Trajectory Diffusion for ObjectGoal Navigation

Xinyao Yu[1,2*], **Sixian Zhang**[1,2*] **Xinhang Song**[1,2], **Xiaorong Qin**[1,2], **Shuqiang Jiang**[1,2,3]
[1]Key Lab of Intelligent Information Processing Laboratory of the Chinese Academy of Sciences (CAS),
Institute of Computing Technology, Beijing, [2]University of Chinese Academy of Sciences, Beijing
[3] Institute of Intelligent Computing Technology, Suzhou, CAS
{xinyao.yu, sixian.zhang, xinhang.song, xiaorong.qin}@vipl.ict.ac.cn
sqjiang@ict.ac.cn

Abstract

Object goal navigation requires an agent to navigate to a specified object in an unseen environment based on visual observations and user-specified goals. Human decision-making in navigation is sequential, planning a most likely sequence of actions toward the goal. However, existing ObjectNav methods, both end-to-end learning methods and modular methods, rely on single-step planning. They output the next action based on the current model input, which easily overlooks temporal consistency and leads to myopic planning. To this end, we aim to learn sequence planning for ObjectNav. Specifically, we propose trajectory diffusion to learn the distribution of trajectory sequences conditioned on the current observation and the goal. We utilize DDPM and automatically collected optimal trajectory segments to train the trajectory diffusion. Once the trajectory diffusion model is trained, it can generate a temporally coherent sequence of future trajectory for agent based on its current observations. Experimental results on the Gibson and MP3D datasets demonstrate that the generated trajectories effectively guide the agent, resulting in more accurate and efficient navigation. The code is available at `https://github.com/sx-zhang/T-diff.git`.

1 Introduction

Embodied AI aims to develop agents with a comprehensive understanding of their environment, capable of interacting with humans, other agents and entities in real physical environments. As a fundamental task of embodied AI, visual object goal navigation (ObjectNav) task involves placing an agent in an unseen environment and tasking it to navigate to a user-specified object (e.g., 'find a toilet') based on visual sensory input. To efficiently complete the ObjectNav task, the agent needs to construct an information-rich memory to store **past** experiences (i.e., what it has previously seen) to avoid redundant searching. Additionally, it needs to learn a planner to plan the **future** (i.e., determine the most efficient sequence of actions to navigate to the target) to avoid unnecessary exploration.

Humans are innately smart navigators, as they encode both short-term memories from current navigation and long-term memories from daily life into a cognitive map. This map records detailed information about the semantics, positions, and relationships within spatial environments [40]. In practice, the human decision-making process is understood as finding the most possible sequence of releases based on cognitive information [41]. Human planning is a sequential process that ensures temporal consistency and global optimality. However, prior methods of ObjectNav employ single-step planning rather than sequential planning for navigation. As illustrated in Fig. 1, prior end-to-end learning methods [66, 45, 62, 28, 18, 33, 34] formulate their planners as end-to-end networks and train them using reinforcement learning (RL) [66, 45, 62, 28, 18] or imitation learning (IL)

*These authors contributed equally to this work.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

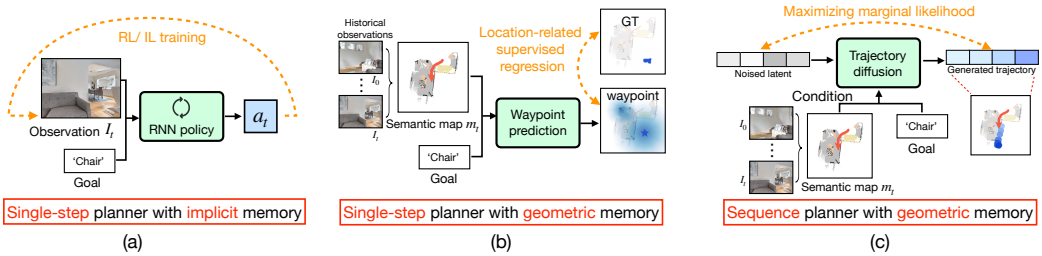

Figure 1: Different planners (denoted as green modules) in existing methods of ObjectNav. (a) represents end-to-end learning methods, and (b) refers to modular methods. (c) denotes the proposed trajectory diffusion, which plans future sequence trajectory based on geometric semantic map.

[33, 34]. These planners perform single-step planning at each moment, outputting an action based on the current egocentric view, which leads to a lack of temporal consistency and interpretability. Furthermore, these methods implicitly encode all past observations, leading to a loss of geometric spatial information, thus limiting their generalization in complex environments [28, 63]. Modular methods [4, 31, 65, 59, 64] attempt to address this by constructing a semantic map during navigation to geometrically store historical observations, helping the agent avoid redundant searching. However, their decision-making process is still single-step. They train a waypoint predictor via supervised learning to plan the next subgoal based on the current semantic map and target. Motivated by learning sequential planning, we propose to model the conditional distribution of the trajectory sequence, i.e., learning to plan a sequence of future trajectory conditioned on the current semantic map and target.

Since both the semantic map and trajectory sequence are high-dimensional, the feature space and computational complexity are significantly high. Thus, directly learning this conditional distribution is challenging [1]. Recently, diffusion models have achieved great success in expressing complex distributions [16, 54] and generating high-quality images [37, 35, 32]. Diffusion models gradually add noise to the data during the diffusion process, transforming the data distribution into a simple distribution (e.g., Gaussian distribution). In the reserve process (i.e., denoising process), noise is iteratively predicted and removed, generating complex data distributions from the simple random distribution. This iterative refinement learning manner enhances the stability and controllability of learning high-dimensional data distributions. Therefore, we utilize diffusion models to learn this conditional distribution and propose the trajectory diffusion (T-diff) model as a navigation planner, which performs sequence planning and generates the future trajectory sequence for the agent.

In this paper, we propose the trajectory diffusion for the ObjectNav task. The trajectory diffusion is a sequence planner designed to generate optimal trajectory sequence for an agent based on its historical observations (i.e., semantic maps) and target object. Specifically, we collect optimal trajectories by using precise maps in training rooms. Then, an agent equipped with a semantic map module is driven to follow these trajectories, gathering semantic maps and poses at each timestep along the way. The collected data are further divided into data pairs consisting of a semantic map at a given moment and the corresponding future trajectory segments. Based on the collected data pairs, we employ DDPM [16] to train our trajectory diffusion. Our trajectory diffusion is implemented by modifying Transformer-based diffusion, which takes noised latent as the starting input and iteratively refines them to produce trajectory sequence. Once the trajectory is generated, we drive the agent to move along the predicted trajectory sequence until the target is found. Evaluations on the Gibson [47] and MP3D [3] simulators show our trajectory diffusion model significantly outperforms baselines. Visualization results confirm effective guidance from generated trajectories. Additionally, we further showcase the scalability of our trajectory diffusion model across different simulators.

2 Related Work

ObjectGoal Navigation. Goal-driven navigation [22, 42, 43, 44, 63] is a fundamental task in embodied AI, and this paper focuses on a specific variant of this task, namely the ObjectNav task, where the goal is defined by object semantics. Current works for ObjectNav task can be categorized into two types: end-to-end learning methods and modular methods. End-to-end learning methods develop a navigation policy by interacting with the environment trained by reinforcement learning (RL) [45, 66, 61] or imitation learning (IL) [34, 33]. These approaches typically take inputs such as

target object categories and extra information, including visual representations[20, 29, 18] and object relationship graphs[51, 62, 11, 55]. Then they predict single-step planning in each timestep. The end-to-end method implicitly encodes all past observations, which results in the loss of geographic spatial information. Consequently, its generalization ability in environments with complex layouts is limited. Modular methods [4, 5, 6] typically preserve a top-down geometric map enriched with semantic details. They are also single-step planners that predict waypoints as sub-goals at each time step based on the constructed semantic map. Through supervised learning, PONI[31] learns to predict the nearest frontier with the current semantic map to infer where to explore, while PEANUT[59] directly predicts the target location in the form of a probabilistic goal map. Current end-to-end learning methods are single-step planners with implicit memory, while modular methods are single-step planners with geometric memory. Alternatively, our trajectory diffusion approach is a sequence planner with geometric memory, predicting future sequential trajectories based on the current semantic map. Sequence planning ensures temporal consistency and interpretability of decisions. The geometric memory prevents the agent from redundant exploration.

