Enhancing Diffuser for Decision-Making with Velocity Prediction with MPPI and Replanning

Kevin Mathew T
New York University
New York, USA
km6748@nyu.edu

Rijul Dahiya
New York University
New York, USA
rd3629@nyu.edu

Grishma Deshmukh
New York University
New York, USA
gd2574@nyu.edu

Utsavi Visaria
New York University
New York, USA
ujv1@nyu.edu

***Abstract*—In this work, we present a set of enhancements to the Diffuser framework for offline reinforcement learning by introducing velocity prediction, Model Predictive Path Integral (MPPI) sampling, and a strategic replanning mechanism. While diffusion models are only recently becoming effective trajectory generation tools by restructuring planning as a generative modeling problem, they are frequently hampered by issues such as trajectory instability, poor temporal coherence, and dynamism in constrained dynamic environments. In response, we provide a velocity-based parameterization of diffusion, which is suitable for image formation, and extend it to sequential decision-making. This leads to smoother, more stable trajectory planning. Going a step further, MPPI sampling achieves robustness through softmax-weighted returns over a collection of candidate trajectories, effectively improving path optimization. Our replanning module also improves adaptability with online corrections for planning over long horizons. Throughout the Maze2D suite of navigation environments, our improved Diffuser model performs by as much as a 21.1% performance gain over the original baseline. Moreover, the entire pipeline Velocity Diffuser using MPPI and Replanning gives rise to a new state-of-the-art baseline bettering both old and new baselines. Most importantly, our approach shortens the planning horizon needed by around 50%, significantly improving computational efficiency without sacrificing trajectory quality. These results demonstrate the potential of our approach to solve difficult long-horizon decision-making tasks with better stability, efficiency, and adaptability.**

***Index Terms*—Diffusion, velocity prediction, MPPI**

I. Introduction

Traditionally planning in reinforcement learning is done through dynamics modeling and trajectory optimization. Model-based reinforcement learning methods typically learn an approximate dynamics model and then use classical trajectory optimization techniques to generate plans. However, this approach often struggles when learned models are exploited by optimizers, resulting in adversarial examples rather than optimal trajectories. Planning with diffusion models represents a paradigm shift to this. This is done through a diffuser model. [1]

Diffuser addresses this limitation by treating planning as a generative modeling problem. Instead of predicting states and actions autoregressively in temporal order, Diffuser models entire trajectories as two-dimensional arrays that represent interleaved states and actions across a planning horizon. The model generates trajectories through an iterative denoising process, from random noise and successively refining the trajectory by imposing local temporal consistency. The approach had several significant advantages. Compared to single-step models suffering from compounding rollout errors, diffuser is trained to produce correct full trajectories and scales better with planning horizon length. In addition, reward functions provide gradients for steering trajectory sampling, thus planning can be constructed up through multiple rewards simultaneously. Diffuser also generates consistent paths through iteratively improving local consistency, enabling generalization to new paths by combining familiar subsequences. And, as a fully convolutional model, Diffuser's planning horizon is determined by input dimensionality rather than architectural constraints. Finally, the diffusion-based technique avoids the shortsighted failure modes found in shooting-based planning algorithms, allowing solutions to sparse-reward, long-horizon problems.

In the Diffuser, planning and sampling are almost similar, with auxiliary guidance functions steering the sampling process toward high-reward locations. This framework has performed well in complicated tasks demanding long-horizon thinking and test-time flexibility, beating both model-free and standard model-based techniques.

In this work, we extend the Diffuser framework by incorporating velocity prediction, Model Predictive Path Integral (MPPI) sampling, and strategic replanning to improve decision-making capabilities.

II. Literature Survey

Denoising Diffusion Probabilistic Models (DDPM), introduced by Sohl-Dickstein et al. (2015) [2], establish a novel generative modeling approach by iteratively refining initially noisy data towards high-quality target samples and are based on a well-defined stochastic diffusion process that gradually corrupts data, and a learned reverse denoising process that systematically restores it. The approach was significantly enhanced by Ho et al. (2020) [3] which further enhanced model stability and sample efficiency with superior performance being realized in tasks for generative synthesis of images. Score-Based Generative Modeling by Song & Ermon (2019) [4] focused more on score matching by computing the gradients (scores) of the data distributions and presented an orthogonal view with solid theoretical justifications for the diffusion models showing the possibility of successful modeling complex distributions using gradient-based approaches.

