Scaling Value Iteration Networks to 5000 Layers for Extreme Long-Term Planning

Yuhui Wang [*1] **Qingyuan Wu** [*2] **Dylan R. Ashley** [1 3] **Francesco Faccio** [1 3] **Weida Li** [4]
Chao Huang [2] **Jürgen Schmidhuber** [1 3]

Abstract

The Value Iteration Network (VIN) is an end-to-end differentiable neural network architecture for planning. It exhibits strong generalization to unseen domains by incorporating a differentiable planning module that operates on a latent Markov Decision Process (MDP). However, VINs struggle to scale to long-term and large-scale planning tasks, such as navigating a 100×100 maze—a task that typically requires thousands of planning steps to solve. We observe that this deficiency is due to two issues: the representation capacity of the latent MDP and the planning module's depth. We address these by augmenting the latent MDP with a dynamic transition kernel, dramatically improving its representational capacity, and, to mitigate the vanishing gradient problem, introduce an "adaptive highway loss" that constructs skip connections to improve gradient flow. We evaluate our method on 2D/3D maze navigation environments, continuous control, and the real-world Lunar rover navigation task. We find that our new method, named *Dynamic Transition VIN (DT-VIN)*, scales to 5000 layers and solves challenging versions of the above tasks. Altogether, we believe that DT-VIN represents a concrete step forward in performing long-term large-scale planning in complex environments.

1. Introduction

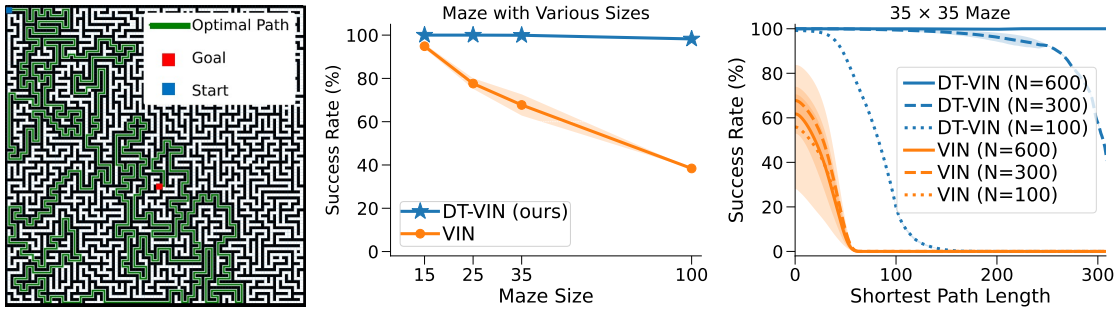
Planning is the problem of finding a sequence of actions that achieves a specific pre-defined goal. As the aim of both some older algorithms (e.g., Dyna (Sutton, 1991), A* (Hart et al., 1968), and others (Schmidhuber, 1990a;b)) and many recent ones (e.g., the Predictron (Silver et al., 2017), the Dreamer family of algorithms (Hafner et al., 2020; 2021; 2023), SoRB (Eysenbach et al., 2019), SA-CADRL (Chen et al., 2017), and the LLM-planner (Song et al., 2023)), effective planning is a long-standing and important challenge in artificial intelligence (AI).

Traditional search-based planning algorithms like A* require an accurate environmental model. Thus, these algorithms are less effective in scenarios with unknown environmental models or when the state and action spaces are large or continuous. In such scenarios, a policy can be learned either through imitation learning (IL), which leverages expert demonstrations, or through trial and error with reinforcement learning (RL). Within RL and IL, the Value Iteration Network (VIN) (Tamar et al., 2016) stands out as quite unique due to its distinctive architecture that integrates a differentiable latent "planning module" into the deep neural network, rather than maintaining an explicit learned environment model like Dreamer (Hafner et al., 2020) or MuZero (Schrittwieser et al., 2020). This integrated planning structure of VINs endows them with powerful generalization capabilities for unseen planning tasks. VINs have been shown to perform well in some small-scale short-term planning situations, like path planning (Pflueger et al., 2019; Jin et al., 2021), autonomous navigation (Wöhlke et al., 2021), and complex decision-making in dynamic environments (Li et al., 2021). However, they still struggle to solve larger-scale and longer-term planning problems. We refer to *large-scale planning tasks* as those with high-dimensional observation space (e.g., the maze size), and *long-term planning tasks* as those necessitating extended planning horizons to achieve the goal. For example, in a 100×100 maze navigation task, the success rate of VINs in reaching the goal drops to well below 40% (see Figure 1(b)). Even in smaller 35×35 mazes, the success rate of VINs drops to 0% when the required planning steps exceed 60 (see Figure 1(c)).

Our work identifies that the principal deficiency causing this is the mismatch between the complexity of planning and the comparatively weak representational capacity of

[*]Equal contribution [1]Center of Excellence for Generative AI, King Abdullah University of Science and Technology [2]The University of Southampton [3]The Swiss AI Lab IDSIA/USI/SUPSI, Switzerland [4]National University of Singapore. Correspondence to: Yuhui Wang <yuhui.wang@kaust.edu.sa>.

Proceedings of the 42^{nd} International Conference on Machine Learning, Vancouver, Canada. PMLR 267, 2025.



(a) 100 × 100 Maze (b) Performance by Domain Scales (c) Performance by Planning Steps

Figure 1: (a) shows an example of 100×100 maze navigation task, where the green line shows the optimal path from the start position (blue) to the goal position (red). See Appendix Figure 8 for more examples of mazes with other sizes. (b) shows the success rate of VIN and DT-VIN on the maze navigation tasks as a function of maze size. The reported results are computed in expectation over different shortest path lengths for each maze size. (c) shows the success rate of VIN (Tamar et al., 2016) and our DT-VIN as a function of planning steps on the 35×35 maze benchmark.

the relatively shallow networks that it uses. And while there has been moderate success in learning more complicated networks (e.g., GPPN (Lee et al., 2018) and Highway VINs (Wang et al., 2024a)), until now, VINs of a scale capable of long-term or large-scale planning have not been computationally tractable due to persistent issues with vanishing and exploding gradients—a fundamental problem of deep learning (Hochreiter, 1991).

In this work, we aim to surgically correct deficiencies in VIN-based architectures to enable large-scale long-term planning. Specifically, we first identify the limitations of the latent MDP in the planning module of VIN and propose a dynamic transition kernel to dramatically increase the representational capacity of the network. We then build on existing work that identifies the connection between network depth and long-term planning (Wang et al., 2024a) and propose an "adaptive highway loss" that selectively constructs skip connections to the final loss according to the actual number of planning steps. This approach helps mitigate the vanishing gradient problem and enables the training of very deep networks. With these changes, we find that our new *Dynamic Transition Value Iteration Network* (*DT-VIN*), is able to be trained with 5000 layers and scale to 1800 planning steps in a 100×100 maze navigation task (compared to the original VIN, which only scaled to 120 planning steps in a 25×25 maze). We apply our method to various challenging tasks, including 2D/3D maze navigation tasks (Wydmuch et al., 2019), continuous control (Foundation, 2022; Fu et al., 2020), and real-world Lunar rover navigation tasks (Berlin, 2018). We find that DT-VINs can solve both despite these problems requiring hundreds to thousands of planning steps. Together, these demonstrate the practical utility of our method on vision-based tasks that previous methods are simply unable to solve. This also serves to highlight the potential of our method to scale to increasingly complex planning tasks alongside the increasing availability of computing power.

2. Preliminaries

Reinforcement Learning (RL) and Imitation Learning (IL). The most common formalism used for RL is that of the Markov Decision Process (MDP) (Bellman, 1957). We consider an MDP—as per Puterman (2014)—to be the 6-tuple $(\mathcal{S}, \mathcal{A}, \mathcal{T}, \mathcal{R}, \gamma, \mu)$, where $\mathcal{S}$ is a countable state space, $\mathcal{A}$ is a finite action space, $\mathcal{T}(s'|s,a)$ represents the transition probability to state $s' \in \mathcal{S}$ from state $s \in \mathcal{S}$ and taking action $a \in \mathcal{A}$, $\mathcal{R}(s,a,s')$ is the reward function, $\gamma \in [0,1)$ is a discount factor, and μ is a distribution over initial states. The behaviour of an artificial agent in an MDP is defined by its policy $\pi(a|s)$, which specifies the probability of taking action a in state s. The state value function $V^\pi(s)$ is the expected discounted sum of rewards from state s and following policy π, i.e., $V^\pi(s) \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \mathcal{R}(s_t, a_t, s_{t+1}) | s_0 = s; \pi\right]$. The goal of RL is usually to find an optimal policy π^* that achieves the highest expected discounted sum of rewards. The value function of an optimal policy is denoted by $V^*(s) = \max_\pi V^\pi(s)$, and satisfies $V^{\pi^*}(s) = V^*(s) \forall s$. The Value Iteration (VI) algorithm iteratively applies the following update to all states to obtain the optimal value function: $V^{(n+1)}(s) = \max_a \sum_{s'} \mathcal{T}(s'|s,a)\left[\mathcal{R}(s,a,s') + \gamma V^{(n)}(s')\right]$, where n is the iteration number. In scenarios where designing a comprehensive reward function is difficult, IL offers a practical alternative. IL enables agents to learn from human or algorithmic demonstrations, with approaches like Behavioral Cloning directly mimicking expert actions in similar states (Bain & Sammut, 1995; Schaal, 1996; Ross et al., 2011).

Value Iteration Networks (VINs). VIN is an end-to-end differentiable architecture that conducts planning on a latent

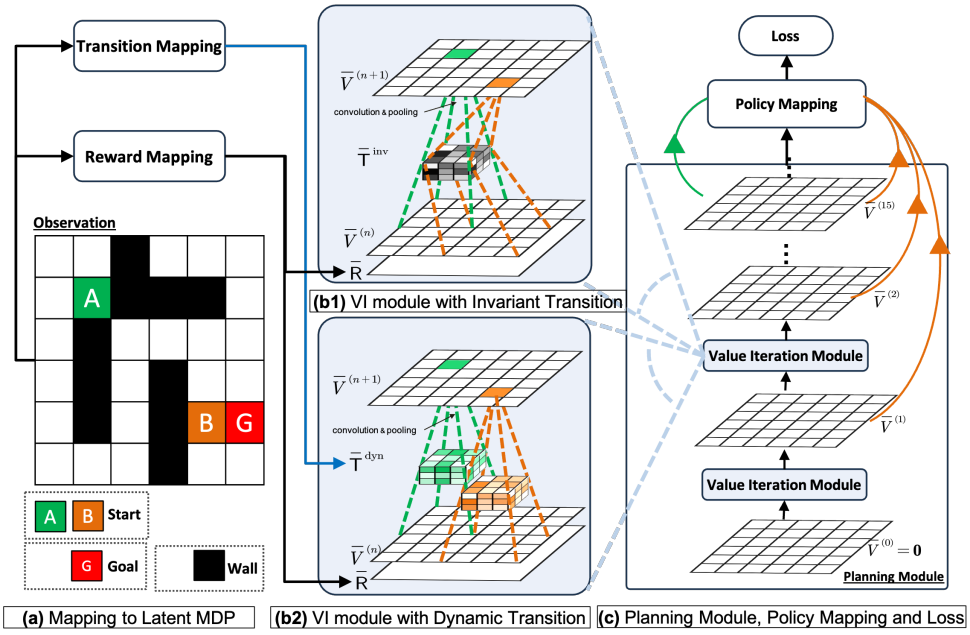


Figure 2: The architecture of VIN and DT-VIN in the maze navigation task. (a) shows the observation of the maze, which is mapped to the latent reward/transition matrix of the latent MDP through the reward/transition mapping module. (c) shows the "planning module", the policy mapping module and the loss. The "planning module" contains numerous stacked Value Iteration (VI) modules. The green and orange connections show an example of adaptive highway loss for planning tasks starting from A and B, respectively. (b1) shows the VI module of the original VIN with invariant transition $\overline{\mathsf{T}}^{\text{inv}} \in \mathbb{R}^{|\mathcal{A}|\times F\times F}$. (b2) shows the VI module of DT-VIN with dynamic transition kernel $\overline{\mathsf{T}}^{\text{dyn}} \in \mathbb{R}^{M\times M\times|\mathcal{A}|\times F\times F}$.