Diffusion Model. Diffusion models (DDPMs [16]) are generative models that learn complex data distributions by iteratively predicting and removing randomly sampled noise to obtain target samples. Diffusion models have been successfully applied in image-related fields, including image generation[32, 37, 30], super-resolution[36, 46], image inpainting[26, 48], and image editing[2, 39]. Diffusion models predominantly use UNet-based [36, 37] and Transformer-based [30, 32, 46] architectures. UNet excels in preserving spatial details for high-resolution images, while Transformers capture global context, suited for sequential data. Therefore, we adopt a Transformer-based diffusion model to implement our trajectory diffusion. Recently, diffusion models are gradually employed in the field of robotics. several works [8, 58, 25, 13] leverage Data collection from real-world often suffers from scarcity or lacks diversity. As natural data synthesizers, diffusion models are leveraged by several works [8, 58, 25, 13] for data augmentation purposes. Furthermore, methods like Diffuser[17], Crossway Diffusion[21] and Diffusion policy [9] utilize diffusion models to fit multi-modal behavioral data of agents. In line with our work, the proposed Diffusion Trajectory aims to learn the distribution of trajectory sequence conditioned on semantic maps and the goal, primarily to address the ObjectNav task.

3 Preliminaries of ObjectNav

The task of ObjectNav involves navigating an agent to a specific type of object (e.g., 'chair') in unseen environments. At the beginning of each episode, the agent is initialized at a random position. During navigation, at every timestep t, the agent receives egocentric RGB-D images I_t, the target object o, and the senor pose p_t, which includes the spatial coordinates and the direction the agent is facing. The agent performs one of several discrete actions, includeing `move_forward`, `turn_left`, `turn_right`, and `stop`. The action `stop` is autonomously initiated by the agent once it determines to complete the task. An episode is considered successful if the agent stops within a preset number of steps at a spot where the target object is within a specified distance (e.g., less than $1m$) and is visible in the agent's field of view.

Existing works for the ObjectNav task can be categorized into end-to-end learning methods and modular methods, as shown in Fig. 1. The end-to-end learning methods [66, 18, 33, 34] generate single-step plans at each time step, formulated as a policy $\pi(a_t|I_t, o)$, where a_t denotes the action at timestep t. They typically utilize RL or IL to train policy functions $\pi(a_t|I_t, o)$. The training objective of RL is to maximize the expected sum of discounted rewards. Let χ denote a sequence of object, action, reward tuples sampled based on π within an episode. The training objective is $\underset{\pi}{argmax}\mathbb{E}_{\chi\sim\pi}\left[R_\chi\right], R_\chi = \sum_{t=1}\gamma^{t-1}r_t$, where γ is a discount factor, and r_t denotes the reward function, typically implemented as a sparse success reward. This is because dense rewards can inhibit agent's exploration [28], thus impairing its generalization in unseen environments [33]. Sparse rewards are desirable, but they result in most sample trajectories having difficulty obtaining positive rewards, making the learning process challenging. Moreover, as the policy parameters are updated, previously sampled trajectories become obsolete, necessitating the collection of new data. Consequently, the low sample efficiency of end-to-end learning methods results in high computational costs for training. As for IL-based methods [33, 34], the training objective is to minimize the difference between the policy's output and human demonstrations (i.e., behavior cloning), summarized as $\underset{\pi}{argmin}\mathbb{E}_{(I_t^d, a_t^d)\sim\mathcal{D}}[-\log(\pi(a_t^d|I_t^d, o))]$, where $\mathcal{D}$ is a dataset of human

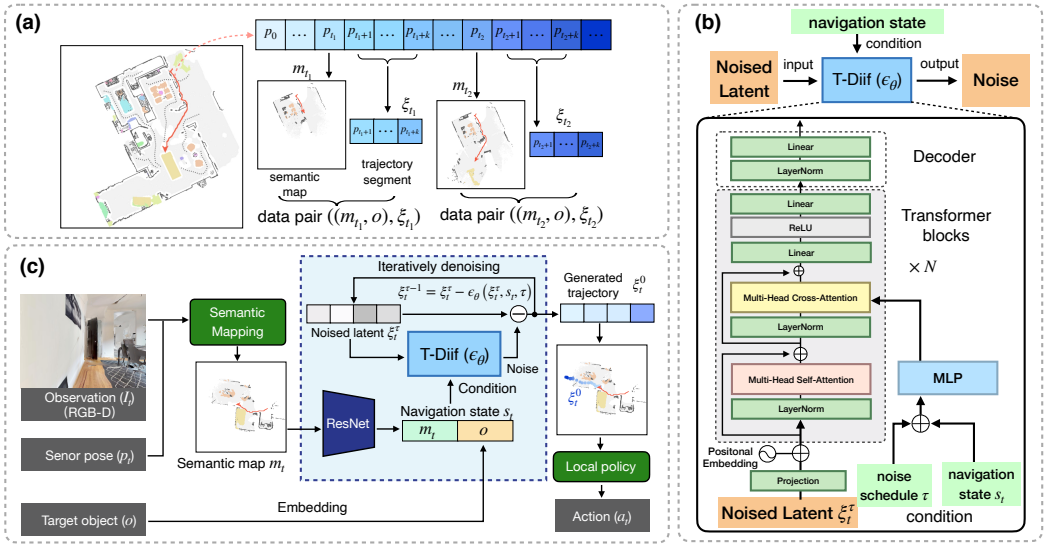

Figure 2: Frameworks of the trajectory diffusion (T-Diff). (a) refers to the process of dividing the collected data into data pairs. (b) shows the implementation structure of T-Diff. (c) illustrates the navigation process guided by trajectories generated by T-Diff.

demonstrations. However, collecting human demonstrations is highly expensive, making IL training costly. Additionally, end-to-end learning methods implicitly encode historical observations, which results in a lack of geometric memory. This also limits the generalization capability of these methods.

Modular methods [4, 31, 65, 59] typically construct a local semantic map (the map module will be detailed in Sec. 4.2) during navigation. The semantic map geometrically stores historical observations, helping the agent avoid redundant exploration. For the navigation planner, they formulate it as $f(\Omega|m_t, o)$, where Ω is a set of points in m_t, e.g., Ω can be defined as frontiers of m_t [31] or unknown regions outside m_t [59]. The module $f(\Omega|m_t, o)$ is trained as a supervised regression task with the training loss $\sum_\Omega \mathcal{L}\left(V\left(\Omega\right), f\left(\Omega|m_t, o\right)\right)$, where $V\left(\Omega\right)$ represents the ground truth, and its values are calculated based on the actual location of the target (i.e. the closer Ω is to the target, the higher value of $V(\Omega)$). Based on the predictions of f, the agent selects the point with the optimal value as a sub-goal to guide the agent. The planner f re-predicts at each timestep, which is also a form of single-step planning. Supervised learning is efficient for training, however, due to the limited room diversity (i.e., current simulators [47, 3, 49] mostly contain fewer than 1000 unique rooms), these location-related supervision constraints lead the learned planner f to overfit to the layouts of the training rooms [63].

In summary, end-to-end learning methods learn a **single-step planner with implicit memory**, and their learning process suffers from sample inefficiency and high training costs. On the other hand, modular methods learn a **single-step planner with geometric memory**, but the generalization of their planner is constrained by location-related supervision. In our work, our trajectory diffusion model is a **sequence planner with geometric memory**, which aims to learn the conditional distribution $p(\tau_t|m_t, o)$, where τ_t represents the planned future sequential trajectory. The sequential plan ensures temporal consistency and interpretability of decisions, while the geometric memory ensures efficient exploration. Furthermore, in terms of model training, we leverage DDPM with automatically collected trajectories to learn this conditional distribution, effectively avoiding sample inefficiency, the necessity for expensive human demonstration collection, and the risk of overfitting location-related information.