Dynamics Modeling for RL, as discussed by Janner et al. (2021) [5], applied advanced Transformer models to address offline reinforcement learning challenges. This research solved major issues like forecasting sequential dynamics with accuracy and dealing with uncertainty in modeling long-term trajectories with great improvement in offline policy optimization and trajectory prediction accuracy.

Trajectory Optimization and Planning have been the foundation of model-based control systems for a long time. Tassa et al. (2012) [7] gave universal solutions for synthesizing and stabilizing complex behaviors through online trajectory optimization. Meanwhile, Kelly (2017) [8] presented the direct collocation approach, giving a practical and theory-established method that has been used predominantly in optimal control and trajectory planning in most reinforcement learning tasks.

The Diffuser model proposed [1] introduced a novel approach to combining diffusion models with offline reinforcement learning, reformulating trajectory generation as a conditional generative modeling problem. Diffuser models generate complete trajectories in parallel by iteratively denoising noisy sequences conditioned on start and goal states, unlike the standard policy optimization method, where actions are generated sequentially at each timestep. This denoising-based formulation enables the model to do global reasoning across whole trajectories, resulting in more coherent and adaptable behavior synthesis. A key benefit of this framework is that it uses the generative powers of diffusion models to be particularly effective in sparse reward situations and long-horizon planning problems. By operating in trajectory space rather than action space, it can better grasp temporal dependencies and work efficiently for global goals. The model is also adaptable, can do fine-grained conditioning and interpolation across objectives making it ideal for multitask and transfer learning environments.

Model Predictive Path Integral (MPPI) introduced by Williams et al. (2015) [6] builds on trajectory optimization by approximating many possible trajectories and their associated costs. MPPI uses sampling-based strategy with probabilistic path integral techniques, balancing exploration and exploitation in an acceptable manner in difficult control environments.

Velocity prediction for diffusion models, as suggested by Ho et al. (2020) [3], is a good substitute for the traditional noise prediction approach in the reverse diffusion process. Rather than predicting the added noise directly, velocity-based parameterization predicts a latent variable that integrates information from the noisy input and the clean target sample. This leads to more stable training dynamics, quicker convergence rates, and overall improved sample quality, especially in high-dimensional data settings such as image creation. While this method has been explored so far for generative vision models, its extension to sequential decision-making tasks is not yet fully established. In reinforcement learning, particularly in trajectory generation tasks, velocity prediction can provide a temporally more coherent and physically more plausible state transition estimate. By learning to predict how states evolve over time rather than merely reconstructing them models are able to better learn system dynamics, leading to improved trajectory prediction and planning performance. Our work intends to extend these benefits to offline RL by demonstrating that velocity-based parameterization improves efficiency and robustness in goal-conditioned planning tasks like Maze2D navigation.

The Maze2D Benchmark for Sparse Reward RL, introduced by Fu et al. (2020) [9], has been a prominent evaluation system specially designed to test reinforcement learning algorithms under long-horizon, sparse-reward conditions. It benchmark tests RL algorithms, notably showing the robustness and generality of diffusion-based planning methods over traditional RL approaches such as Conservative Q-Learning (CQL) and Implicit Q-Learning (IQL).

Experimental Validation and Benchmarking performed extensively on Maze2D environments consistently demonstrate the advantages of diffusion-based planning methods, especially their capability to handle sparse reward conditions and long-term planning horizons effectively. These experimental studies provide substantial evidence validating diffusion-based models' improved robustness, adaptability, and effectiveness over traditional reinforcement learning approaches.

III. Recent Work

New developments in reinforcement learning (RL) methods have increasingly moved towards methods that can deal with intricate, long-horizon decision-making problems. From all the developing techniques, diffusion probabilistic models have received much interest because of their success in modeling high-dimensional data distributions elegantly. One pivotal development in this area is the Diffuser framework introduced by Janner et al. [1], which fundamentally redefines model-based reinforcement learning by treating the planning problem as a generative modeling task. Unlike traditional model-based RL approaches that utilize learned models solely for predicting next-step dynamics, Diffuser simultaneously generates entire trajectories through an iterative denoising process. This trajectory level of generation introduces a much decreased cumulative error traditionally associated with AR prediction methods such that it increases the coherence as well as overall global consistency when generating plans. Additionally, this Diffuser introduces flexibility which allows conditioning such that the guidance signals condition or steer trajectory samples toward regions under higher expected reward. This aspect proves particularly advantageous in sparse-reward environments, such as maze navigation tasks, where goal states are infrequent and the reward signal sparse.

