectory planning while reducing the need for extensive trajectory supervision. Instead of learning from expert trajectories, we consider a setting where only state observations are available, i.e., a generative model is trained to capture the distribution of valid states (Fig. 1, bottom row). Under the manifold hypothesis [39, 17, 16, 11], these states are assumed to lie on a low-dimensional manifold embedded in the ambient space, i.e., a space that can be curved or complex overall, but looks locally like a flat Euclidean space. While learning such state distributions is well understood, a key challenge remains: how can we leverage this learned representation to construct feasible trajectories? We address this by interpreting trajectory planning

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

as searching for valid paths on the learned state manifold, and propose a method that constructs trajectories by explicitly exploiting its geometry.

We view trajectory construction as following a vector field defined over the state space, where a trajectory corresponds to an integral curve of this field. Intuitively, at each state, the vector field specifies a local direction of motion, and integrating these directions produces a path connecting the start and goal states. This vector field is constrained to lie on the underlying state manifold, ensuring that the resulting trajectory remains in feasible regions of the state space. This perspective raises three key challenges: identifying the local geometry of the manifold, defining a vector field that respects this geometry, and ensuring meaningful notions of distance for projection.

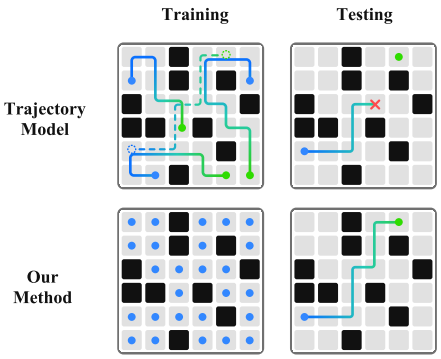

Figure 1: Trajectory-based model can't generalize towards unseen combination of start and goal state even if both states are presented in the training data. In contrast, our method is able to generalize under such condition without seeing any trajectory data.

Our key insight is that generative models provide rich geometric information about the data manifold. In particular, the score function captures local first-order structure, while higher-order information reveals the full normal space. Leveraging this, we construct a geometry-aware vector field that remains on the manifold and connects given endpoints. To enable stable projection, we further derive a Riemannian metric from the score function [4, 13].

We first verify the idea in the classical planning domain where a 2D maze is considered. We demonstrate that with only state information provided during training a generative model, our method is able to extract the geometry and perform sequence modeling by providing feasible planned path towards the goal. Our method therefore overcomes a key limitation of learning-based sequence modeling, where models only perform well on start-goal pairs whose trajectories were seen during training. We further demonstrate that our method can also be adopted to higher-dimensional space where a 6 DoF robot arm is tasked to avoid obstacles in a table top settings. Our method is also extended to two 7 DoF robot arms control problem where the two arms need not only to avoid obstacles but also avoid each other and achieves comparable results to the State-of-the-Art trajectory-learning-based planning method. Overall, our contributions are as follows:

- We extend the manifold hypothesis on score-accessible generative models to not only confirm the existence but also extract general geometry without strong assumptions from it.
- We propose a novel trajectory planning framework which requires only state-space training data by utilizing the geometry and introducing an endpoint constraint by connecting the non-linear interpolation formulation with iterative projection method.
- We evaluate **Ariadne** against classical search, trajectory-supervised diffusion planners, and geometry-based trajectory optimization, and show competitive planning performance from state-only supervision across Maze2D and robotic motion-planning tasks.

2 Background and Motivation

Learning-based trajectory planning is typically formulated as learning from trajectory datasets. Let a *state* be denoted by $x \in \mathcal{X}$, where $\mathcal{X}$ is the state space. A *trajectory* is a time-indexed sequence $\tau = (x_0, x_1, \ldots, x_T)$ describing a feasible transition between a start state x_0 and a goal state x_T. A *task* specifies trajectory-level conditions, such as initial states, goal states, or intermediate constraints. A dataset $\mathcal{D}_{\text{traj}} = \{\tau_i\}_{i=1}^{N}$ consists of sampled trajectories collected from demonstrations or rollouts. During training, a model learns to approximate the distribution over trajectories $p(\tau)$ from $\mathcal{D}_{\text{traj}}$. At inference time, the model is conditioned on task specifications and generates a trajectory that is evaluated by task completion and feasibility. The most prominent requirements [20] for task specifications are that the trajectory terminates at a specified goal state. This start-goal formulation is fundamental in robotics and motion planning and our work specifically focus on this setting.

As illustrated in Fig. 1 (top-left), trajectory datasets inevitably provide incomplete coverage of the space of feasible trajectories. Certain valid trajectories, although physically realizable, may be absent from the dataset (shown as dashed curves), due to limited data collection or biased sampling.

This incomplete coverage leads to a well-recognized generalization issue in generative planning models [20, 6, 26, 7]. Because the model learns a trajectory-level distribution, it becomes biased toward motion patterns observed during training, i.e., biased patterns in the training set. At inference time, when tasked with generating trajectories that were not explicitly present in $\mathcal{D}_{\text{traj}}$, the model often fails to produce valid solutions (Fig. 1, top-right). This failure arises from trajectory-level distribution mismatch: the required trajectory lies outside the support of the learned trajectory distribution, even though it may be perfectly feasible in the underlying environment.

We propose a new paradigm to address this limitation. We argue that many trajectory-level out-of-distribution (OOD) failures originate from bias in trajectory data rather than from insufficient state coverage. While trajectory samples may be sparse or incomplete, the underlying state distribution often exhibits significantly broader coverage (Fig. 1, bottom-left). That is, even if certain trajectories are missing, their constituent states may still lie within the observed state space. This observation motivates a fundamental question: **can we learn planning from state data alone, without relying on trajectory supervision?**

Eliminating trajectory supervision immediately raises a key challenge: without access to explicit path information, how can trajectories be reconstructed at inference time? We observe that, for most planning problems, valid trajectories correspond to smooth curves in the state space that remain within dynamically feasible, low-energy regions. If a model can learn the geometric and energetic structure of the state space from state-only data, then trajectory construction can be framed as a curve generation problem under task constraints (e.g., specified endpoints). Moreover, collecting state data is often substantially easier and less restrictive than collecting full trajectory demonstrations, making this paradigm more scalable in practice.

Formally, given a learned state-space model and task-level conditions such as start and goal states, our objective becomes: how can we construct a feasible curve at inference time that obeys these conditions while remaining in valid regions of the state space? We present **Ariadne**[1] below, which exploits the geometry of the learned data manifold and formulates trajectory construction as integrating an ordinary differential equation (ODE) that evolves along it.

3 Trajectory Construction on the Learned Manifold

Before proceeding, we briefly clarify the notion of the data manifold. Let $\mathcal{X} \subset \mathbb{R}^d$ denote the ambient state space. Given state-only training data sampled from valid system behaviors, we assume that feasible states concentrate around a lower-dimensional subset $\mathcal{M} \subset \mathcal{X}$, which we refer to as the *data manifold*. Intuitively, $\mathcal{M}$ represents the set of dynamically feasible and physically valid states of the system.

Rather than assuming explicit access to $\mathcal{M}$, we rely on a learned generative model to implicitly characterize its local geometry. In particular, the local geometry, i.e. normal and tangent spaces at a point, are defined with respect to this learned geometric structure.

We first consider the problem of constructing a trajectory that remains on the data manifold $\mathcal{M}$, ignoring endpoint constraints for now. A natural approach is to iteratively follow a reference direction while projecting the update back onto the local tangent space of the manifold.

Let $x_0 \in \mathcal{M}$ be the initial state and x_1 be the desired endpoint. We define a constant direction vector $x_1 - x_0$. If no manifold constraint were imposed, integrating this direction would produce the straight-line path

$$x(t) = x_0 + t(x_1 - x_0),$$

which exactly reaches the endpoint.

However, this straight-line path will in general leave the manifold. To enforce the manifold constraint, we adopt a local move-then-project strategy. At each step, we first move along the constant direction:

$$x'_t = x_{t-1} + (x_1 - x_0)\Delta t. \tag{1}$$

[1] Ariadne refers to Ariadne's thread in Greek mythology, which guides Theseus through the labyrinth. We use it to reflect our goal of constructing a path through complex state spaces.

