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

Yuhui Wang [*1] **Qingyuan Wu** [*2] **Weida Li** [3] **Dylan R. Ashley** [1,4,5,6]
Francesco Faccio [1,4,5,6] **Chao Huang** [2] **Jürgen Schmidhuber** [1,4,5,6,7]

1. Center of Excellence in GenAI, King Abdullah University of Science and Technology, Saudi Arabia.
2. The University of Southampton, United Kingdom.
3. National University of Singapore, Singapore.
4. Dalle Molle Institute for Artificial Intelligence Research, Switzerland.
5. Università della Svizzera italiana, Switzerland.
6. Scuola universitaria professionale della Svizzera italiana, Switzerland.
7. NNAISENSE, Switzerland.

Abstract

The Value Iteration Network (VIN) is an end-to-end differentiable architecture that performs value iteration on a latent MDP for planning in reinforcement learning (RL). However, VINs struggle to scale to long-term and large-scale planning tasks, such as navigating a 100×100 maze—a task which 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 both 2D maze navigation environments and the ViZDoom 3D navigation benchmark. We find that our new method, named *Dynamic Transition VIN (DT-VIN)*, easily scales to 5000 layers and casually 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 RL environments.

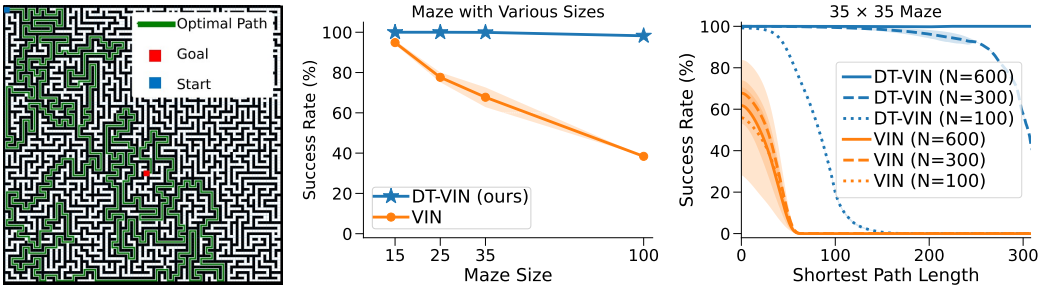
1 Introduction

Planning is the problem of finding a sequence of actions that achieve a specific pre-defined goal. As the aim of both some older algorithms (e.g., Dyna [1], A* [2], and others [3], [4]) and many recent ones (e.g., the Predictron [5], the Dreamer family of algorithms [6]–[8], SoRB [9], SA-CADRL [10], and the LLM-planner [11]), effective planning is a long-standing and important challenge in artificial intelligence (AI). Within reinforcement learning (RL), one particularly notable method is the Value Iteration Network (VIN) proposed by Tamar, Levine, Abbeel, *et al.* [12].

VIN is an artificial neural network (NN) architecture designed for planning, incorporating a differentiable "planning module" that performs value iteration [13] on a "latent MDP." VINs have been shown to perform exceptionally well in some small-scale short-term planning situations, like path planning [14], [15], autonomous navigation [16], and complex decision-making in dynamic environments [17]. However, they still struggle to solve larger-scale and longer-term planning problems. For example, in a 100×100 maze navigation task, the success rate of VINs in reaching the goal drops to

[*]Equal Contribution. Correspondence to yuhui.wang@kaust.edu.sa

17th European Workshop on Reinforcement Learning (EWRL 2024).



(a) 100×100 Maze Navigation (b) Performance for Different Domain Scales (c) Performance for Different 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 Figure 6 in Appendix B 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 the size of the maze. The reported results are computed in expectation over different shortest path lengths for each maze size. (c) shows the success rate of VIN [12] and our DT-VIN as a funciton of the planning steps on the 35×35 maze benchmark.

well below 40% (see Figure 1(b)). Even in smaller 35×35 mazes, the success rate of VINs drop 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 the relatively shallow networks that it uses. And while there has been moderate success in learning more complicated networks (e.g., GPPN [18] and Highway VINs [19]), 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 [20].

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 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 [19] 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 easily 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 top-down image-based maze navigation tasks and the first-person image-based ViZDoom benchmark [21]. We find that DT-VINs can easily 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). The most common formalism used for RL is that of the Markov Decision Process (MDP) [22]. We consider an MDP—as per Puterman [23]—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 probability of transitioning to state $s' \in \mathcal{S}$ when being in state $s \in \mathcal{S}$ and taking action $a \in \mathcal{A}$, $\mathcal{R}(s,a,s')$ is the scalar 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.

Convolutional Neural Networks (CNNs). CNNs are neural networks that specialize in processing data with a grid structure, such as images [24]–[26]. A CNN forward pass typically involves several convolutional layers, where a learnable filter is used to slide across the input data and create a feature map, and max-pooling layers, where the dimension of the feature map is reduced. Formally, a stacked max-pooling and convolutional layer performs the following operation: $X_{c',i,j} = \max_{i',j' \in N(i,j)} \sigma\left(\sum_{c,i,j} W^{c'}_{c,i,j} X_{c,i'-i,j'-j}\right)$, where σ is an activation function, X is an image comprising c channels, $W^1, \ldots, W^{c'} \in \mathbb{R}^{c \times F \times F}$ are kernels, and $N(i,j)$ denotes a $F \times F$ patch centered around pixel (i,j).

