Highway Value Iteration Networks

Yuhui Wang [*1] **Weida Li** [*2] **Francesco Faccio** [1 3] **Qingyuan Wu** [4] **Jürgen Schmidhuber** [1 3]

Abstract

Value iteration networks (VINs) enable end-to-end learning for planning tasks by employing a differentiable "planning module" that approximates the value iteration algorithm. However, long-term planning remains a challenge because training very deep VINs is difficult. To address this problem, we embed highway value iteration—a recent algorithm designed to facilitate long-term credit assignment—into the structure of VINs. This improvement augments the "planning module" of the VIN with three additional components: 1) an "aggregate gate," which constructs skip connections to improve information flow across many layers; 2) an "exploration module," crafted to increase the diversity of information and gradient flow in spatial dimensions; 3) a "filter gate" designed to ensure safe exploration. The resulting novel *highway VIN* can be trained effectively with hundreds of layers using standard backpropagation. In long-term planning tasks requiring hundreds of planning steps, deep highway VINs outperform both traditional VINs and several advanced, very deep NNs.

1. Introduction

Planning is a search for action sequences that are predicted to achieve specific goals. The value iteration network (VIN) (Tamar et al., 2016) is a neural network (NN) architecture that enables end-to-end training for planning tasks using an embedded "planning module," a differentiable approximation of the value-iteration algorithm (Bellman, 1966). VINs have exhibited remarkable proficiency in various tasks, including 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., 2021b).

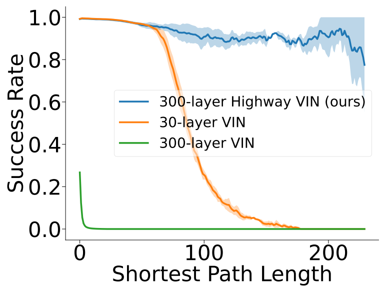


Figure 1: Success rates of reaching the goal in a 25×25 maze problem. The success rate of a 30-layer VIN considerably decreases as the shortest path length increases, and training a 300-layer VIN is difficult and exhibits poor performance.

However, VIN encounters significant challenges in long-term planning. For example, in path planning tasks where the shortest path length exceeds 120, the success rate of VINs in reaching the goal drastically decreases below 10% (Fig. 1). A promising approach to improve the long-term planning capabilities of VIN is increasing the depth of its embedded "planning module." A deeper planning module can integrate more planning steps in VINs, potentially improving their ability to perform long-term planning.

Nonetheless, increasing the depth of an NN can introduce complications, such as vanishing or exploding gradients, which is a fundamental problem in deep learning (Hochreiter, 1991). Although very deep NNs can be effectively trained in classification tasks using different methods (Srivastava et al., 2015b;a; He et al., 2016; Huang et al., 2017), these methods have not been equally successful in planning tasks (Table 1 in Section 5). This disparity may arise from the unique architecture of the VIN, particularly its planning module, which incorporates a specific inductive bias from reinforcement learning (RL). This inductive bias is grounded in the value iteration algorithm, known for its theoretical soundness (Bellman, 1966; Sutton & Barto, 2018). Herein, additional RL-relevant prior knowledge is integrated into the VIN architecture to address these challenges.

This study aims to integrate well-established techniques from both the fields of NNs and RL to create an effective and theoretically sound method. Central to the foundation

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

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

of this proposed approach is highway value iteration (Wang et al., 2024). This method was designed to facilitate efficient long-term credit assignment in the context of RL. We also leverage the architectural innovations of highway networks (Srivastava et al., 2015b;a) and their variants residual networks (He et al., 2016) and DenseNets (Huang et al., 2017), particularly their use of skip connections, which are advantageous for training exceptionally deep NNs.

Building on this foundation, the VIN is enhanced by embedding the highway value iteration algorithm into its core "planning module." This integration introduces three key innovations: (1) an "*aggregate gate*" for creating skip connections and improving information flow between layers; (2) an "*exploration module*" that injects controlled stochasticity during training, thereby diversifying information and gradient flow across spatial dimensions; (3) and a "*filter gate*" designed to "filter out" useless exploration paths, ensuring safe and efficient exploration. These improvements result in the *highway value iteration network (highway VIN)*, a new VIN variant specifically tailored for long-term planning. Remarkably, highway VINs can be efficiently trained with hundreds of layers using standard backpropagation techniques. This study highlights the connections between highway RL and highway networks and their combined potential to advance the capabilities of deep learning models in complex planning tasks. Notably, in scenarios requiring extensive planning, highway VINs outperform traditional VINs and several advanced deep NN models. This showcases their superior capability in handling complex, long-term planning challenges. The source code of highway VIN is available at https://github.com/wangyuhuix/HighwayVIN.

2. Related Work

Variants of Value Iteration Networks. VINs (Tamar et al., 2016) are important architectures that integrate planning capabilities directly into NNs. VINs have a notable advantage over classical RL methods, as they learn policies that generalize better on novel tasks. However, VINs are challenged by issues such as training instability, hyperparameter sensitivity, and overestimation bias. These issues can be addressed using gated path-planning networks (GPPN) (Lee et al., 2018), a recurrent version of the VIN, which replaces convolutional networks with gated recurrent networks, resulting in more stable and effective learning. For higher-dimensional planning tasks, AVINs (Schleich et al., 2019) extend VINs with multi-level abstraction modules. These modules can capture various types of useful information during learning. Another significant challenge arises in generalizing VINs to target domains with limited data. Transfer VINs (Shen et al., 2020) tackle this issue by proposing a transfer learning approach, effectively adapting VINs to different, unseen target domains. Unfortunately, existing methods are considerably limited to extend to real-world and large-scale planning problems as shallow NNs lack long-term planning ability.

Neural Networks with Deep Architectures. Deep learning involves assigning credits to NN components that affect the performance of the NN across multiple layers, or in the case of sequential data, over several time steps. In 1965, Ivakhnenko & Lapa (1965) introduced the first learning algorithms for deep feedforward NNs (FNNs) with any number of hidden layers. However, training FNNs with more than six layers by gradient descent remained a challenge until the early 2010s (Ciresan et al., 2010). Similarly, in the 1980s, recurrent neural networks (RNNs) were limited to problems spanning fewer than ten time steps due to the "vanishing gradient problem" (Hochreiter, 1991), the fundamental problem of deep learning. In very deep NNs, and in RNNs processing sequences with significant time lags between relevant events, the backpropagated gradients tend to either explode or vanish. In 1991, advances in history compression and neural sequence chunking through self-supervised pre-training enabled training RNNs over hundreds or thousands of steps (Schmidhuber, 1991; 1992). However, this worked only for sequences with predictable regularities. This limitation was overcome using residual recurrent connections (Hochreiter, 1991) in long short-term memory (LSTM) RNNs (Hochreiter & Schmidhuber, 1997). This and the later gated LSTM version (Gers et al., 2000) informed the first very deep FNNs called highway networks (Srivastava et al., 2015b). LSTM RNNs are particularly well-suited for tasks involving credit assignment over thousands of steps, whereas similar highway networks were the earliest FNNs with hundreds of layers (previous FNNs had at most tens of layers). Following the same principle, the popular ResNet FNN architecture (He et al., 2016) keeps the highway gates permanently open, allowing uninterrupted information flow from the first to the last layer. Residual connections (Hochreiter, 1991) have become essential for many successful deep-learning architectures (Huang et al., 2017), including graph neural networks with hundreds of layers (Li et al., 2019; 2021a).