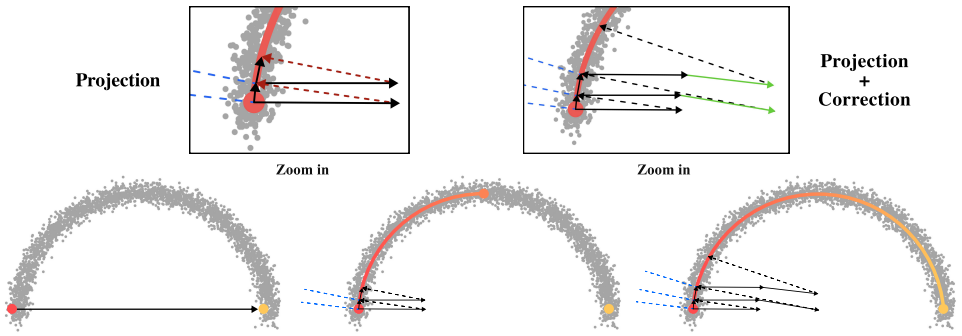

Figure 2: Method overview. We take Uniform Circular Gaussian (UCG) as an example. We assume we have a diffusion model trained on such data and aim to use it for constructing a sequence of states that goes from the red dot to the yellow dot. **Left:** straight line interpolation between the two points doesn't consider the information of the manifold, but enforce the endpoint constraint that the line would start from the red dot and end at the yellow dot. **Middle:** We can iteratively move along the straight line and use the geometry stored in the diffusion model to project the movement back towards the manifold. However, the projection would shrink the actual moving distance towards the goal, therefore resulting in satisfying manifold constraint but overlooking the endpoint constraint. **Right:** In order to satisfy both the manifold constraint and the endpoint constraint, we add a correction term derived from an ODE perspective and successfully construct a path represented by the ODE that satisfies both the constraints.

The intermediate point x_t' may lie off the manifold. We then remove the normal component w.r.t the manifold of the displacement to project the motion back onto the tangent space at x_{t-1}.

Let $\mathbf{V}$ denote an orthonormal basis spanning the normal space at x_{t-1}. The projected update is given by

$$x_t = x_t' + \sum_{n \in \mathbf{V}} \big[-\langle x_t' - x_{t-1}, n \rangle n \big], \tag{2}$$

which subtracts the normal components of the displacement. Equivalently, this corresponds to projecting the update onto the tangent space $T_{x_{t-1}}\mathcal{M}$.

By repeatedly applying Eq. (1) and Eq. (2), the trajectory remains locally tangent to the manifold at every step. Intuitively, each iteration removes the component that would push the state away from $\mathcal{M}$. As illustrated in Fig. 2 (middle), this generates a smooth curve that stays on the manifold while being guided by the global direction from x_0 to x_1. While this construction encourages the trajectory to remain on the manifold, it does not strictly guarantee it. In particular, if the reference direction aligns with the normal space, the update may push the state away from the manifold, and the projection step may not fully compensate for this effect. As a result, deviations from the manifold can accumulate over time. Nevertheless, we find that this formulation works well in practice in most cases.

More importantly, even when the trajectory approximately stays on the manifold, this construction does not guarantee satisfaction of task-level endpoint constraints. The projection step can alter the global direction of motion, causing the trajectory to deviate from the intended target and fail to reach the desired endpoint. This reveals a key limitation: local geometric corrections alone are insufficient to control the global behavior of the trajectory. In the next section, we address this issue by explicitly incorporating endpoint constraints into the trajectory construction.

4 Incorporating Endpoint Constraints

The original move-then-project procedure uses endpoint information through the direction vector pointing toward the target. However, once this direction is projected onto the tangent space, its

magnitude and orientation are altered. In particular, the projection removes the normal component of motion, but this removal is not controlled with respect to the remaining progress toward the goal. As a result, although each infinitesimal update locally moves the state "toward" the endpoint in a geometric sense, the accumulated effect may significantly distort the global trajectory. In extreme cases, the projected dynamics may stall or converge to a point that satisfies the manifold constraint but fails to reach the specified endpoint.

Roadmap. In its discrete iterative form, the move-then-project procedure enforces the manifold constraint but does not guarantee satisfaction of the endpoint condition. To address this issue, we first reinterpret the move-then-project algorithm as the discretization of a specific ordinary differential equation (ODE). In particular, we show that the iterative projection update corresponds to an operator-splitting scheme for a certain projected flow.

Building on this formulation, we then analytically modify the induced ODE so that its integral curves both (i) remain on the manifold and (ii) satisfy the prescribed start and endpoint constraints. In this way, endpoint satisfaction is no longer an emergent property of an iterative projection algorithm, but a structural property of the designed dynamics.

We'll briefly introduce the operator-splitting methods and then discuss the specific ODE to which the move-and-project procedure corresponds.

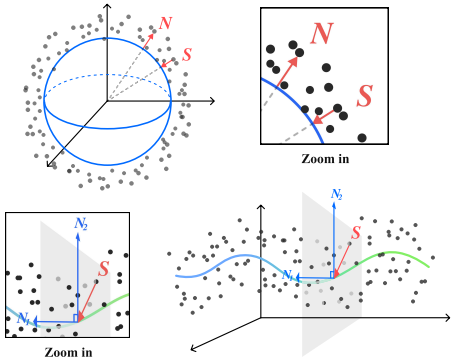

Figure 3: While the score function provided by the diffusion model points toward the data manifold and can be viewed as an approximation of the normal direction of a (D-1)-dim manifold embedded in D-dimensional space, it no longer holds when we're facing a more general and realistic case where the manifold might be even lower dimensional. We therefore propose to use the Jacobian of the score function for normal estimation as shown on the right.

4.1 Operator Splitting

Operator splitting provides a convenient way to decompose a dynamical system into multiple sub-dynamics that can be evolved sequentially. For an ODE

$$\frac{dx(t)}{dt} = A(x(t)) + B(x(t)), \tag{3}$$

a first-order splitting scheme alternates between the dynamics induced by A and B over each integration step [40, 36, 30].

In our formulation, operator splitting is used primarily as a conceptual tool to separate two naturally distinct operations in trajectory construction: reference motion and manifold projection. Specifically, the reference motion corresponds to A, while the projection onto the learned local manifold geometry corresponds to B. This decomposition allows us to reinterpret the iterative move-then-project procedure as the discretization of an ODE, and subsequently introduce the endpoint correction in a structured way. The explicit Euler realization and full derivation are provided in Appendix B.1.

4.2 Interpreting Iterative Projection as ODE

Following the previous discussion, we give the ODE view of the move-then-project procedure in Eq. (1) and Eq. (2).

$$\frac{dx}{dt} = x_1 - x_0 + 2t(1-t)\frac{d}{dt}\varphi(x_t), \tag{4}$$

where

$$\varphi(x_t) = \sum_i \sum_{n \in \mathbf{N_i}} \frac{(-nn^T)dx}{2t(1-t)\epsilon}\Delta t. \tag{5}$$

Here i denotes the discrete integration step when iterating the ODE to x_t, ϵ is a factor that controls the scale of projection (also cancels out the Δt that is not in the original projection form) and $\mathbf{V}_i$ denotes

an orthonormal basis of the normal subspace at the current state x_t. Intuitively, this formulation views move as A and projection as B in Eq. (8). Please find the full derivation in Appendix B.

This perspective reveals that endpoint satisfaction is not automatically guaranteed: the induced ODE enforces invariance to the manifold but does not, by itself, enforce endpoint conditions.

To properly incorporate endpoint constraints, we must analytically design a vector field whose integral curves both (i) remain on the manifold and (ii) satisfy the prescribed endpoint conditions. In other words, we need to define a velocity field that contains the RHS terms in Eq. (4), but also enforces the integral of the velocity field to be a curve anchored at the start and end points.