The Diffuser framework naturally facilitates task compositionality and temporal compositionality by enabling tasks with different rewards or constraints to be addressed well through the composition of pertinent objectives at the trajectory sampling phase. This feature significantly promotes the model's versatility in a range of different and unseen task settings without requiring retraining, hence highlighting its capabilities for intricate environments that require dynamic decision-making abilities. Janner et al. empirically showed the effectiveness of Diffuser over a range of reinforcement learning benchmarks, particularly performing well on long-horizon Maze2D navigation tasks where classical reinforcement learning approaches face huge challenges in the presence of sparse feedback and large state-action spaces.

Based on this initial work we built upon the capabilities of Diffuser by adding more techniques, such as explicit velocity prediction, Model Predictive Path Integral (MPPI) sampling, and strategic replanning. These changes as a whole aimed to solve some of the remaining issues in diffusion-based planning models, including trajectory stability, temporal consistency, and responsiveness to dynamic changes. We incorporated velocity prediction into the denoising process, inspired by successes observed in generative vision models such as Stable Diffusion, where incorporating velocity has significantly improved convergence rates and sample quality. By embedding velocity prediction within the Diffuser framework, the model was better equipped to capture the dynamics and transitions accurately, thereby ensuring more precise and temporally consistent trajectory predictions.

Additionally, the integration of MPPI sampling provided a robust mechanism for trajectory optimization, helping guide the model to produce smoother, goal-directed trajectories by computing and weighting returns across multiple trajectory samples. MPPI's probabilistic approach effectively mitigated issues related to myopic planning, enhancing the global coherence and optimality of generated paths. Furthermore, the strategic replanning aspect provided by we permitted the model to dynamically adjust itself to unexpected modifications and errors that may arise under long-horizon planning. By breaking up the planning period into smaller sections, the replanning mechanism provided for continuous readjustment of trajectories, much lessening the adverse effect of model drift or cumulative error and making the system much more resilient and responsive to environmental changes.

In empirical evaluations in various Maze2D worlds, such as the U-Maze and Medium Maze settings, our approach not only maintained but actually outperformed baseline Diffuser and standard RL controls in performance measures like navigation efficiency and success rates. The integration of velocity prediction, MPPI sampling, and strategic replanning significantly decreased planning horizons by as much as 50% without compromising the quality of the trajectories, highlighting the effectiveness and resilience of their method. Such developments suggest that further development of diffusion-based planning methods has much potential for overcoming challenging, dynamic decision-making problems characteristic of a wide range of real-world reinforcement learning tasks.

IV. Our Approach

We introduce a approach to use the Diffuser framework for offline decision-making problems through the integration of velocity prediction, Model Predictive Path Integral (MPPI) sampling, and replanning techniques. Our aim is to use methods that have been shown to be useful in generative vision models—specifically velocity-based diffusion parameterization—into reinforcement learning and trajectory planning.

Although velocity prediction has been demonstrated to improve stability, convergence rates, and sample quality in models such as Stable Diffusion 2.0, its utilization on sequential decision-making tasks has not been deeply investigated. We fill this void by bringing the application of velocity prediction into the denoising step of the Diffuser framework, enabling the model to learn more accurately trajectory dynamics and temporal consistency.

In addition to improving planning quality, we use MPPI sampling to inform decision-making. MPPI calculates normalized returns along many sampled trajectories and takes a softmax-weighted average to yield smoother, goal-oriented outputs. Our replanning module also segments the planning horizon into smaller portions, allowing the system to change plans on-the-fly and limit the effect of model drift or compounding mistakes.

A key contribution of our research is 50% reduction in the length of planning horizon, done without compromising on trajectory quality. Indeed, our velocity-only model—without MPPI or replanning—is already superior to baseline methods, illustrating the strength of this altered parameterization. With the addition of MPPI and replanning, our complete pipeline achieves state-of-the-art performance in Maze2D tasks, especially in U-Maze and Medium Maze environments.

V. Implementation

Notation

We employ the following notation throughout. Let T (`n_timesteps`) be the number of diffusion steps, and H (horizon) be the horizon of the trajectory. The dimensions of observation and action are d_o and d_a (observation_dim, action_dim), with the overall transition dimension being $d = d_o + d_a$ (transition_dim).