Value Iteration Networks (VINs). VIN is an end-to-end differentiable neural network architecture for planning which demonstrates strong generalization to unseen domains through the incorporation of an explicit planning module [12]. The main idea of VIN is to map observations into a latent MDP $\overline{\mathcal{M}}$ and then use the embedded planning module to perform value iteration (VI) on this latent MDP. 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 $x = \phi(s)$, e.g., an image of a maze and the current position of the agent, to $\overline{\mathcal{M}}$. $\overline{\mathcal{M}}$ is described by the latent state space $\overline{\mathcal{S}} = \{(i,j)\}_{i,j \in [m]}$; a fixed discrete latent action space $\overline{\mathcal{A}}$; a latent reward matrix $\overline{\mathsf{R}} = f^{\overline{\mathsf{R}}}(\phi(s)) \in \mathbb{R}^{m \times m}$, where $f^{\overline{\mathsf{R}}}$ is a learnable NN called a *reward mapping module*; and a latent transition matrix (or kernel) $\overline{\mathsf{T}}^{\text{inv}} \in \mathbb{R}^{|\overline{\mathcal{A}}| \times F \times F}$. The latent transition matrix is a parameter matrix that is *invariant* for each latent state (i,j), independent of the observation[*] x, and not restricted to satisfy the probabilistic property, i.e., its elements are not required to represent probabilities or sum to one. Next, VIN conducts VI on the latent MDP $\overline{\mathcal{M}}$ to approximate the latent optimal value function $\overline{V}^*$. To ensure the differentiability of the VI computation, a differentiable VI module is proposed. This module simulates VI computation using differentiable CNN operations, i.e., convolutional and max-pooling operations: $\overline{V}^{(n+1)}_{i,j} = \max_{\overline{a}} \sum_{i',j'} \overline{\mathsf{T}}^{\text{inv}}_{\overline{a},i',j'} \left(\overline{\mathsf{R}}_{i-i',j-j'} + \overline{V}^{(n)}_{i-i',j-j'}\right)$, $i,j \in [m]$. 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 $\mathcal{M}$. Finally, the model can be trained by standard RL and IL algorithms with the following general loss: $\mathcal{L}(\theta) = \frac{1}{|\mathcal{D}|} \sum_{(x,y) \in \mathcal{D}} \ell\left(f^\pi\left(\overline{V}^{(N)}(x)\right), y\right)$, 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, e.g., in imitation learning, the label y is the optimal action and ℓ is the cross-entropy loss.

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 Increasing the Representation Capacity of the Latent MDP

Motivation. VIN utilizes the computational similarities between VI and CNNs to directly implement VI through a CNN-based VI module, as described in Section 2. However, there is a discrepancy between the CNN-based VI module and the general VI computation process.

[*] Although the original VIN paper proposes a general framework where the latent transition kernel depends on the observation, i.e., $\overline{\mathsf{T}} = f^{\overline{\mathsf{T}}}(\phi(s))$, it implements it as an independent parameter in practice.

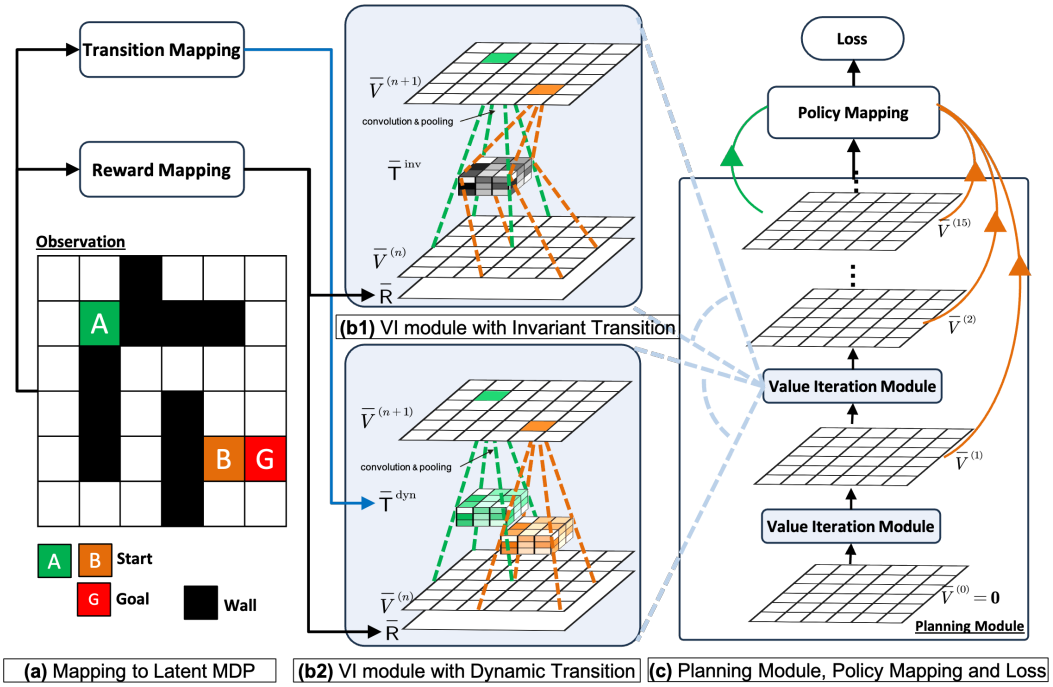


Figure 2: The architecture of VIN and DT-VIN in the maze navigation task. (a) shows the observation of the maze, which is mapped to the latent reward/transition matrix of the latent MDP through the reward/transition mapping module. (c) shows the "planning module", the policy mapping module and the loss. 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 $\overline{\mathsf{T}}^{\text{dyn}} \in \mathbb{R}^{m \times m \times |\mathcal{A}| \times F \times F}$.

CNN-based VIN uses an *invariant* latent transition kernel $\overline{\mathsf{T}}^{\text{inv}} \in \mathbb{R}^{|\overline{\mathcal{A}}| \times F \times F}$ as a learnable parameter, which is the same for each latent state $\overline{s} = (i, j)$ and independent of the current observation, e.g., the map of the maze. This severely limits the representational capacity of the latent MDP which, to be effective, should model what will in practice be the complex and state-dependent transition function of the actual MDP. For example, in the maze navigation problem shown in Figure 1(a), the transition probabilities are quite different if the adjacent cell is a wall versus an empty cell. Additionally—as the latent transition kernel of VIN is independent of the real observation—VIN is unable to exploit any information in the observations to simultaneously model the different transition dynamics of different environments. In the maze example, this means that it will greatly struggle because the model is employed to plan on completely different mazes.