We therefore propose an ODE that obeys the two rules

$$\frac{dx}{dt} = x_1 - x_0 + (2-4t)\varphi(x_0, x_1, t) + 2t(1-t)\frac{d}{dt}\varphi(x_0, x_1, t). \tag{6}$$

This new formulation would still satisfy the manifold constraint since we can view $x_1 - x_0 + (2 - 4t)\varphi(x_0, x_1, t)$ as A and $2t(1-t)\frac{d}{dt}\varphi(x_0, x_1, t)$ as B in the operator-splitting method. As long as there exists a projection step, i.e. $2t(1-t)\frac{d}{dt}\varphi(x_0, x_1, t)$ or B, in our formulation, the constructed path represented by the ODE will always be on the manifold. In the continuous formulation, the projection component is designed to preserve local manifold adherence while the correction term enforces the endpoint constraint. In practice, numerical discretization and imperfect geometry estimation can introduce small residual manifold drift and endpoint error, which we quantify experimentally in Tab. 2.

Intuitively speaking, the added term $(2-4t)\varphi(x_0, x_1, t)$ gradually corrects/adds back the removed components from the projection when evolving the ODE. The full derivation can be found in the Appendix B. We name Eq. 6 a Correction-Projection ODE. As can be seen from Fig. 2, the resulting path constructed using this method indeed stays on the semicircle and it starts from the given start point and ends at the given endpoint.

4.3 Numerically evolving CP-ODE

Given the above analysis and derivation, we now give the overview of our proposed method.

We use Euler method with operator-splitting to solve the ODE Eq. 6, namely

$$\begin{aligned} x'_t &= x_{t-1} + (x_1 - x_0)\Delta t, \\ x''_t &= x'_t + (2-4t)\varphi(x'_t)\Delta t, \\ x_t &= x''_t + 2t(1-t)\frac{d}{dt}\varphi(x''_t)\Delta t. \end{aligned}$$

The pseudo-code can be found in Algo. 1. In practice, there are three hyperparameters to set to evolve the CP-ODE, namely the initial arc-length (or step size for evolving the ODE in state space), the projection scale ϵ and the normal selection threshold (how negative an eigenvalue can inform a normal direction). We provide ablation studies on these three factors in Sec. 5.2.

Manifold adherence and endpoint satisfaction. We quantitatively verify the two roles of the Correction-Projection dynamics on the UCG example in Fig. 2. Since the ground-truth semicircular manifold is known, we measure the average Euclidean distance of the generated path to the manifold, together with the final endpoint error. As shown in Table 2, on UCG, the straight-line reference satisfies the endpoint constraint by construction but deviates substantially from the ground-truth manifold. Projection-Only exhibits the opposite behavior: it remains closest to the manifold but fails to reach the target. **Ariadne** balances the two objectives, substantially reducing manifold deviation relative to the straight-line reference while reducing endpoint error by an order of magnitude relative to Projection-Only. This supports the complementary roles of the projection and correction terms in the proposed dynamics.

4.4 Manifold Hypothesis and Geometry from Generative Models

The only missing component in our formulation is the normal directions required for projection. We aim to extract this information from a generative model trained on state-space data. Let $p(x)$ denote a

Table 1: Maze2D planning results measured by the standard D4RL normalized score. Diffuser uses full trajectory supervision, while Diffuser* excludes trajectories reaching the evaluation target region. **Ariadne** is trained only on state observations.

METHOD	MEDIUM ↑	LARGE ↑
MPPI	10.20	5.10
CQL	5.00	12.50
IQL	34.90	58.60
RRT	45.95	10.70
FOLLOW-THE-ENERGY	16.29	16.41
DIFFUSER*	52.70	41.98
DIFFUSER	**120.52**	**120.37**
ARIADNE (OURS)	55.69	50.63

data distribution in $\mathbb{R}^D$, with samples concentrated near a k-dimensional manifold $\mathcal{M}$. We assume access to the score function $s(x) = \nabla_x \log p(x)$.

The score captures first-order geometry and typically points toward higher-density regions, providing a useful approximation of normal directions when the manifold has co-dimension one. However, for general manifolds with higher co-dimension, the normal space is multi-dimensional and cannot be characterized by the score alone.

To recover the full normal subspace, we instead consider the Hessian

$$H(x) := \nabla_x^2 \log p(x),$$

which encodes second-order geometric information. In particular, under the manifold hypothesis, directions with large negative curvature correspond to the normal space, while near-zero curvature directions correspond to the tangent space. We defer detailed analysis to Appendix D.1.

Riemannian metric from the score. In addition to identifying normal directions, projection requires a notion of distance. We derive a Riemannian metric from the score function of a pretrained generative model. Under the assumption that the score magnitude is small on the data manifold and increases away from it, we define a position-dependent metric

$$G(x) = \frac{\exp(\|s(x)\|)}{Z} I, \tag{7}$$

which remains close to identity on the manifold and grows off it, thereby penalizing deviations. Further details are provided in Appendix D.2.

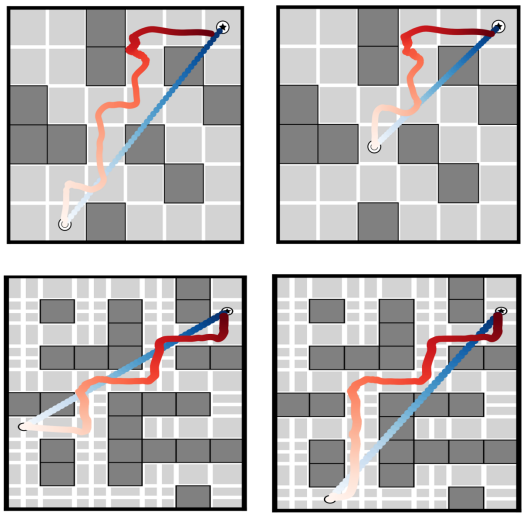

Figure 4: Visualization of the planned trajectory on maze. The blue straight line indicates the reference path and the red curve is the output of **Ariadne**. Light color corresponds to the starting point.

5 Experiments

Practical considerations. In practice, we use a straight-line reference path between endpoints, which we find sufficient for stable performance. We further adopt arc-length-based step sizes to ensure consistent progression along the trajectory, and adapt the implementation for different generative models (e.g., diffusion and flow matching) accordingly. Detailed discussions are provided in Appendix E.

Experimental Setup. We evaluate **Ariadne** on three planning benchmarks: Maze2D [20, 15], YAM 6D tabletop manipulation, and DualKuka [29]. In all settings, the model is trained using only

Table 2: **Left:** YAM 6D motion planning success rate (%). **Right:** manifold adherence and endpoint accuracy on UCG. M-Dist. denotes the average Euclidean distance from the generated path to the ground-truth manifold, and E-Err. denotes the Euclidean distance between reference endpoint and the resulted endpoint from different methods.

Method	SR (%) ↑
Straight Line	12.31
Ariadne	**60.77**

Method	M-Dist. ↓	E-Err. ↓
Straight Line	0.8800	0.0000
Projection-Only	0.0322	3.3652
Ariadne	0.0798	0.3998

state-space data without trajectory or transition supervision. At test time, we generate state sequences connecting start-goal pairs and evaluate performance using standard environment metrics, including task score and collision-free success rate. Detailed dataset construction and evaluation protocols are provided in Appendix F. We compare against classical, trajectory-supervised, and geometry-based planning baselines, including RRT, Diffuser [20], Follow-the-Energy [4], and PBDMP [29]. We provide detailed descriptions of their supervision and inference assumptions in Appendix L.

5.1 Experimental Results

Maze2D. Table 1 compares **Ariadne** against classical, trajectory-supervised, and geometry-based planners on Maze2D. **Ariadne** outperforms Diffuser* under the state-only generalization setting, where the constituent states are observed during training but trajectories reaching the evaluation targets are withheld. This supports our hypothesis that broad state-space coverage can enable planning for unseen start-goal combinations even when the corresponding trajectories are absent from the training data. When full expert trajectory supervision is provided, Diffuser achieves substantially higher D4RL normalized scores [20, 15], highlighting the advantage of directly observing solutions to the evaluation planning tasks.