3. Preliminaries

Reinforcement Learning. RL is usually formalized as a Markov decision process (MDP) problem (Puterman, 2014). An MDP comprises states $s \in \mathcal{S}$, actions $a \in \mathcal{A}$, a reward function $\mathcal{R}(s, a, s')$, and a transition function $\mathcal{T}(s'|s, a)$ that represents the likelihood of transitioning to the next state s' from the current state and action (s, a). We assume that the action space is finite and the state space is countable. A policy $\pi(a|s)$ defines a probability distribution over actions for each state. The value function $V^{\pi}(s)$ is defined as the expected discounted sum of rewards for following policy π from state s, i.e., $V^{\pi}(s) \triangleq \mathbb{E}\left[\sum_{t=0}^{\infty}\gamma^t\mathcal{R}(s_t,a_t,s_{t+1})|s_0=s;\pi\right]$, where $\gamma \in [0,1)$ is a discount factor. It is also convenient to define the action-value function $Q^{\pi}(s,a) \triangleq \sum_{s'}\mathcal{T}(s'|s,a)\left[\mathcal{R}(s,a,s')+\gamma V^{\pi}(s')\right]$. The objective of RL is to find a policy yielding the maximum expected sum of rewards. To achieve this, the optimal value function is defined as follows: $V^*(s) = \max_\pi V^\pi(s)$ and $Q^*(s,a) = \max_\pi Q^\pi(s,a)$, which allow us to construct an optimal policy $\pi^*(s) = \arg\max_a Q^*(s,a)$ that satisfies $V^{\pi^*}(s) = V^*(s)\forall s$. The Bellman optimality operator and Bellman expectation operator are commonly used to obtain these value functions as follows:

$$(\mathcal{B}V)(s) \triangleq \max_a \sum_{s'} \mathcal{T}(s'|s,a)\left[\mathcal{R}(s,a,s')+\gamma V(s')\right], \quad (1)$$

$$(\mathcal{B}^\pi V)(s) \triangleq \sum_a \pi(a|s) \sum_{s'} \mathcal{T}(s'|s,a)\left[\mathcal{R}(s,a,s')+\gamma V(s')\right]. \quad (2)$$

Iteratively applying $\mathcal{B}$ and $\mathcal{B}^\pi$ to any initial value function $V^{(0)}$ will result in the convergence to V^* and V^π, respectively. The *value iteration (VI)* algorithm is a concrete example of such a convergence, which iteratively applies the Bellman optimality operator as $V^{(n+1)} = \mathcal{B}V^{(n)}$.

Value Iteration Networks. VINs are NNs that integrate the process of planning into the learning architecture. VINs feature a "planning module", which approximates the VI process based on a learned latent MDP $\overline{\mathcal{M}}$. Below, we will use $\overline{\cdot}$ to denote all the terms associated with the latent MDP $\overline{\mathcal{M}}$. VINs use learnable mapping functions to transit an observation $\phi(s)$ to a latent MDP by $\overline{\mathsf{R}} = f^{\overline{\mathsf{R}}}(\phi(s))$ and $\overline{\mathsf{T}} = f^{\overline{\mathsf{T}}}(\phi(s))$. Then, it implements the VI update in Eq. (1) using a *Value Iteration module*, which applies a convolutional operation along with a max-pooling operation:

$$\overline{Q}^{(n)}_{\overline{a},i,j} = \sum_{i',j'}\left(\overline{\mathsf{T}}_{\overline{a},i',j'}\overline{\mathsf{R}}_{i-i',j-j'} + \overline{\mathsf{T}}_{\overline{a},i',j'}\overline{V}^{(n-1)}_{i-i',j-j'}\right), \quad (3)$$

$$\overline{V}^{(n)}_{i,j} = \max_{\overline{a}} \overline{Q}^{(n)}_{\overline{a},i,j}. \quad (4)$$

Here, the indices i,j correspond to the coordinates of the latent state, and $\overline{a}$ is the index of the action in the latent MDP $\overline{\mathcal{M}}$. Eq. (3) sums over a matrix patch centered around position (i,j). Fig. 4(a) shows the computation process of the VI module. By stacking the VI module for several layers, it approximates the optimal value function $\overline{V}^*$, which is then mapped to a policy applicable to the actual MDP $\mathcal{M}$. Fig. 2 shows the VIN architecture. As each component of the architecture is differentiable, VINs can be trained end-to-end.

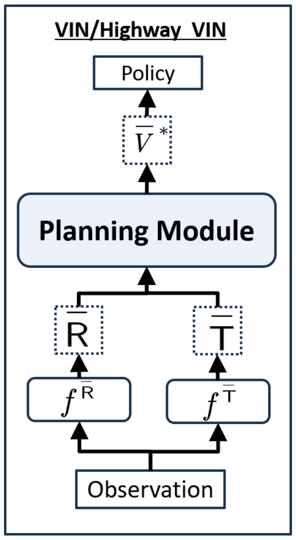
Highway Value Iteration. Highway value iteration (highway VI) is an algorithm derived from the theory of highway RL (Wang et al., 2024), which is a framework for improving the efficiency of long-term credit assignment. This approach introduces a multi-step operator, which averages n-step bootstrapping values using various policies, termed *lookahead policies*, each executed for different n time steps. Here, n represents the *lookahead depth*. This operator is formally defined as follows:

$$\mathcal{G}^{\Pi,\tilde{\alpha}}_{\mathcal{N},\alpha}V \triangleq \underset{\pi\in\Pi}{smax}^{\tilde{\alpha}}\,\underset{n\in\mathcal{N}}{smax}^{\alpha}\max\left\{(\mathcal{B}^\pi)^{\circ(n-1)}\mathcal{B}V, \mathcal{B}V\right\}. \quad (5)$$

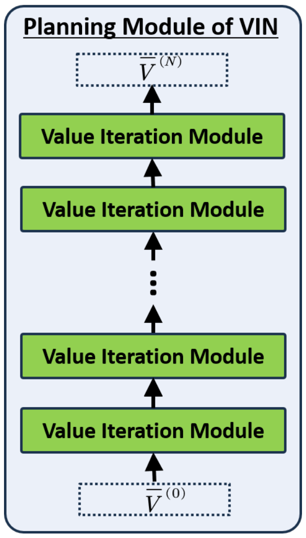
In this formula, Π denotes the set of lookahead policies and $\mathcal{N}$ denotes the set of lookahead depths. The softmax function, with a softmax temperature α, is defined as: $\underset{x\in\mathcal{X}}{smax}^{\alpha} f(x) \triangleq \sum_{x\in\mathcal{X}}\frac{\exp(\alpha f(x))}{\sum_{x'\in\mathcal{X}}\exp(\alpha f(x'))}f(x)$. Here, $(\cdot)^{\circ k}$ indicates the composition of operator $(\cdot)$ for k times. Based on this operator, the algorithm *highway VI* iteratively updates the value function as $V^{(n+1)} = \mathcal{G}^{\Pi,\tilde{\alpha}}_{\mathcal{N},\alpha}V^{(n)}$. Two critical aspects of highway VI are highlighted:

Remark 1. (Convergence to the Optimal Value Function) The highway VI algorithm is proved to converge to the optimal value function V^* regardless of the choice of lookahead policies Π, lookahead depths $\mathcal{N}$, and softmax temperatures $\tilde{\alpha}$ and α. For a detailed formal statement, please refer to their Theorem A.2 (Wang et al., 2024).

Remark 2. (Importance of the Maximization Operation) The maximization operation, $\max\left\{(\mathcal{B}^\pi)^{\circ(n-1)}\mathcal{B}V, \mathcal{B}V\right\}$, is crucial for ensuring convergence to V^*. Convergence is not guaranteed without this component. For a detailed



(a) VIN/HighwayVIN



(b) Planning Module of VIN

Figure 2: (a): Architecture of VIN and highway VIN. (b): Architecture of the planning module of VIN, which includes N layers of value iteration modules. The architecture of the value iteration module is detailed in Fig. 4(a).

formal statement, please refer to their Theorem 1, 2 (Wang et al., 2024).