Altogether, this lack of representation capacity does not affect 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. However, we found it to be a major barrier to VIN's effectiveness in large-scale, long-term planning tasks. As we have shown in Figure 1(c), VIN fails on large-scale 100×100 maze navigation tasks and long-term planning tasks requiring more than 60 steps.

Method. Due to the above, we aim to increase the representation capacity of VIN's latent MDP. To this end, we propose a new architecture called *Dynamic Transition VINs (DT-VINs)*. Instead of using an invariant latent transition kernel, DT-VINs employ a dynamic latent transition kernel $\overline{\mathsf{T}}^{\text{dyn}} = f^{\overline{\mathsf{T}}}(\phi(s)) \in \mathbb{R}^{m \times m \times |\overline{\mathcal{A}}| \times F \times F}$, which inputs the observation into a learnable *transition mapping module* $f^{\overline{\mathsf{T}}}$ and dynamically outputs the latent transition kernel for each latent state. The augmented *dynamic transition VI module* is computed as follows:

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

The transition mapping module $f^{\overline{\mathsf{T}}}$ can be any type of neural network, such as CNNs or fully connected networks. In our Maze Navigation tasks, $f^{\overline{\mathsf{T}}}$ includes only one convolutional layer with a kernel size of $F \times F$, which iteratively maps each local patch of the maze to a $|\overline{\mathcal{A}}| \times F \times F$ latent transition kernel for each latent state. This architecture requires $|\overline{\mathcal{A}}|F^4$ number of parameters, compared to the original VIN's $|\overline{\mathcal{A}}|F^2$. Note that in practice, a small kernel size F of 3 is used and is sufficient to produce strong performance. Thus, this alternative module greatly improves the representation capacity of VIN, but typically does not introduce a significant change in training cost.

3.2 Increasing Depth of Planning Module

Motivation. Recent work on Highway VIN has demonstrated the relationship between the depth of VIN's planning module and its planning ability [19]. A deeper planning module implies more iterations of the value iterations process, which is proved to result in a more accurate estimation of the optimal value function (see Theorem 1.12 [27]). However, training very deep neural networks is challenging due to the vanishing or exploding gradient problem [20]. Highway VINs address this issue by incorporating skip connections within the context of reinforcement learning, showing similarities to existing works for classification tasks [28], [29]. 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 more simple, easy-to-implement method for training very deep VINs.

Method. 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. We achieve this by introducing the following *adaptive highway loss*:

$$\mathcal{L}(\theta) = \frac{1}{K} \sum_{(x,y,l)\in\mathcal{D}} \sum_{1\le n\le N} \mathbb{1}_{\{n\ge l\}} \mathbb{1}_{\{n \bmod l_j = 0\}} \ell\left(f^{\pi}\left(\overline{V}^{(n)}(x) \right), y \right), \tag{2}$$

Here, $K = \sum_{(x,y,l)\in\mathcal{D}} \sum_{1\le n\le N} \mathbb{1}_{\{n\ge l\}} \mathbb{1}_{\{n \bmod l_j = 0\}}$, $\mathbb{1}$ is the indicator function, $l_j \in \mathbb{Z}^+$ is a hyperparameter, and l is the number of actual planning steps required for the task, which can be inferred from the training data. For example, in the imitation learning of the maze navigation task, for each maze in the dataset, l is the length of the provided shortest path from the start to the goal.

As Equation (2) implies, it constructs skip connections for the hidden layers to enhance information flow, similar to existing works such as Highway Nets and Residual Nets [28], [29]. However, we connect the hidden layers directly to the final loss, while existing works typically connect skip connections between the intermediate layers. Note that we construct skip connections for each layer $n \ge l$ rather than at the specific layer $n = l$. This is because it would be beneficial for a relatively deeper VIN with depth $n > l$ to also output the correct action in short-term planning tasks. Additionally, during the execution phase, the actual planning steps are unknown, so only the output of the last layer of the VIN will be used. 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 $n \bmod l_j = 0$.

To avoid the gradient exploding problem, we enforce a softmax operation on the values of the latent transition kernel for each latent state $\overline{s}$. This gives a statistical semantic meaning to the latent transition kernel. This change is simple but critical to training stability, as will be shown in experimental results in Section 4.1 and Figure 4(d).

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. In line with previous works (e.g., [18]), we assess our planning algorithms on navigation tasks within 2D mazes and 3D ViZDoom [21] environments. Each task includes a start position and a goal position, and the agent navigates the four adjacent cells by moving one step at a time in any of the four cardinal directions. Our experiments look at each method's effectiveness over several versions of the tasks with the different versions having different *shortest path lengths (SPLs)*. The SPLs are precomputed using Dijkstra's algorithm and serve as a good proxy measure for the complexity of the planning task. We say that an agent has succeeded in a task if it generates a path from the start position to the goal position within a predetermined number of steps (m^2 in our paper). We further say that the agent has found an optimal path if the corresponding path has a minimal length. 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 generates an 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 [12], GPPNs [18], and Highway VIN [19]. The models are trained through imitation learning using a labelled 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 2D Maze Navigation