Comparison with geometric path optimization. Follow-the-Energy [4] provides the closest geometric comparison to **Ariadne** because both methods derive planning information from a learned state-space model. The key distinction is how this geometry is converted into a trajectory. Follow-the-Energy optimizes an entire trajectory interpolant under the learned metric, whereas **Ariadne** constructs the trajectory online through local Correction-Projection updates.Follow-the-Energy degrades substantially on Maze2D and on two of the three DualKUKA settings. We observe that trajectory-level interpolation can produce overly smooth paths in environments with multiple local routes, causing the optimized path to leave narrow feasible regions. In contrast, **Ariadne** repeatedly applies local manifold-guided corrections during path construction. This comparison suggests an advantage of local Correction-Projection construction over trajectory-level geometric optimization in these settings.

YAM 6D. We further evaluate **Ariadne** on a higher-dimensional tabletop manipulation task. As shown in Table 2, **Ariadne** increases collision-free planning success from 12.31% for the straight-line reference to 60.77%, demonstrating that the state-space geometric construction extends beyond the 2D maze setting.

DualKUKA. We next evaluate **Ariadne** in the higher-dimensional DualKUKA environments, where two 7-DoF robot arms must avoid both workspace obstacles and inter-arm collisions. Table 3 compares against classical planning, trajectory-supervised diffusion planning, geometric interpolation, and PBDMP. **Ariadne** achieves the highest average success rate across the three environments while using only state observations during training.

Compared with Diffuser, **Ariadne** obtains substantially higher success rates on all three settings despite Diffuser receiving full trajectory supervision. One possible explanation is that the quality of trajectory supervision becomes important in this domain: the expert trajectories are produced by a classical planner and need not provide short or optimal solutions, whereas **Ariadne** does not imitate their trajectory distribution and instead constructs paths from the distribution of feasible states. **Ariadne** is also competitive with PBDMP, although PBDMP additionally uses scene information to support generalization across environments while our setting learns one planner per scene.

Table 3: Success rate (%) on DualKUKA across three environments. PBDMP takes the scene layout as input, whereas **Ariadne** learns one planner per scene from state-only observations.

METHOD	$n = 7$ $s = 0.14$	$n = 5$ $s = 0.14$	$n = 5$ $s = 0.22$	AVG.
STRAIGHT LINE	23.0	14.5	26.5	21.3
RRT	47.0	36.5	50.5	44.7
DIFFUSER	20.5	29.0	45.5	31.7
FOLLOW-THE-ENERGY	30.0	44.0	53.0	42.3
PBDMP*	55.0	**50.5**	**54.0**	53.2
ARIADNE (OURS)	**57.5**	**50.5**	52.0	**53.3**

RRT remains a strong classical baseline, but its success rate degrades as the search problem becomes more difficult and it requires a large number of online collision checks. A detailed efficiency comparison is provided in Appendix K.

We additionally evaluate sensitivity to score-model retraining across five independently trained models and observe consistent performance; full results are provided in Appendix M.

5.2 Ablation Studies

We ablate key hyperparameters of **Ariadne**, including projection scale, step size, and normal selection threshold as shown in Algo. 1, on the DualKUKA scenario.

We ablate the normal-selection threshold, integration step size, and projection scale in Table 4. We make the following observations for our method under different hyperparameter settings. An overly loose normal threshold introduces weak or noisy directions into the estimated normal space, whereas an overly restrictive threshold produces an incomplete estimate. Similarly, large integration steps can move the state outside the region where the local manifold approximation is accurate, while an excessively strong projection can dominate the reference motion and hinder progress toward the goal. Overall, **Ariadne** does not rely on a narrowly tuned hyperparameter configuration. A comprehensive analysis can be found in Appendix I.

6 Related Work

For a more detailed review of the related work, please see Appendix A.

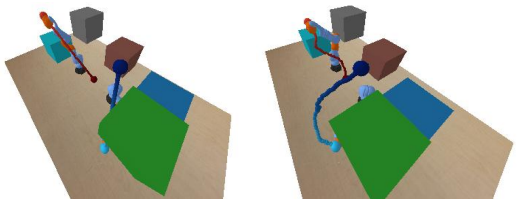

Figure 5: Dualkuka planned trajectory. Dark colors represent start and light colors represent end.

Learning-based Planning. Recent advances have introduced sequence modeling techniques into the planning domain [12, 14]. The key insight is that a trajectory can be viewed as a structured time series, allowing planning to be reformulated as a sequence generation problem. In this paradigm, a model is trained to learn the distribution over trajectories from data, capturing both dynamics and behavioral regularities. At inference time, planning is performed by conditioning the generative model on task-specific information, such as initial states, goals, or constraints, and sampling trajectories that satisfy these conditions.

A central idea behind these approaches is to reinterpret planning as probabilistic inference. Instead of explicitly searching over action sequences, sequence models are trained to approximate the distribution of feasible trajectories. Diffusion-based methods such as Diffuser [20] parameterize this distribution via a denoising process in trajectory space. During inference, planning reduces to conditional sampling, i.e. the model iteratively refines a noisy trajectory while enforcing constraints such as initial states or goal conditions. In this way, trajectory optimization is replaced by guided

Table 4: Ablation of the three main hyperparameters of **Ariadne** on DualKUKA ($n = 7$, $s = 0.14$). We vary one parameter at a time while fixing the others to their default values. Success rate (%) is reported.

Normal Threshold		Step Size		Projection Scale	
Value	SR ↑	Value	SR ↑	Value	SR ↑
-12	49.5	0.01	54.5	0.4	52.0
-20	**57.5**	0.02	**57.5**	0.5	**57.5**
-25	53.5	0.03	**57.5**	0.6	54.5
-30	51.5	0.04	54.0	0.7	53.0
-35	49.0	–	–	0.9	50.0

generation. Subsequent works have improved this framework for long-horizon planning [28] and compositionality [29], demonstrating the flexibility of generative sequence modeling for planning.

The Manifold Hypothesis. The manifold hypothesis [17, 39, 25] claims that high dimensional data actually live on a much lower-dimensional manifold embedded in that high-dimensional space, which inspires a series of machine learning techniques [39, 17]. With the development of generative model, it has been proposed to use diffusion models for learning the manifold [8, 21], as well as inducing manifold constraints into deep diffusion models [16, 10]. In this work, we extend prior assumptions, i.e. linear subspace, $D - 1$-dim manifold, towards a more general setting by using eigendecomposition on the Jacobian of diffusion models for local geometry extraction. Unlike prior works, we do not have a strong assumption over the structure of the data manifold and purely extract the geometry from the learned diffusion model.

7 Conclusion

In this work, we introduce **Ariadne**, a framework for trajectory planning without trajectory supervision. By extracting geometric information from a pretrained state-space diffusion model, **Ariadne** constructs trajectories under manifold constraints and endpoint constraints. Unlike trajectory-based generative approaches, our method learns only from feasible states and constructs trajectories during inference. Experiments on Maze2D and robotic motion planning benchmarks demonstrate that **Ariadne** can synthesize feasible trajectories without observing trajectory data during training, while generalizing to unseen start-goal configurations. These results suggest that state-space geometry may provide an alternative inductive bias for trajectory planning beyond explicit trajectory imitation.

Limitations. **Ariadne** assumes that sufficiently nearby states on the learned manifold admit valid local transitions. State validity alone does not guarantee transition validity in systems with irreversible, discontinuous, or strongly action-dependent dynamics; extending the model to state-action or transition tuples is therefore an important direction. Planning quality also depends on the accuracy of the learned local geometry and can degrade under insufficient state coverage, multimodal connectivity, or long-horizon problems where a simple reference path does not capture the correct global topology. In addition, estimating the normal space requires derivatives of the learned score field, making inference more computationally expensive as the state dimension grows. Developing more informative reference paths and extending the framework to more complex environments remain important directions for future work.

Acknowledgments and Disclosure of Funding

This work has been funded in part by the Army Research Laboratory (ARL) award W911QX-24-F-0049, DARPA award FA8750-23-2-1015, ONR award N00014-23-1-2840, ONR MURI grant N00014-25-1-2116, and the National University of Singapore under its Start-up Grant (Award No: SUG-251RES2505. The author would like to thank Xinran Zhao for the valuable feedback on the manuscript and Julia Du for the help on visual illustrations.