4 Trajectory Diffusion

4.1 Diffusion Model

Diffusion models are probabilistic generative models trained to learn data distributions by iteratively denoising variables sampled from Gaussian distributions. The forward process of diffusion models (DDPMs [16]) is defined as the diffusion process, which is implemented via a Markov chain that

gradually applies Gaussian noise to the real data which can be formulated as follows:

$$q\left(x_{1:T}|x_0\right) = \prod_{\tau=1}^{T} q\left(x_\tau|x_{\tau-1}\right),\ q\left(x_\tau|x_{\tau-1}\right) = \mathcal{N}\left(x_\tau; \sqrt{\bar{\alpha}_\tau}, \left(1-\bar{\alpha}_\tau\right)\mathbf{I}\right) \tag{1}$$

where constants $\bar{\alpha}_\tau$ are hyper-parameters. x_0 is the real data, while $x_1, \ldots, x_T$ are noised latent data. The dimension of both latent and real data are the same, i.e., $x_{0:T} \in \mathbb{R}^d$. By applying the reparameterization trick, the noised data can be sampled by $x_\tau = \sqrt{\bar{\alpha}_\tau}x_0 + \sqrt{1-\bar{\alpha}_\tau}\epsilon_\tau$, where $\epsilon_\tau \sim \mathcal{N}(0, \mathbf{I})$. Diffusion models learn the real data distribution by reversing the diffusion Markov chain, denoted as $p(x_{\tau-1}|x_\tau)$, which is referred to as denoising process. Theoretically, this process reduces to predict the noise added to x_τ. The noise prediction network ϵ_θ is trained using the mean-squared error between the predicted noise and the ground truth sampled Gaussian noise ϵ_τ. Additionally, diffusion models can be conditioned on other inputs (e.g., text [2, 30] or image [9, 52]), then the training objective is formulated by

$$\mathcal{L}_\theta = \mathbb{E}_{x,c,\epsilon,\tau}[\|\epsilon_\tau - \epsilon_\theta\left(x_\tau, c, \tau\right)\|_2^2] \tag{2}$$

where c is the embeddings of input condition. Once the model ϵ_θ is trained, real data are iteratively generated starting from random noise.

4.2 Navigating with Trajectory Diffusion

Our trajectory diffusion aims to generate the optimal future trajectory for the ObjectNav agent based on its current state, thereby assisting the agent in efficiently navigating to the target object.

Trajectory collection. To train the trajectory diffusion, we collect a set of data pairs $((m_t, o), \xi_t)$, where m_t is the semantic map at time t, o denotes target object, and $\xi_t \in \mathbb{R}^{2\times k} = [p_{t+1}, \ldots, p_{t+k}]$ represents the trajectory segment, defined as the concatenation of the senor poses from time $t+1$ to $t+k$. Each sensor pose p_t is recorded in 2 dimensions representing the 2D coordinates. To obtain efficient trajectories and corresponding semantic maps, we first randomly initialize a start position in the training room. Then, the Fast Marching Method (FMM) [38] is utilized to compute an optimal path from the start position to a specific target o based on the precise collision maps of the current training room. Note that such precise maps are unavailable during testing since the test rooms are unseen. Subsequently, we drive an agent equipped with semantic mapping module along this optimal path, recording the semantic map m_t at each timestep. Based on the collected data, we sample a series of data pairs for training our trajectory diffusion. Furthermore, the collected trajectories are segmented into sub-trajectories ξ_t of length k. Together with the corresponding semantic maps m_t and target object o, these data pairs $((m_t, o), \xi_t)$ ultimately constitute the training data for our trajectory diffusion, as illustrated in Fig. 2 (a).

Trajectory diffusion model. We utilize DDPM to train our trajectory diffusion to estimate the conditional distribution $p(\xi_t|m_t, o)$. In the diffusion process, we sample noised data by $\xi_t^\tau = \sqrt{\bar{\alpha}_\tau}\xi_t^0 + \sqrt{1-\bar{\alpha}_\tau}\epsilon_\tau$, where $\epsilon_\tau \sim \mathcal{N}(0, \mathbf{I})$ represents Gaussian noise and ξ_t^0 is the real data ξ_t, i.e., the trajectory segments in the collected data pairs $((m_t, o), \xi_t)$. $\bar{\alpha}_\tau$ and τ are hyper-parameters that control the variance schedule. Note that two timesteps (t and τ) are involved here, where t represents a specific moment in navigation, and τ denotes the noise schedule in diffusion or denoising processes. We train the model to predict the added noise, and the training objective is modified from Eq. 2

$$\mathcal{L}_\theta = \mathbb{E}_{\xi,s,\epsilon,\tau,t}[\|\epsilon_\tau - \epsilon_\theta\left(\xi_t^\tau, s_t, \tau\right)\|_2^2] \tag{3}$$

where θ represents the parameter for trajectory diffusion. The navigation state s_t of the current agent acts as the condition for trajectory diffusion. The state s_t is the concatenation of the embedding of the semantic map m_t and the target object o. The semantic map m_t is encoded by a ResNet18 (without pretrained), while the target object o is encoded using linear projection.

Regarding the implementation of trajectory diffusion, since the navigation trajectory ξ_t is a sequence of k tokens, each with a dimension of m, our trajectory diffusion follows DiT [30], which is a diffusion model based on the Transformer architecture. As illustrated in Fig. 2 (b), the noised latent ξ_t^τ is encoded via a linear projection with added positional embeddings, and then processed through a series of transformer blocks. In addition to the noised latent input, trajectory diffusion also conditions on the diffusion timestep τ and the navigation state s_t. The condition information interacts with the encoded noised latent through a multi-head cross-attention layer. After passing through N

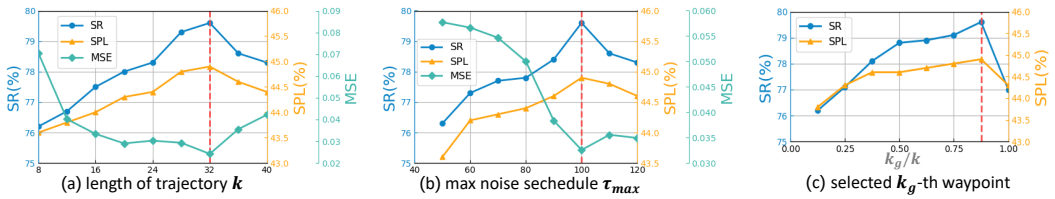

Figure 3: The impact of three hyper-parameters. (a) represents the impact of the length of generated trajectories k during training. (b) reflects the impact of maximum noise schedule τ_{max} in diffusion and denoising process of T-diff. (c) shows the impact of selected proportion of generated trajectory length for navigation performance.

Transformer blocks, we utilize a standard linear decoder to output the noise prediction. Note that the output has a shape equal to the original noised latent.

Navigating with generated trajectory. During navigation, at each timestep t, the agent receives the current egocentric RGB-D view I_t, the current sensor pose p_t, and the target object o as shown in Fig. 2 (c). The **semantic mapping module** aids the agent in constructing an incrementally growing semantic map m_t. At each timestep during navigation, the received RGB-D image is segmented into semantic categories using a pre-trained segmentation model [14]. Subsequently, the depth image is employed to project each pixel, along with its semantic label, into 3D space. Points within a specific height range (relative to the agent's height) are designated as obstacles. Then, This point cloud is transformed into a voxel occupancy grid, which is integrated across the height dimension to create an egocentric map. The egocentric map is then transformed into an allocentric coordinate system based on the agent's pose and is aggregated with the pre-existing global map. Consequently, the semantic map $m_t \in \mathbb{R}^{(4+n)\times h\times w}$ comprises $(n+4)\times h\times w$ elements, where n denotes the number of semantic categories, and h, w are the map size. Channels 1 and 2 depict the obstacle map and the explored regions, respectively. Channels 3 and 4 represent the agent's current position and all previous positions.