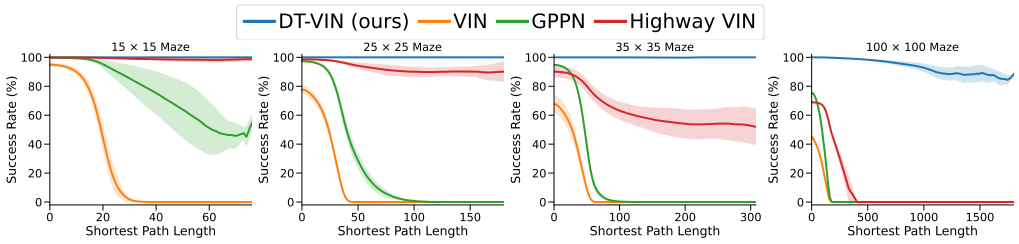
Setting. In our evaluation, we use 2D maze navigation tasks with sizes M set to 15, 25, 35, and 100. Many of these mazes require hundreds or thousands of planning steps to be solved. To assess the performance of each algorithm, we test various neural network depths N. Specifically, for mazes of size $M = 15$ and $M = 25$, we examine depths in $N = 30, 100, 200$. For $M = 35$, we examine depths in $N = 30, 100, 300, 600$. For the largest mazes, $M = 100$, we examine depths of $N = 600, 5000$, with the exception of GPPN, which is limited to $N = 600$ due to GPU resource constraints. For each maze size, we generate a dataset following the methodology in GPPN [18]. Each sample has a starting position, a visual representation of the $m \times m$ map, and an $m \times m$ matrix indicating the position of the goal. For more details, see Appendix B.1.

Results and Discussion. Figure 3(a) and Table 1 show 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 7 in Appendix C for other values of N).

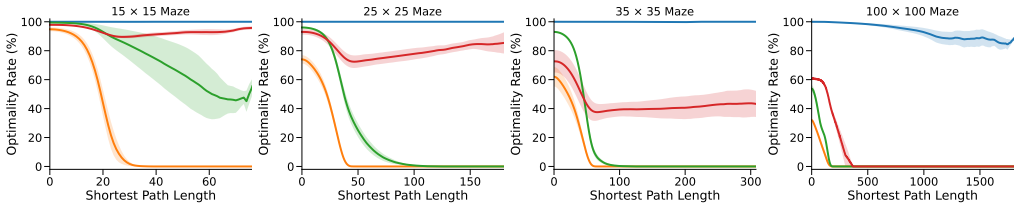
Here, DT-VIN outperforms all the other methods on all the maze navigation tasks under all the various sizes M and SPLs. Notably, on small-scale mazes with size in $M = 15, 25, 35$, DT-VIN achieves approximately 100% SRs on all the tasks. For the most challenging environment with $M = 100$, DT-VIN performs best with the full 5000 layers, and it maintains an SR of approximately 100% on

Table 1: 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 7 in Appendix C.

Maze Size	35×35			100×100		
SPL	[1,100]	[100, 200]	[200, 300]	[1,600]	[600, 1200]	[1200, 1800]
VIN [12]	68.41 ± 6.25	0.00 ± 0.00	0.00 ± 0.00	45.05 ± 0.04	0.00 ± 0.00	0.00 ± 0.00
GPPN [18]	95.71 ± 0.33	0.39 ± 0.27	0.00 ± 0.00	75.72 ± 0.64	0.00 ± 0.00	0.00 ± 0.00
Highway VIN [19]	90.67 ± 3.92	65.50 ± 5.59	54.40 ± 10.2	69.12 ± 0.02	0.00 ± 0.00	0.00 ± 0.00
DT-VIN (ours)	$\mathbf{100.00 \pm 0.00}$	$\mathbf{99.99 \pm 0.01}$	$\mathbf{99.77 \pm 0.23}$	$\mathbf{99.98 \pm 0.00}$	$\mathbf{99.56 \pm 0.20}$	$\mathbf{88.65 \pm 4.76}$



(a) Success Rate



(b) Optimality Rate

Figure 3: SRs and ORs for different algorithms as a function of the shortest path length and the maze size. 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 and 25, 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 7 and Figure 8 in Appendix C for additional results at other depths.

short-term planning tasks with SPL ranging in $[1, 200]$ and an SR of approximately 88% on tasks with SPLs over 1200.

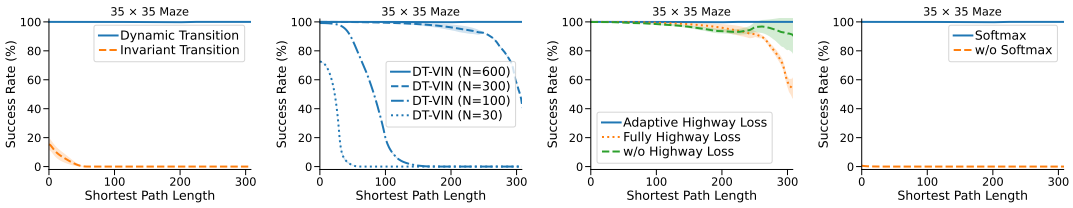
Comparatively, VIN performs well on small-scale and short-term planning tasks. However, even on a small-scale maze with size $M = 15$, VIN's SRs drop to 0% when the SPL exceeds 30. Moreover, when the maze size increases to 100, VIN only achieves an SR of less than 40%—even on short-term planning tasks 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 $M = 15, 25$. However, it nevertheless 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 although these models can generate a path that can achieve the goal, the path is often non-optimal.

Ablation Study. We perform multiple ablation studies with a $M = 35$ maze and an NN with depth $N = 600$ to assess the impact on DT-VIN of (1) the dynamic latent transition kernel, as described in Section 3.1; (2) the network depth, as outlined in Section 3.2; (3) the adaptive highway loss, also covered in Section 3.2; and (4) the softmax function on the latent transition kernel, as mentioned in Section 3.2. Unless otherwise mentioned, all these elements are included.