4. Method

Motivation. NNs with increased depth have superior representational and generalization capabilities (Szegedy et al., 2014; Ciresan et al., 2011; 2012; Telgarsky, 2016). Building on this knowledge, we propose that increasing the depth of VIN can considerably boost their long-term planning abilities in the context of RL. This proposition is grounded in the intrinsic design of VINs, which includes a value iteration planning module. A theoretical study (see Theorem 1.12, (Agarwal et al., 2019)) indicates that increased iterations in this module can result in a more accurate estimation of the optimal value function, subsequently improving the policy performance.

Overview. The traditional VIN (Section 3) propagates information layer-by-layer, based on the step-by-step approach of the VI process. The proposed novel method, i.e., *highway VINs*, enhances VINs by incorporating a distinct planning module inspired by highway VI. As detailed in Section 3, highway VI uses information from various policies and multiple steps ahead, forming a new VIN architecture that facilitates the information flow from various dimensions. The planning module of highway VINs follows the computational process of highway VI in Eq. (5). Below, we detail the transition from the n-th activation $\overline{V}^{(n)}$, representative of the iterative process of the proposed planning module.

First, $\overline{V}^{(n)}$ is fed into the value iteration module to generate a new activation $\overline{V}^{(n+1)}$, as in VINs (Eq. 3 and 4). This step corresponds to the Bellman optimality operator $\mathcal{B}(\cdot)$ in highway VI (see Eq. 5).

Then, to facilitate information flow over spatial dimensions, we introduce a new *value exploration (VE module) module*. Each VE module is equipped with an *embedded policy* $\overline{\pi}$, defined on the latent MDP $\overline{\mathcal{M}}$ and determining the path of the information flow. Conceptually, it corresponds to one application of the Bellman Expectation operator $\mathcal{B}^{\overline{\pi}}(\cdot)$ in highway VI.

Then, to further facilitate spatial information flow in depth, we stack N_p parallel VE modules for $N_b - 1$ layers, corresponding to multiple compositions $(\mathcal{B}^{\overline{\pi}})^{\circ(N_b-1)}(\cdot)$ in highway VI. These stacked and parallel VE modules process the input $\overline{V}^{(n)}$, leading to new activations $\{\overline{\mathcal{V}}_{n_p}^{(n+n_b)}\}_{n_p,n_b}$ for various indexes of the parallel modules $n_p = 1, \cdots, N_p$ and various depths $n_b = 1, \cdots, N_b$, where the initial $\overline{\mathcal{V}}_{n_p}^{(n+1)} = \overline{V}^{(n+1)}$ for each n_p.

Finally, the outputs from these parallel and stacked VE modules are combined using an *aggregate gate* and a *filter gate*. This combination forms a skip connection architecture, which eases the training of very deep NNs. These gates mirror the operations $\underset{\pi}{smax}^{\tilde{\alpha}}\underset{n}{smax}^{\alpha}\max\{\cdot\}$ in highway VI.

We term the above four procedures as a *highway block*. The planning module of highway VIN comprises N_B such highway blocks. Fig. 3 overviews this planning module. The subsequent sections detail the components of the VE module, filter gate, and aggregate Gate.

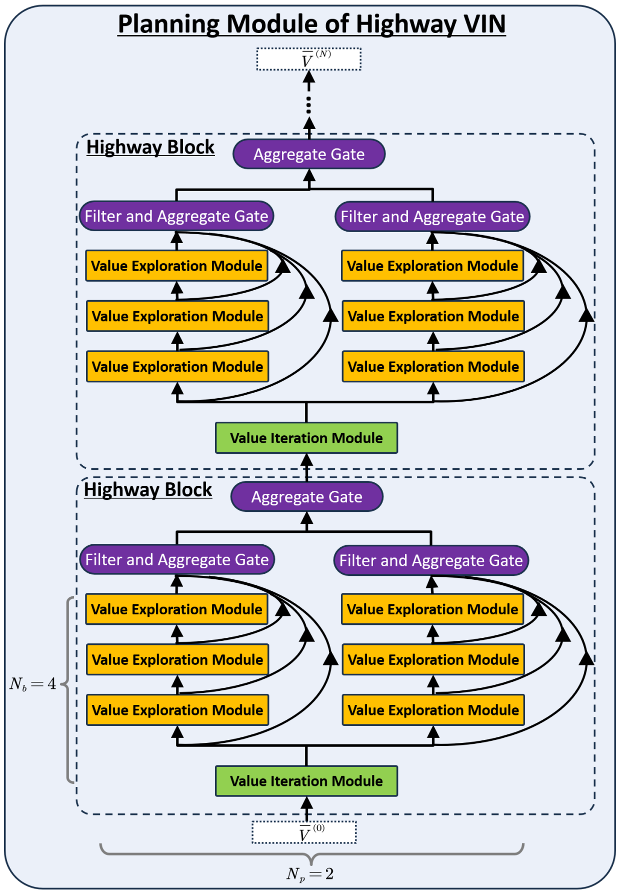
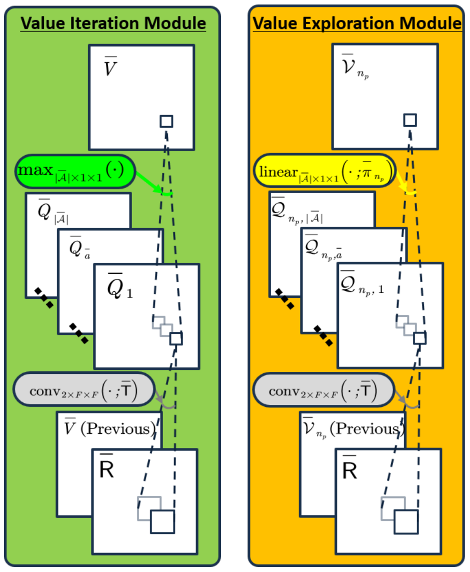


Figure 3: Planning module of highway VIN. Here, we demonstrate the planning module of highway VIN using a highway block of depth $N_b = 4$ and incorporating $N_p = 2$ embedded policies.

4.1. Value Exploration Modules

The value iteration module in VINs greedily takes the largest Q value, as shown in Eq. (4). Consequently, this mechanism can result in a distinctive information flow for each layer, thereby channeling gradients toward certain specific neurons. To facilitate the information and gradient flow across spatial dimensions, we introduce a new *VE module*. Each


Value Iteration Module
$\overline{V}$
$\max_{|\overline{\mathcal{A}}|\times 1\times 1}(\cdot)$
$\overline{Q}_{|\overline{\mathcal{A}}|}$
$\overline{Q}_{\bar a}$
$\overline{Q}_1$
$\mathrm{conv}_{2\times F\times F}(\cdot;\overline{\mathsf{T}})$
$\overline{V}$ (Previous)
R
Value Exploration Module
$\overline{\mathcal{V}}_{n_p}$
$\mathrm{linear}_{|\overline{\mathcal{A}}|\times 1\times 1}(\cdot;\overline{\pi}_{n_p})$
$\overline{\mathcal{Q}}_{n_p,|\overline{\mathcal{A}}|}$
$\overline{\mathcal{Q}}_{n_p,\bar a}$
$\overline{\mathcal{Q}}_{n_p,1}$
$\mathrm{conv}_{2\times F\times F}(\cdot;\overline{\mathsf{T}})$
$\overline{\mathcal{V}}_{n_p}$ (Previous)
R

(a) Value Iteration Module (b) Value Exploration Module

Figure 4: Architecture of the value iteration module and VE module, respectively. The operation $\max_{\overline{\mathcal{A}}\times 1\times 1}$ denotes a max operation over the action axis, as shown in Eq. (4). The operation $\mathrm{linear}_{\overline{\mathcal{A}}\times 1\times 1}$ represents a linear combination of the input Q matrix $\overline{\mathcal{Q}}_{n_p}$ and the policy matrix $\overline{\pi}_{n_p}$ over the action axis, as shown in Eq. (7).