References

[1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[2] Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025.

[3] Karpathy Andrej. Walk in stable diffusion. `https://gist.github.com/karpathy/00103b0037c5aaea32fe1da1af553355`, 2022.

[4] Louis Béthune, David Vigouroux, Yilun Du, Rufin VanRullen, Thomas Serre, and Victor Boutin. Follow the energy, find the path: Riemannian metrics from energy-based models. *arXiv preprint arXiv:2505.18230*, 2025.

[5] Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

[6] Chang Chen, Fei Deng, Kenji Kawaguchi, Caglar Gulcehre, and Sungjin Ahn. Simple hierarchical planning with diffusion, 2024.

[7] Chang Chen, Hany Hamed, Doojin Baek, Taegu Kang, Samyeul Noh, Yoshua Bengio, and Sungjin Ahn. Extendable planning via multiscale diffusion, 2025.

[8] Ricky TQ Chen and Yaron Lipman. Flow matching on general geometries. *arXiv preprint arXiv:2302.03660*, 2023.

[9] 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*, 44(10-11):1684–1704, 2025.

[10] Hyungjin Chung, Jeongsol Kim, Geon Yeong Park, Hyelin Nam, and Jong Chul Ye. Cfg++: Manifold-constrained classifier free guidance for diffusion models. *arXiv preprint arXiv:2406.08070*, 2024.

[11] Hyungjin Chung, Byeongsu Sim, Dohoon Ryu, and Jong Chul Ye. Improving diffusion models for inverse problems using manifold constraints. *Advances in Neural Information Processing Systems*, 35:25683–25696, 2022.

[12] Murtaza Dalal, Jiahui Yang, Russell Mendonca, Youssef Khaky, Ruslan Salakhutdinov, and Deepak Pathak. Neural mp: A generalist neural motion planner. *arXiv preprint arXiv:2409.05864*, 2024.

[13] Yilun Du, Conor Durkan, Robin Strudel, Joshua B Tenenbaum, Sander Dieleman, Rob Fergus, Jascha Sohl-Dickstein, Arnaud Doucet, and Will Sussman Grathwohl. Reduce, reuse, recycle: Compositional generation with energy-based diffusion models and mcmc. In *International conference on machine learning*, pages 8489–8510. PMLR, 2023.

[14] Yilun Du, Toru Lin, and Igor Mordatch. Model based planning with energy based models, 2021.

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

[16] Yutong He, Naoki Murata, Chieh-Hsin Lai, Yuhta Takida, Toshimitsu Uesaka, Dongjun Kim, Wei-Hsiang Liao, Yuki Mitsufuji, J Zico Kolter, Ruslan Salakhutdinov, et al. Manifold preserving guided diffusion. *arXiv preprint arXiv:2311.16424*, 2023.

[17] Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

[18] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[19] Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David J Fleet. Video diffusion models. *Advances in neural information processing systems*, 35:8633–8646, 2022.

[20] Michael Janner, Yilun Du, Joshua B Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. *arXiv preprint arXiv:2205.09991*, 2022.

[21] Kacper Kapusniak, Peter Potaptchik, Teodora Reu, Leo Zhang, Alexander Tong, Michael Bronstein, Joey Bose, and Francesco Di Giovanni. Metric flow matching for smooth interpolations on the data manifold. *Advances in Neural Information Processing Systems*, 37:135011–135042, 2024.

[22] Lydia E Kavraki, Petr Svestka, J-C Latombe, and Mark H Overmars. Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE transactions on Robotics and Automation*, 12(4):566–580, 2002.

[23] James J Kuffner and Steven M LaValle. Rrt-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium conference. IEEE international conference on robotics and automation. Symposia proceedings (Cat. No. 00CH37065)*, volume 2, pages 995–1001. IEEE, 2000.

[24] Vladimir Kulikov, Matan Kleiner, Inbar Huberman-Spiegelglas, and Tomer Michaeli. Flowedit: Inversion-free text-based editing using pre-trained flow models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19721–19730, 2025.

[25] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[26] Zhixuan Liang, Yao Mu, Mingyu Ding, Fei Ni, Masayoshi Tomizuka, and Ping Luo. Adaptdiffuser: Diffusion models as adaptive self-evolving planners, 2023.

[27] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023.

[28] Yunhao Luo, Utkarsh A Mishra, Yilun Du, and Danfei Xu. Generative trajectory stitching through diffusion composition. *arXiv preprint arXiv:2503.05153*, 2025.

[29] Yunhao Luo, Chen Sun, Joshua B Tenenbaum, and Yilun Du. Potential based diffusion motion planning. *arXiv preprint arXiv:2407.06169*, 2024.

[30] Shev MacNamara and Gilbert Strang. Operator splitting. In *Splitting methods in communication, imaging, science, and engineering*, pages 95–114. Springer, 2017.

[31] Elio Moreau, Florentin Coeurdoux, Grégoire Ferre, and Eric Vanden-Eijnden. Probing the geometry of diffusion models with the string method. *arXiv preprint arXiv:2602.22122*, 2026.

[32] Michael Posa, Cecilia Cantu, and Russ Tedrake. A direct method for trajectory optimization of rigid bodies through contact. *The International Journal of Robotics Research*, 33(1):69–81, 2014.

[33] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[34] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

[35] 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.

[36] Gilbert Strang. On the construction and comparison of difference schemes. *SIAM journal on numerical analysis*, 5(3):506–517, 1968.

[37] Łukasz Struski, Michał Sadowski, Tomasz Danel, Jacek Tabor, and Igor T Podolak. Feature-based interpolation and geodesics in the latent spaces of generative models. *IEEE Transactions on Neural Networks and Learning Systems*, 35(9):12068–12082, 2023.

[38] Yuval Tassa, Tom Erez, and Emanuel Todorov. Synthesis and stabilization of complex behaviors through online trajectory optimization. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 4906–4913. IEEE, 2012.

[39] Joshua B Tenenbaum, Vin de Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *science*, 290(5500):2319–2323, 2000.

[40] Hale F. Trotter. On the product of semi-groups of operators. 1959.

A Extented Related Works

A.1 Generative Models for Sequence Modeling

Generative models have seen remarkable progress in recent years [33, 9, 18, 1]. Diffusion models, in particular, have gained significant attention [20, 33, 18, 34, 35]. While initially advanced in image domain, works have been done on extending it to sequence modeling including video [19, 5], robotics [9] and planning [20, 28]. In addition, works have also tried to explore the geometry underlying the generative models [16, 4, 21, 8]. While prior work assumes a learned metric model, we focus on directly extracting the geometry from a trained diffusion model.

A.2 Latent Interpolation

Latent interpolation has been widely studied in generative modeling as a tool for exploring learned representations and constructing smooth transitions between data samples [2, 3].

While the standard way is to do linear interpolation, several advanced methods on latent interpolation [4, 31, 24, 3, 37], considering geodesic paths or spherical interpolation (slerp), have been proposed in diffusion-based models to demonstrate continuity and semantic smoothness of learned latent spaces.

However, latent interpolation primarily operates in the noise space rather than directly enforcing geometric or dynamical feasibility in the ambient data space. The resulting trajectories are smooth in latent coordinates but are not explicitly constrained to remain connected on a learned data manifold.

In contrast, **Ariadne** constructs trajectories directly in the data space by explicitly modeling the local geometry of the data manifold and formulating trajectory generation as integrating a constrained ODE. Rather than interpolating between latent embeddings, we design a dynamical system whose integral curves both remain on the data manifold and satisfy endpoint constraints by construction.

B Math formulation for ODE construction

B.1 Operator Splitting Methods

Operator Splitting methods [40, 36, 30] are a family of methods that are proposed to solve a specific type of ODE. Consider an ODE whose vector field can be written as the sum of two components:

$$\frac{dx(t)}{dt} = A(x(t)) + B(x(t))\,, \tag{8}$$

where $A(\cdot)$ and $B(\cdot)$ represent two functions that are easier to handle separately. Instead of solving Eq. (8) in one shot, Operator splitting suggests that over each time interval of length Δt, we first evolve the state under A and then evolve it under B.