The embeddings of the semantic map m_t and target object o are concatenated to form the navigation state s_t, serving as generation condition. The trained trajectory diffusion model ϵ_θ begins with an initialized Gaussian noise $\xi_t^{\tau_{max}}$ as the initial input, and predicts the noise $\epsilon_\theta\left(\xi_t^\tau, s_t, \tau\right)$ contained within the noised latent ξ_t^τ. Then, the noised latent trajectory is denoised by $\xi_t^{\tau-1} = \xi_t^\tau - \epsilon_\theta\left(\xi_t^\tau, s_t, \tau\right)$. This denoising process is repeated for τ_{max} steps, iteratively generated the final trajectory ξ_t^0.

Once the generated trajectory ξ_t^0 (i.e., ξ_t) is obtained, a local policy [4, 6] is employed to drive the agent along this trajectory. To prevent the agent from encountering unreachable points (i.e., obstacles) in the generated trajectory, we select the k_g-th waypoint of ξ_t as the navigation goal, where k_g is a hyper-parameter discussed in Sec. 5.2. The local policy converts the navigation goal into low-level actions by computing a collision-free path using the FFM method based on the obstacle channel from the semantic map m_t. It then determines deterministic actions according to the agent's step distance to navigate agent towards the navigation goal (i.e., k_g-th waypoint of ξ_t). The trajectory diffusion generates a new trajectory every t_{T-diff} step, while the local policy replans deterministic action at each step of navigation.

5 Experiments

5.1 Experimental Setup

Dataset. We evaluate the performance of our model on standard ObjectNav datasets, including Gibson [47] and Matterport3D (MP3D) [3] , in the Habitat simulator. For Gibson, we use 25 train / 5 val scenes from the tiny-split, following the settings of [31], with 1000 validation episodes containing 6 target object categories. For MP3D, we utilize 56 train / 11 val scenes, with 2195 validation episodes containing 21 target object categories. The detailed goal categories are mentioned in Appendix.

Implementation Details. For the training of trajectory diffusion model, we sample 84k and 465k data pairs from the training scenes in Gibson and MP3D (85% train / 15% val), respectively. The semantic maps are resized to 224 $\times$ 224. We implement the trajectory diffusion model based on the DiT [30] structure. Additionally, the semantic map in condition information is encoded by

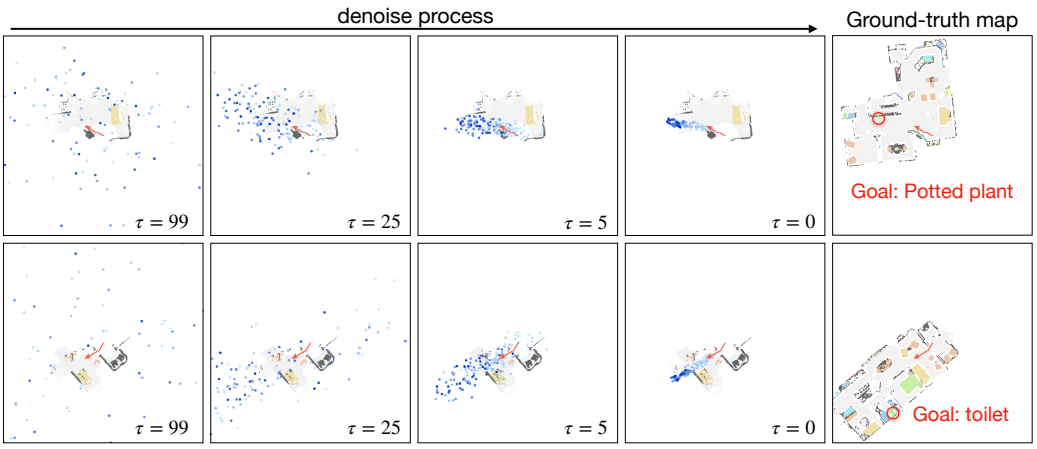

Figure 4: Visualization of trajectory generation. Each row in the figure overlays the results of five repeated experiments, where the same semantic map and target are used as conditions but different random initialization noises are used as inputs. The color intensity of trajectory points represents temporal order, with lighter colors indicating points closer in time to the current navigation timestep.

a ResNet-18 [15] with the first convolutional layer's input channel adjusted to accommodate the dimension of the semantic map. The diffusion process contains $N = 8$ Transformer blocks. Training is performed using AdamW optimizer[19, 24] with a base learning rate of 1e-4, warmed up for 1000 steps using linear warmup and cosine schedule. After the warmup steps, the learning rate for the diffusion model is decayed by a factor of 1e-3, and the learning rate of the semantic map encoder is decayed by a factor of 1e-6. Each model is trained for 200 epochs. Following some common generative methods, exponential moving average(EMA) is used with a maximum decay of 0.9999 during training. All reported results are obtained using EMA model. The maximum noise schedule τ_{max} is set to 100. The length of the predicted trajectory $k = 32$ and the selected k_g-th point is set to 28. The experiments on these three hyper-parameters are detailed in Sec. 5.2. The agent's turn angle is fixed at 30 degrees and each $Forward$ step distance is 25 cm. The maximum timestep limit is set to 500 during navigation and t_{T-diff} is set to 5. Note that, since the navigation performance of multiple repeated experiments does not show significant differences, error bars are not reported.

Evaluation metrics. Three standard metrics are utilized to quantify the performance of the model in ObjectNav task, following [31, 59, 4]. **SR** (Success Rate) indicates the proportion of success episodes. **SPL** (Success weighted by Path Length) represents the success rate of episodes weighted by path length, thereby reflecting the efficiency of the agent's path relative to the shortest path. **DTS** (Distane To Goal) denotes the distance of the agent from the goal when the episode ends. In addition, we use **MSE** (Mean Squared Error), which reflects the distance between the denoised generated trajectory and the real trajectory, to assess generation accuracy of our T-Diff.

5.2 Evaluation Results

Hyper-parameter tuning of T-Diff. Hyper-parameter k determines the length of generated trajectory. We evaluate the impact of parameter k on the generated trajectory (on MSE metric) and navigation performance (on SR and SPL metrics), as shown in Fig. 3 (a). The results indicate that both overly short and overly long generated trajectories are suboptimal. We infer that when the trajectory length is too short, original sequence-planning gradually turns into step-planning, which undermines the temporal consistency of the planning. Conversely, if the trajectory is too long (i.e., greater than 32), the distribution $p(\xi_t|m_t, o)$ becomes more complex, making the learning process more difficult. Based on experimental results, we determine that $k = 32$ is the optimal value.

The hyper-parameter τ_{max} represents the maximum noise schedule, determining the upper limit of iterations in both the diffusion and denoising processes. As evaluation results shown in Fig. 3 (b), the noise schedule τ_{max} is empirically set to 100.

Visualization of trajectory generation. We visualize the iterative trajectory generation process, as shown in Fig. 4. Initially, the trajectories are initialized as random coordinate points. As the denoising process progresses, these points gradually converge, forming the final trajectory at $\tau = 0$. Note that

Table 1: Ablation study on minimal trajectory length for T-Diff training in Gibson (val). When the trajectory length is 1 (rows 2-6), T-Diff is trained to directly predict a waypoint. $i\text{-}th$ denotes that the training ground truth is the point at the i-th step from the current position on the optimal trajectory.