VE module is equipped with an *embedded policy*, which determines the path of the information flow. The VE module computes values according to Bellman expectation operator $\mathcal{B}^{\overline{\pi}}(\cdot)$ in highway VI:

$$\overline{\mathcal{Q}}^{(n+n_b)}_{\overline{\pi},\bar a,i,j} = \sum_{i',j'} \left(\overline{\mathsf{T}}_{\bar a,i',j'} \overline{\mathsf{R}}_{i-i',j-j'} + \overline{\mathsf{T}}_{\bar a,i',j'} \overline{\mathcal{V}}^{(n+n_b-1)}_{\overline{\pi},i-i',j-j'} \right) \qquad (6)$$

$$\overline{\mathcal{V}}^{(n+n_b)}_{n_p,i,j} = \sum_{\bar a} \overline{\pi}^{(n+n_b)}_{n_p,\bar a,i,j} \overline{\mathcal{Q}}^{(n+n_b)}_{n_p,\bar a,i,j} \qquad (7)$$

where $\overline{\pi}^{(n+n_b)}_{n_p} \in \mathbb{R}^{|\mathcal{A}|\times m\times m}$ represents the n_p-th embedded policy for $(n+n_b)$-th layer. Here, the value functions are denoted as $\overline{\mathcal{V}}$ and $\overline{\mathcal{Q}}$ to distinguish them from the value functions $\overline{V}$ and $\overline{Q}$ of the VI module. As implied by Eq. (7), instead of taking maximization over the actions as in the VI module (see Eq. 4), the proposed VE module takes expectation over actions based on the distribution of the embedded policy $\overline{\pi}_{n_p}$. The computation process of the VE module is shown in Fig. 4(b).

Note that, as stated in Remark 1, convergence in highway VI is assured regardless of the chosen lookahead policies. These policies correspond to embedded policies within several VE modules of highway VINs. Generally, the embedded policy can be generated using a learnable mapping function $\overline{\pi}^{(n+n_b)}_{n_p} = f^{\overline{\pi}}\left(\overline{\mathcal{Q}}^{(n+n_b)}_{n_p}, \phi(s); W^{(n+n_b)}_{n_p}\right)$, where $W^{(n+n_b)}_{n_p}$ are learnable parameters. Drawing inspiration from the dropout technique, which introduces stochasticity into the activations to increase robustness (Srivastava et al., 2014; Hanson, 1990; Hertz et al., 1991; Baldi & Sadowski, 2013), we randomly generate multiple embedded policies as follows:

$$\overline{\pi}^{(n+n_b)}_{n_p,\bar a,i,j} = \begin{cases} 1, & \bar a = \widehat{\bar a} \sim P\left(\cdot; \overline{\mathcal{Q}}^{(n+n_b)}_{n_p,\cdot,i,j}, \epsilon\right) \\ 0, & \text{otherwise,} \end{cases} \qquad (8)$$

where ϵ is the *embedded exploration rate*, and $\widehat{\bar a}$ is a sampled action drawn from the ϵ-greedy distribution $P\left(\cdot; \overline{\mathcal{Q}}^{(n+n_b)}_{n_p,\cdot,i,j}, \epsilon\right)$, defined as:

$$P\left(\bar a; \overline{\mathcal{Q}}^{(n+n_b)}_{n_p,\cdot,i,j}, \epsilon\right) = \begin{cases} 1-\epsilon+\frac{\epsilon}{|\mathcal{A}|}, & \bar a = \arg\max_{\bar a'} \overline{\mathcal{Q}}^{(n+n_b)}_{n_p,\bar a',i,j} \\ \frac{\epsilon}{|\mathcal{A}|}, & \text{otherwise.} \end{cases}$$

During the training phase, the embedded policies are generated randomly for each iteration. In contrast, we adopt greedy policies during the evaluation phase, where trained models are applied in realistic environments, defined as follows:

$$\overline{\pi}^{(n+n_b)}_{n_p,\bar a,i,j} = \begin{cases} 1, & \bar a = \arg\max_{\bar a'} \overline{\mathcal{Q}}^{(n+n_b)}_{n_p,\bar a',i,j} \\ 0, & \text{otherwise.} \end{cases}$$

This equation indicates that, during the evaluation phase, VE modules function in the same way as VI modules. In practice, we observe that incorporating this stochastic mechanism substantially improves robustness and is essential for achieving high performance. Moreover, this stochasticity does not impact the convergence to the optimal value function. As stated by the theory of highway VI in Remark 1, convergence is guaranteed irrespective of any chosen embedded policies, even if they are fully stochastic.

4.2. Aggregate and Filter Gates

We aim to integrate the proven efficacy of skip connections in training very deep NNs (Srivastava et al., 2015b; He et al., 2016). However, theoretically validating this architecture within the RL framework and ensuring its compatibility with the stochasticity in the VE modules are challenging. The highway VI algorithm offers a guiding principle for this design. By implementing the $\underset{\pi}{smax}^{\tilde\alpha}\,\underset{n}{smax}^{\alpha}\max\{\cdot\}$ operations of highway VI, the activations are aggregated as follows:

$$\overline{V}^{(n+N_b)}_{i,j} = \sum_{n_p=1}^{N_p} \overline{\mathsf{A}}^{(n+N_b)}_{n_p,i,j} \underbrace{\sum_{n_b=1}^{N_b} \mathsf{A}^{(n+n_b)}_{n_p,i,j} \overbrace{\max\left\{\overline{\mathcal{V}}^{(n+n_b)}_{n_p,i,j}, \overline{V}^{(n+1)}_{i,j}\right\}}^{\overline{V}'^{(n+n_b)}_{n_p,i,j}}}_{\overline{V}''^{(n+N_b)}_{n_p,i,j}}.$$

Here, $\widetilde{\mathsf{A}}_{n_p,i,j}^{(n+N_b)}$ and $\mathsf{A}_{n_p,i,j}^{(n+n_b)}$ are termed as the *aggregate gate*, reflecting the degree to which activations contribute to the output, similar to the concept in highway networks (Srivastava et al., 2015b;a). The aforementioned equation illustrates the aggregation of activations from various layers across multiple parallel VE modules, i.e., $\left\{\overline{\mathcal{V}}_{n_p}^{(n+n_b)}\right\}_{n_p,n_b}$, for $n_p = 1, \cdots, N_p$ and $n_b = 1, \cdots, N_b$. Fig. 3 illustrates the information flow of this computation process. Aggregate gates can be generally generated using mapping functions as follows: $\mathsf{A}_{n_p}^{(n+n_b)} = f^{\mathsf{A}}\left(\left\{\overline{V}_{n_p}^{(n+n_b')}\right\}_{n_b'}, \phi(s); U_{n_p}^{(n+n_b)}\right)$. Here, $U_{n_p}^{(n+n_b)}$ are learnable parameters. For simplicity and consistency with highway VI, we use softmax weights in the following form:

$$\widetilde{\mathsf{A}}_{n_p,i,j}^{(n+N_b)} = \frac{\exp\left(\alpha_{\widetilde{\mathsf{A}}} \overline{V}''^{(n+N_b)}_{n_p,i,j}\right)}{\sum_{n_p'} \exp\left(\alpha_{\widetilde{\mathsf{A}}} \overline{V}''^{(n+N_b)}_{n_p',i,j}\right)}, \text{for various } n_p,$$

$$\mathsf{A}_{n_p,i,j}^{(n+n_b)} = \frac{\exp\left(\alpha_{\mathsf{A}} \overline{V}'^{(n+n_b)}_{n_p,i,j}\right)}{\sum_{n_b'} \exp\left(\alpha_{\mathsf{A}} \overline{V}'^{(n+n_b')}_{n_p,i,j}\right)}, \text{for various } n_b,$$

where $\alpha_{\widetilde{\mathsf{A}}}$ and α_{A} are the softmax temperatures that vary for each highway block and learnable via backpropagation.