Split subproblems. Specifically, for each step we solve the two sub-ODEs sequentially:

$$\frac{dx(t)}{dt} = A(x(t))\,, \tag{9}$$

$$\frac{dx(t)}{dt} = B(x(t))\,. \tag{10}$$

Forward Euler discretization. Starting from x_{t-1}, we apply a forward Euler step for Eq. (9):

$$x_t^{(A)} = x_{t-1} + \Delta t\, A(x_{t-1}), \tag{11}$$

followed by a forward Euler step for Eq. (10) initialized at $x_t^{(A)}$:

$$x_t = x_t^{(A)} + \Delta t\, B\Big(x_t^{(A)}\Big)\,. \tag{12}$$

Equivalently, the full update can be written as

$$x_t = x_{t-1} + \Delta t\, A(x_{t-1}) + \Delta t\, B(x_{t-1} + \Delta t\, A(x_{t-1}))\,. \tag{13}$$

Operator Splitting works when alternating between sub-dynamics does not introduce uncontrolled curvature or instability relative to the combined flow. We refer the reader to the book about the method for rigorous explanation [30].

B.2 Iterative Projection

The iterative projection procedure that we used can be written as follows,

$$\begin{aligned} x'_t &= x_{t-1} + (x_1 - x_0)\Delta t, \\ x_t &= x'_t + \sum_{n \in \mathbf{V}} [-\langle x'_t - x_{t-1}, n \rangle n], \end{aligned} \tag{14}$$

We can rewrite the second line as

$$x_t = x'_t + \frac{1}{\epsilon} \sum_{n \in \mathbf{V}} [-\langle x'_t - x_{t-1}, n \rangle n] \Delta t \tag{15}$$

where ϵ is a small constant that cancels out the effect of the step size Δt, i.e., $\epsilon = \Delta t$, $\mathbf{V}$ is the local normal subspace and n are the basis of that subspace. It corresponds to an operator-splitting numerical method for solving the following ODE,

$$\frac{dx}{dt} = x_1 - x_0 + \sum_{n \in \mathbf{V}} \frac{(-nn^T)dx}{\epsilon}, \tag{16}$$

where $\epsilon = O(dt)$, $A(x_t) = x_1 - x_0$ and $B(x_t) = \sum_{n \in \mathbf{V}} \frac{(-nn^T)dx}{\epsilon}$.

B.3 Correction-Projection ODE

We start our derivation with a well-established non-linear interpolation formula since it naturally guarantees the start and end points.

$$x_t = (1-t)x_0 + tx_1 + 2t(1-t)\varphi(x_t), \tag{17}$$

where $\varphi(x_t)$ is the non-linear correction term and $(1-t)x_0 + tx_1$ represents a straight-line interpolation. Taking derivative w.r.t t on both sides, we get

$$\frac{dx}{dt} = x_1 - x_0 + (2-4t)\varphi(x_t) + 2t(1-t)\frac{d}{dt}\varphi(x_t). \tag{18}$$

We now find a correspondence between the iterative-projection ODE and Eq. (18).

Notice that since $\epsilon = O(dt)$, we can draw similarity between the term $\sum_{n \in \mathbf{V}} \frac{(-nn^T)dx}{\epsilon}$ in Eq. (16) and $2t(1-t)\frac{d}{dt}\varphi(x_t)$ in Eq. (18). We can set

$$\sum_{n \in \mathbf{V}} \frac{(-nn^T)dx}{\epsilon} = 2t(1-t)\frac{d}{dt}\varphi(x_t). \tag{19}$$

Therefore by canceling out ϵ and dt, we have

$$d\varphi(x_t) = \sum_{n \in \mathbf{V}} \frac{(-nn^T)dx}{2t(1-t)} \tag{20}$$

and

$$\varphi(x_t) = \varphi(x_0) + \int_0^t \frac{d\varphi(x_t)}{ds} ds. \tag{21}$$

Since we solve the ODE using Euler method, we can set $\varphi(x_0) = \mathbf{0}$ and compute the integral in Eq. (21) as

$$\varphi(x_t) = \sum_i \sum_{n \in \mathbf{N_i}} \frac{(-nn^T)dx}{2t(1-t)\epsilon} \Delta t, \tag{22}$$

where i denotes the discrete integration step when iterating the ODE to x_t, and $\mathbf{V}_i$ denotes an orthonormal basis of the normal subspace at the current state x_t.

C Pseudo Code for the proposed Correction-Projection ODE

Please refer to Algo. 1 for full procedure of evolving our algorithm. Notice that there are three hyperparameters that has to be set for evolving the Correction-Projection ODE, i.e. the initial arc-length (or step size for evolving the ODE in state space), the projection scale and the normal selection threshold. We provide ablation studies on these three factors in Sec. 5.2.

Algorithm 1 Correction Projection ODE

Input: initial arc-length C, projection scale ϵ, Normal selection threshold T_{normal}, start point x_0, end point x_1, a flow model learned on state distribution F
Output: A feasible plan P with every state in the state space.
Initialize parameters $t = 0$
Initialize state $x \leftarrow x_0$
Initialize past state $x_p \leftarrow x_0$
Initialize post-projection state $x_{pro} \leftarrow x_0$
Initialize PHI list $d\varphi = []$
Initialize State Path $P = [x_0]$
while $t < 1$ **do**
  if $t = 0$ **then**
    $dt = C$
  else
    $dt = \frac{C}{x - x_p}$
    $dt = \min\{dt, 1 - t\}$
  end if
  $V_{ref} = x_1 - x_0$
  Manifold correction scale $M = \min\{\exp(|x - x_0|), \exp(|x - x_1|)\}$
  $S = F(x, t = 0.98)$
  eigendecomposition: $\frac{\nabla_x S + (\nabla_x S)^T}{2} = Q\Sigma Q^T$
  $n = \{\frac{Q_i}{M} | \Sigma_i < T_{normal}\}$, $u = sum(\{\Sigma_i | \Sigma_i > 0\})$
  $x_p \leftarrow x$
  for n_i in n **do**
    $k = -\Sigma_i$ if $-\Sigma_i > -1$ else $k = -\frac{1}{\Sigma_i}$
    $x = x - \epsilon \frac{u^k M(n_i n_i^T)(x - x_{pro})}{||n_i||^2}$
    $d\varphi \leftarrow \epsilon \frac{u^k M(n_i n_i^T)(x - x_{pro})}{2t_i(1 - t_i)||n_i||^2}$
  end for
  $x_{pro} \leftarrow x$
  $\varphi = \sum d\varphi$
  $x = x - 2(1 - 2t)\varphi dt$
  $x = x + V_{ref} dt$
  $t = t + dt$
  $P \leftarrow x$
end while
Return: P

D Geometry from score function

D.1 Hessian and local geometry.

We analyze the second-order structure of the log-density via the Hessian

$$H(x) = \nabla_x^2 \log p(x).$$

Under the manifold-plus-noise assumption, the log-density is approximately flat along tangent directions of $\mathcal{M}$ and exhibits strong negative curvature along normal directions. Consequently, eigenvectors corresponding to large negative eigenvalues span the normal space, while those with near-zero eigenvalues align with the tangent space.

This decomposition provides a local characterization of the manifold geometry. Deviations from this structure, such as the presence of positive eigenvalues, indicate more complex local structures (e.g., multi-modality or stratified manifolds).

D.2 Riemannian metric from score function.

In addition to identifying normal directions of the data manifold, performing projection also requires a notion of distance, i.e., a Riemannian metric defined over the ambient space. We observe that a learned energy function $E_\theta(x)$ from diffusion or score-based models naturally provides such geometric information [4, 13].

Ideally, the learned energy satisfies two properties: (1) $E_\theta(x)$ is small on the data manifold and increases as x moves away from it; and (2) the gradient magnitude $|\nabla_x E_\theta(x)|$ is close to zero when x lies on the manifold. Under these assumptions, the norm of the score function serves as a proxy for the distance to the manifold since $s(x) = -\nabla_x E_\theta(x)$.