ID	Method	Trajectory			Navigation (Gibson)		
		Length	$i\text{-}th$	MSE ↓	SR(%) ↑	SPL(%) ↑	DTS(m) ↓
1	Single-step (PONI)	-	-	-	73.6	41	1.25
2	T-Diff	1	1	0.2247	71.0	37.6	1.49
3	T-Diff	1	8	0.2253	72.8	40.4	1.44
4	T-Diff	1	16	0.2308	72.6	40.8	1.45
5	T-Diff	1	24	0.2291	74.2	42.1	1.39
6	T-Diff	1	32	0.2456	73.8	41.3	1.40
7	T-Diff	4	1	0.1042	75.9	43.1	1.32
8	T-Diff	32	1	0.0357	79.6	44.9	1.00

the visualization is conducted on the validation set of Gibson, where scene layout is unknown to our trajectory diffusion method. Despite this, the trajectory diffusion is still capable of generating the most efficient path to the target, as indicated by the goal position marked on the ground-truth map.

Ablation study on minimal trajectory length for T-Diff training. We conduct an ablation study using minimal trajectory lengths (e.g., 1 and 4) for training T-Diff, as shown in Tab. 1. Note that when the length is set to 1, the prediction of T-Diff is a single waypoint. The results indicate that when the length is set to 1, T-Diff's performance is influenced by the choice of ground truth point (i.e., the i-th point from current position on the optimal trajectory). The performance with shorter lengths (1 or 4) is lower compared to longer lengths (32).

We hypothesize that predicting a sequence of trajectories, as opposed to a single point, allows each predicted point to receive contextual information from neighboring points. This helps correct and smooth out prediction errors of individual points, reducing the sensitivity of the results to single-point errors. Consequently, this ensures more stable predictions and enhances overall accuracy of trajectory prediction. This finding further supports our motivation for using sequence planning.

Evaluations of T-Diff variants. We compare different variants of T-Diff as shown in Tab. 2, where rows 2-4 represent different variants of T-Diff (i.e., sequence planner with geometric memory). The comparison in row 1 uses an enhanced FBE method (proposed by PONI [31]) combined with a local policy (i.e., single-step planner with geometric memory). The 'X' marks in row 1 indicate that T-Diff is not used, but this alternative still employs semantic map and goal for navigation. Row 0 refers to the case where navigation process does not use any map or goal. The comparison between row 1 and T-Diff variants (rows 2-4) demonstrates that sequence planning achieves superior navigation performance. Moreover, compared to different T-Diff variants, row 2 represents the variant that utilizes only a single timestep observation I_t for trajectory generation. Rows 3 and 4 represent variants that use a semantic map m_t encompassing all historical observations as conditions for trajectory generation. Comparing rows 2 and 3, the results indicate that using m_t not only improves trajectory generation but also enhances navigation performance. We hypothesize that navigation is a sequential decision-making task, and relying solely on current single-step observations can lead to suboptimal decisions due to a lack of temporal consistency, thereby reducing overall navigation efficiency.

Table 2: Comparison with different variants of T-Diff on Gibson (val). I_t means using RGB information as condition while m_t refers to employing semantic map as condition. Here, LP denotes local policy, and FBE corresponds to the area potential function proposed by [31].

ID	Method	T-Diff variants	Trajectory	Navigation (Gibson)		
			MSE ↓	SR(%) ↑	SPL(%) ↑	DTS(m) ↓
0	Random policy	✗	-	0.4	0.4	3.89
1	FBE+LP	✗	-	72.3	38.5	1.32
2	T-Diff+LP	Visual(I_t)	0.2058	73.5	41.6	1.23
3	T-Diff+LP	Visual (m_t)	0.0546	76.9	44.1	1.08
4	T-Diff+LP	Visual (m_t)+Goal	0.0357	79.6	44.9	1.00

Furthermore, comparing rows 3 and 4, the inclusion of target object improves navigation accuracy, demonstrating that the agent's trajectory is goal-driven rather than aimless. Finally, compared to the baseline (comparing rows 1 and 4), integrating our trajectory diffusion method improves navigation performance by 7.3% in SR, 6.4% in SPL, and reduces DTS by 0.32m. These results validate the effectiveness of the proposed trajectory diffusion.

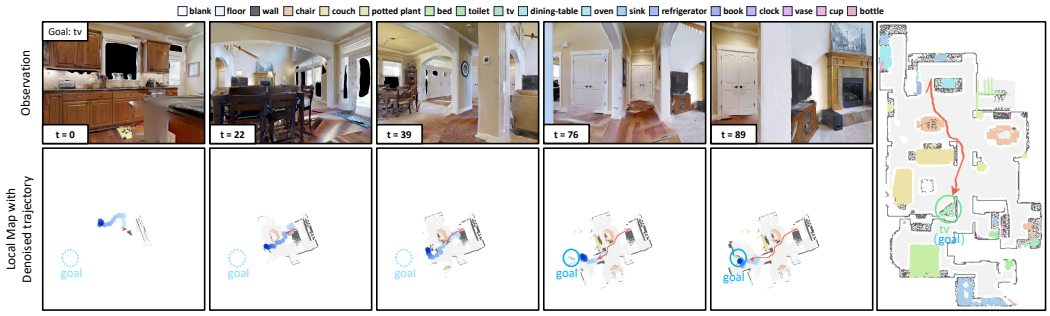

Figure 5: Visualization of navigating with generated trajectory on Gibson (val). The top row shows the agent's first-person RGB observation and the bottom row displays the local semantic map along with generated trajectory by T-Diff. The right figure shows the ground truth map with the actual trajectory of the agent and the target location marked. The generated trajectory effectively guides the agent towards the target, even when the target remains unseen.

Comparisons with simpler model for trajectory generation. We consider the following simple decoder to learn the trajectory generation $P(\tau_t|m_t, o)$ as a comparison. It adopts a similar Transformer-based architecture to T-Diff, with comparable parameters and the same conditional inputs. However, unlike T-Diff, which is trained using DDPM, this competitor is trained with MSE loss. The results are shown in the Tab. 3. The results indicate that directly learning $P(\tau_t|m_t, o)$ through supervised training yields poor performance. Our analysis suggests that since both m_t and τ_t are high-dimensional, the target distribution $P(\tau_t|m_t, o)$ is also high-dimensional. Given the limited number of training rooms (less than 100), $P(\tau_t|m_t, o)$ is sparse and difficult to learn directly. In contrast, the diffusion model (DDPM), through its diffusion and denoising process, gradually simplifies complex distribution into multiple simpler distributions. This allows for better learning of $P(\tau_t|m_t, o)$ distribution. Therefore, our experiments and analysis confirm the necessity of using diffusion models for learning trajectory generation.

Table 3: Comparisons with simpler model for trajectory generation. MSE measures the quality of generated trajectories, while SR, SPL, and DTS indicate navigation performance.

	MSE ↓	SR(%) ↑	SPL(%) ↑	DTS(m) ↓
Simple decoder	0.6541	59.2	33.5	2.05
T-diff (Ours)	0.0357	79.6	44.9	1.00

Scalability. We compare the performance of our trajectory diffusion method with the modular method (i.e., PONI [31]) across different simulators, as shown in Tab. 4. When training and testing are performed on the same simulator, despite the testing rooms being unseen, the layouts of the training and testing rooms are still similar. In this scenario, the modular method, which uses location-related information for supervision, achieves relatively good performance, as shown in line 1 of Tab. 4. However, when the training and testing rooms come from different simulators, the performance of the modular method significantly deteriorates (line 2). In contrast, our trajectory diffusion method demonstrates better scalability. Even when the training and testing rooms come from different simulators, it maintains good navigation performance (compare lines 3 and 4).

Table 4: Comparisons of navigation performance across different training and testing simulators.