The maximization operation $\max\{\cdot\}$, which we term *filter gate*, compares the $(n + n_b)$-th activation $\overline{\mathcal{V}}_{n_p}^{(n+n_b)}$ with the $(n + 1)$-th one $\overline{V}^{(n+1)}$, selecting the maximum value. This filter gate is essential for discarding any activations $\overline{\mathcal{V}}_{n_p}^{(n+n_b)}$ that are lower than $\overline{V}^{(n+1)}$, effectively filtering out explorations in VE modules that do not contribute to convergence. Furthermore, as suggested in Remark 2, this operation is crucial for ensuring convergence.

4.3. Relation to Highway Networks

Section 4.2 and Fig. 3 illustrate the planning module of highway VIN, which features an architecture with skip connections similar to those found in established NNs such as highway networks (Srivastava et al., 2015b;a) and their variants residual networks (He et al., 2016) and DenseNets (Huang et al., 2017). A straightforward approach is to directly implement skip connections in VINs as follows:

$$\overline{V}_{i,j}^{(n+N_b)} = \sum_{n_b=1}^{N_b} \mathsf{A}_{i,j}^{(n+n_b)} \overline{V}_{i,j}^{(n+n_b)}. \quad (9)$$

Here, $\overline{V}_{i,j}^{(n+n_b)}$ is derived from the VI module (Eqs. 3 and 4), using $\overline{V}_{i,j}^{(n+n_b-1)}$ as the input. While this method helps to address optimization challenges in training very deep VINs, it has not shown effectiveness in improving long-term planning capabilities (Table 1 in Section 5). Highway VINs include two additional critical components: a VE module that improves the diversity of information and gradient flow, and an innovative filter gate designed to eliminate useless information generated by the VE modules. This study also reveals the underlying connections between the highway RL and highway networks, which were initially proposed under different contexts and purposes but share fundamental similarities.

5. Experiments

We conduct a series of experiments to evaluate how highway VINs can improve the long-term planning capabilities of VINs for complex tasks. We also explore the significance of each component within the highway VINs.

5.1. 2D Maze Navigation

We evaluate the algorithms on 2D maze navigation tasks of various sizes $m \times m$, specifically 15×15 and 25×25. In these tasks, the agent can move forward, turn 90 degrees left or right, and has four orientations. We follow the experimental setup described in the paper on GPPN (Lee et al., 2018). We train the models for 30 epochs using imitation learning on a labeled training dataset, select the best model based on validation dataset performance, and test it on a separate test dataset. Each dataset contains numerous planning tasks, each involving a maze with a start position, an image of the $m \times m$ map, and a goal position represented by a $4 \times m \times m$ matrix (where 4 corresponds to the four orientations). These datasets involve tasks requiring planning over hundreds of steps to reach the goal. For more detailed information about the dataset, please refer to Appendix A.

We measure the planning abilities of an agent based on the *success rate (SR)*, which is defined as the ratio of the number of successfully completed tasks to the total number of planning tasks. The agent is considered to succeed in a task if it generates a path from the start position to the goal position within a limited number of steps. To assess the planning ability of the algorithms across different scales, we evaluate them on navigation tasks with varying *shortest path lengths (SPLs)* from start to goal. These lengths are calculated in advance using Dijkstra's algorithm with access to the maze's underlying structure. Tasks with longer SPLs typically demand greater long-term planning capabilities. We follow the GPPN paper's setting, evaluating each algorithm with 3 random seeds. We report the mean and standard deviation on the test dataset.

Subsequently, we compare the highway VINs against several advanced NNs for planning tasks. The baseline model

Table 1: The success rates for each algorithm with various depths under 2D maze navigation tasks with different ranges of shortest path length. Please also refer to Appendix Table 4 for the results of all the other depths.

Maze Size	15×15				25×25			
Shortest Path Length		$[1, 30]$	$[30, 60]$	$[60, 100]$		$[1, 60]$	$[60, 130]$	$[130, 230]$
VIN (Tamar et al., 2016)	$N = 20$	99.83 ± 0.11	96.48 ± 0.58	63.03 ± 3.20	$N = 30$	98.84 ± 0.16	49.25 ± 4.16	2.96 ± 0.66
	$N = 40$	99.79 ± 0.10	95.84 ± 0.69	76.16 ± 1.87	$N = 60$	96.47 ± 1.33	48.26 ± 4.21	7.87 ± 3.54
	$N = 100$	0.80 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	$N = 150$	0.22 ± 0.08	0.00 ± 0.00	0.00 ± 0.00
	$N = 200$	0.56 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$N = 300$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GPPN (Lee et al., 2018)	$N = 20$	99.98 ± 0.01	92.68 ± 1.07	51.12 ± 5.00	$N = 30$	98.98 ± 0.25	25.98 ± 5.78	2.76 ± 1.68
	$N = 40$	$\mathbf{99.99 \pm 0.01}$	96.16 ± 3.56	65.17 ± 12.4	$N = 60$	$\mathbf{99.09 \pm 0.19}$	28.87 ± 1.47	1.32 ± 0.55
	$N = 100$	99.95 ± 0.05	93.34 ± 4.16	60.57 ± 13.6	$N = 150$	98.51 ± 0.31	21.62 ± 3.50	0.73 ± 0.68
	$N = 200$	99.98 ± 0.01	92.79 ± 1.28	50.88 ± 3.59	$N = 300$	95.38 ± 2.01	6.29 ± 4.35	0.02 ± 0.03
Highway network (Srivastava et al., 2015b)	$N = 40$	99.65 ± 0.17	96.04 ± 0.63	75.86 ± 10.0	$N = 60$	97.93 ± 0.56	62.95 ± 8.79	17.46 ± 5.45
	$N = 100$	99.36 ± 0.32	91.11 ± 2.64	60.32 ± 8.87	$N = 150$	85.42 ± 4.20	12.55 ± 3.89	0.35 ± 0.23
	$N = 200$	0.73 ± 0.12	0.00 ± 0.00	0.00 ± 0.00	$N = 300$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Highway VIN (ours)	$N = 40$	99.77 ± 0.09	98.83 ± 0.25	90.00 ± 2.12	$N = 60$	97.87 ± 0.60	77.02 ± 6.30	20.68 ± 9.89
	$N = 100$	99.93 ± 0.03	$\mathbf{99.52 \pm 0.12}$	$\mathbf{98.61 \pm 0.66}$	$N = 150$	97.77 ± 0.48	89.56 ± 0.95	75.42 ± 10.1
	$N = 200$	99.94 ± 0.01	99.13 ± 0.12	98.20 ± 1.75	$N = 300$	98.73 ± 0.50	$\mathbf{92.28 \pm 3.50}$	$\mathbf{90.06 \pm 3.13}$

is the original VIN (Tamar et al., 2016). We also compare highway VINs against GPPNs (Lee et al., 2018), which improve the training stability of VINs using gated recurrent operators such as LSTM updates and the highway networks (Srivastava et al., 2015b), which incorporate skip connections for training of very deep NNs (adapted here for VINs, Section 4.3). We follow the hyperparameter settings listed in the paper on GPPNs (Lee et al., 2018). To ensure a fair comparison, we set the number of parallel VE modules of highway VINs to $N_p = 1$ unless stated otherwise. The embedded exploration rate is set to $\epsilon = 1$ (defined in Eq. 8). Note that this setting does not result in a sub-optimal solution of the latent value function due to the filter gate, which excludes actions detrimental to convergence. The baseline uses a 20-layer VIN and GPPN for the 15×15 Maze, and a 30-layer VIN and GPPN for the 25×25 Maze. For both highway networks and highway VINs, we set $N_B = 20$ highway blocks for the 15×15 maze and $N_B = 30$ for the 25×25 maze. We set various highway block depths $N_b \geq 2$, which yields various total depths: $N = N_b * N_B$, specifically $N \in \{40, 60, 80, 120, 160, 200\}$ for the 15×15 maze and $N \in \{60, 90, 120, 150, 180, 240, 300\}$ for the 25×25 maze. The baselines VIN and GPPN are also tested using the same depths.