Motivated by this observation, we define a Riemannian metric

$$G(x) = \frac{\exp\left(|s(x)|\right)}{Z} I, \tag{23}$$

where Z is a normalization constant and I is the identity matrix.

By construction,

$$G(x) \approx I \quad \text{when } x \in \mathcal{M}, \tag{24}$$

and $G(x)$ increases as x moves away from the manifold. This induces a geometry in which off-manifold directions incur larger metric cost, effectively penalizing deviations from $\mathcal{M}$.

Therefore, under the assumption that $|\nabla_x E_\theta(x)| \approx 0$ on the data manifold, a Riemannian metric can be directly derived from a learned generative model without explicit manifold supervision.

E Practical Considerations Full Details

Reference path selection. Notice that $x_1 - x_0$ is a direct outcome of the linear interpolation between x_1 and x_0. We may also use different parametrizations for the interpolation such as SLERP [3], non-perfect trajectories generated by trajectory-based generative models [29, 20], or parametrized curve like

$$x(t) = (1-t)x_0 + tx_1 + (\sum_{k=1}^{m} \alpha_k \sin(k\pi t))d, \tag{25}$$

where m and α_k are hyperparameters deciding the shape of the curve and d is chosen to be a perpendicular direction w.r.t the straight line $(1-t)x_0 + tx_1$.

However, we do find that in practice straight-line reference path is sufficient enough for **Ariadne** to provide reasonable results and we stick to the straight line for our experiments.

Step size and parameter t. In order to ensure the connectivity in the state space, instead of choosing a fixed time interval dt, we should instead do a change of variable and consider a fixed step size ds, the time interval should then be $dt = \frac{ds}{||f||}$, where f is the RHS of Eq. 6.

Flow Matching and Diffusion. While diffusion models directly predict the score function and thus lead to simple Hession computation, the implementation over a Flow-Matching model should slightly be changed to use $\frac{(J+J^T)}{2}$ as the approximation of the Hession matrix where J is the Jacobian of the output of Flow-Matching w.r.t its input.

F Experimental Setup Details

We test our algorithm in the Maze 2D environment and only train our model using the state-space data of the maze without any trajectory data that inform connectivity. We report the standard score gathered in the environment as the evaluation metric.

We further validate **Ariadne** on the YAM 6D tabletop setup, where a 6-DoF robotic arm operates in a 3D workspace containing tabletop obstacles. We evaluate the success rate of collision-free motion planning. Similar to Maze 2D, our model is trained using state-only supervision, with no trajectory data and no transition information.

We construct the training set by sampling arm configurations and applying forward kinematics to obtain end-effector and link poses. We then run a collision filter against the tabletop and obstacles, and retain only collision-free states as feasible training data.

For each test instance (start-goal pair), we plan a state sequence using the learned model and execute the resulting plan under the environment's standard collision and validity checks. We report the task success / score defined by the benchmark (same scoring protocol as the environment), emphasizing that performance is achieved despite training without trajectories.

For all experiments, we use 0.02 as the step size for evolving the ODE. We use -0.0001 as normal threshold and 1.0 as projection scale for Maze 2D, -1.0 as normal threshold and 0.5 as projection scale for YAM 6D and -20.0 as normal threshold and 0.5 as projection scale for DualKUKA 14D. We found that as the dimension of the problem goes up, the absolute value of eigenvalues goes up, and therefore bigger normal threshold. For close manifold normal direction estimation, we use the timestep 0.98 as input to the flow-matching model that we used as it is trained to estimate a time-dependent vector field and close to 1 could provide a reasonable normal estimation for the real data manifold.

We further adopt DualKuka [29], a more challenging scenario where two 7-Dof robot arm is tasked to avoid obstacles with more complex layout as well as not colliding with each other. The state space is defined to be all the collision-free end effector pose. Similar to the YAM 6D setting, each test instance is a given start-goal pair for both the arm and we aim to generate a state sequence using our algorithm. The success rate is computed as the **one-shot** success of a collision-free and goal-reaching path generation. We spawn 7 and 5 obstacles following the standard data generation process of DualKUKA for quantitative evaluation. The half size of the block is set to 0.14, 0.22 respectively for the three scnarios reported in Tab. 3. Qualitative results are given with 5 obstacles with 0.22 as half size as it gives better visualization of how our method provides obstacle avoidance capability, i.e. manifold following, for a path that has collision. The ablation study is done on 7 obstacles with 0.14 as the half size. For our results, we use the end effector space with XYZ position and euler angle for the state representation. We found that as this representation reveals the essential problem dimension, our method is able to provide comparable results to State-of-the-Art trajectory-based methods.

For baselines, since we follow the tradition as Diffuser [20] to train single planner for a single scene, we restrict the evaluation to a single scene and evaluate the baselines also on this restricted scene even if they offer generalizable capabilities to take as input different scenarios' layout.

G Training Details

For training the flow-matching model, we use a MLP with 4 hidden layers and 128 as the latent dimension for Maze 2D and YAM 6D. For DualKUKA, we use MLP with 8 hidden layers and 1024 as hidden dimension. The learning rate for Maze 2D medium, Maze 2D large, YAM 6D and DualKUKA are respectively $1e-4$, $5e-4$, $1e-4$ and $2e-5$. The batch size for Maze 2D and YAM 6D is 64 and for DualKUKA is 1024. We transform the joint angle representation of DualKUKA into end effector space and represent it by euler angle. The inference of our algorithm is conducted in end effector space and use IK with the joint angle reference of the start point to derive the joint angles for both the KUKA arms. All the training is conducted on a single NVIDIA RTX 6000 Ada Generation GPU and requires from 2 hours to 12 hours depending on the tasks and dataset size. We follow the standard way to generate the Maze data, randomly uniformly sampled for the YAM 6D environment and uniformly sampled for the DualKUKA environment. The GPU memory needed for training such model is only at around 400M.

H More Quatitative Results

In Fig. 6, we visualize a representative planned trajectory produced by **Ariadne** in the YAM 6D scene: the plotted path shows the end-effector (and/or configuration) sequence staying within the collision-free manifold and routing around obstacles to connect the specified start and goal.

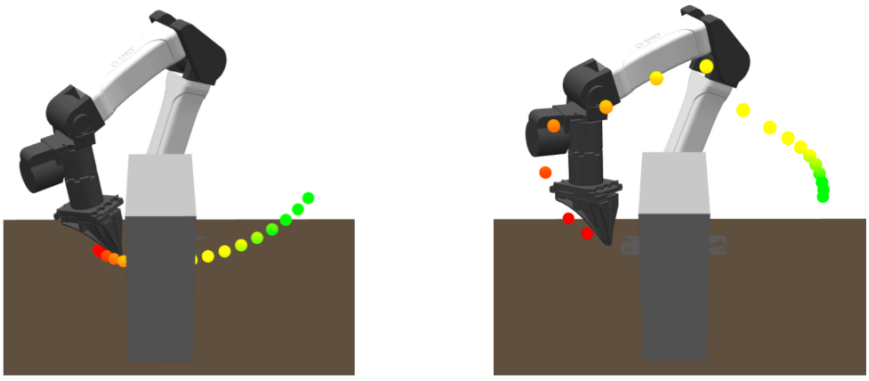

Figure 6: Visualization of the planned trajectory. Color of the way-points specify task progress, where initial state is green and final state is red.

I Extented Analysis on Ablation Studies

Projection scale. We compare multiple scaling factors in Tab. 4 and observe that performance peaks at a moderate scale, while both smaller and larger values lead to degraded results. This can be attributed to a trade-off between following the reference direction and enforcing manifold constraints. When the projection scale is too small, the update is dominated by the reference direction, causing the trajectory to drift away from the manifold and resulting in infeasible states. In contrast, when the projection scale is too large, the accumulated projection corrections dominate the update, significantly altering the direction of motion and causing the trajectory to drift away from the reference path. This hinders effective progress toward the goal. A moderate projection scale balances these effects, leading to more stable and effective trajectory construction.