Dynamic Latent Transition Kernel. Figure 4(a) shows the SRs of our method with the proposed dynamic and the original invariant latent transition kernel. The performance with the invariant transition significantly decreases, highlighting the importance of the dynamic transition. It is notable that this variant incorporates the additional adaptive highway loss to the original VIN, which adversely affects the performance when the representation capacity of the latent MDP is limited.

Depth of Planning Module. Figure 4(b) shows the SRs of our DT-VIN with various depths. Here, increasing the depth dramatically enhances the long-term planning ability. For example, for tasks with an SPL of 200, the variant with depth $N = 300$ performs much better than the variant with depth $N = 100$. Moreover, for tasks with an SPL of 300, the deeper variant with depth $N = 600$ performs



(a) Dynamic Transition (b) Depth N (c) Adaptive Highway Loss (d) Softmax Operation

Figure 4: The results of ablation studies of our DT-VIN with 600 layers. (a) shows the performance of DT-VIN using a dynamic versus an invariant latent transition kernel. (b) shows the performance of DT-VIN over various depths of the planning module. (c) shows the performance of DT-VIN over different loss functions. (d) shows the performance of DT-VIN with and without the softmax operation on the latent transition kernel.

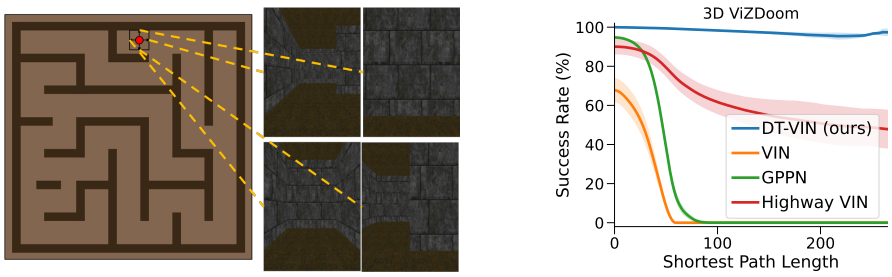
much better. Other methods like VIN and GPPN do not show a clear performance improvement when the depth increases. Figure 7 in Appendix C shows the performance of other methods over all depths.

Adaptive Highway Loss. We evaluate two variants of our DT-VIN, the first without the highway loss, and the second with a "fully highway loss," where the latter enforces a highway loss for each hidden layer without adaptive adjustment based on the actual planning steps. As shown in Figure 4(c), the variant without the highway loss suffers a decrease in performance, and the one with the fully highway loss performs even worse. These results imply that enforcing additional loss on hidden layers without any adjustment could harm performance.

Softmax Latent Transition Kernel. As shown in Figure 4(d), 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.

4.2 3D ViZDoom Navigation

Following the methodology of the GPPN paper, we test our method on 3D ViZDoom [21] 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 5(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 (see Appendix B.2 for more implementation details). 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. Figure 5(b) shows the SRs on 3D ViZDoom mazes with size $M = 35$. Predictably, the performance of all the baselines decreases compared to the 2D maze environments due to the additional noise introduced by the predictions. Here, DT-VIN outperforms all the methods compared to the task over all the various SPLs.



(a) 3D ViZDoom (b) Performance

Figure 5: (a) shows an example of a ViZDoom maze and the first-person view of the environment with each of the corresponding four orientations. (b) shows the success rates of the algorithms on the 3D ViZDoom environments over various SPLs.

5 Related Work

Variants of Value Iteration Networks (VINs). Several variants of VIN [12] 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 [18]. 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 [15]. For addressing challenges in irregular spatial graphs, Generalized VINs adopt a graph convolution operator, extending the traditional convolution operator used in VINs [30]. To improve scalability, AVINs introduce an abstraction module that extracts higher-level information from the environment and the goal [31]. 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 [32]. 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 [33]. 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 [34].

Most related to DT-VIN is other recent work that focused on developing very deep VINs for long-term planning. Specifically, Highway VIN [19] incorporates the theory of Highway Reinforcement Learning [35] 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 Long Short-Term Memory (LSTM) architecture and its gated residual connections, which help alleviate the "vanishing gradient problem" [20], [36]. For feedforward NNs, a similar gated residual connection architecture was used in Highway Networks [28] and later in the ResNet architecture [29], where the gates were kept open. Such residual connections are still ubiquitous in modern language architectures, such as the Generative Pre-trained Transformer (GPT) [37]. 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 [38] or the coordinates of an image [39], have been previously used in Computer Vision.

6 Conclusions

Planning is a long-standing challenge in the field of artificial intelligence and its subfield: reinforcement learning. Previous work proposed VIN as an end-to-end differentiable neural network architecture for this task. While VINs have been successful at short-term small-scale planning, they start to fail quite rapidly as the horizon and the scale of the planning 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 maze environments and 3D ViZDoom environments. We find that DT-VINs scale to exponentially longer-term and exponentially 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, consequentially, the scale of the planning ability) remains unknown. As our experiments were limited mostly by computational cost and not observed instability, we expect that with the growth of available computational power, our method will scale to even longer-term and larger-scale planning.

7 Limitations and Future Work

The principal limitation of our work compared to VIN and Highway VIN is the increased computational cost (see Appendix B.3). This is a consequence of the scale of the network. The past decades have seen AI dominated by the trend of scaling up systems [40], so this is not likely a long-term issue. Other limitations include the requirement to know the length of the shortest path l in the highway loss. In a general RL problem, such a quantity could be estimated online. 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.

Acknowledgements

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. This work was supported by the European Research Council (ERC, Advanced Grant Number 742870). The authors would additionally like to thank both the NVIDIA Corporation for donating a DGX-1 as part of the Pioneers of AI Research Award and IBM for donating a Minsky machine.

References

[1] R. S. Sutton, "Dyna, an integrated architecture for learning, planning, and reacting," *ACM SIGART Bulletin*, vol. 2, no. 4, pp. 160–163, 1991. DOI: 10.1145/122344.122377.