MDP $\overline{\mathcal{M}}$ (Tamar et al., 2016). Below, we use $\overline{\cdot}$ to denote all the terms associated with the latent MDP $\overline{\mathcal{M}}$. For each decision, VIN first maps an observation x, e.g., an image of a maze and the agent's position, to $\overline{\mathcal{M}}$. $\overline{\mathcal{M}}$ is described by the latent state space $\overline{\mathcal{S}} = \{(i,j)\}_{i,j\in[M]}$, where M denotes its size; a fixed discrete latent action space $\overline{\mathcal{A}}$; a latent reward matrix $\overline{\mathsf{R}} = f^{\overline{\mathsf{R}}}(x) \in \mathbb{R}^{M\times M}$, where $f^{\overline{\mathsf{R}}}$ is a learnable NN called a *reward mapping module*; and a latent transition kernel $\overline{\mathsf{T}}^{\text{inv}} \in \mathbb{R}^{|\overline{\mathcal{A}}|\times F\times F}$ with F representing the dimension of the kernel. The latent transition kernel is a learnable parameter matrix that is invariant of both the latent state and the observation x. Next, VIN conducts VI on the latent MDP $\overline{\mathcal{M}}$ to approximate the latent optimal value function $\overline{V}^*$. To ensure the differentiability, a differentiable VI module is proposed, simulating VI computation using CNN operations, i.e., convolutional and max-pooling operations:

$$\overline{V}^{(n)}_{i,j} = \max_{\overline{a}} \sum_{i',j'} \overline{\mathsf{T}}^{\text{inv}}_{\overline{a},i',j'} \left(\overline{\mathsf{R}}_{i-i',j-j'} + \overline{V}^{(n-1)}_{i-i',j-j'}\right) \quad (1)$$

This equation sums over a matrix patch centered around position (i,j). After the above, by stacking the VI module for N layers, the latent value function is then fed to a policy mapping module by f^π to represent a policy that is applicable to the actual MDP $\mathcal{M}$. Here, $f^\pi\left(\overline{V}^{(n)}(x), a\right)$ represents the probability of taking action a given observation x. Finally, the model can be trained by standard RL and IL algorithms with the following general loss: $\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|}\sum_{(x,y)\in\mathcal{D}} \ell\left(f^\pi\left(\overline{V}^{(N)}(x), \cdot\right), y\right)$, where $\mathcal{D} = \{(x,y)\}$ is the training data, x is the observation, y is the label, and ℓ is the sample-wise loss function. The specific meaning of these items varies depending on the task. For example, in imitation learning, where the expert data is provided, the label y is the expert action and ℓ is the cross-entropy loss, i.e., $\ell\left(f^\pi\left(\overline{V}^{(N)}(x)\right), y\right) = -\sum_{a\in\mathcal{A}} \mathbb{1}_{\{a=y\}} \log f^\pi\left(\overline{V}^{(n)}(x), a\right)$, where $\mathbb{1}$ is the indicator function.

3. Method

In this section, we discuss how to train scalable VINs for long-term large-scale planning tasks. Our method addresses the two key issues with VIN that are identified as hampering its scalability: the capacity of the latent MDP representation and the depth of the planning module.

3.1. Improving Latent MDP's Representation Capacity

VIN utilizes CNNs to simulate the VI computation process, where the latent transition kernel is implemented as a learn-

able parameter $\overline{\mathsf{T}}^{\text{inv}} \in \mathbb{R}^{|\overline{\mathcal{A}}| \times F \times F}$, as described in Section 2. However, there is a discrepancy between the computation process of CNNs and the general VI.

First, the latent transition kernel of VIN is invariant for each latent state $\overline{s} = (i, j)$. This severely limits the latent MDP's capacity to represent the complex, state-dependent transitions of the real MDP–an ability that is essential for its effectiveness. For example, in the maze navigation problem shown in Figure 2 (a), the transition probabilities are quite different if the adjacent cell is a wall versus an empty cell.

Therefore, we propose to use a *latent state-dynamic transition kernel* $\overline{\mathsf{T}}^{\text{dyn}} \in \mathbb{R}^{M \times M \times |\overline{\mathcal{A}}| \times F \times F}$ and the augmented VI module is computed as follows:

$$\overline{V}^{(n)}_{i,j} = \max_{\overline{a}} \sum_{i',j'} \overline{\mathsf{T}}^{\text{dyn}}_{i,j,\overline{a},i',j'} \left(\overline{\mathsf{R}}_{i-i',j-j'} + \overline{V}^{(n-1)}_{i-i',j-j'} \right). \quad (2)$$

Different from VIN's VI module in Eq. (1), which uses an invariant kernel $\overline{\mathsf{T}}^{\text{inv}} \in \mathbb{R}^{|\overline{\mathcal{A}}| \times F \times F}$ that is same across all latent states (i, j), here Eq. (2) utilizes a dynamic kernel $\overline{\mathsf{T}}^{\text{dyn}}_{i,j}$ that adjusts dynamically to each latent state $\overline{s} = (i, j)$.

Second, although the original VIN paper proposes a general framework where the latent transition kernel depends on the observation, i.e., $\overline{\mathsf{T}}^{\text{inv}} = f^{\overline{\mathsf{T}}}(x)$, with $f^{\overline{\mathsf{T}}}$ as a learnable transition mapping module, *the latent transition kernel in all VIN's experiments is implemented as a learnable parameter independent of the observation*. Intuitively, ignoring observations prevents VIN from exploiting any information in the observation to model varying transition dynamics across different environments. While this limitation does not impact VIN's performance in small-scale, short-term planning tasks (as were tested on in the original work) where the state space is limited and only a few steps are needed to reach the goal, we found it to be a major barrier to VIN's effectiveness when employed to plan on large-scale, long-term planning tasks with completely different observations. Therefore, we propose using observation-dynamic transition kernels, which employ a learnable transition mapping module $f^{\overline{\mathsf{T}}}$ to dynamically generate the latent transition kernel for each observation.

The notions of latent state-dynamic and observation-dynamic features are orthogonal, each capturing an independent aspect of variation in the latent transition kernel. Their combination yields four distinct configurations, along with corresponding implementations, as summarized in Table 1. Our experiment in Section 4.4 shows that the fully dynamic kernel—which varies with both the latent state and the observation—achieves the best performance. To implement this kernel, we design a CNN-based architecture for the transition mapping module f^{T}, which maps each local $F' \times F'$ maze patch to a $|\overline{\mathcal{A}}| \times F \times F$ latent transition kernel

Table 1: Types of latent transition kernels, categorized by whether they are *latent state-dynamic* (i.e., vary with the latent state) and *observation-dynamic* (i.e., vary with the observation).

Types	Latent state-dynamic?	Observation-dynamic?	Implementation
fully invariant	✗	✗	learnable parameter $\overline{\mathsf{T}} \in \mathbb{R}^{\lvert\overline{\mathcal{A}}\rvert \times F \times F}$
latent state-dynamic only	✓	✗	learnable parameter $\overline{\mathsf{T}} \in \mathbb{R}^{M \times M \times \lvert\overline{\mathcal{A}}\rvert \times F \times F}$
observation-dynamic only	✗	✓	learnable model $f^{\overline{\mathsf{T}}}(x)$ which output $\overline{\mathsf{T}} \in \in \mathbb{R}^{\lvert\overline{\mathcal{A}}\rvert \times F \times F}$
fully dynamic	✓	✓	learnable model $f^{\overline{\mathsf{T}}}(x)$ which output $\overline{\mathsf{T}} \in \mathbb{R}^{M \times M \times \lvert\overline{\mathcal{A}}\rvert \times F \times F}$

for each latent state, where F' is the convolutional kernel size of the transition mapping module, and F is the size of the latent transition kernel. This architecture only requires $|\overline{\mathcal{A}}| F'^2 F^2$ additional parameters. Note that in practice, setting both F' and F to the small value 3 suffices to achieve strong performance. Thus, this alternative module greatly improves the representation capacity of VIN, but typically does not introduce a significant change in training cost.

Lastly, we further enforce a softmax operation on the values of the latent transition kernel for each latent state $\overline{s}$ to avoid the gradient exploding problem. This not only provides a statistical semantic meaning to the kernels but also plays a crucial role in ensuring the training stability, as will be demonstrated in Section 4.4.

Altogether, these above components form our proposed method, *Dynamic Transition VINs (DT-VINs)*, which employs latent transition kernels that dynamically adapt based on both the latent state and observation, along with a softmax normalization operation.

3.2. Increasing Depth of Planning Module

Recent work on Highway VIN has demonstrated the relationship between the depth of VIN's planning module and its planning ability (Wang et al., 2024a). A deeper planning module implies more iterations of the value iterations process, which is proven to yield a more accurate estimation of the optimal value function (see Theorem 1.12) (Agarwal et al., 2019). However, training very deep neural networks is challenging due to the vanishing or exploding gradient problem (Hochreiter, 1991). Highway VINs address this issue by incorporating skip connections within the context of reinforcement learning, showing similarities to existing works for classification tasks (Srivastava et al., 2015b; He et al., 2016). Although Highway VINs can be trained with up to 300 layers, they still fail to achieve perfect scores in larger-scale and longer-term planning tasks and necessitate a more intricate implementation. Here, we present a simpler, easy-to-implement method for training very deep VINs.

To facilitate the training of very deep VINs, we also adopt the skip connections structure but implement it differently. Our central insight is that short-term planning tasks generally require fewer iterations of value iteration compared to long-term planning tasks. This is because the information from the goal position propagates to the start position in fewer steps when their distance is short. Therefore, we propose adding additional loss to shallower layers directly when the task requires only a few steps during the training process. We achieve this by introducing the following *adaptive highway loss*:

$$\mathcal{L}(\theta) = \frac{1}{K|\mathcal{D}|} \cdot \sum_{(x,y,l)\in\mathcal{D}} \sum_{1\le n\le N} \mathbb{1}_{\{n\ge l\}} \mathbb{1}_{\{n \bmod J=0\}} \ell\left(f^{\pi}\left(\overline{V}^{(n)}(x), \cdot\right), y\right) \tag{3}$$

Here, $K = \sum_{(x,y,l)\in\mathcal{D}} \sum_{1\le n\le N} \mathbb{1}_{\{n\ge l\}} \mathbb{1}_{\{n \bmod J=0\}}$, $\mathbb{1}$ is the indicator function, and l is the length of the planning path or trajectory, which can be computed from the training data. For example, in the imitation learning of the maze navigation task, the length l of the provided expert path from start to goal is inherently known for each maze in the dataset. Here, Eq. (3) presents a general form of $\mathcal{L}$ with an arbitrary single-term loss function ℓ (e.g., cross-entropy loss); see Appendix B.1 for its specific form in different cases.