Peformance with the Best NN Depth. Fig. 5 shows the SRs of various algorithms under tasks with different SPLs. To ensure a fair comparison, we select the best results from varying NN depths for each algorithm as each algorithm may perform optimally at different depths (please refer to Fig. 9 in Appendix for a comprehensive view of the results across all depths). The results demonstrate that the SRs for all compared methods considerably decrease with increasing SPLs. Remarkably, when the SPL exceeds 200 in the 25×25 Maze, the SRs of all methods nearly drop to $\mathbf{0\%}$. In contrast, highway VIN maintains an impressive $\mathbf{98\%}$ SR with an SPL of 100 in the 15×15 Maze and $\mathbf{90\%}$ SR with an SPL of 200 in the 25×25 Maze.

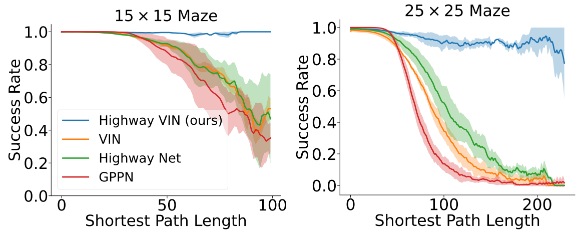
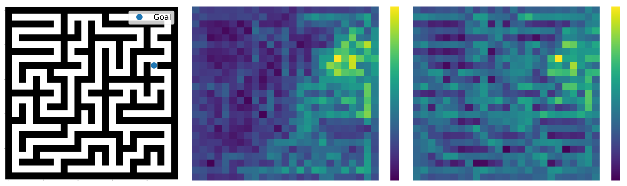


Figure 5: Success rates of the algorithms are presented as a function of varying shortest path length. For each algorithm, the optimal result from a range of depths is selected. For a comprehensive view of the results across all depths, please see Fig. 9 in the Appendix.



(a) 25×25 Maze (b) VIN (c) Highway VIN

Figure 6: (a) An example of 25×25 Maze. (b) and (c) learned feature maps of VIN and highway VIN, respectively.

Additionally, Fig. 6 shows the feature map of the VIN and highway VIN, which can conceptually be understood as the learned value function. The figure reveals that the learned values of highway VIN for states distant from the goal are larger than those of VIN, implying that highway VIN learns an effective value function for long-term planning.

Peformance with Various Depths. Table 1 shows the SRs of each algorithm across various depths and tasks with different SPL ranges (see Table 4 in the Appendix for results across all depths). The table shows that the highway VINs generally perform better with increased depth. Notably, we observe a **+69.38%** improvement in the SR of the 25×25 maze with an SPL range of $[130, 230]$ as the depth N increases from 60 to 300.

The performance of the VIN decreases with increasing depth. Specifically, the SR of VINs drops to nearly **0%** for all tasks at a depth of $N > 150$. The highway network, in contrast, maintains its performance even at greater depths. However, an increase in depth does not considerably improve its long-term planning capabilities. We hypothesize that integrating the skip connections of highway networks into VINs does not introduce additional architectural inductive bias beneficial for planning.

The GPPN effectively mitigates the challenges of training very deep models and performs robustly across various depths. In particular, the GPPN excels in tasks with short SPLs, achieving a **99.09%** SR on a 25×25 maze with an SPL range of $[1, 30]$. However, the GPPN does not show a notable improvement in long-term planning capabilities with an increased depth. For instance, the SR of the GPPN drops to less than **3%** on a 25×25 maze with an SPL range of $[130, 230]$. This might be because the GPPN, as a black box method with less inductive bias towards planning, is more suited to learning patterns for short-term planning tasks rather than those requiring long-term planning skills.

5.2. Ablation Study

Several ablation studies were conducted to evaluate: 1) the effectiveness of the filter gate (Section 4.2); 2) the impact of the VE module (Section 4.1). We also evaluate the influence of the number of parallel VE modules in Appendix B.2. In the highway VIN experiments, the default hyperparameters include an exploration rate ϵ of 1, a single parallel VE module ($N_p = 1$), and depth configurations of 200 for the 15×15 mazes and 300 for the 25×25 mazes.

Filter Gate. The SRs of highway VINs, with and without the filter gate, are shown in Fig. 7. The performance considerably decreases when the filter gate is absent. This is because, without a filter gate, highway VINs could easily suffer the adverse effects of exploration in VE modules, which could prevent convergence.

VE Modules. We also evaluate a variant of highway VIN without the VE module (referred to as *w/o VE modules*), which means that all VE modules are replaced with VI modules. Fig. 8 shows the SRs for the variants with and without the VE module. Without VE modules, the performance of highway VINs notably diminishes in the 25×25 maze. In comparison, the variant equipped with the VE modules performs much better. This improvement is likely due to the use of diverse latent actions, which could facilitate information flow among various neurons in the NN. Table 2 lists the entropy of the selected latent actions of VIN and highway VIN. For VIN, it always selects the action that leads to the maximum Q value. Although the selected latent actions generally vary for each layer because the Q values dynamically change for each layer, they will converge to the same actions when the Q values converge. Therefore, the diversity of the selected actions for VIN is much more limited. Instead, in the proposed highway VIN, we employ the value exploration module to maintain diversity.

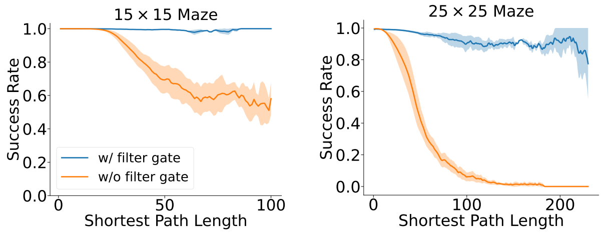


Figure 7: Success rates of highway VIN with and without the filter gate.

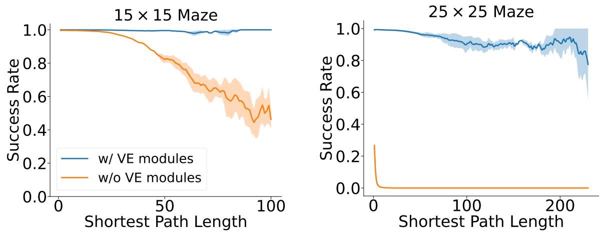


Figure 8: Success rates of highway VIN with and without the VE modules,

Table 2: The entropy of the selected latent actions of VIN and highway VIN with various depths. The entropy is computed by $\sum_{\bar{a} \in \bar{\mathcal{A}}} [-p(\bar{a}) \log p(\bar{a})]$, where $p(\bar{a}) = \frac{\mathrm{cnt}(\bar{a})}{\sum_{\bar{a} \in \bar{\mathcal{A}}} \mathrm{cnt}(\bar{a})}$ and $\mathrm{cnt}(\bar{a})$ is the number of selected latent actions equals to $\bar{a}$ over the latent actions across all hidden layers.