[2] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968. DOI: 10.1109/TSSC.1968.300136.

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

[4] J. Schmidhuber, "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*, vol. 2, 1990, pp. 253–258.

[5] D. Silver, H. van Hasselt, M. Hessel, *et al.*, "The predictron: End-to-end learning and planning," *Proceedings of the 34th International Conference on Machine Learning*, vol. 70, pp. 3191–3199, 2017. [Online]. Available: http://proceedings.mlr.press/v70/silver17a/silver17a.pdf.

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

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

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

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

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

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

[12] A. Tamar, S. Levine, P. Abbeel, Y. Wu, and G. Thomas, "Value iteration networks," *Advances in Neural Information Processing Systems*, vol. 29, pp. 2146–2154, 2016. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2016/file/c21002f464c5fc5bee3b98ced83963b8-Paper.pdf.

[13] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[14] M. Pflueger, A. Agha, and G. S. Sukhatme, "Rover-irl: Inverse reinforcement learning with soft value iteration networks for planetary rover path planning," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 1387–1394, 2019.

[15] X. Jin, W. Lan, T. Wang, and P. Yu, "Value iteration networks with double estimator for planetary rover path planning," *Sensors*, vol. 21, no. 24, p. 8418, 2021.

[16] J. Wöhlke, F. Schmitt, and H. van Hoof, "Hierarchies of planning and reinforcement learning for robot navigation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2021, pp. 10 682–10 688.

[17] W. Li, B. Yang, G. Song, and X. Jiang, "Dynamic value iteration networks for the planning of rapidly changing uav swarms," *Frontiers of Information Technology & Electronic Engineering*, vol. 22, no. 5, pp. 687–696, 2021.

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

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

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

[21] M. Wydmuch, M. Kempka, and W. Jaśkowski, "ViZDoom Competitions: Playing Doom from Pixels," *IEEE Transactions on Games*, vol. 11, no. 3, pp. 248–259, 2019. DOI: 10.1109/TG.2018.2877047.

[22] R. E. Bellman, "A markovian decision process," *Journal of Mathematics and Mechanics*, vol. 6, no. 5, pp. 679–684, 1957. [Online]. Available: http://www.jstor.org/stable/24900506.

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

[24] K. Fukushima, "Neural network model for a mechanism of pattern recognition unaffected by shift in position - Neocognitron," *Trans. IECE*, vol. J62-A(10), pp. 658–665, 1979.

[25] A. Waibel, "Phoneme recognition using time-delay neural networks," in *Meeting of the IEICE*, Tokyo, Japan, 1987.

[26] W. Zhang, J. Tanida, K. Itoh, and Y. Ichioka, "Shift invariant pattern recognition neural network and its optical architecture," in *Proceedings of Annual Conference of the Japan Society of Applied Physics*, vol. 6p-M-14, 1988, p. 734.

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

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

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

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

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

[32] J. Shen, H. H. Zhuo, J. Xu, B. Zhong, and S. Pan, "Transfer value iteration networks," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, 2020, pp. 5676–5683.

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

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

[35] Y. Wang, H. Liu, M. Strupl, *et al.*, *Highway Reinforcement Learning*. arXiv, 2024. [Online]. Available: `https://arxiv.org/abs/2405.18289`.

[36] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[37] J. Achiam, S. Adler, S. Agarwal, *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[38] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, and Z. Liu, "Dynamic convolution: Attention over convolution kernels," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 030–11 039.

[39] R. Liu, J. Lehman, P. Molino, *et al.*, "An intriguing failing of convolutional neural networks and the coordconv solution," *Advances in neural information processing systems*, vol. 31, 2018.

[40] R. S. Sutton, *The Bitter Lesson*. Mar. 13, 2019. [Online]. Available: `http://incompleteideas.net/IncIdeas/BitterLesson.html`.

[41] G. Lample and D. S. Chaplot, "Playing fps games with deep reinforcement learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, 2017.

A Broad Impact

Our work principally involves fundamental research and does not have a clear negative societal impact in excess of those held by all scientific advancements.

B Experimental Details

The below subsections detail specific information about the experiments that have been deemed too minor to appear in the main text.

B.1 2D Maze Navigation

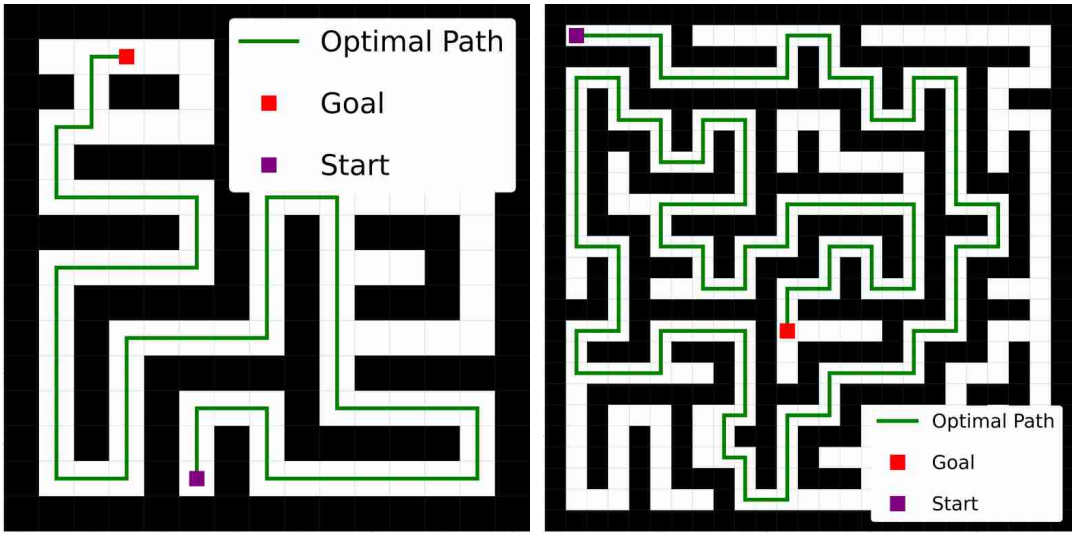
Figure 6 shows some visualizations of some of the different 2D maze navigation tasks we experiment with. Our experimental setup follows the guidelines established in the GPPN paper [18]. For these tasks, the datasets for training, validation, and testing comprise 25000, 5000, and 5000 mazes, respectively. 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 2 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.