As Eq. (3) implies, it constructs skip connections for the hidden layers to improve information flow, similar to existing works such as Highway Nets and subsequently, Residual Nets (Srivastava et al., 2015b; He et al., 2016). However, we connect hidden layers directly to the final loss, while existing works typically connect skip connections between the intermediate layers. Besides, we construct skip connections for each layer $n \ge l$ rather than at the specific layer $n = l$, as the value iteration process should converge to the correct solutions even with additional iterations. Note that this additional loss will not alter the inherent structure of the value iteration process and will be removed during the execution phase. Moreover, to decrease computational complexity, we only apply adaptive highway loss to the layers that satisfy the condition $n \bmod J = 0$, where $J \ge 1$ is a hyperparameter.

4. Experiments

We perform several experiments to test if our modifications to VIN's planning module allow training very deep DT-VINs for large-scale long-term planning tasks. Following previous work (Tamar et al., 2016; Lee et al., 2018), we focus on the imitation learning scenario, where we leverage expert demonstrations to evaluate planning capabilities.

In line with previous works (e.g., (Lee et al., 2018)), we assess our planning algorithms on maze navigation tasks, encompassing both 2D and 3D environments (Wydmuch et al., 2019), as detailed in Section 4.1. To demonstrate the potential for complex action spaces, we further evaluate the generality of DT-VIN on continuous control tasks (Section 4.2), which present more challenges than the continuous control tasks described in the original VIN paper due to increased complexity of the maze and the physics (Fu et al., 2020). Additionally, we assess DT-VIN in rover navigation tasks (Section 4.3), where the agent must perform planning based on orthomosaic images (Berlin, 2018).

Each task includes a start position and a goal position. We say that an agent has succeeded in a task if it can navigate from the start position to the goal position within a predetermined number of steps (M^2 in our paper). To assess planning capabilities across tasks of varying complexities, our experiments evaluate various tasks distinguished by their *shortest path lengths (SPLs)*. The SPL represents the length of the expert path, derived from either human or algorithmic experts. We further define a path as relatively optimal if it has the shortest length among all solutions provided by various models, including the expert solution. We follow GPPN and use these for the *success rate (SR)*, which is the rate at which the algorithm succeeds in the task, and the *optimality rate (OR)*, which is the rate at which the algorithm provides a relatively optimal path.

On the above tasks, we compare our DT-VIN method with several advanced neural networks designed for planning tasks, including the original VIN (Tamar et al., 2016), GPPNs (Lee et al., 2018), and Highway VIN (Wang et al., 2024a). The models are trained through imitation learning using a labeled dataset. We then identify the best-performing model based on its results on a validation dataset and evaluate it on a separate test dataset. Following the methodology from the GPPN paper, we conduct evaluations using three different random seeds for each algorithm. This is sufficient to provide a reliable performance estimate here due to the low standard deviation we observe in the tasks. All figures that show learning curves report the mean and standard deviation on the test set.

4.1. Maze Navigation

Setting. We first evaluate 2D maze navigation tasks with sizes M set to 15, 35, and 100, where the agent moves one step at a time to any of the four adjacent cells. Each task is defined by a starting position, a visual representation of the $M \times M$ *map design* matrix (with 0 and 1 representing obstacles and roads, respectively), and an $M \times M$ goal matrix indicating the position of the goal. These mazes require hundreds to thousands of planning steps, as shown in Figure 1(a) and further detailed in Appendix Figure 8. For each maze size, we generate a dataset following the methodology in GPPN (Lee et al., 2018) and ensure no

overlap between training and test datasets. Specifically, each unique obstacle arrangement—with its varying start positions and goals—is exclusive to either the training or the test dataset, preventing information leakage. To assess the performance of each algorithm, we test various neural network depths N. For the largest mazes ($M = 100$), we specifically examine $N = 600$ and $N = 5000$, while for $M = 35$, we test multiple depths: $N = 30$, 100, 300, and 600. For more details, see Appendix C.1.

Results and Discussion. Figure 3(a) shows the success rates (SRs) of our method and the baseline methods, as a function of the SPLs. For each algorithm and environment configuration, we report the performance of the NN with the best depth N across the ranges specified in the previous paragraph (see Figure 9 in Appendix C.2 for additional results concerning different values of N).

Here, DT-VIN outperforms all the other methods on all the maze navigation tasks under all the various sizes M and SPLs. Notably, on small-scale mazes with size in $M = 15, 35$, DT-VIN achieves approximately 100% SRs on all the tasks. For the most challenging environment with $M = 100$, DT-VIN performs best with the full 5000 layers, and it maintains an SR of approximately 100% on short-term planning tasks with SPL ranging in $[1, 200]$ and an SR of approximately 88% on tasks with SPLs over 1200. Comparatively, VIN performs well on small-scale and short-term planning tasks. However, even on a small-scale maze with size $M = 15$, VIN's SRs drop to 0% when the SPL exceeds 30. Moreover, when the maze size increases to 100, VIN only achieves an SR of less than 40%—even on short-term planning tasks with SPL within $[1, 100]$. GPPN performs well on short-term planning tasks, but it fails to generalize well on long-term planning tasks, which also decreases to an SR of 0% as the SPL increases. Highway VIN performs well across tasks with various SPLs on a small-scale maze with $M = 15$. However, it shows a performance decrease on larger-scale maze tasks with $M = 35, 100$. Figure 3(b) shows the optimality rates (ORs) of the algorithms, which measure the rate at which the model outputs the optimal path. Our DT-VIN maintains consistent ORs compared to SRs. However, some other methods—especially Highway VIN—exhibit a clear decrease in ORs, indicating that the paths generated by these methods is often sub-optimal.

Addtional Experiments. Due to space constraints, we evaluate challenging cases with noisy maze observations in Appendix C.4 and different transition types (agents moving to any of the eight Moore neighborhood cells) in Appendix C.3. We also evaluate 3D ViZDoom navigation with first-person inputs (Wydmuch et al., 2019), where the model plans based on noisy maze matrix predictions (Appendix C.5). DT-VIN outperforms all compared methods in these cases.

Table 2: The success rates of the methods on Point Maze and Ant Maze tasks.

	Point Maze		**Ant Maze**	
	35×35	100×100	35×35	100×100
VIN	$67.42_{\pm 7.56}$	$38.19_{\pm 9.73}$	$64.85_{\pm 9.34}$	$31.67_{\pm 11.28}$
GPPN	$91.21_{\pm 1.94}$	$69.37_{\pm 2.85}$	$86.09_{\pm 3.46}$	$61.78_{\pm 6.12}$
Highway VIN	$89.63_{\pm 2.71}$	$66.45_{\pm 4.83}$	$82.34_{\pm 4.19}$	$52.91_{\pm 8.55}$
DT-VIN (ours)	$\mathbf{99.87_{\pm 0.12}}$	$\mathbf{98.04_{\pm 1.93}}$	$\mathbf{96.57_{\pm 2.61}}$	$\mathbf{93.26_{\pm 3.14}}$

Table 3: The success rates of the algorithms on rover navigation tasks with various image sizes.

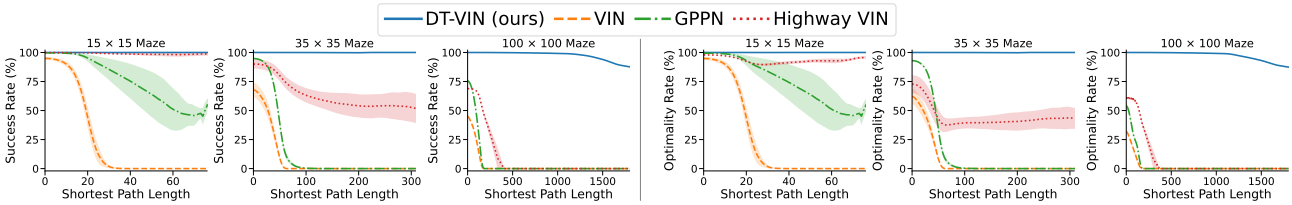
Image Sizes	270×270	450×450	630×630
VIN	$85.32_{\pm 0.14}$	$82.43_{\pm 0.74}$	$71.71_{\pm 3.48}$
GPPN	$85.79_{\pm 0.31}$	$81.72_{\pm 0.22}$	$76.31_{\pm 0.75}$
Highway VIN	$85.81_{\pm 0.6}$	$81.88_{\pm 0.86}$	$73.21_{\pm 0.81}$
DT-VIN (ours)	$\mathbf{86.54_{\pm 0.5}}$	$\mathbf{82.78_{\pm 0.6}}$	$\mathbf{77.4_{\pm 0.98}}$
CNN+A^*	$84.42_{\pm 0.67}$	$82.11_{\pm 0.74}$	$76.19_{\pm 1.23}$

4.2. Continuous Control

We evaluate the algorithms on two continuous control tasks: Point Maze and Ant Maze (Foundation, 2022; Fu et al., 2020). These tasks are more challenging than simple maze navigation: the agent must not only conduct (explicit or implicit) planning over the complex maze but also output the torques to direct the controlled agent (a 2-DOF ball or a 3D quadruped robot) toward the correct direction, as shown in Figure 4. The input includes the maze's top-down view, agent and goal locations, positional values, and velocities of the agent's body parts. The network architecture integrates a planning module structurally similar to that used in 2D maze navigation. The outputs of this module, combined with additional state information, are aggregated and passed to the policy mapping module to produce the control action. We train the model using the D4RL offline Point/Ant Maze dataset, which features a fixed 9×12 maze layout across all tasks, varying only in start and goal positions. However, during evaluation, the model is tested on substantially larger and more complex mazes (35×35 and 100×100) with previously unseen layouts and start/goal configurations, significantly increasing task complexity and demanding strong generalization capabilities. To address this challenge, we pretrain the planning module on a diverse and readily available 2D maze navigation dataset to endow the model with transferable planning skills. Please refer to Appendix D for more details. Table 2 shows the results of this experiment. DT-VIN solves the mazes at a much higher rate than all the baseline methods. Specifically, in the challenging 100×100 Ant Maze, DT-VIN achieves a notable 51% higher success rate than the second-best baseline.

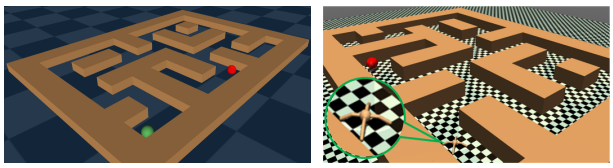
4.3. Rover Navigation

We further evaluate the algorithms on the rover navigation task, where the algorithm conduct planning based on a *ter-*



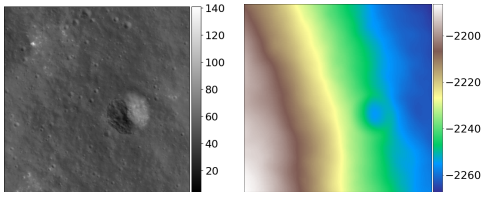
(a) Success Rate (b) Optimality Rate

Figure 3: SRs and ORs for different algorithms as a function of the shortest path length on 2D maze navigation tasks with various sizes. For each algorithm, we select the best result across various depths. Specifically, for our DT-VIN, the optimal depth consistently corresponds to the maximum value in the range: 200 for mazes of size 15, 600 for size 35, and 5000 for size 100. For other methods, the optimal depth differs per task. In the maze of size 100, the optimal depth for all the baselines is 600. See Figure 9 and Figure 10 in Appendix C.2 for additional results at other depths.



(a) Point Maze (b) Ant Maze

Figure 4: The Point Maze and Ant Maze environment in the training dataset from D4RL (Foundation, 2022; Fu et al., 2020), where map size is 9×12. Please refer to Appendix D for larger (e.g., 100×100) mazes in our testing dataset.



(a) Terrain image (b) Elevation data

Figure 5: (a) and (b) show a patch of terrain image and elevation data from Apollo 17 landing tasks.