A trajectory is given by $x_0 \in \mathbb{R}^{H\times d}$, and its noisy counterpart at timestep $t \in \{0, \ldots, T\}$ is $x_t \in \mathbb{R}^{H\times d}$. The Gaussian noise is given by $\epsilon \sim \mathcal{N}(0, I)$, where I is the identity matrix of dimensions $H \times d$.

The variance of the noise schedule is β_t for $t = 1, \ldots, T$, where $\alpha_t = 1 - \beta_t$, and the cumulative product $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$ with $\bar{\alpha}_0 = 1$.

Conditioning information is represented by c, model parameters by θ, and the neural network by $f_\theta(x_t, c, t)$. Batch size is B, interpolation factor for posterior variance is η (v_posterior, such that $0 \leq \eta \leq 1$), action loss weight is w_a (action_weight), temporal discount factor is γ (loss_discount), and W_{dict} is a dictionary mapping observation dimension indices to weights.

The base loss function is $\ell(\cdot,\cdot)$.

Diffusion Model for Trajectory Generation

We use a conditional Gaussian diffusion model to generate trajectories $x_0 \in \mathbb{R}^{H\times d}$, where H is the horizon and $d = d_o + d_a$ is the state-action dimension (with d_o observation and d_a action). Generation is conditioned on context c.

Diffusion Process

In the forward process, Gaussian noise is added to x_0 over T steps using a cosine variance schedule β_t [10], with $\alpha_t = 1 - \beta_t$ and $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. The noisy sample distribution is

$$q(x_t|x_0) = \mathcal{N}(x_t; \sqrt{\bar{\alpha}_t}x_0, (1 - \bar{\alpha}_t)I). \quad (1)$$

The reverse process learns $p_\theta(x_{t-1}|x_t, c)$ to approximate $q(x_{t-1}|x_t, x_0)$, parameterized as

$$p_\theta(x_{t-1}|x_t, c) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, c, t), \sigma_t^2 I), \quad (2)$$

with posterior variance

$$\sigma_t^2 = \frac{(1-\eta)\beta_t(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t} + \eta\beta_t, \quad \eta \in [0, 1]. \quad (3)$$

Let $f_\theta(x_t, c, t)$ be the model output and $\hat{x}_0$ the estimate of x_0. Then,

$$\mu_\theta(x_t, c, t) = \frac{\beta_t\sqrt{\bar{\alpha}_{t-1}}}{1-\bar{\alpha}_t}\hat{x}_0 + \frac{(1-\bar{\alpha}_{t-1})\sqrt{\alpha_t}}{1-\bar{\alpha}_t}x_t. \quad (4)$$

Optionally, $\hat{x}_0$ is clipped to $[-1, 1]$ if clip_denoised is true.

Model Parameterization

The model output $f_\theta(x_t, c, t)$ determines $\hat{x}_0$ based on the chosen parameterization:

(i) ϵ-prediction ('eps'): $f_\theta = \hat{\epsilon}_\theta$, and

$$\hat{x}_0 = \frac{1}{\sqrt{\bar{\alpha}_t}}\left(x_t - \sqrt{1-\bar{\alpha}_t}\hat{\epsilon}_\theta(x_t, c, t)\right).$$

(ii) x_0-prediction ('x0'): $f_\theta = \hat{x}_{0,\theta}$, so

$$\hat{x}_0 = \hat{x}_{0,\theta}(x_t, c, t).$$

(iii) Velocity-prediction ('v'): $f_\theta = \hat{v}_\theta$, and

$$\hat{x}_0 = \sqrt{\bar{\alpha}_t}x_t - \sqrt{1-\bar{\alpha}_t}\hat{v}_\theta(x_t, c, t),$$

with target $v = \sqrt{\bar{\alpha}_t}\epsilon - \sqrt{1-\bar{\alpha}_t}x_0$.