B.2 3D ViZDoom

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 [18], [41]. Specifically, for each point 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 5(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 3.

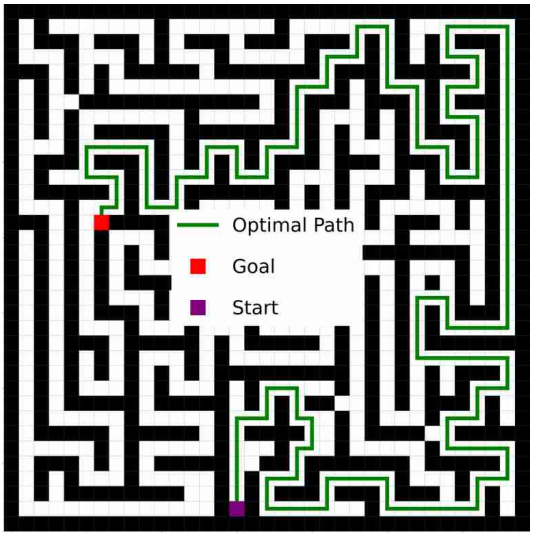
B.3 Computational Complexity

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 4, shows the GPU memory consumption and training time on an NVIDIA A100 GPU for DT-VIN and the baselines when using 600 layers and training for 30 epochs 35×35 maze.

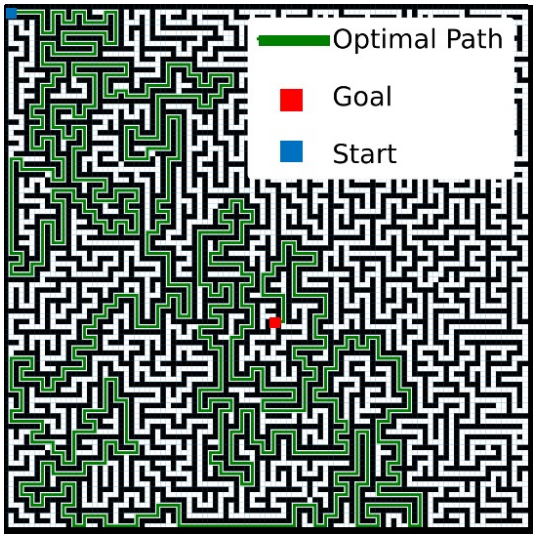


(a) 15×15 Maze

(b) 25×25 Maze



(c) 35×35 Maze



(d) 100×100 Maze

Figure 6: Some examples of the 2D maze navigation tasks.

Table 2: 2D Maze Navigation Hyperparameters

Hyperparameter	Value
Transition Mapping Module	Conv with 3×3 kernel
Reward Mapping Module	Conv with 1×1 kernel
Latent Transition Kernel Size (F)	3
Latent Action Space Size ($\|\overline{\mathcal{A}}\|$)	4
Optimizer	RMSprop
Learning Rate	1e-3
Batch Size	32
Skip Size for Adaptive Highway Loss l_j	10
Depth of Planning Module	15×15 maze: 200 25×25 maze: 200 35×35 maze: 600 100×100 maze: 5000

Table 3: 3D ViZDoom Preprocessing Network

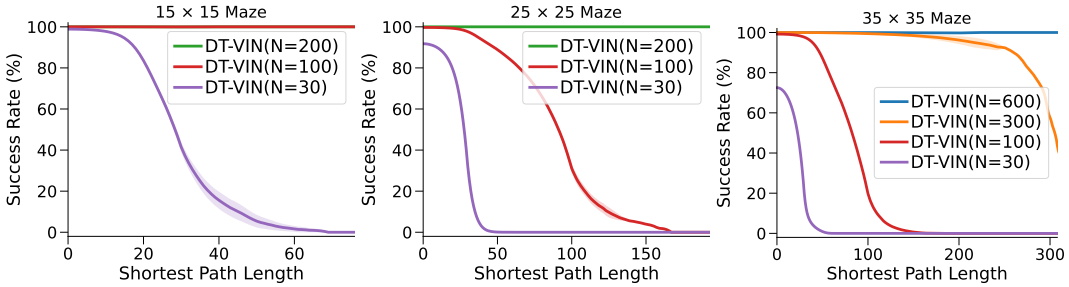
Hyperparameter	Value
Batch Size (B)	32
Image Directions (D)	4
Image Channels (C)	3
Image Width (W)	24
Image Height (H)	32
Input Size	(B, M, M, D, W, H, C)
Layer 1 (Convolution)	$(3, 32, 8, 4, 1)$
Layer 2 (Convolution)	$(32, 64, 4, 2, 1)$
Layer 3 (Linear)	$(384, 256)$
Layer 4 (Convolution)	$(1024, 64, 3, 1, 1)$
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 4: Compuational Complexity