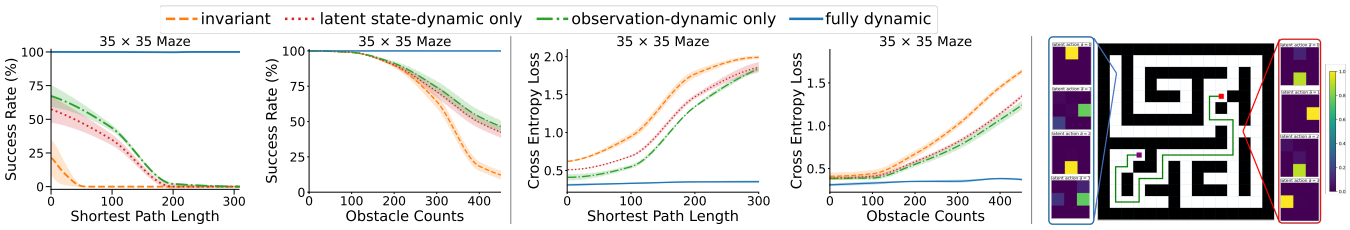
rain image (e.g., Figure 5(a)) rather than the *elevation data* (e.g., Figure 5(b)). While terrain images, visual representations from aerial photographs, typically lack elevation information, they are generally more readily available or less expensive to obtain. We evaluate the Apollo 17 landing tasks, featuring images with a resolution of 0.5 meters per pixel (Berlin, 2018). We crop the terrain image into various-sized patches, each 18×18 patch defining a cell. The cell is considered an obstacle if the associated area exhibits an elevation angle exceeding 10 degrees. Analogous to supervised learning, the training phase uses external elevation data to generate the expert path, while in the testing phase, this data is inaccessible to the models and only used to assess performance metrics. Please refer to Appendix E for details on the task setting and the models' architecture.

As shown in Table 3, our DT-VIN outperforms all compared methods across various terrain image sizes. Notably, with larger image sizes (particularly 630×630), our DT-VIN outperforms VIN by more than 5%. We also compare with a special baseline, CNN+A^*, which combines classification and classical planning algorithms. This method trains a CNN to classify whether an 18×18 image patch is an obstacle using elevation data, then uses A^* to conduct planning based on this prediction. While this unsurprisingly is able to outperform VIN, it is still outperformed by DT-VIN.

4.4. Ablation Study

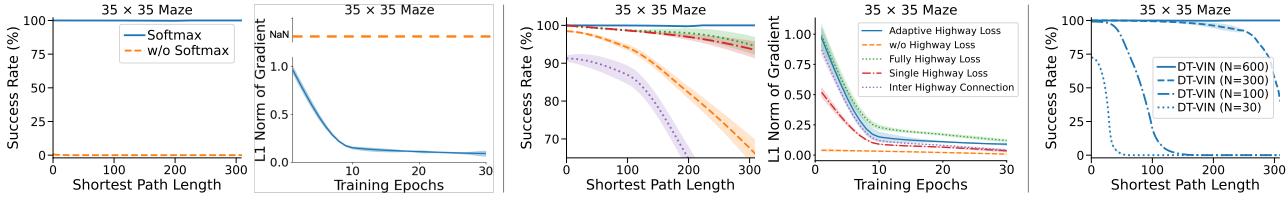
We perform multiple ablation studies with a $M = 35$ maze and an NN with depth $N = 600$ to assess the impact of DT-VIN components, including the dynamic latent transition kernel and its associated softmax operation, the adaptive highway loss and the network depth. Unless otherwise indicated, all these elements are included.

Dynamic Latent Transition Kernel. We evaluate four variants of dynamic latent transition kernels: *fully invariant*, *observation-dynamic only*, *latent state-dynamic only*, *fully dynamic* (incorporating both observation and latent state-dynamic). Figure 6(a) and (b) show the SRs of all these variants w.r.t. the SPLs and obstacle counts respectively. The variant with invariant kernels performs the worst. Both observation-dynamic and latent state-dynamic components contribute to the performance of DT-VIN – removing either results in a substantial performance drop. It's worth noting that the performance gap between the dynamic kernel and invariant kernel grows with increasing the SPL or obstacle counts. This improvement is due to the improved model's representational capacity by the dynamic kernel, reducing compounded errors over extended planning horizons and complex environments with increased obstacle counts, as shown in Figure 6(c) and (d). Figure 6(e) gives an illustra-



(a) SRs w.r.t. SPLs (b) SRs w.r.t. Obstacle Counts (c) Loss w.r.t. SPLs (d) Loss w.r.t. Obstacle Counts (e) Learned Kernels

Figure 6: Ablation studies on the **dynamic latent transition kernel**. (a) and (b) show the success rates (SRs) w.r.t. to various shortest path lengths (SPLs) and obstacle counts of the tasks, respectively. (c) and (d) show the losses. (e) shows the learned latent transition kernels of DT-VIN.



(a) Softmax Operation: Success Rate(Left), Gradient Norm(Right) (b) Adaptive Highway Loss: Success Rate(Left), Gradient Norm(Right) (c) Depths of Model

Figure 7: (a) and (b) show ablation on **softmax operation** and **adaptive highway loss**, respectively. The left and right sides of each sub-figure show the success rate, and the averaged L1 norm of the gradient over the first 10 layers, respectively. (c) show the the success rate across various **depths of model**.

tion of DT-VIN's dynamic transition kernels.

Softmax Operation on Latent Transition Kernel. As shown in Figure 7(a) (left), the variant without the softmax operation on the latent transition kernel fails on all the tasks. This failure is due to exploding gradients, wherein the gradient becomes extremely large, eventually resulting in the model's parameters overflowing and becoming a NaN (Not a Number) value, as depicted in Figure 7(a) (right).

Adaptive Highway Loss. We evaluate several additional variants to verify the effectiveness of adaptive highway loss. Figure 7(b) (left) shows the success rate of all variants. **(a)** The variant *without highway loss* suffer a decrease in performance, as shown in Figure 7(b). **(b)** We also evaluate a variant named *fully highway loss*, which enforces a highway loss on each hidden layer without adaptive adjustment by removing the filter $\mathbb{1}_{\{n\geq l\}}$ in Eq. (3). This variant performs worse than the one with adaptive highway loss, highlighting the necessity of adaptive adjustment. **(c)** We assess a variant called *single highway loss*, which only apply highway loss to a specific layer n where $n = l$ by substituting $\mathbb{1}_{\{n\geq l\}}$ with $\mathbb{1}_{\{n=l\}}$ in Eq. (3). This variant performs worse than the adaptive highway loss, demonstrating the critical role of the component $n > l$ in performance. **(d)** Finally, we evaluate a variant named *inter highway connection*, which constructs skip connections across the intermediate layers of the planning module, akin to techniques used in Highway Nets (Srivastava et al., 2015a). This variant performs the worst because the skip connections over intermediate layers can disrupt the structure of value iteration. These results are consistent with those in existing work (Wang et al., 2024a).

Depth of Planning Module. Figure 7(c) shows the SRs of DT-VIN with various depths. Here, increasing the depth dramatically improves the long-term planning ability. For example, for tasks with an SPL of 200, the variant with depth $N = 300$ outperforms the variant with depth $N = 100$. Moreover, for tasks with an SPL of 300, the deeper variant with depth $N = 600$ excels greatly. Other methods like VIN and GPPN do not show a clear performance improvement when the depth increases, please refer to Figure 9 in Appendix C.2 for more details.

Additional Ablation Studies. Due to space constraints, please refer to Appendix F and G, which show the robustness of DT-VIN with 50% of the training dataset, the choice of the length of the expert path l, impact of jumping hyperparameter J, and the scaling of computation complexity, dataset requirements, and model size with task scale.

5. Related Work

Variants of Value Iteration Networks (VINs). Several variants of VIN (Tamar et al., 2016) have been proposed in recent years. Gated Path Planning Networks employ gating recurrent mechanisms to reduce the training instability and hyperparameter sensitivity seen in VINs (Lee et al., 2018).

To mitigate overestimation bias (which is detrimental to learning here), dVINs were proposed and use a weighted double estimator as an alternative to the maximum operator (Jin et al., 2021). For addressing challenges in irregular spatial graphs, Generalized VINs adopt a graph convolution operator, extending the traditional convolution operator used in VINs (Niu et al., 2018). To improve scalability, AVINs introduce an abstraction module that extracts higher-level information from the environment and the goal (Schleich et al., 2019). To improve scalability, Zhao et al. proposed implicit differentiable planners that decouple forward and backward passes (Zhao et al., 2023). In contrast, our work takes a complementary architectural approach, redesigning VINs for deeper and more expressive planning over larger-scale tasks. For transfer learning, Transfer VINs address the generalization of VINs to target domains where the action space or the environment's features differ from those of the training environments (Shen et al., 2020). More recently, VIRN was proposed and employs larger convolutional kernels to plan using fewer iterations as well as self-attention to propagate information from each layer to the final output of the network (Cai et al., 2022). Similarly, GS-VIN also uses larger convolutional kernels but to stabilize training and also incorporates a gated summarization module that reduces the accumulated errors during value iteration (Cai et al., 2023). Most related to DT-VIN is other recent work that focused on developing very deep VINs for long-term planning. Specifically, Highway VIN (Wang et al., 2024a) incorporates the theory of Highway Reinforcement Learning (Wang et al., 2024b) to create deep planning networks with up to 300 layers for long-term planning tasks. Highway VIN modifies the planning module of VIN by introducing an exploration module that injects stochasticity in the forward pass and uses gating mechanisms to allow selective information flow through the network layers. Our method, however, achieves even deeper planning by incorporating a dynamic transition matrix in the latent MDP and adaptively weighting each layer's connection to the final output.

Neural Networks with Deep Architectures. There is a long history of developing very deep neural networks (NNs). For sequential data, this prominently includes the LSTM architecture and its gated residual connections, which help alleviate the "vanishing gradient problem" (Hochreiter & Schmidhuber, 1997; Hochreiter, 1991). For feedforward NNs, a similar gated residual connection architecture was used in Highway Networks (Srivastava et al., 2015b) and later in the ResNet architecture (He et al., 2016), where the gates were kept open. Such residual connections are still ubiquitous in modern language architectures, such as the Generative Pre-trained Transformer (GPT) (Achiam et al., 2023). Our method dynamically employs skip connections from select hidden layers to the final loss, utilizing a state and observation map-dependent transition kernel. This approach is more closely aligned with the computation of the true VI algorithm. Similar kernels, dependent on an input image (Chen et al., 2020) or the coordinates of an image (Liu et al., 2018), have been previously used in Computer Vision.

6. Conclusions

Previous work introduced VIN as a differentiable neural network for planning in artificial intelligence and reinforcement learning. While VINs have been successful at short-term small-scale planning, they start to fail quite rapidly as the planning horizon and the scale of the tasks grows. We observed that this decay in performance is principally due to limitations in the (1) representational capacity of their network and (2) its depth. To alleviate these problems, we propose several modifications to the architecture, including a dynamic transition kernel to increase the representation capacity and an adaptive highway loss function to ease the training of very deep models. Altogether, these modifications have allowed us to train networks with 5000 layers. In line with previous work, we evaluate the efficacy of our proposed Dynamic Transition VINs (DT-VINs) on 2D/3D maze navigation, continuous control, and rover navigation. We find that DT-VINs scale to longer-term and larger-scale planning problems than previous attempts. To the best of our knowledge, DT-VINs is, at the time of publication, the current state-of-the-art planning solution for these specific environments. We note that the upper bound for this approach (i.e., the scale of the network and, consequently, the scale of the planning ability) remains unknown. As our experiments were limited mostly by computational cost and did not observe instability, we expect that with the growth of available computational power, our method will scale to even longer-term and larger-scale planning.