ID	Method	Train	Test	SR(%) ↑	SPL(%) ↑	DTS(m) ↓
1	PONI [31]	Gibson	Gibson	73.6	41.0	1.25
2	PONI [31]	MP3D	Gibson	43.9	26.3	2.56
3	T-Diff (Ours)	Gibson	Gibson	79.6	44.9	1.00
4	T-Diff (Ours)	MP3D	Gibson	78.2	45.2	1.07

Navigation with trajectory diffusion. The generated trajectory ξ_t represents the future location points of the agent from time $t+1$ to $t+k$. As mentioned in Sec.4.2, instead of directly using the first points on the generated trajectory as the waypoint to guide the agent, we select the k_g-th point as the guidance.The impact of the hyper-parameter k_g is evaluated as shown in Fig. 3 (c), where the horizontal axis represents k_g/k. The results indicate that the optimal value is achieved when $k_g/k = 0.875$, i.e., when $k = 32$, $k_g = 28$. We attribute this to the fact that, while the generated trajectory follows the correct overall trend, it still fluctuates within a certain range, as shown in Fig. 4. Therefore, if the selected point is too close to the current coordinate, the planning of near-term actions is more frequently affected by these fluctuations, leading to a performance decline.

Table 5: Comparing ObjectNav performance on Gibson and MP3D of related studies. Note that Red-Rabbit [53] utilizes auxiliary tasks for training, while THDA [28] and Habitat-Web [34] incorporate additional training data. Results of SemExp [4], L2M [12] and Stubborn [27] are reported from [60]. Results marked with * indicate our implementation.

ID		Method	Gibson			MP3D		
			SR(%) ↑	SPL(%) ↑	DTS(m) ↓	SR(%) ↑	SPL(%) ↑	DTS(m) ↓
	1	Random	0.4	0.4	3.89	0.5	0.5	8.05
I	2	DD-PPO [44]	15.0	10.7	3.24	8.0	1.8	6.94
	3	Red-Rabbit [53]	-	-	-	34.6	7.9	-
	4	THDA [28]	-	-	-	28.4	11.0	5.62
	5	SSCNav* [23]	-	-	-	27.1	11.2	5.71
	6	EmbCLIP* [18]	68.1	39.5	1.15	29.2	10.1	5.40
	7	Habitat-Web [34]	-	-	-	35.4	10.2	-
	8	ENTL [20]	-	-	-	17.0	5.0	-
	9	OVG-Nav [56]	-	-	-	35.8	12.3	5.69
II	10	FBE [50]	64.3	28.3	1.78	22.7	7.2	6.70
	11	ANS [6]	67.1	34.9	1.66	27.3	9.2	5.80
	12	SemExp [4]	71.1	39.6	1.39	28.3	10.9	6.06
	13	PONI [31]	73.6	41.0	1.25	31.8	12.1	5.10
	14	L2M [12]	-	-	-	32.1	11.0	5.12
	15	Stubborn [27]	-	-	-	31.2	13.5	5.01
	16	3D-aware [60]	74.5	42.1	1.16	34.0	14.6	**4.78**
	17	L3MVN [57]	76.9	38.8	1.01	-	-	-
	18	SGM [64]	78.0	44.0	1.11	37.7	14.7	4.93
	19	**T-Diff (Ours)**	**79.6**	**44.9**	**1.00**	**39.6**	**15.2**	5.16

Additionally, we visualize the navigation process of the agent, as shown in Fig. 5. According to the ground truth map and target location on the right side, it is evident that during navigation, the trajectories generated by our trajectory diffusion are consistently correct and efficient, even when the target object is not visible in the early stages (t<76). Consequently, the agent efficiently finds the target guided by the generated trajectories. More visualizations can be found in the Appendix.

Comparisons with the related works. We evaluate the performance of our T-Diff on ObjectNav task by comparing it with related baselines, categorizing into end-to-end [44, 53, 28, 23, 18, 34, 20, 56] and modular [50, 5, 4, 31, 12, 27, 60, 57] methods. It is worth noting that some methods incorporate additional information [28, 10, 34] or auxiliary tasks [53, 7], making it challenging to achieve a fair comparison. Therefore, we primarily focus on the following baselines: SemExp [4], PONI [31], OVG-Nav [56], L3MVN [57], L2M [12], and SSCNav [23]. PONI enhances SemExp by introducing supervised learning to predict goal-related information. OVG-Nav uses semantic topological maps to plan sub-goal nodes at a high level for the agent. L3MVN leverages LLM to infer unknown regions based on semantic information of the current boundary. L2M and SSCNav improve navigation by learning to build a top-down map that is egocentric in a single timestep. Since L2M and SSCNav only report results on their custom datasets, we use either re-implementation results from other work [60] with similar experimental settings or our own implementation results for comparison.

We compare our T-Diff with these methods on the validation sets of Gibson and MP3D, as shown in Tab. 5. On the Gibson, our T-Diff outperforms the current state-of-the-art method [57] by 2.7%, 6.1%, and -0.01m in SR, SPL, and DTS metrics, respectively. On the MP3D, T-Diff improves by 3.8%, 2.9%, and -0.53m in the same metrics compared to the current state-of-the-art method [56].

6 Conclusion

We propose a trajectory diffusion model (T-Diff) as a sequence planner for ObjectNav task. Our method leverages agent's historical observations and target object as conditions to iteratively generate the future sequential trajectory. Experimental results on standard datasets, including Gibson and MP3D, demonstrate that our T-Diff effectively improves navigation performance compared to the baselines. Furthermore, additional visualizations and experiments, detailed in the Appendix and supplementary materials, show that the future trajectory generated by T-Diff provides effective guidance for the agent and exhibits scalability and generalizability across different simulators.

Acknowledgements

This work was supported by the National Natural Science Foundation of China under Grant 62125207, 62032022, 62272443 and U23B2012, in part by Beijing Natural Science Foundation under Grant JQ22012 and L242020.

References

[1] Richard Bellman. *Adaptive Control Processes - A Guided Tour (Reprint from 1961)*, volume 2045 of *Princeton Legacy Library*. Princeton University Press, 2015.

[2] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. Instructpix2pix: Learning to follow image editing instructions. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 18392–18402. IEEE, 2023.

[3] Angel X. Chang, Angela Dai, Thomas A. Funkhouser, Maciej Halber, Matthias Nießner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3d: Learning from RGB-D data in indoor environments. In *2017 International Conference on 3D Vision, 3DV 2017, Qingdao, China, October 10-12, 2017*, pages 667–676. IEEE Computer Society, 2017.

[4] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Russ R. Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[5] Devendra Singh Chaplot, Dhiraj Gandhi, Saurabh Gupta, Abhinav Gupta, and Ruslan Salakhutdinov. Learning to explore using active neural SLAM. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[6] Devendra Singh Chaplot, Ruslan Salakhutdinov, Abhinav Gupta, and Saurabh Gupta. Neural topological SLAM for visual navigation. In *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 12872–12881, 2020.

[7] Shizhe Chen, Thomas Chabal, Ivan Laptev, and Cordelia Schmid. Object goal navigation with recursive implicit maps. *CoRR*, abs/2308.05602, 2023.

[8] Zoey Qiuyu Chen, Shosuke C. Kiami, Abhishek Gupta, and Vikash Kumar. Genaug: Retargeting behaviors to unseen situations via generative augmentation. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.

[9] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.

[10] Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Jordi Salvador, Kiana Ehsani, Winson Han, Eric Kolve, Ali Farhadi, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *NeurIPS*, 2022. Outstanding Paper Award.

[11] Heming Du, Xin Yu, and Liang Zheng. Learning object relation graph and tentative policy for visual navigation. In *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VII*, pages 19–34, 2020.

[12] Georgios Georgakis, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, and Kostas Daniilidis. Learning to map for active semantic goal navigation. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

[13] Haoran He, Chenjia Bai, Kang Xu, Zhuoran Yang, Weinan Zhang, Dong Wang, Bin Zhao, and Xuelong Li. Diffusion model is an effective planner and data synthesizer for multi-task reinforcement learning. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[14] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*, pages 2980–2988, 2017.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 770–778, 2016.

[16] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

[17] Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 9902–9915. PMLR, 2022.

[18] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: CLIP embeddings for embodied AI. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 14809–14818. IEEE, 2022.

[19] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[20] Klemen Kotar, Aaron Walsman, and Roozbeh Mottaghi. Entl: Embodied navigation trajectory learner. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10863–10872, October 2023.