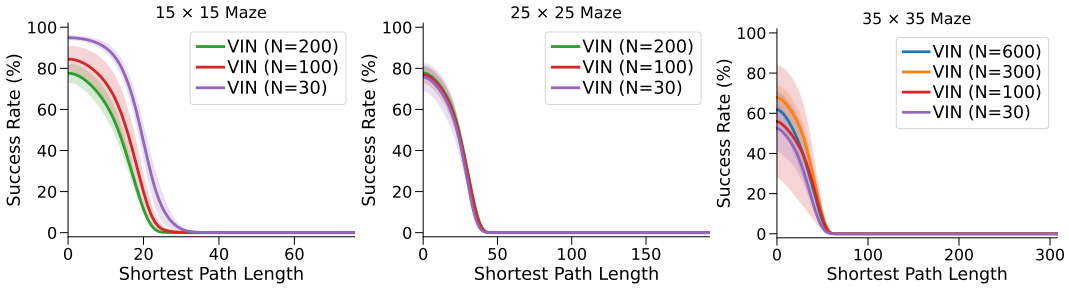
Method	GPU Memory (GB)	Training Time (hours)
VIN	4.2	8.4
GPPN	182	4.2
Highway VIN	41.3	14.3
DT-VIN	53.3	12.1

C Additional Experimental Results

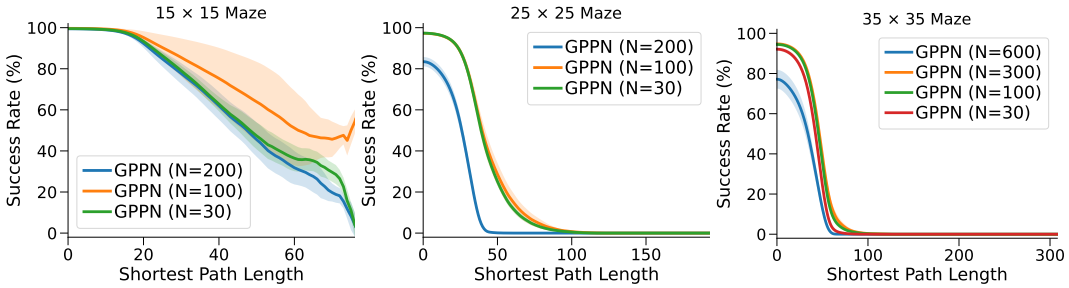
Due to space constraints, the below results could not appear in the main text. Figure 7 shows the success rate of all the algorithms on the 15×15, 25×25, and 35×35 mazes as a function of the shortest path length and the depth of the network. Similarly, Figure 8 shows the corresponding optimality rates.



(a) DT-VIN (ours)

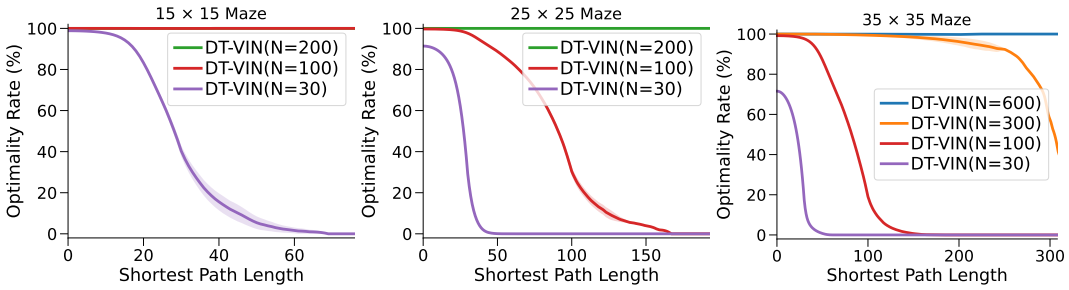


(b) VIN

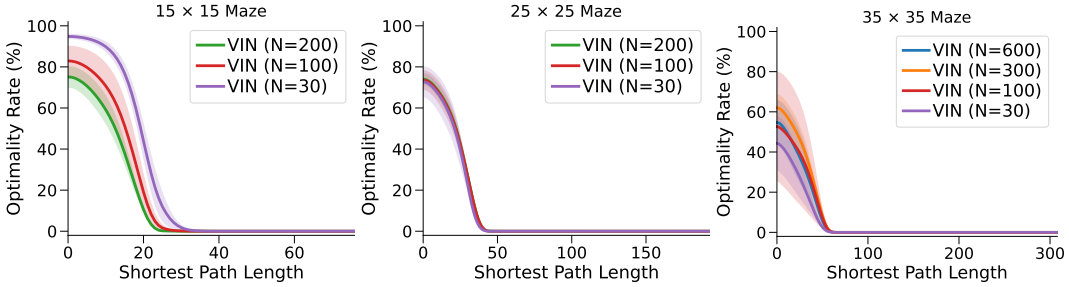


(c) GPPN

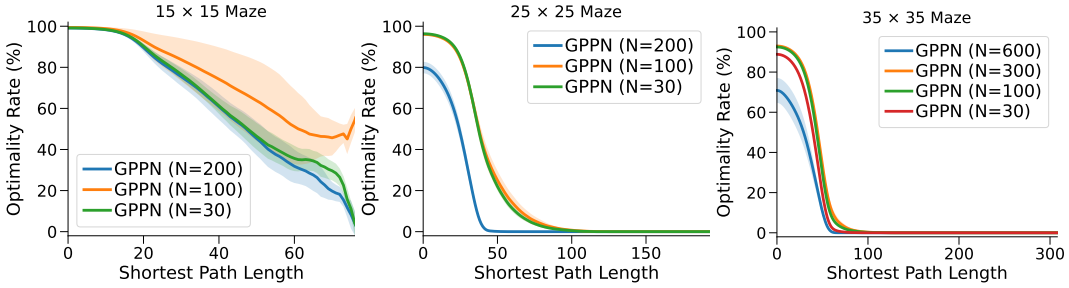
Figure 7: The success rate of each method as a function of shortest path length and network depth. The green and red curves overlap in the top-left plot.



(a) DT-VIN (ours)



(b) VIN



(c) GPPN

Figure 8: 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.