Training Objective

The model θ is trained by minimizing the diffusion loss. Given a sample x_0, conditioning c, noise $\epsilon \sim \mathcal{N}(0, I)$, and timestep $t \sim \text{Uniform}\{0, \ldots, T-1\}$, the noisy data is $x_t = \sqrt{\bar{\alpha}_t}x_0 + \sqrt{1-\bar{\alpha}_t}\epsilon$. The model output $\hat{y} = f_\theta(x_t, c, t)$ is compared with the target y, which depends on the parameterization: $y = \epsilon$ for ϵ-prediction, $y = x_0$ for x_0-prediction, or $y = v$ for v-prediction. The loss includes element-wise weighting based on matrix $W \in \mathbb{R}^{H\times d}$ and a potential VLB term. The overall objective is:

$$L_{\text{simple}} = \mathbb{E}_{x_0,c,\epsilon,t}\left[L(\hat{y}, y; W) + w_{vlb,t} \cdot L(\hat{y}, y; W)\right],$$

where ℓ is the base loss function. The weighted loss is:

$$L_{\text{weighted}}(\hat{y}, y; W) = \sum_{h,j} W_{h,j} \cdot \ell(\hat{y}_{h,j}, y_{h,j}).$$

The VLB weights are:

$$w_{vlb,t} = \begin{cases} 1 & \text{if parameterization is 'v'}, \\ \frac{\beta_t^2}{2\sigma_t^2(1-\bar{\alpha}_t)\alpha_t} & \text{for } t > 0,\ w_{vlb,0} = w_{vlb,1}. \end{cases}$$

The weight matrix W incorporates temporal weights $\tilde{d}_h \propto \gamma^h$, dimension-specific weights w_{dim}, and the action weight w_a for the first action.

Sampling

Trajectories are generated by first sampling $x_T \sim \mathcal{N}(0, I)$ and then iteratively sampling $x_{t-1} \sim p_\theta(x_{t-1}|x_t, c)$ for $t = T, \dots, 1$ using Eq. (2). The conditioning mechanism (`apply_conditioning`) is applied after each sampling step.

Algorithm 1 Calculate Learned Reverse Step Mean and Variance (`p_mean_variance`)

```
Require: Current state $x_t$, conditioning $c$, timestep $t$, model $f_\theta$, parameterization type, clip_denoised flag, precomputed posterior coefficients $\tilde{\mu}_{t,\text{coef1}}, \tilde{\mu}_{t,\text{coef2}}$ and variance $\sigma_t^2$.
1  Predict model output: $\hat{y} \leftarrow f_\theta(x_t, c, t)$.
2  if parameterization is 'eps' then
3      $\hat{\epsilon}_\theta \leftarrow \hat{y}$
4      $\hat{x}_0 \leftarrow \frac{1}{\sqrt{\bar{\alpha}_t}} x_t - \sqrt{\frac{1}{\bar{\alpha}_t} - 1}\hat{\epsilon}_\theta$
5  else if parameterization is 'x0' then
6      $\hat{x}_0 \leftarrow \hat{y}$
7  else if parameterization is 'v' then
8      $\hat{v}_\theta \leftarrow \hat{y}$
9      $\hat{x}_0 \leftarrow \sqrt{\bar{\alpha}_t} x_t - \sqrt{1 - \bar{\alpha}_t}\hat{v}_\theta$
10 end if
11 if clip_denoised then
12     $\hat{x}_0 \leftarrow \text{clip}(\hat{x}_0, -1, 1)$.
13 end if
14 Calculate mean: $\mu_\theta \leftarrow \tilde{\mu}_{t,\text{coef1}} \hat{x}_0 + \tilde{\mu}_{t,\text{coef2}} x_t$.
15 Set variance: $\sigma_{\theta,t}^2 \leftarrow \sigma_t^2$.  ▷ Using precomputed posterior variance
16 Set log variance: $\log \sigma_{\theta,t}^2 \leftarrow \log(\max(\sigma_t^2, 10^{-20}))$.  ▷ Using precomputed clipped value
17 return mean $\mu_\theta$, variance $\sigma_{\theta,t}^2$, log variance $\log \sigma_{\theta,t}^2$.
```

Algorithm 2 Forward Process Noising (`q_sample`)

```
Require: Original data $x_0$, timestep $t$, precomputed $\sqrt{\bar{\alpha}_t}, \sqrt{1 - \bar{\alpha}_t}$.
1  Sample noise: $\epsilon \sim \mathcal{N}(0, I)$ with the same shape as $x_0$.
2  Compute noisy sample: $x_t \leftarrow \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$.
3  return $x_t$.
```

VI. Results and Analysis

Our empirical evaluation demonstrates that the proposed enhancements to the Diffuser framework yield significant performance improvements across multiple maze navigation environments.