Acknowledgements

We gratefully acknowledge the insightful comments from the ICML 2025 reviewers, as well as the constructive feedback received during earlier stages of peer review. We are especially thankful to Yanning Dai for her valuable support with the continuous control experiments. The research reported in this publication was supported by funding from King Abdullah University of Science and Technology (KAUST) - Center of Excellence for Generative AI, under award number 5940. We also acknowledge the support by the grant EP/Y002644/1 under the EPSRC ECR International Collaboration Grants program, funded by the International Science Partnerships Fund (ISPF) and the UK Research and Innovation. This work was additionally supported by the European Research Council (ERC, Advanced Grant Number 742870). For computer time, this research used Ibex managed by the Supercomputing Core Laboratory at King Abdullah University of Science and Technology (KAUST) in Saudi Arabia.

Impact Statement

This paper aims to advance the field of Machine Learning. While acknowledging there are many potential societal consequences of our work, we believe that none need to be specially highlighted here.

References

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

Agarwal, A., Jiang, N., Kakade, S. M., and Sun, W. Reinforcement learning: Theory and algorithms. *CS Dept., UW Seattle, Seattle, WA, USA, Tech. Rep*, 32, 2019.

Bain, M. and Sammut, C. A framework for behavioural cloning. In *Machine Intelligence 15*, pp. 103–129, 1995.

Bellman, R. E. A markovian decision process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. URL http://www.jstor.org/stable/24900506.

Berlin, T. U. Moon apollo 17 lroc nac landing site orthomosaic 50cm, 2018. URL https://astrogeology.usgs.gov/search/map/moon_apollo_17_lroc_nac_landing_site_orthomosaic_50cm. Accessed: 2024-10-02.

Cai, J., Li, J., Mao, Z., and Tei, K. Value iteration residual network with self-attention. In *International Conference on Intelligent Systems Design and Applications*, pp. 16–24. Springer, 2022.

Cai, J., Li, J., Zhang, M., and Tei, K. Value iteration networks with gated summarization module. *IEEE Access*, 2023.

Chen, Y., Dai, X., Liu, M., Chen, D., Yuan, L., and Liu, Z. Dynamic convolution: Attention over convolution kernels. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 11030–11039, 2020.

Chen, Y. F., Everett, M., Liu, M., and How, J. P. Socially aware motion planning with deep reinforcement learning. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1343–1350. IEEE, 2017.

Eysenbach, B., Salakhutdinov, R. R., and Levine, S. Search on the replay buffer: Bridging planning and reinforcement learning. *Advances in neural information processing systems*, 32, 2019.

Foundation, F. Gymnasium: A toolkit for developing and comparing reinforcement learning environments. https://github.com/Farama-Foundation/Gymnasium, 2022.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Hafner, D., Lillicrap, T. P., Ba, J., and Norouzi, M. Dream to control: Learning behaviors by latent imagination. *Proceedings of the 8th International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=S1lOTC4tDS.

Hafner, D., Lillicrap, T. P., Norouzi, M., and Ba, J. Mastering atari with discrete world models. *Proceedings of the 9th International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=0oabwyZbOu.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. P. *Mastering Diverse Domains through World Models*. arXiv, 2023. URL https://arxiv.org/abs/2301.04104.

Hart, P. E., Nilsson, N. J., and Raphael, B. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. doi: 10.1109/TSSC.1968.300136.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Hochreiter, S. Untersuchungen zu dynamischen neuronalen Netzen. Diploma thesis, Institut für Informatik, Lehrstuhl Prof. Brauer, Technische Universität München, 1991. Advisor: J. Schmidhuber.

Hochreiter, S. and Schmidhuber, J. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 1997.

Janner, M., Du, Y., Tenenbaum, J., and Levine, S. Planning with diffusion for flexible behavior synthesis. In *International Conference on Machine Learning*, pp. 9902–9915. PMLR, 2022.

Jin, X., Lan, W., Wang, T., and Yu, P. Value iteration networks with double estimator for planetary rover path planning. *Sensors*, 21(24):8418, 2021.

Lample, G. and Chaplot, D. S. Playing fps games with deep reinforcement learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

Lee, L., Parisotto, E., Chaplot, D. S., Xing, E., and Salakhutdinov, R. Gated path planning networks. In *International Conference on Machine Learning*, pp. 2947–2955. PMLR, 2018.

Li, W., Yang, B., Song, G., and Jiang, X. Dynamic value iteration networks for the planning of rapidly changing uav swarms. *Frontiers of Information Technology & Electronic Engineering*, 22(5):687–696, 2021.

Liu, R., Lehman, J., Molino, P., Petroski Such, F., Frank, E., Sergeev, A., and Yosinski, J. An intriguing failing of convolutional neural networks and the coordconv solution. *Advances in neural information processing systems*, 31, 2018.

Mishra, U. A., Xue, S., Chen, Y., and Xu, D. Generative skill chaining: Long-horizon skill planning with diffusion models. In *Conference on Robot Learning*, pp. 2905–2925. PMLR, 2023.

Niu, S., Chen, S., Guo, H., Targonski, C., Smith, M., and Kovačević, J. Generalized value iteration networks: Life beyond lattices. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Pflueger, M., Agha, A., and Sukhatme, G. S. Rover-irl: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning. *IEEE Robotics and Automation Letters*, 4(2):1387–1394, 2019.

Puterman, M. L. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014.

Ross, S., Gordon, G., and Bagnell, D. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Schaal, S. Learning from demonstration. *Advances in neural information processing systems*, 9, 1996.

Schleich, D., Klamt, T., and Behnke, S. Value iteration networks on multiple levels of abstraction. *arXiv preprint arXiv:1905.11068*, 2019.

Schmidhuber, J. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. Technical Report FKI-126-90 (revised), Institut für Informatik, Technische Universität München, November 1990a. (Revised and extended version of an earlier report from February.).

Schmidhuber, J. An on-line algorithm for dynamic reinforcement learning and planning in reactive environments. In *Proc. IEEE/INNS International Joint Conference on Neural Networks, San Diego*, volume 2, pp. 253–258, 1990b.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020.

Shen, J., Zhuo, H. H., Xu, J., Zhong, B., and Pan, S. Transfer value iteration networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 5676–5683, 2020.

Silver, D., van Hasselt, H., Hessel, M., Schaul, T., Guez, A., Harley, T., Dulac-Arnold, G., Reichert, D. P., Rabinowitz, N. C., Barreto, A., and Degris, T. The predictron: End-to-end learning and planning. *Proceedings of the 34th International Conference on Machine Learning*, 70:3191–3199, 2017. URL http://proceedings.mlr.press/v70/silver17a/silver17a.pdf.

Song, C. H., Wu, J., Washington, C., Sadler, B. M., Chao, W.-L., and Su, Y. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 2998–3009, 2023.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015a.

Srivastava, R. K., Greff, K., and Schmidhuber, J. Training very deep networks. *Advances in neural information processing systems*, 28, 2015b.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM SIGART Bulletin*, 2(4): 160–163, 1991. doi: 10.1145/122344.122377.

Sutton, R. S. *The Bitter Lesson*. 2019. URL http://incompleteideas.net/IncIdeas/BitterLesson.html.

Tamar, A., Levine, S., Abbeel, P., Wu, Y., and Thomas, G. Value iteration networks. *Advances in Neural Information Processing Systems*, 29:2146–2154, 2016.

Wang, Y., Li, W., Faccio, F., Wu, Q., and Schmidhuber, J. Highway value iteration networks. *Proceedings of the 41st International Conference on Machine Learning*, 2024a. URL https://arxiv.org/abs/2406.03485.

Wang, Y., Liu, H., Strupl, M., Faccio, F., Wu, Q., Tan, X., and Schmidhuber, J. *Highway Reinforcement Learning*. arXiv, 2024b. URL https://arxiv.org/abs/2405.18289.

Wöhlke, J., Schmitt, F., and van Hoof, H. Hierarchies of planning and reinforcement learning for robot navigation. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 10682–10688. IEEE, 2021.

Wydmuch, M., Kempka, M., and Jaśkowski, W. ViZDoom Competitions: Playing Doom from Pixels. *IEEE Transactions on Games*, 11(3):248–259, 2019. doi: 10.1109/TG.2018.2877047.

Zhao, L., Xu, H., and Wong, L. L. Scaling up and stabilizing differentiable planning with implicit differentiation. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PYbe4MoHf32.

A. Limitations and Future Works

The principal limitation of our work compared to VIN and Highway VIN is the increased computational cost (see Appendix G). This is a consequence of the scale of the network. The past decades have seen AI dominated by the trend of scaling up systems (Sutton, 2019), so this is not likely a long-term issue.

Future work will explore the impact of a more sophisticated transition mapping module (this work uses a single CNN layer for this purpose) in more challenging real-world applications, such as real-time robotics navigation in dynamic and unpredictable environments. Additionally, recent diffusion-based planners (Janner et al., 2022; Mishra et al., 2023) offer a complementary approach to long-horizon reasoning, which may be integrated with VIN-style architectures to improve planning capacity and temporal abstraction.

B. Method: Additional Details

B.1. Loss Function

In imitation learning, the loss function in Eq. (3) can be written as

$$\mathcal{L}(\theta) = \frac{1}{K|\mathcal{D}|} \sum_{(x,y,l)} \sum_{n} \mathbb{1}_{\{n \geq l\}} \mathbb{1}_{\{n \bmod J = 0\}} \left(- \sum_{a} \mathbb{1}_{\{a=y\}} \log f^{\pi} \left(\overline{V}^{(n)}(x), a \right) \right).$$

In RL, where the policies are learned through policy gradient, the loss function can be rewritten as

$$\mathcal{L}(\theta) = \frac{1}{K|\mathcal{D}|} \sum_{(x,y,R,l) \in \mathcal{D}} \sum_{1 \leq n \leq N} \mathbb{1}_{\{n \geq l\}} \mathbb{1}_{\{n \bmod J = 0\}} \Big(- R \log f^{\pi} \left(\overline{V}^{(n)}(x), y \right) \Big),$$

where y is the excuted action, and R the cumulative future reward.

B.2. Translation Equivariance

Although our dynamic latent transition kernel is input-dependent and breaks the conventional weight-sharing scheme, it still preserves translation equivariance—a desirable property in spatial planning tasks. In other words, shifting the input (e.g., the maze image) leads to a corresponding shift in the output. While this may seem counterintuitive at first, it is in fact a direct consequence of our design: the kernel $\overline{\mathsf{T}}^{\mathrm{dyn}} = f^{\overline{\mathsf{T}}}(x)$ is produced by a CNN, which is itself translation equivariant. As a result, the downstream value iteration process retains this property (see Eq. 2).