Step size. We evaluate step sizes of 0.01, 0.02, 0.03 and 0.04 in Tab. 4. We observe that moderate step sizes achieve the best performance, while both smaller and larger values lead to degraded results. This can be attributed to a trade-off in the discretized update dynamics. Smaller step sizes provide a more accurate approximation to the underlying continuous trajectory, but require more iterations to make progress, which can accumulate errors. In contrast, larger step sizes may overshoot the local region where the manifold approximation is valid, moving the state far from the manifold before projection is applied. This weakens the effectiveness of the projection step and degrades performance. Moderate step sizes balance these effects, leading to more stable and effective trajectory construction.

Normal threshold. We vary the threshold for selecting normal directions in Tab. 4 and observe that performance peaks at a moderate threshold, while both smaller and larger values lead to degraded results. This can be attributed to a trade-off in estimating the normal subspace. When the threshold is too low, many weak or noisy directions are included as normals, leading to inaccurate projections that distort the trajectory and introduce instability. In contrast, when the threshold is too high, only a small subset of directions is retained, resulting in an incomplete characterization of the normal space. This weakens the constraint and allows the trajectory to drift away from the manifold. A moderate threshold strikes a balance between these effects, yielding a more accurate and stable approximation of the normal space.

J Extended Discussion

A central observation behind **Ariadne** is that trajectory incompleteness does not necessarily imply state incompleteness. While trajectory datasets often sparsely cover start-goal combinations, feasible states themselves may still be broadly represented. Rather than explicitly learning trajectory distributions, **Ariadne** constructs trajectories from the geometry of feasible states, enabling planning from state distribution learning.

Table 5: Planning efficiency comparison with RRT. We report success rate (SR, %), number of collision checks, and planning time in seconds. For DualKUKA, n denotes the number of obstacles and s denotes the obstacle half-size. **Ariadne** does not invoke a collision checker during online trajectory construction.

ENVIRONMENT	METHOD	SR ↑	COLLISION CHECKS ↓	TIME (S) ↓
MAZE-MEDIUM	RRT	45.95	47,374.67	3.08
	ARIADNE	55.69	0	0.00145
MAZE-LARGE	RRT	10.70	90,849.51	4.50
	ARIADNE	50.63	0	0.00049
DUALKUKA ($n = 7,\ s = 0.14$)	RRT	47.0	21,039.0	8.26
	ARIADNE	57.5	0	1.482
DUALKUKA ($n = 5,\ s = 0.14$)	RRT	36.5	56,311.8	25.78
	ARIADNE	50.5	0	1.122
DUALKUKA ($n = 5,\ s = 0.22$)	RRT	50.5	21,862.0	7.92
	ARIADNE	52.0	0	2.040

While our proposed method takes the form of ODE construction, there are several ways that it can be interpreted and connected to existing methods like the transition kernel interpretation of diffusion models [18], and transportation between arbitrary distributions [24, 27].

Ariadne may also be viewed as inducing deterministic local transition dynamics over the learned state manifold. Conceptually related to probability flow ODEs in diffusion models [18, 35], **Ariadne** instead considers transport between feasible states while remaining consistent with the learned state distribution.

Transition Kernel. The evolution of the ODE that we constructed can also be viewed as a deterministic transition kernel

$$K(x_{t+dt}|x_t) = \delta(x_{t+dt} - v(x_t, t)dt), \tag{26}$$

where $v(x_t, t)$ is the RHS of our constructed ODE and δ is the Dirac delta function.

We can view the constructed kernel as a replacement to the Gaussian kernel used in the diffusion process. **Ariadne** therefore can be viewed as constructing a kernel that we can sample a real data from. Such interpretation can further be extended to stochastic kernels to introduce exploration and multi-modality to the method.

Transportation between Two Distributions. Similarly, the two endpoint constraints specified in trajectory planning can also be viewed as two Dirac delta function centered at the start and end point respectively. Therefore **Ariadne** can also be explored together with existing methods like FlowEdit [24] for high-dimensional sequence generation with only distribution-level constraints.

More broadly, our results raise the question of whether the manifold hypothesis and its geometry may extend beyond trajectory planning to broader sequential domains, which we leave to future work.

K Efficiency Comparison

In addition to planning success, we compare **Ariadne** with RRT in terms of online planning efficiency. RRT performs explicit sampling-based search and repeatedly queries a collision checker to identify feasible paths, whereas **Ariadne** directly constructs trajectories from the learned state-space geometry and does not invoke a collision checker during online path construction. For DualKUKA, n denotes the number of obstacles and s denotes the obstacle half-size, consistent with Table 3. We report success rate, the number of collision checks, and wall-clock planning time in Table 5.

Table 6: Robustness across five independently trained score models on DualKUKA. We report mean $\pm$ standard deviation of success rate (%).

METHOD	$n = 7$ $s = 0.14$	$n = 5$ $s = 0.14$	$n = 5$ $s = 0.22$
ARIADNE	56.90 ± 2.22	50.70 ± 3.25	51.30 ± 1.99

L Detailed Explanation of Baselines

We compare **Ariadne** against planning methods with different supervision and inference assumptions. RRT is a classical sampling-based planner that explicitly searches the configuration space and queries a collision checker during planning. Diffuser is a trajectory-supervised generative planner trained on complete state trajectories. Diffuser* denotes the restricted variant used in our original Maze2D experiments: the model observes the evaluation target states during training, but never observes them as trajectory endpoints, and therefore does not receive trajectory supervision for the corresponding target-reaching planning problems.

Follow-the-Energy [4] is the closest geometry-based baseline to **Ariadne**. It learns an energy model from state observations, derives a Riemannian metric from the learned energy landscape, and optimizes a trajectory interpolant under this metric. PBDMP is a trajectory-based diffusion motion planner that additionally takes the scene layout as input to enable generalization across environments. In contrast, **Ariadne** is trained only on state observations and constructs trajectories at inference time directly from the learned local geometry of the state space.

M Robustness to Score-Model Retraining

Although trajectory construction is deterministic given a trained generative model, the learned state-space geometry may vary with model initialization and optimization. We therefore independently train five score models and evaluate **Ariadne** with each model on the three DualKUKA settings. The results are reported in Table 6.

The relatively small variation across independently trained models indicates that **Ariadne**'s planning performance is not strongly dependent on a particular score-model initialization.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes], [No], or [N/A].
- [N/A] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for [N/A]).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will also be asked to include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While [Yes] is generally preferable to [No], it is perfectly acceptable to answer [No] provided a proper justification is given (e.g., error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering [No] or [N/A] is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

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

   Answer: [Yes]

   Justification: The contributions and scope are well covered in the following sections of Introduction.

   Guidelines:

   - The answer [N/A] 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 [N/A] 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: Limitation is discussed in Conclusion.

Guidelines:

- The answer [N/A] 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: [N/A]

   Justification: The paper contains no theoretical results but only uses theoretical hypotheses to guide the development of the methodology.

   Guidelines:

   - The answer [N/A] 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: The hyperparameters are provided in the Appendix and source code will be made public upon acceptance.

   Guidelines:

- The answer [N/A] 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: [No]

   Justification: We will release the code upon acceptance.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.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://neurips.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, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Experimental details are provided in the Appendix

   Guidelines:

   - The answer [N/A] 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: [Yes]

   Justification:We report mean and standard deviation across five independently trained score models for the DualKUKA experiments in Appendix M.

   Guidelines:

   - The answer [N/A] 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: The full details of compute resources is provided in the Appendix.

   Guidelines:

   - The answer [N/A] 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: Code of Ethics is fully understood and conformed.

   Guidelines:

   - The answer [N/A] 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: [N/A]

    Justification: The work includes no social aspects but focuses on core deep learning problems.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] 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., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper poses no such risks.

Guidelines:

- The answer [N/A] 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: All the code and data used are properly cited in the paper.

    Guidelines:

    - The answer [N/A] 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: [N/A]

    Justification: Code will be made public upon acceptance. Other than the code, the paper contains no new assets.

    Guidelines:

    - The answer [N/A] 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: [N/A]

Justification: No human is included in the experiments.

Guidelines:

- The answer [N/A] 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: [N/A]

Justification: No human subjects or crowdsourcing is included in the paper.

Guidelines:

- The answer [N/A] 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 rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer [N/A] 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 in the NeurIPS handbook for what should or should not be described.