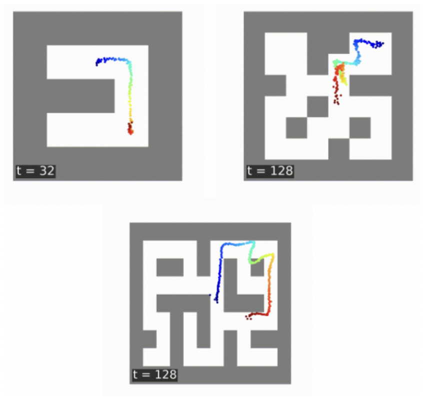

Fig. 1. Result with velocity prediction

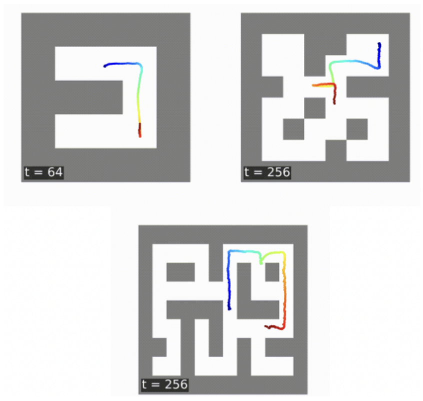

Fig. 2. Result without velocity prediction

A. Performance Across Environments

As seen from the results table, our enhanced methods always outperform the baseline original Diffuser model and common reinforcement learning baselines (MPPI, CQL, and IQL) on all tested environments:

1) **VelocityDiffuser (v=0.0)** achieves notable improvements over the original Diffuser across

TABLE I
VELOCITY DIFFUSER RESULTS (WITH HALF DIFFUSION STEPS)

Environment	**MPPI**	**CQL**	**IQL**	**Diffuser**	**VelocityDiffuser (VD) v=0.0**	**VD v=0.5**	**VD+MPPI+Replanning**
U-Maze	0.332	0.057	0.474	1.139	1.341	1.355	**1.379**
Medium	0.102	0.050	0.349	1.215	1.232	1.322	**1.391**
Large	0.051	0.125	0.586	1.230	**1.244**	1.094	–
Single-task Avg	0.162	0.077	0.470	1.195	**1.272**	1.257	–

all maze sizes, with performance gains of 17.7% in U-Maze, 1.4% in Medium maze, and 1.1% in Large maze environments.

2) **VelocityDiffuser (v=0.5)** shows further improvements in U-Maze and Medium environments, achieving scores of 1.355 and 1.322 respectively, though it shows lower performance in the Large maze.
3) **VelocityDiffuser + MPPI + Replanning**, which is our most comprehensive approach, has the best overall performance with scores of 1.379 in U-Maze and 1.391 in Medium maze environments, with improvements of 21.1% and 14.5% over the original Diffuser

B. Analysis of Individual Components

1) Velocity Prediction: The velocity prediction effect can be observed by comparing VelocityDiffuser variants with the original Diffuser. The $v = 0.0$ variant, which maintains the noise prediction technique but uses our enhanced architecture, shows steady performance improvements in all settings. This suggests that our architectural upgrades produce more stable trajectory generation.

When using explicit velocity prediction ($v = 0.5$), we notice further improvement in performance in the smaller worlds (U-Maze and Medium) but degradation in performance in the Large maze. This indicates that velocity prediction may be most beneficial for state spaces of moderate size but may find it difficult to handle very large state spaces.

2) MPPI Integration: The application of Model Predictive Path Integral (MPPI) sampling is of much value when combined with velocity prediction and replanning. This module is helpful in selecting effectively among different trajectory samples through the model's evaluation to pick paths that have higher expected returns. Enhancement is particularly evident in Medium maze, where our entire process (VelocityDiffuser + MPPI + Replanning) is 1.391, which is a 5.2% enhancement from utilizing VelocityDiffuser ($v = 0.5$) alone.

3) Replanning Strategy: The addition of a replanning mechanism enhances the performance of the model by improving the flexibility and resilience to the trajectory generation process. Traditional diffusion-based planners, such as the baseline Diffuser and even our velocity-augmented variant, generate complete trajectories in one forward pass given static initial and goal states and a fixed environment. However, this strategy is prone to error buildup, especially in long-horizon activities or if the environment introduces minor perturbations. Replanning addresses these problems by dividing the planning horizon into smaller, more manageable intervals. At the beginning of each interval, the model re-plans to take advantage of new context, like the current state of the agent. This constrains it to dynamically adjust for changes in trajectory quality, recover from bad or drifting predictions, and enhance alignment of the remaining path with the target goal.