C. Maze Navigation: Additional Experimental Details and Results

C.1. 2D Maze Navigation: Experimental Details

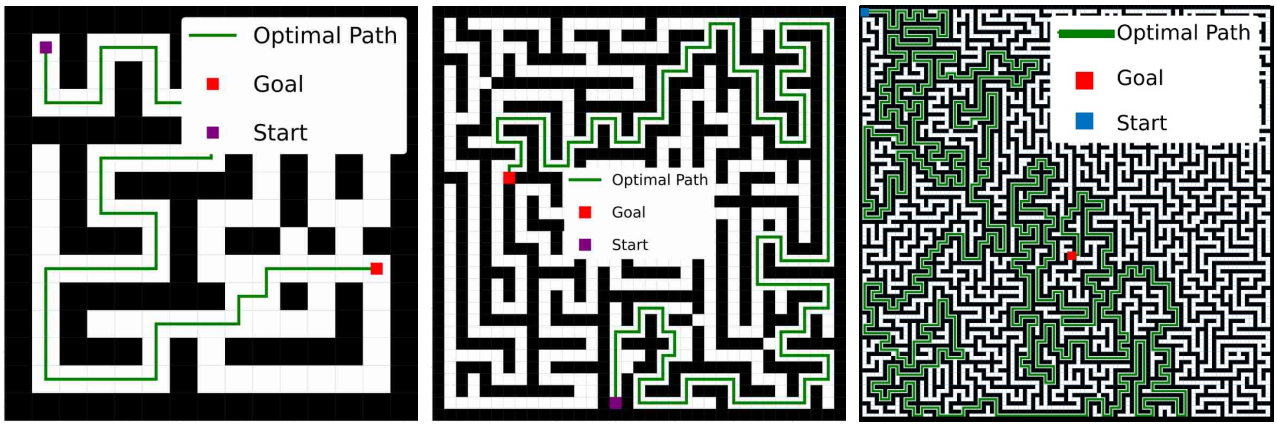
Figure 8 shows some visualizations of some of the different 2D maze navigation tasks we experiment with. Our experimental setup follows the guidelines established in the GPPN paper (Lee et al., 2018). For these tasks, the datasets for training, validation, and testing comprise 25000, 5000, and 5000 mazes, respectively. Each maze features a goal position, with all reachable positions selected as potential starting points. Note that this setting, as done by GPPN, produces a distribution of mazes with non-uniform SPLs, which is skewed towards shorter SPLs. Table 4 shows the hyperparameters used by our method. Note that, while DT-VIN consistently uses 3 for the size of the latent transition kernel F and 4 for the size of the latent action space $|\overline{\mathcal{A}}|$, other methods instead used their best-performing sizes from between 3 and 5, and between 4 and 150, respectively.

C.2. 2D Maze Navigation with Different Depths of Models

Figure 9 shows the success rate of all the algorithms on the 15×15, 35×35, 100×100 mazes as a function of the shortest path length and the depth of the network. Similarly, Figure 10 shows the corresponding optimality rates.

C.3. 2D Maze Navigation with Different Transition Types

Following the GPPN paper (Lee et al., 2018), we run additional experiments using different transition type: the *Differential Drive* transition, where the agent can move forward along its orientation or rotate 90 degrees left or right, and the *MOORE*



(a) 15×15 Maze (b) 35×35 Maze (c) 100×100 Maze

Figure 8: Examples of the 2D maze navigation tasks.

Table 4: 2D Maze Navigation Hyperparameters

Hyperparameter	Value
Transition Mapping Module	A 1-layer CNN with 3×3 kernel
Reward Mapping Module	A 1-layer CNN with 1×1 kernel
Policy Mapping Module	A 1-layer FCN
Latent Transition Kernel Size (F)	3
Latent Action Space Size ($\lvert\overline{\mathcal{A}}\rvert$)	4
Optimizer	RMSprop
Learning Rate	1e-3
Batch Size	32
Depth of Planning Module	15×15 maze: 200 35×35 maze: 600 100×100 maze: 5000

transition, where the agent can relocate to any of the eight adjacent cells that comprise its Moore neighborhood. As shown in Table 8 and 9, DT-VIN consistently outperforms all the compared methods regardless of the transition type used.

C.4. 2D Maze Navigation with Controlled Noisy Maze

We evaluated the methods under controlled noise. To emulate the prediction noise, we add Gaussian noise to the original maze and enforce the value within $(0, 1)$ by $clip(maze + \epsilon, 0, 1)$, where ϵ is a noise sampled from Gaussian distribution. The results of this are below. DT-VIN seems more robust in handling noise than Highway VIN and VIN. GPPN shows robustness in the short term setting, but is not able to solve the task when long term planning is required.

C.5. 3D ViZDoom Navigation

Following the methodology of the GPPN paper, we test our method on 3D ViZDoom (Wydmuch et al., 2019) environments. Here, instead of directly using the top-down 2D maze as in the previous experiments, we use the observation consists of RGB images capturing the first-person perspective of the environment, as illustrated in Figure 11(a). Then, a CNN is trained to predict the maze map from the first-person observation. The map is then given as input to the planning model, using the same architecture and hyperparameters as the 2D maze environments. For each algorithm, we select the best result across the various network depths $N = 30, 100, 300, 600$. We find that the optimal depth for DT-VIN is 600, for GPPN is 300, for VIN is 300, and for Highway VIN is 300. We evaluated the algorithm on 3D ViZDoom mazes with grid 35×35, where each cell in the grid corresponds to a 64×64 map unit area, the standard spatial measurement in the game engine. Figure 11(b) shows the SRs. Predictably, the performance of all the baselines decreases compared to the 2D maze environments due to

Table 5: Rover Navigation Hyperparameters

Hyperparameter	Value
Transition Mapping Module	A 10-layer CNN
Reward Mapping Module	A 10-layer CNN (sharing the first 8 layers with Transition Mapping Module)
Policy Mapping Module	A 1-layer FCN
Latent Transition Kernel Size (F)	3
Latent Action Space Size ($\lvert\overline{\mathcal{A}}\rvert$)	4
Optimizer	RMSprop
Learning Rate	1e-3
Batch Size	32
Depth of Planning Module	$270 \times 270 : 50$ $450 \times 450 : 100$ 630×630: 200

Table 6: 3D ViZDoom Preprocessing Network

Hyperparameter	Value
Batch Size (B)	32
Image Directions (D)	4
Image Channels (C)	3
Image Width (W)	24
Image Height (H)	32
Input Size	(B, M, M, D, W, H, C)
Layer 1 (Convolution)	$(3, 32, 8, 4, 1)$
Layer 2 (Convolution)	$(32, 64, 4, 2, 1)$
Layer 3 (Linear)	$(384, 256)$
Layer 4 (Convolution)	$(1024, 64, 3, 1, 1)$
Layer 5 (Convolution)	$(64, 1, 3, 1, 1)$
Output Size	(B, M, M)
Optimizer	Adam
Learning Rate	1e-3
Betas	$(0.9, 0.999)$

Table 7: The success rates for each method under tasks with different ranges of shortest path length. For each algorithm, we choose the best result from a range of depths. Specifically, for our DT-VIN, the optimal depth consistently corresponds to the maximum value in the range: 600 for size 35, and 5000 for size 100. For other compared methods, the optimal depth differs depending on the task. In the maze of size 100, the optimal depth for all the baselines is 600. For additional results, see Figure 9 in Appendix C.2.

Maze Size	35×35			100×100		
Ranges of Shortest Path Lengths	[1,100]	[100, 200]	[200, 300]	[1,600]	[600, 1200]	[1200, 1800]
VIN (Tamar et al., 2016)	$68.41_{\pm 6.25}$	$0.0_{\pm 0.00}$	$0.00_{\pm 0.00}$	$45.05_{\pm 0.04}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
GPPN (Lee et al., 2018)	$95.71_{\pm 0.33}$	$0.39_{\pm 0.27}$	$0.00_{\pm 0.00}$	$75.72_{\pm 0.64}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
Highway VIN (Wang et al., 2024a)	$90.67_{\pm 3.92}$	$65.50_{\pm 5.59}$	$54.40_{\pm 10.2}$	$69.12_{\pm 0.02}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
DT-VIN (ours)	$\mathbf{100.00_{\pm 0.00}}$	$\mathbf{99.99_{\pm 0.01}}$	$\mathbf{99.77_{\pm 0.23}}$	$\mathbf{99.98_{\pm 0.00}}$	$\mathbf{99.56_{\pm 0.20}}$	$\mathbf{88.65_{\pm 4.76}}$

Table 8: The success rate (%) for each method in 35×35 2D maze navigation with *Differential Drive* transition type, where the agent can move forward along its orientation or rotate 90° left or right.

Shortest Path Length	[1,150]	[150,300]	[300,500]
VIN	$68.44_{\pm 3.12}$	$0.03_{\pm 0.01}$	$0.00_{\pm 0.00}$
GPPN	$83.1_{\pm 1.23}$	$0.31_{\pm 0.01}$	$0.0_{\pm 0.0}$
Highway VIN	$87.1_{\pm 3.73}$	$57.1_{\pm 3.98}$	$49.1_{\pm 8.73}$
DT-VIN (ours)	$\mathbf{100.00_{\pm 0.00}}$	$\mathbf{100.00_{\pm 0.00}}$	$\mathbf{99.99_{\pm 0.01}}$

the additional noise introduced by the predictions. Here, DT-VIN outperforms all the methods compared to the task over all the various SPLs.

To be in line with previous work, we use a state representation preprocessing stage for the 3D ViZDoom environment similar to that used in the GPPN paper and others (Lee et al., 2018; Lample & Chaplot, 2017). In 3D ViZDoom, a maze is designed on a grid of $M \times M$ cells. Each cell in this grid corresponds to an area of 64×64 map units within the 3D ViZDoom environment. The map unit is the basic measure of space used in the ViZDoom game engine to define distances and sizes. Specifically, for each cell in the $M \times M$ 3D maze, the RGB first-person views for each of the four cardinal directions are given as state to a preprocessing network (see Figure 11(a)). This network then encodes this state and produces an $M \times M$ binary maze matrix. The hyperparameters and exact specification of the network are given in Table 6.

D. Continuous Control: Additional Experimental Details and Results

In this section, we present experimental details of continuous control, including Point Maze and Ant Maze. Figure 12 demonstrates Point Maze with larger sizes 35×35 and 100×100. For continuous control tasks, the input includes a $4M \times 4M$ top-down view of the maze, agent and goal locations, positional values, and velocities of the agent's body parts. Network architecture comprises 2-layer CNNs for reward and transition mapping, and a 3-layer fully connected network for policy mapping, which takes outputs from the planning module and additional state information to produce controlled actions. The planning module in this architecture overlaps with that in the 2D maze navigation task. Therefore, we use pre-trained parameters from 2D maze navigation tasks. Table 12 lists the hyperparameters of DT-VIN for the continuous control tasks. We train using PointMaze-Large and AntMaze-Large-Play datasets from D4RL (Fu et al., 2020) and pre-train on the 100×100 2D maze.

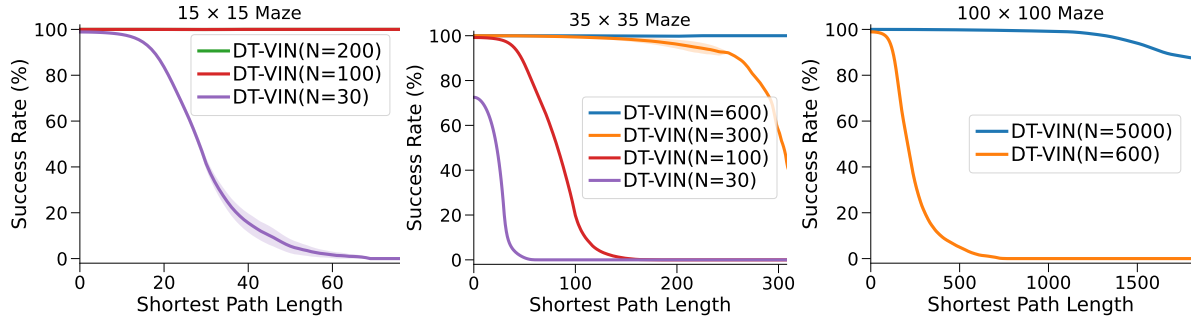
E. Rover Navigation: Additional Experimental Details and Results

Table 5 shows the hyperparameters of DT-VIN for the rover navigation tasks. For the transition and reward mapping modules, we employ 10-layer CNNs, with the first 8 layers shared between them.