Maze Size	15×15			25×25		
Depth	40	100	200	60	150	300
VIN	0.51	0.14	0.00	0.63	0.07	0.00
Highway VIN (ours)	2.04	3.53	4.17	2.27	3.77	4.35

5.3. 3D ViZDoom Navigation

We evaluate the proposed approach in 3D ViZDoom environments (Wydmuch et al., 2019). Following the experimental setting of the GPPN paper (Lee et al., 2018), the input to the model consists of RGB images capturing the first-person view, rather than the top-down 2D maze. Based on these observations, a CNN is employed to predict the maze map, which is then inputted into the planning model (such as VIN or highway VIN), using an architecture and hyperparameters similar to the 2D maze setup. We select the optimal depth from $N = 40, 100, 200$ for each algorithm. Highway VIN performs optimally at depth 100, while other methods perform best at depth 40. Table 3 shows the SRs in 15×15 3D ViZDoom maze navigation. Our highway VINs excel in tasks with SPLs exceeding 30.

Table 3: Success rates of each algorithm with various depths under 3D ViZDoom maze navigation tasks with different ranges of SPLs.

	[1, 30]	[30, 60]	[60,100]
VIN	98.57 ± 1.54	92.03 ± 2.03	69.37 ± 2.63
GPPN	$\mathbf{99.91 \pm 0.10}$	89.95 ± 9.21	44.42 ± 8.14
Highway network	97.82 ± 1.01	91.99 ± 2.54	63.78 ± 11.49
Highway VIN (ours)	99.43 ± 0.18	$\mathbf{98.70 \pm 0.38}$	$\mathbf{96.98 \pm 1.20}$

5.4. Computational Complexity

The proposed approach introduces a minimal number of additional parameters to the existing VIN architecture, specifically the softmax temperatures for each highway block denoted as $\{(\alpha_{\tilde{A}}^{(n_B)}, \alpha_{A}^{(n_B)})\}_{n_B=1}^{N_B}$. Note that N_B indicates the total number of highway blocks, which are set to $N_B = 20$ for the 15×15 maze and $N_B = 30$ for the 25×25 maze. Additionally, the following table details the GPU memory consumption and training duration for each method when employing 300 layers on NVIDIA A100 GPUs.

	VIN	GPPN	Highway network	Highway VIN (ours)
GPU Memory	3.1G	103.0G	3.3G	15.0G
Training Time	7.5 hours	6.5 hours	7.7 hours	9.0 hours

6. Conclusions

This paper presents a general framework based on highway RL to improve the long-term planning ability of VINs. We improve traditional VINs by incorporating three key components: an aggregate gate, which establishes skip connections and facilitates long-term credit assignment; an exploration module, crafted to diversify information and gradient flow between neurons; and a filter gate, designed to eliminate non-essential information. Our experiments demonstrate that highway VINs enable long-term planning by training neural networks with hundreds of layers, surpassing the performance of several advanced methods. Future research will investigate the integration of multiple parallel VE modules with various types of embedded policies to improve performance. Additionally, future work will focus on scaling up to larger tasks.

Acknowledgement

We sincerely thank the authors of the GPPN paper (Lee et al., 2018) for providing experimental details and Prof. Chao Huang for his valuable suggestions. This work was supported by the European Research Council (ERC, Advanced Grant Number 742870) and the Swiss National Science Foundation (SNF, Grant Number 200021 192356).

Impact Statement

This paper introduces research aimed at advancing the field of machine learning. While acknowledging numerous potential societal consequences, we believe that none need to be specifically emphasized here.

References

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.

Baldi, P. and Sadowski, P. J. Understanding dropout. *Advances in neural information processing systems*, 26, 2013.

Bellman, R. Dynamic programming. *Science*, 153(3731): 34–37, 1966.

Ciresan, D. C., Meier, U., Gambardella, L. M., and Schmidhuber, J. Deep big simple neural nets for handwritten digit recogntion. *Neural Computation*, 22(12):3207–3220, 2010.

Ciresan, D. C., Meier, U., Masci, J., Gambardella, L. M., and Schmidhuber, J. Flexible, high performance convolutional neural networks for image classification. In *Intl. Joint Conference on Artificial Intelligence IJCAI*, pp. 1237–1242, 2011.

Ciresan, D. C., Meier, U., Masci, J., and Schmidhuber, J. Multi-column deep neural network for traffic sign classification. *Neural Networks*, 32:333–338, 2012.

Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: Continual prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 2000.

Hanson, S. J. A stochastic version of the delta rule. *Physica D: Nonlinear Phenomena*, 42(1):265–272, 1990.

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.

Hertz, J., Krogh, A., and Palmer, R. *Introduction to the Theory of Neural Computation*. Addison-Wesley, Redwood City, 1991.

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.

Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 4700–4708, 2017.

Ivakhnenko, A. G. and Lapa, V. G. *Cybernetic Predicting Devices*. CCM Information Corporation, 1965.

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.

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, G., Muller, M., Thabet, A., and Ghanem, B. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9267–9276, 2019.

Li, G., Müller, M., Ghanem, B., and Koltun, V. Training graph neural networks with 1000 layers. In *International conference on machine learning*, pp. 6437–6449. PMLR, 2021a.

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, 2021b.

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.

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

Schmidhuber, J. Neural sequence chunkers. Technical Report FKI-148-91, Institut für Informatik, Technische Universität München, April 1991.

Schmidhuber, J. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

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.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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. and Barto, A. G. *Reinforcement learning: An introduction*. MIT press, 2018.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. Going deeper with convolutions. Technical Report arXiv:1409.4842 [cs.CV], Google, 2014.

Tamar, A., Wu, Y., Thomas, G., Levine, S., and Abbeel, P. Value iteration networks. *Advances in neural information processing systems*, 29, 2016.

Telgarsky, M. Benefits of depth in neural networks. In *Conference on learning theory*, pp. 1517–1539. PMLR, 2016.

Wang, Y., Strupl, M., Faccio, F., Wu, Q., Liu, H., Grudzień, M., Tan, X., and Schmidhuber, J. Highway reinforcement learning. *arXiv preprint arXiv:2405.18289*, 2024.

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.

A. Experimental Details

Our experimental settings follow those outlined in the paper on GPPN (Lee et al., 2018). For maze navigation tasks, the training, validation, and test datasets comprise $25K$, $5K$, and $5K$ mazes, respectively.

All models are trained for 30 epochs using the RMSprop optimizer with a learning rate of 0.001 and a batch size of 32. We also specify a kernel size of 5 for convolutional operations in the planning module, as mentioned in Eq. (3). For the neural network that maps the observation to the latent MDP, we set the hidden dimension to 150.

B. Additional Experimental Results

B.1. Various Depths of Highway VIN

Table 4 and Fig. 9 show the SRs of various algorithms across different depths. For each algorithm, we also provide the rate at which it can plan the optimal path that yields the shortest path length, summarized in Table 5.

B.2. Number of Parallel VE Modules

We evaluate highway VIN with varying numbers of parallel VE modules N_p under varying depths N. As shown in Fig. 10, under different depths N, the number of parallel VE modules has a different effect on the performance of highway VIN. For example, under depth $N = 300$, with fewer parallel VE modules, i.e., $N_p = 1$, highway VIN performs the best. While under depth $N = 100$, with more VE modules, $N_p = 3$, highway VIN performs the best. These results imply that the additional parallel VE modules may be detrimental to the performance of very deep networks.

B.3. Examples of 2D Maze Navigation

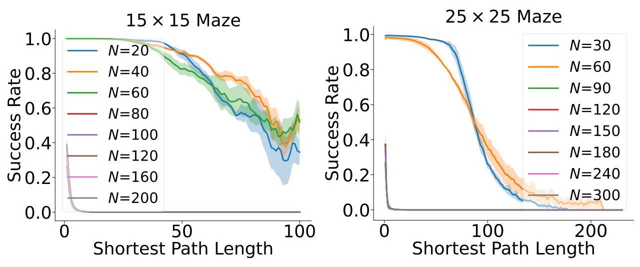
In Fig. 11, we show examples where highway VIN succeeds, but other methods fail.