[21] Xiang Li, Varun Belagali, Jinghuan Shang, and Michael S. Ryoo. Crossway diffusion: Improving diffusion-based visuomotor policy via self-supervised learning. *CoRR*, abs/2307.01849, 2023.

[22] Xiangyang Li, Zihan Wang, Jiahao Yang, Yaowei Wang, and Shuqiang Jiang. Kerm: Knowledge enhanced reasoning for vision-and-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2583–2592, 2023.

[23] Yiqing Liang, Boyuan Chen, and Shuran Song. Sscnav: Confidence-aware semantic scene completion for visual semantic navigation. In *IEEE International Conference on Robotics and Automation, ICRA 2021, Xi'an, China, May 30 - June 5, 2021*, pages 13194–13200. IEEE, 2021.

[24] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

[25] Cong Lu, Philip J. Ball, Yee Whye Teh, and Jack Parker-Holder. Synthetic experience replay. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023.

[26] Andreas Lugmayr, Martin Danelljan, Andrés Romero, Fisher Yu, Radu Timofte, and Luc Van Gool. Repaint: Inpainting using denoising diffusion probabilistic models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 11451–11461. IEEE, 2022.

[27] Haokuan Luo, Albert Yue, Zhang-Wei Hong, and Pulkit Agrawal. Stubborn: A strong baseline for indoor object navigation. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*, pages 3287–3293. IEEE, 2022.

[28] Oleksandr Maksymets, Vincent Cartillier, Aaron Gokaslan, Erik Wijmans, Wojciech Galuba, Stefan Lee, and Dhruv Batra. THDA: treasure hunt data augmentation for semantic navigation. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 15354–15363, 2021.

[29] Bar Mayo, Tamir Hazan, and Ayellet Tal. Visual navigation with spatial attention. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 16898–16907, 2021.

[30] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 4172–4182. IEEE, 2023.

[31] Santhosh Kumar Ramakrishnan, Devendra Singh Chaplot, Ziad Al-Halah, Jitendra Malik, and Kristen Grauman. PONI: potential functions for objectgoal navigation with interaction-free learning. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 18868–18878. IEEE, 2022.

[32] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *CoRR*, abs/2204.06125, 2022.

[33] Ram Ramrakhya, Dhruv Batra, Erik Wijmans, and Abhishek Das. Pirlnav: Pretraining with imitation and RL finetuning for OBJECTNAV. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 17896–17906. IEEE, 2023.

[34] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 5163–5173. IEEE, 2022.

[35] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE, 2022.

[36] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2022, New Orleans, LA, USA, June 18-24, 2022*, pages 10674–10685. IEEE, 2022.

[37] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L. Denton, Seyed Kamyar Seyed Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Photorealistic text-to-image diffusion models with deep language understanding. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[38] James A. Sethian. Fast marching methods. *SIAM Rev.*, 41(2):199–235, 1999.

[39] Shelly Sheynin, Adam Polyak, Uriel Singer, Yuval Kirstain, Amit Zohar, Oron Ashual, Devi Parikh, and Yaniv Taigman. Emu edit: Precise image editing via recognition and generation tasks. *CoRR*, abs/2311.10089, 2023.

[40] Edward C Tolman. Cognitive maps in rats and men. *Psychological review*, 55(4):189, 1948.

[41] Yingxu Wang and Günther Ruhe. The cognitive process of decision making. *Int. J. Cogn. Informatics Nat. Intell.*, 1(2):73–85, 2007.

[42] Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, Junjie Hu, Ming Jiang, and Shuqiang Jiang. Lookahead exploration with neural radiance representation for continuous vision-language navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13753–13762, 2024.

[43] Zihan Wang, Xiangyang Li, Jiahao Yang, Yeqi Liu, and Shuqiang Jiang. Gridmm: Grid memory map for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15625–15636, 2023.

[44] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. DD-PPO: learning near-perfect pointgoal navigators from 2.5 billion frames. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net, 2020.

[45] Mitchell Wortsman, Kiana Ehsani, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Learning to learn how to learn: Self-adaptive visual navigation using meta-learning. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 6750–6759, 2019.

[46] Chanyue Wu, Dong Wang, Yunpeng Bai, Hanyu Mao, Ying Li, and Qiang Shen. Hsr-diff: Hyperspectral image super-resolution via conditional diffusion models. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, pages 7060–7070. IEEE, 2023.

[47] Fei Xia, Amir Roshan Zamir, Zhi-Yang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*, pages 9068–9079. IEEE Computer Society, 2018.

[48] Shaoan Xie, Zhifei Zhang, Zhe Lin, Tobias Hinz, and Kun Zhang. Smartbrush: Text and shape guided object inpainting with diffusion model. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 22428–22437. IEEE, 2023.

[49] Karmesh Yadav, Ram Ramrakhya, Santhosh Kumar Ramakrishnan, Théophile Gervet, John M. Turner, Aaron Gokaslan, Noah Maestre, Angel Xuan Chang, Dhruv Batra, Manolis Savva, Alexander William Clegg, and Devendra Singh Chaplot. Habitat-matterport 3d semantics dataset. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 4927–4936. IEEE, 2023.

[50] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97 - Towards New Computational Principles for Robotics and Automation, July 10-11, 1997, Monterey, California, USA*, pages 146–151. IEEE Computer Society, 1997.

[51] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019.

[52] Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. Ip-adapter: Text compatible image prompt adapter for text-to-image diffusion models. *CoRR*, abs/2308.06721, 2023.

[53] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectgoal navigation. In *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*, pages 16097–16106. IEEE, 2021.

[54] Mao Ye, Lemeng Wu, and Qiang Liu. First hitting diffusion models for generating manifold, graph and categorical data. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022.

[55] Xin Ye and Yezhou Yang. Hierarchical and partially observable goal-driven policy learning with goals relational graph. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*, pages 14101–14110, 2021.

[56] Hwiyeon Yoo, Yunho Choi, Jeongho Park, and Songhwai Oh. Commonsense-aware object value graph for object goal navigation. *IEEE Robotics Autom. Lett.*, 9(5):4423–4430, 2024.

[57] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3MVN: leveraging large language models for visual target navigation. In *IROS*, pages 3554–3560, 2023.

[58] Tianhe Yu, Ted Xiao, Jonathan Tompson, Austin Stone, Su Wang, Anthony Brohan, Jaspiar Singh, Clayton Tan, Dee M, Jodilyn Peralta, Karol Hausman, Brian Ichter, and Fei Xia. Scaling robot learning with semantically imagined experience. In Kostas E. Bekris, Kris Hauser, Sylvia L. Herbert, and Jingjin Yu, editors, *Robotics: Science and Systems XIX, Daegu, Republic of Korea, July 10-14, 2023*, 2023.

[59] Albert J. Zhai and Shenlong Wang. Peanut: Predicting and navigating to unseen targets. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 10926–10935, October 2023.

[60] Jiazhao Zhang, Liu Dai, Fanpeng Meng, Qingnan Fan, Xuelin Chen, Kai Xu, and He Wang. 3d-aware object goal navigation via simultaneous exploration and identification. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6672–6682, June 2023.

[61] Sixian Zhang, Weijie Li, Xinhang Song, Yubing Bai, and Shuqiang Jiang. Generative meta-adversarial network for unseen object navigation. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXIX*, volume 13699 of *Lecture Notes in Computer Science*, pages 301–320.

[62] Sixian Zhang, Xinhang Song, Yubing Bai, Weijie Li, Yakui Chu, and Shuqiang Jiang. Hierarchical object-to-zone graph for object navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15130–15140, October 2021.

[63] Sixian Zhang, Xinhang Song, Weijie Li, Yubing Bai, Xinyao Yu, and Shuqiang Jiang. Layout-based causal inference for object navigation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, pages 10792–10802. IEEE, 2023.

[64] Sixian Zhang, Xinyao Yu, Xinhang Song, Xiaohan Wang, and Shuqiang Jiang. Imagine before go: Self-supervised generative map for object goal navigation. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 16414–16425. IEEE, 2024.