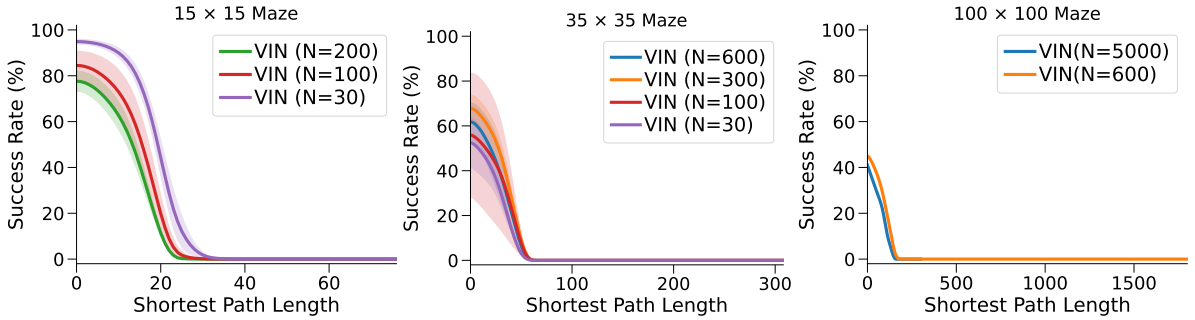
F. Ablation Studies: Additional Experimental Details and Results

F.1. Ablation on the Jumping Hyperparameter J

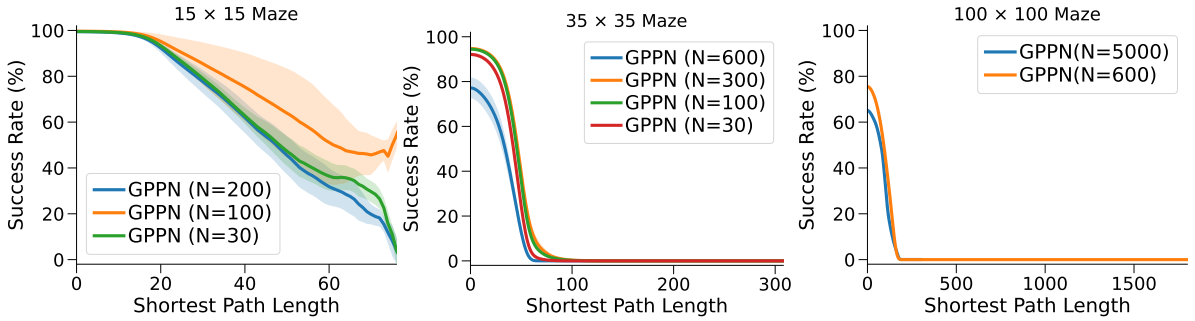
To reduce the computational complexity of highway loss, we apply adaptive highway loss only to layers n satisfying the condition $n \bmod J = 0$, where J is a hyperparameter set to 10 in our experiments. Here, the main idea is to build the



(a) DT-VIN (ours)



(b) VIN



(c) GPPN

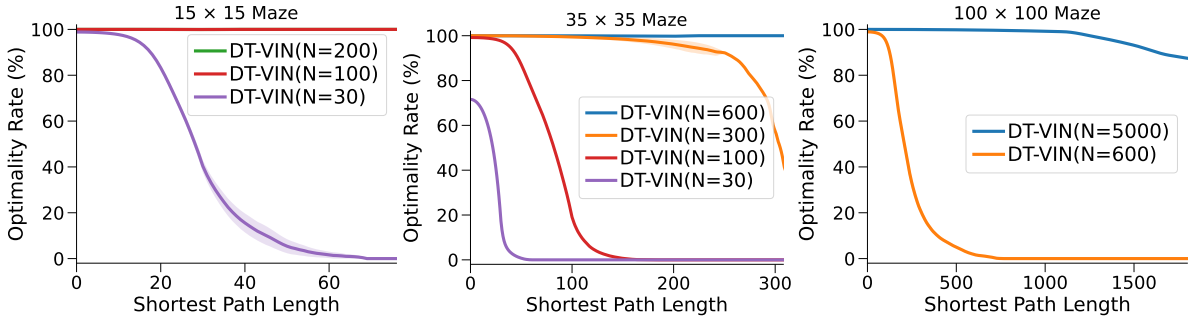
Figure 9: The success rate of each method as a function of shortest path length and network depth. The green and red curves overlap in the first plot of (a).

highway connections at an interval J — for example, every 10 neural network layers, as in our setting where $J = 10$. Using this, the number of the loss terms will reduce to only $1/J$ of the original one. Table 13 shows the magnitude of the computational speedup as a consequence of this implementation detail.

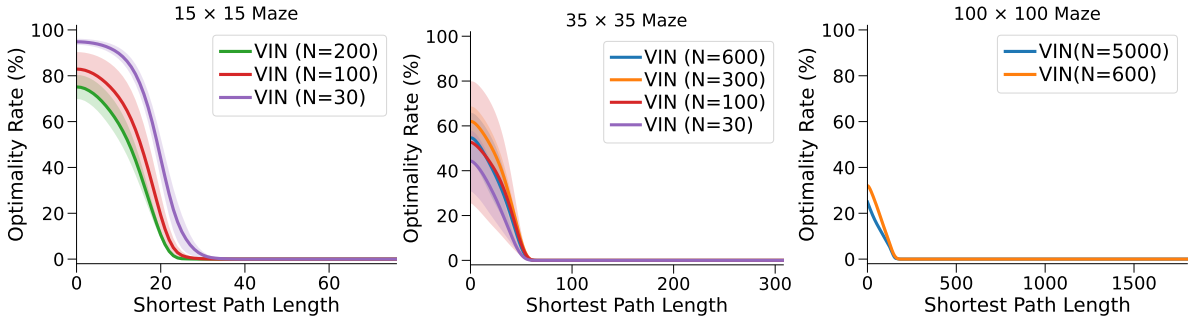
F.2. Ablation on the Choice of l

The knowledge of the length l of the expert path naturally exists in the imitation learning case. However, for the case where such information is unknown, one can use either the length of non-expert data or some heuristic methods to estimate l when the actual l is completely unknown, e.g., using the distance between the start and the goal position.

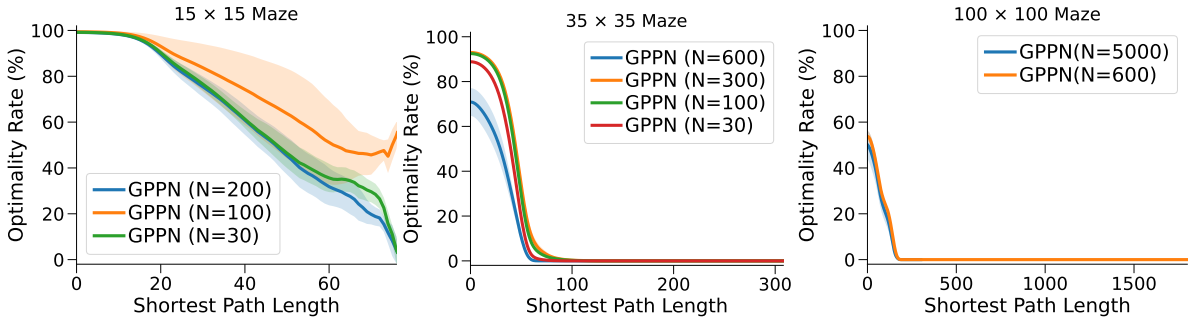
To measure the effect of overestimation/underestimation, we experiment with various estimated values of the length of the shortest path $\widehat{l}$, which are $0, l/2, l, 2l, N$ (where l is the actual length of the shortest path, N is the depth of the planning module). Second, to evaluate the case when the estimation of l has variance, we use $l \cdot \max(\epsilon, 0)$ as the estimation, with ϵ sampled from a Gaussian distribution $\mathcal{N}(1, 1)$. Third, we also assess two additional variants for estimating l: (a) One variant that utilizes the length of non-expert trajectories for l; (b) Another variant that estimates the shortest path length



(a) DT-VIN (ours)



(b) VIN



(c) GPPN

Figure 10: The optimality rate of each method as a function of shortest path length and network depth. The green and red curves overlap in the top-left plot.

heuristically using the L1 distance between the start (x_s, y_s) and the goal (x_g, y_g), i.e., $D = |x_s - x_g| + |y_s - y_g|$.

As indicated in Table 14, both overestimation and underestimation lead to a performance degradation of no more than 7%. Additionally, we find that leveraging non-expert data or the heuristic L1 distance only yields a nearly 3% degradation in performance, and performs better than the case when the optimal length is extremely overestimated/underestimated. These results imply that employing the information from non-expert data or heuristic estimation could be taken as an alternative when the optimal length is not available.

F.3. Ablation on the Size of Training Dataset

Even in situations where data is rare, DT-VIN still outperforms compared methods. As shown in Table 15, with only 50% of the original dataset, DT-VIN greatly outperforms existing methods. We also highlight the changes compared to the performance with a full-sized dataset in Table 15, where DT-VIN results in less than a 0.2% degradation for tasks within the range [1, 100], while the best-performing comparison method, GPPN, incurs a degradation of nearly 12%.

Table 9: The success rate for each method in 35×35 2D maze navigation with *Moore* transition type, where the agent can relocate to any of the eight adjacent cells that comprise its Moore neighborhood.

Shortest Path Length	[1,100]	[100,200]	[200,250]
VIN	$66.44_{\pm 3.21}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
GPPN	$89.94_{\pm 1.31}$	$0.04_{\pm 0.01}$	$0.00_{\pm 0.00}$
Highway VIN	$83.14_{\pm 2.21}$	$37.1_{\pm 1.98}$	$25.1_{\pm 3.28}$
DT-VIN (ours)	$\mathbf{100.0_{\pm 0.00}}$	$\mathbf{98.9_{\pm 0.72}}$	$\mathbf{96.7_{\pm 1.23}}$

Table 10: The success rate for each method in 35×35 2D maze navigation, with maze noise $\epsilon \sim \mathcal{N}(0, 0.01)$.

Shortest Path Length	[1,100]	[100,200]	[200,300]
VIN	$66.41_{\pm 7.25}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
GPPN	$91.71_{\pm 1.33}$	$0.21_{\pm 0.27}$	$0.00_{\pm 0.00}$
Highway VIN	$86.67_{\pm 4.92}$	$57.50_{\pm 6.59}$	$46.40_{\pm 11.2}$
DT-VIN (ours)	$\mathbf{99.21_{\pm 0.00}}$	$\mathbf{98.17_{\pm 0.01}}$	$\mathbf{97.77_{\pm 0.23}}$

G. Scaling Experiments

In this section, we examine how the methods scale in terms of computational complexity, dataset requirements, and model size relative to task scale.

G.1. Required Computational Resources

As we have discussed in Section 3.1, our approaches only require $|\overline{\mathcal{A}}| \times F^4$ parameters, where we set $|\overline{\mathcal{A}}| = 4$ and $F = 3$ in our experiments. Table 17 shows the memory consumption and training time on NVIDIA A100 GPUs for DT-VIN and the compared methods when using 5000 layers and training for 90 epochs on 100×100 maze. As shown in the table below, DT-VIN incurs only slightly higher GPU memory and time costs compared to VIN, while being far more memory-efficient than GPPN and Highway VIN. These results are generally consistent with those observed in the 35×35 2D maze in Table 16.