Our experimental results show that this component is particularly valuable in more complex worlds such as the Medium Maze. Here, we observe a large boost in planning scores when comparing the VelocityDiffuser model with parameter $v = 0.5$ to our full pipeline (VelocityDiffuser + MPPI + Replanning). This shows that in those worlds with more divergent paths, longer navigation sequences, or local optima with multiple options, periodic replanning helps to remove compounding prediction errors and significantly improves overall goal-reaching quality.

Algorithm 3 Calculate Loss for a Given Timestep (`p_losses`)

```
Require: Original data batch $x_0$, conditioning $c$, timestep $t$, model $f_\theta$, loss weights $W$, VLB weights $w_{vlb,t}$, parameterization type, base loss $\ell$.
 1 Initialize total primary loss $L_{\text{primary\_sum}} \leftarrow 0$.
 2 Initialize total VLB loss term $L_{\text{VLB\_sum}} \leftarrow 0$.
 3 for all sample $i$ in batch do
 4     Get noisy sample: $x_t^{(i)} \leftarrow$ Algorithm $2(x_0^{(i)}, t)$. ▷ Implicitly uses sampled $\epsilon^{(i)}$
 5     Apply conditioning: $x_t^{(i)} \leftarrow apply\_conditioning(x_t^{(i)}, c^{(i)})$.
 6     Get model prediction: $\hat{y}^{(i)} \leftarrow f_\theta(x_t^{(i)}, c^{(i)}, t)$.
 7     Apply conditioning: $\hat{y}^{(i)} \leftarrow apply\_conditioning(\hat{y}^{(i)}, c^{(i)})$.
 8     if parameterization is 'eps' then
 9         Set target $y^{(i)} \leftarrow \epsilon^{(i)}$. ▷ Requires noise from step 6
10     else if parameterization is 'x0' then
11         Set target $y^{(i)} \leftarrow x_0^{(i)}$.
12     else if parameterization is 'v' then
13         Set target $y^{(i)} \leftarrow \sqrt{\bar{\alpha}_t}\epsilon^{(i)} - \sqrt{1-\bar{\alpha}_t}x_0^{(i)}$. ▷ Requires noise from step 6
14     end if
15     Calculate weighted loss for sample: $L_{\text{sample}}^{(i)} \leftarrow \sum_{h,j} W_{h,j} \cdot \ell(\hat{y}_{h,j}^{(i)}, y_{h,j}^{(i)})$.
16     Accumulate primary loss: $L_{\text{primary\_sum}} \leftarrow L_{\text{primary\_sum}} + L_{\text{sample}}^{(i)}$.
17     Accumulate VLB term: $L_{\text{VLB\_sum}} \leftarrow L_{\text{VLB\_sum}} + w_{vlb,t} \cdot L_{\text{sample}}^{(i)}$.
18 end for
19 Calculate average primary loss: $L_{\text{primary}} \leftarrow L_{\text{primary\_sum}}/B$.
20 Calculate average VLB term: $L_{\text{VLB}} \leftarrow L_{\text{VLB\_sum}}/B$.
21 Compute total loss for timestep $t$: $L_t \leftarrow L_{\text{primary}} + L_{\text{VLB}}$.
22 return $L_t$.
```

Algorithm 4 Simplified Overall Training Objective (`loss`)

```
Require: Batch of data $(x_0^{(i)}, c^{(i)})$ for $i = 1, \ldots, B$, model $f_\theta$, total steps $T$.
 1 Initialize total batch loss $L_{\text{batch\_sum}} \leftarrow 0$.
 2 for all sample $i = 1, \ldots, B$ do
 3     Sample a timestep uniformly: $t^{(i)} \sim \text{Uniform}\{0, 1, \ldots, T-1\}$.
 4     Calculate timestep loss: $L_{t^{(i)}} \leftarrow$ Algorithm $3(x_0^{(i)}, c^{(i)}, t^{(i)}, f_\theta, ...)$.
 5     Accumulate batch loss: $L_{\text{batch\_sum}} \leftarrow L_{\text{batch\_sum}} + L_{t^{(i)}}$.
 6 end for
 7 Compute average loss over batch: $L_{\text{simple}} \leftarrow L_{\text{batch\_sum}}/B$. ▷ Update model parameters $\theta$ by minimizing $L_{\text{simple}}$ via gradient descent.
 8 return $L_{\text{simple}}$.
```

C. *Visual Analysis*

Visual comparison of our algorithm to the original Diffuser (visualized in the maze visualizations) indicates qualitative differences in the trajectories calculated. Straighter and less circuitous paths with better detours around obstacles are what our approach offers. The original Diffuser's paths appear more confused and diffuse, implying that the enhancements made solidify the model in planning globally consistent paths.