[65] Minzhao Zhu, Binglei Zhao, and Tao Kong. Navigating to objects in unseen environments by distance prediction. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2022, Kyoto, Japan, October 23-27, 2022*, pages 10571–10578. IEEE, 2022.

[66] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven visual navigation in indoor scenes using deep reinforcement learning. In *2017 IEEE International Conference on Robotics and Automation, ICRA 2017, Singapore, Singapore, May 29 - June 3, 2017*, pages 3357–3364, 2017.

Table 6: Chosen object categories in Gibson [47] and MP3D [3].

Dataset	Training	Evaluating
Gibson	chair, couch, potted plant, bed, toilet, dining-table, tv, oven, sink, refrigerator, book, clock, vase, cup, bottle	chair, couch, tv, bed, toilet, potted plant
MP3D	char, table, picture, cabinet, cushion, sofa, bed, chest of drawers, plant, sink, toilet, stool, towel, tv monitor, shower, bathtub, counter, fireplace, gym equipment, seating, clothes	

Table 7: FLOPs(Floating Point Operations) of SemExp [4], PONI [31], 3D-aware [60] and our T-Diff.

Method	SemExp	PONI	3D-Aware	T-Diff (Ours)
FLOPs (M)	3080.41	46621.90	11617.01	16078.40

A Appendix / supplemental material

A.1 Experiments Setup

Evaluation metrics. We employ SR, SPL and DTS metrics to evaluate the ObjectNav performance and assess the trajectory generation accuracy by MSE metric.

SR (Success Rate). SR measures the success rate of the agent in successfully finding the target object. It is defined as $SR = \frac{1}{N}\sum_{i=1}^{N} S_i$, where N is the total number of validation episodes and S_i is an indicator that representing whether the i-th episode is successful or not.

SPL (Success weighted by Path Length). SPL considers whether the path length of navigation is efficient based on success rate which is formulated as $SPL = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i^*}{max(l_i, l_i^*)}$, where l_i^* means the shortest path length calculated by the simulator and l_i refers the actual path length of i-th episode.

DTS (Distance to Goal). DTS evaluates the distance $L_{i,g}$ of the agent towards the target object when the episode ends. It can be calculated as $DTS = \frac{1}{N}\sum_{i=1}^{N} max(L_{i,g} - \xi, 0)$, where success threshold $\xi = 1m$. DTS is 0 when an episode is successful.

MSE (Mean Squared Error). MSE assesses the accuracy of the generated trajectory which is defined as $MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - \hat{x_i})^2$, where n is the number of validation set of our collected trajectory data pairs. x_i refers the ground truth trajectory and $\hat{x_i}$ is the generated trajectory.

Object Categories. Our experimental setup follows previous settings [31, 60, 4], and the adopted object categories are detailed in Tab. 6. For Gibson, we choose 15 categories for training and 6 for evaluating. For MP3D, there are 21 categories for both training and evaluating.

More Visualizations. Fig. 6 illustrates more visualizations of the generated trajectory by our T-Diff during navigation process. As shown in Fig. 6, T-Diff precisely generates the agent's future trajectory conditioned on the current state, thus effectively steering the agent towards the target object. The accuracy of the generated future trajectory reduces unnecessary exploration and guides the agent along an almost optimal path, ulimately navigating it to stop directly in front of the target object, demostrating that our T-Diff significantly improves the efficiency of navigation.

Furthermore, we provide a video demo that presents a more intuitive view of the process of trajectory denoising and navigation. Please refer to the MP4 file in the supplements zip archive.

A.2 Computation Complexity

To compare the computational complexity of our T-Diff with the modular baselines(SemExp [4], PONI [31] and 3D-aware [60]), we utilize FLOPs(Floating Point Operations) metric to assess the computational complexity, as shown in Tab. 7, where a higher value of FLOPs refers greater computational complexity. Note that, T-Diff iterates $\tau_{max} = 100$ steps for each time step to generate the denoised future trajectory, and this trajectory is generated every $t_{T-diff} = 5$ steps during navigation. Thus, we compute the computation complexity of 100 iterations and average it over every 5 steps. The results in Tab. 7 indicate that the computation complexity of our T-Diff is higher than

Figure 6: More visualizations. At each timestep, we visualize RGB observation, local semantic map along with the previous path and the generated future trajectory. Note that the target object location is marked with a circle.

that of SemExp, comparable to that of 3D-aware, but significantly lower than that of PONI. Although T-Diff requires multiple iteration, its operating latents is relatively small (when $k = 32$, the input dimension is $B \times 2 \times 32$) compared to PONI's input (the input dimension is $B \times (4+n) \times 480 \times 480$, where n refers to the number of semantic categories). Therefore, the compuation complexity of T-Diff is considered acceptable.

A.3 Limitations

(1) The method of generating future trajectories based on diffusion has prediction biases. The proposed trajectory diffusion model primarily considers the current semantic map, historical trajectory information, and target category. However, other information such as the egocentric view and the relationships between object semantics, may help generate better trajectories for the ObjectNav task. Therefore, in future work, we plan to apply more conditions into the trajectory diffusion model.

(2) The training data for the trajectory diffusion model mainly comes from the collection of optimal paths, which are relatively straight and short. The types of training data pairs are not yet diverse enough, and the collection of complex trajectories requiring multiple turns and detours is still

insufficient. In future work, we plan to introduce data pairs of complex paths to enhance the model's applicability to more complex scenarios.

A.4 Broader Impacts

Our trajectory diffusion model is a general method applied to the ObjectNav task. Although our training and testing are currently limited to the simulator stage, the trajectory diffusion model can be deployed on real robots. However, prediction errors in the model may lead to incorrect actions by the robot, potentially causing damage to personal or social property. Therefore, it must be used cautiously to ensure safety in real-world applications.

A.5 Data License

We use two datasets (Gibson [47] and MP3D [3]), and employ Habitat simulator. All of them are published on the official papers with no licenses stated on their papers and websites. Thus, we just cite all corresponding papers without licenses.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: We have clearly delineated the main claims of our proposed method in both the abstract and introduction.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of the proposed method in Sec. A.3.

   Guidelines:

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

3. **Theory Assumptions and Proofs**

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

   Answer: [NA]

Justification: The paper does not include theoretical results.

Guidelines:

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

4. **Experimental Result Reproducibility**

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

   Answer: [Yes]

   Justification: We provide all the essential details for replicating the main experimental results, with thorough descriptions of the experimental procedures in Sec. 5.1. In addition, an anonymous code repository is provided in Sec. **??**.

   Guidelines:

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

5. **Open access to data and code**

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

Answer: [Yes]

Justification: We provide an anonymous code repository in Sec. **??**, containing comprehensive data, code, and relication instructions.

Guidelines:

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

6. **Experimental Setting/Details**

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

   Answer: [Yes]

   Justification: We specify all the training and test details in Sec. 5.1.

   Guidelines:

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

7. **Experiment Statistical Significance**

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

   Answer: [No]

   Justification: We mention error bars in Sec. 5.1 and explain that due to the relatively small variance in the experiments conducted in our paper, error bars are not reported.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).

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

8. **Experiments Compute Resources**

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

   Answer: [Yes]

   Justification: We provide the related information in Sec. A.2.

   Guidelines:

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

9. **Code Of Ethics**

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

   Answer: [Yes]

   Justification: The research conforms with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader Impacts**

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

    Answer: [Yes]

    Justification: We discuss both potential positive societal impacts and negative impacts in Sec. A.4.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.

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

11. **Safeguards**

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

    Answer: [NA]

    Justification: The paper poses no such risks.

    Guidelines:

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

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: See in Sec. A.5.

    Guidelines:

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

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

13. **New Assets**

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

    Answer: [NA]

    Justification: The paper does not release new assets.

    Guidelines:

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

14. **Crowdsourcing and Research with Human Subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

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

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

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