G.2. Required Size of Model

In our experiments, the depth of the network required to solve the problem is close to linear with the number of planning steps required by the problem. For maze size $M = 15, 25, 35$, we test DT-VIN models at increasing depths in increments of 100 until the optimal performance is achieved. For instance, for mazes of size 25×25, we assess depths of $100, 200, 300, 400$. For maze size $M = 100$, we assess depths of $4000, 5000, 6000$. As Table 18 illustrates, the depth of the smallest network that can solve the task increases slightly more than linearly with the required planning steps. Therefore, it might be feasible to continue increasing the network depth as the problems become more complex.

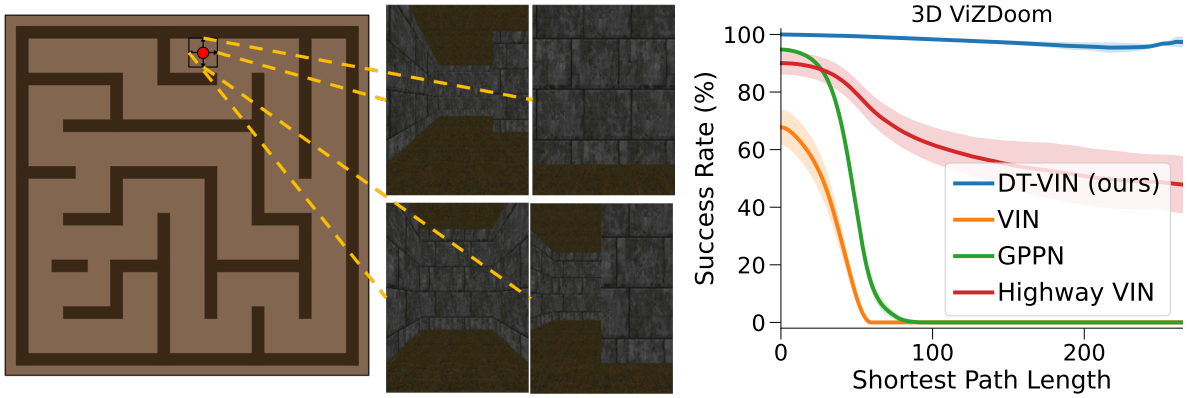
G.3. Required Size of Training Data

We evaluate the models on the mazes of size 15×15, 25×25, and 35×35, on increasingly larger numbers of expert steps (a step here is one transition from a cell to another cell). Specifically, for each maze size, we look for the smallest n such that, with $5000 \times n$ different mazes (with each maze having a number of expert trajectories and larger mazes having more expert trajectories), DT-VIN can achieve a $\geq 98\%$ success rate on the longest length planning problem for each maze. The results of this experiment are below (and we will add this to a camera-ready version of the paper). While this is a bit of a coarse result, comparing the length column to the expert data column, we see a growth rate that looks a more than linear, but still indicates a manageable degree of complexity increase. We would like to note that using more mazes in the training set with fewer expert trajectories for each maze might further increase the sample efficiency of our method and the baselines. Designing an appropriate curriculum for this could be an exciting area of future work.

The scale of the dataset needs to scale up with the complexity of the problem rather than the model depth. Under the same

Table 11: The success rate for each method in 35×35 2D maze navigation, with maze noise $\epsilon \sim \mathcal{N}(0, 0.04)$.

Shortest Path Length	[1,100]	[100,200]	[200,300]
VIN	$59.11_{\pm 7.94}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
GPPN	$87.43_{\pm 1.34}$	$0.15_{\pm 0.48}$	$0.00_{\pm 0.00}$
Highway VIN	$81.76_{\pm 4.74}$	$49.5_{\pm 6.05}$	$35.7_{\pm 11.3}$
DT-VIN (ours)	$\mathbf{95.17}_{\pm \mathbf{0.00}}$	$\mathbf{92.29}_{\pm \mathbf{0.57}}$	$\mathbf{91.99}_{\pm \mathbf{0.27}}$



(a) 3D ViZDoom

(b) Performance

Figure 11: (a) an example of a ViZDoom 3D maze and the first-person view of the environment with each of the corresponding four orientations. (b) the success rates of the algorithms over various SPLs.

scale of the problem, we didn't find that increasing model depth requires additional data. As shown in Table 20, without expanding the dataset, increasing the model depth does not reduce the performance.

Table 12: Continuous Control Hyperparameters

Hyperparameter	Value
Transition Mapping Module	A 2-layer CNN with 3×3 kernel
Reward Mapping Module	A 2-layer CNN (sharing the first layers with Transition Mapping Module)
Policy Mapping Module	A 3-layer FCN
Latent Transition Kernel Size (F)	3
Latent Action Space Size ($\lvert\overline{\mathcal{A}}\rvert$)	4
Optimizer	RMSprop
Learning Rate	1e-3
Batch Size	32
Depth of Planning Module	35×35 maze: 600 100×100 maze: 5000

Table 13: Training time and success rate (%) across different ranges of SPLs for DT-VIN with different J values.

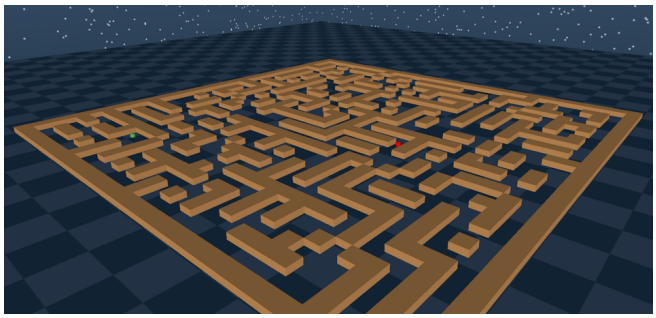
	Wall-Clock Time (hours)	[1,100]	[100,200]	[200,300]
$J = 1$	37	$\mathbf{100.00_{\pm 0.00}}$	$\mathbf{99.99_{\pm 0.01}}$	$\mathbf{99.78_{\pm 0.21}}$
$J = 10$	12.1	$\mathbf{100.00_{\pm 0.00}}$	$\mathbf{99.99_{\pm 0.01}}$	$99.77_{\pm 0.23}$
$J = 50$	7.1	$100.00_{\pm 0.00}$	$99.98_{\pm 0.02}$	$99.69_{\pm 0.27}$

Table 14: Ablation study for using various estimated lengths of optimal paths for adaptive highway loss, under 35×35 ViZDoom navigation. The best results are highlighted.

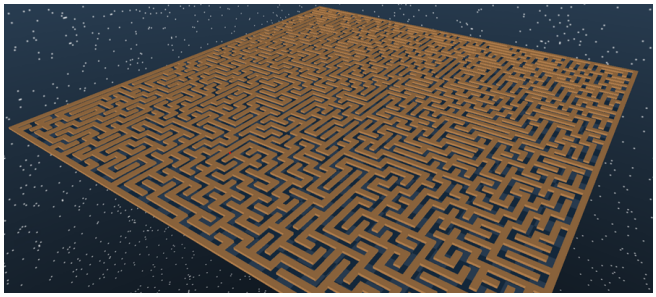
Shortest Path Length	**[1,100]**	**[100, 200]**	**[200,300]**
$\widehat{l} = 0$ (connected to all hidden layers)	$99.49_{\pm 0.35}$	$94.51_{\pm 0.77}$	$89.9_{\pm 3.76}$
$\widehat{l} = l/2$	$99.62_{\pm 0.91}$	$96.21_{\pm 0.44}$	$91.24_{\pm 1.68}$
$\widehat{l} = l$	$\mathbf{99.67_{\pm 0.22}}$	$\mathbf{97.92_{\pm 0.11}}$	$\mathbf{96.41_{\pm 0.37}}$
$\widehat{l} = 2 * l$	$99.61_{\pm 0.18}$	$96.29_{\pm 0.48}$	$93.12_{\pm 0.73}$
$\widehat{l} = N$ (connected to only last layer)	$99.52_{\pm 0.29}$	$95.52_{\pm 0.86}$	$91.12_{\pm 1.64}$
$\widehat{l} = l \cdot \max(\epsilon, 0), \epsilon \sim \mathcal{N}(1, 1)$	$99.62_{\pm 0.50}$	$96.19_{\pm 0.15}$	$93.21_{\pm 0.92}$
$\widehat{l} = len$(non-expert path)	$99.62_{\pm 0.12}$	$97.01_{\pm 0.69}$	$93.31_{\pm 0.31}$
$\widehat{l} = D$ (L1 distance)	$99.64_{\pm 0.49}$	$96.92_{\pm 0.05}$	$93.52_{\pm 0.87}$

Table 15: The success rates and the changes in success rates for each method, using a dataset reduced to 50% of the original size, are presented. These changes are compared to the results from the full-sized dataset, with more negative values indicating worse performance.

	Success Rate			Changes		
Shortest Path Length	[1,100]	[100,200]	[200,300]	[1,100]	[100,200]	[200,300]
VIN	$32.41_{\pm 4.25}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$	$-36.00_{\pm 3.12}$	$0.00_{\pm 0.00}$	$0.00_{\pm 0.00}$
GPPN	$83.11_{\pm 1.33}$	$0.01_{\pm 0.01}$	$0.00_{\pm 0.00}$	$-12.60_{\pm 1.29}$	$-0.38_{\pm 0.11}$	$0.00_{\pm 0.00}$
Highway VIN	$45.41_{\pm 4.13}$	$37.41_{\pm 3.25}$	$21.41_{\pm 6.98}$	$-45.26_{\pm 3.48}$	$-28.09_{\pm 2.98}$	$-32.99_{\pm 3.11}$
DT-VIN (ours)	$\mathbf{99.96_{\pm 0.01}}$	$\mathbf{99.8_{\pm 0.12}}$	$\mathbf{96.01_{\pm 0.32}}$	$\mathbf{-0.04_{\pm 0.01}}$	$\mathbf{-0.19_{\pm 0.04}}$	$\mathbf{-3.76_{\pm 0.31}}$



(a) 35×35 Maze



(b) 100×100 Maze

Figure 12: Some examples of the Point Maze tasks.

Table 16: Computational complexity during training of each method, employing 600 layers and trained over 30 epochs, evaluated in a 35×35 2D maze navigation.

Method	GPU Memory (GB)	Wall-Clock Time (h)	GPU Hours (h)
VIN	4.2	8.4	8.4
GPPN	182	4.2	12.6
Highway VIN	41.3	14.3	14.3
DT-VIN	7.2	12.3	12.3

Table 17: Computational complexity during training of each method using 5000 layers and training for 90 epochs, evaluated on a 100×100 2D maze navigation.

Method	GPU Memory (GB)	Wall-Clock Time (h)	GPU Hours (h)
VIN	35	36	36
GPPN	710	31	310
Highway VIN	111	112	224
DT-VIN (ours)	61.2	97	97

Table 18: Minimal depths of DT-VIN model across various maze sizes.

Maze Size	Longest Length of Optimal Path	Minimal Depth of DT-VIN
15	80	100
25	200	300
35	300	500
100	1800	5000

Table 19: Minimal required size of training data for DT-VIN across various maze sizes.

Maze Size	Longest Length of Optimal Path	Required Number of Mazes	Required Amount of Expert Steps
15	80	15K	4M
25	200	15K	24M
35	300	10K	45M

Table 20: The success rate of DT-VIN across various model depths N, maintaining the same size as the original dataset.

Shortest Path Length	[1,100]	[100,200]	[200,300]
$N = 300$	$99.99_{\pm 0.01}$	$99.81_{\pm 0.13}$	$92.11_{\pm 1.31}$
$N = 600$	$100.00_{\pm 0.00}$	$99.99_{\pm 0.01}$	$99.77_{\pm 0.23}$
$N = 1200$	$100.00_{\pm 0.00}$	$99.99_{\pm 0.01}$	$99.81_{\pm 0.11}$