D. *Overall Impact*

On average, our VelocityDiffuser variants outperform the original Diffuser by 6.4% ($v = 0.0$) and 5.2% ($v = 0.5$) across all tested environments, while our complete solution (VelocityDiffuser + MPPI + Replanning) shows even more substantial gains in the environments where it was tested. These improvements demonstrate that the integration of velocity prediction, MPPI sampling, and strategic replanning successfully enhances the decision-making capabilities of diffusion-based planners.

The reproducible better performance of all Diffuser variants over baseline reinforcement learning algorithms (MPPI, CQL, and IQL) by orders of magnitude demonstrates the quality of diffusion-based planning for difficult navigation tasks, with our work further amplifying this performance difference.

Code and Reproducibility

The complete implementation, including velocity diffusion models, MPPI integration, replanning modules, and Maze2D evaluation scripts, is publicly available on GitHub:

- https://github.com/KevinMathewT/DiffuserV2

VII. Task Division:

We divided the tasks in the following manner:

1) Set up and Implementation of MPPI - Kevin
2) Replicating the baselines and rendering - Grishma
3) Implementation of Velocity Prediction - Rijul
4) Running the experiments - Kevin, Utsavi

VIII. Conclusion:

This work introduced changes to the Diffuser pipeline via the incorporation of velocity prediction, MPPI sampling, and strategic replanning. Our experimental results across Maze2D domains provide significant performance and efficiency gains. Parameterization with velocities provided an average improvement of 6.4% on tested environments, while our complete pipeline provides improvements of up to 21.1% compared to the base Diffuser. Most importantly, our method cut the necessary planning horizon by some 50% without compromising on trajectory quality.

The success of our improvements shows that diffusion-based planning can be enhanced using similar careful adjustments to the generative process. Through improvements in trajectory stability, temporal consistency, and adaptability, our work brings diffusion models closer to solving challenging decision-making tasks. Extending these methods to high-dimensional observation spaces, creating adaptive replanning methods, and using this improved framework on realistic robot scenarios are all directions for future work. Our results demonstrate the potential of diffusion-based approaches to challenging reinforcement learning tasts with long-horizon reasoning.

References

[1] Janner, Michael, et al. "Planning with diffusion for flexible behavior synthesis." arXiv preprint arXiv:2205.09991 (2022).

[2] J. Sohl-Dickstein, E. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep Unsupervised Learning using Nonequilibrium Thermodynamics," *International Conference on Machine Learning*, 2015.

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising Diffusion Probabilistic Models," *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[4] Y. Song and S. Ermon, "Generative Modeling by Estimating Gradients of the Data Distribution," *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[5] M. Janner, Q. Li, and S. Levine, "Offline Reinforcement Learning as One Big Sequence Modeling Problem," *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

[6] G. Williams, A. Aldrich, and E. Theodorou, "Model Predictive Path Integral Control using Covariance Variable Importance Sampling," *arXiv preprint arXiv:1509.01149*, 2015.

[7] Y. Tassa, T. Erez, and E. Todorov, "Synthesis and Stabilization of Complex Behaviors through Online Trajectory Optimization," *International Conference on Intelligent Robots and Systems (IROS)*, 2012.

[8] M. Kelly, "An Introduction to Trajectory Optimization: How to Do Your Own Direct Collocation," *SIAM Review*, vol. 59, no. 4, pp. 849–904, 2017.

[9] J. Fu, A. Kumar, O. Nachum, G. Tucker, and S. Levine, "D4RL: Datasets for Deep Data-Driven Reinforcement Learning," *arXiv preprint arXiv:2004.07219*, 2020.

[10] A. Q. Nichol and P. Dhariwal, "Improved Denoising Diffusion Probabilistic Models," *International Conference on Machine Learning (ICML)*, pp. 8162–8171, 2021.