Table 4: Success rates of each algorithm with various depths under 2D maze navigation tasks with different ranges of shortest path lengths.

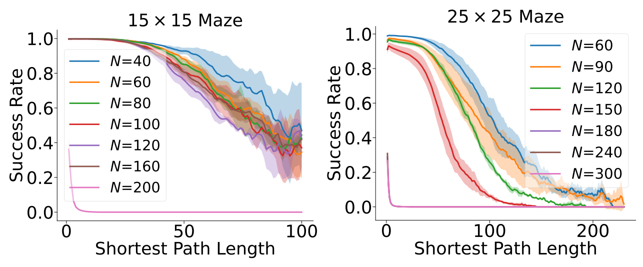
Maze Size	15×15				25×25			
Shortest Path Length		$[1, 30]$	$[30, 60]$	$[60, 100]$		$[1, 60]$	$[60, 130]$	$[130, 230]$
VIN (Tamar et al., 2016)	$N = 20$	99.83 ± 0.11	96.48 ± 0.58	63.03 ± 3.20	$N = 30$	98.84 ± 0.16	49.25 ± 4.16	2.96 ± 0.66
	$N = 40$	99.79 ± 0.10	95.84 ± 0.69	76.16 ± 1.87	$N = 60$	96.47 ± 1.33	48.26 ± 4.21	7.87 ± 3.54
	$N = 60$	99.83 ± 0.03	92.53 ± 1.33	66.18 ± 6.91	$N = 90$	0.21 ± 0.08	0.00 ± 0.00	0.00 ± 0.00
	$N = 80$	0.65 ± 0.16	0.00 ± 0.00	0.00 ± 0.00	$N = 120$	0.21 ± 0.08	0.00 ± 0.00	0.00 ± 0.00
	$N = 100$	0.80 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	$N = 150$	0.22 ± 0.08	0.00 ± 0.00	0.00 ± 0.00
	$N = 120$	0.80 ± 0.03	0.00 ± 0.00	0.00 ± 0.00	$N = 180$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	$N = 160$	0.64 ± 0.12	0.00 ± 0.00	0.00 ± 0.00	$N = 240$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	$N = 200$	0.56 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$N = 300$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GPPN (Lee et al., 2018)	$N = 20$	99.98 ± 0.01	92.68 ± 1.07	51.12 ± 5.00	$N = 30$	98.98 ± 0.25	25.98 ± 5.78	2.76 ± 1.68
	$N = 40$	$\mathbf{99.99 \pm 0.01}$	96.16 ± 3.56	65.17 ± 12.4	$N = 60$	$\mathbf{99.09 \pm 0.19}$	28.87 ± 1.47	1.32 ± 0.55
	$N = 60$	99.96 ± 0.02	91.47 ± 3.50	54.52 ± 7.32	$N = 90$	98.59 ± 0.06	25.35 ± 2.66	0.86 ± 0.59
	$N = 80$	99.97 ± 0.03	95.44 ± 4.48	66.85 ± 15.5	$N = 120$	98.67 ± 0.37	25.60 ± 4.87	1.35 ± 0.98
	$N = 100$	99.95 ± 0.05	93.34 ± 4.16	60.57 ± 13.6	$N = 150$	98.51 ± 0.31	21.62 ± 3.50	0.73 ± 0.68
	$N = 120$	$\mathbf{99.99 \pm 0.01}$	95.57 ± 3.27	66.99 ± 15.2	$N = 180$	90.49 ± 8.62	7.40 ± 8.56	0.41 ± 0.58
	$N = 160$	99.96 ± 0.01	95.51 ± 3.13	66.74 ± 12.8	$N = 240$	93.98 ± 2.48	8.64 ± 5.21	0.15 ± 0.11
	$N = 200$	99.98 ± 0.01	92.79 ± 1.28	50.88 ± 3.59	$N = 300$	95.38 ± 2.01	6.29 ± 4.35	0.02 ± 0.03
Highway network (Srivastava et al., 2015b)	$N = 40$	99.65 ± 0.17	96.04 ± 0.63	75.86 ± 10.0	$N = 60$	97.93 ± 0.56	62.95 ± 8.79	17.46 ± 5.45
	$N = 60$	99.69 ± 0.11	94.31 ± 0.55	64.94 ± 5.61	$N = 90$	94.59 ± 1.51	49.91 ± 11.8	13.98 ± 5.86
	$N = 80$	99.70 ± 0.05	93.50 ± 1.15	62.22 ± 5.87	$N = 120$	93.65 ± 0.81	38.79 ± 2.68	4.05 ± 0.61
	$N = 100$	99.36 ± 0.32	91.11 ± 2.64	60.32 ± 8.87	$N = 150$	85.42 ± 4.20	12.55 ± 3.89	0.35 ± 0.23
	$N = 120$	99.51 ± 0.17	88.45 ± 2.60	51.88 ± 4.24	$N = 180$	0.23 ± 0.01	0.00 ± 0.00	0.00 ± 0.00
	$N = 160$	99.50 ± 0.05	90.11 ± 0.93	60.57 ± 3.33	$N = 240$	0.25 ± 0.04	0.00 ± 0.00	0.00 ± 0.00
	$N = 200$	0.73 ± 0.12	0.00 ± 0.00	0.00 ± 0.00	$N = 300$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Highway VIN (ours)	$N = 40$	99.77 ± 0.09	98.83 ± 0.25	90.00 ± 2.12	$N = 60$	97.87 ± 0.60	77.02 ± 6.30	20.68 ± 9.89
	$N = 60$	99.83 ± 0.10	98.53 ± 0.72	94.35 ± 4.15	$N = 90$	95.31 ± 1.69	80.57 ± 7.40	34.72 ± 6.27
	$N = 80$	99.76 ± 0.02	98.03 ± 0.02	94.79 ± 0.67	$N = 120$	96.37 ± 1.82	84.81 ± 2.12	61.09 ± 3.50
	$N = 100$	99.93 ± 0.03	$\mathbf{99.52 \pm 0.12}$	$\mathbf{98.61 \pm 0.66}$	$N = 150$	97.77 ± 0.48	89.56 ± 0.95	75.42 ± 10.1
	$N = 120$	99.88 ± 0.04	98.62 ± 0.35	96.72 ± 1.76	$N = 180$	95.99 ± 1.75	85.18 ± 2.28	75.40 ± 4.05
	$N = 160$	99.86 ± 0.04	98.81 ± 0.24	96.76 ± 1.02	$N = 240$	97.64 ± 1.49	90.12 ± 3.68	82.40 ± 8.95
	$N = 200$	99.94 ± 0.01	99.13 ± 0.12	98.20 ± 1.75	$N = 300$	98.73 ± 0.50	$\mathbf{92.28 \pm 3.50}$	$\mathbf{90.06 \pm 3.13}$

Table 5: Optimality rates of each algorithm with various depths under 2D maze navigation tasks with different ranges of shortest path lengths. The optimality rate is defined by the ratio of tasks completed within the steps of the shortest path length to the total number of tasks.

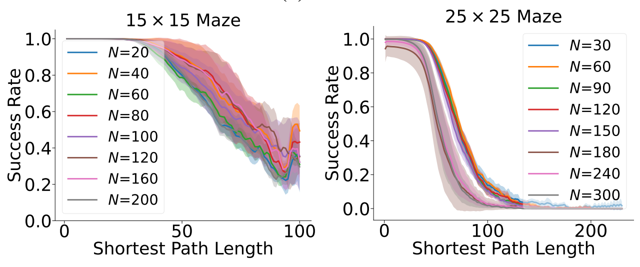
Maze Size	15×15				25×25			
Shortest Path Length		$[1, 30]$	$[30, 60]$	$[60, 100]$		$[1, 60]$	$[60, 130]$	$[130, 230]$
VIN (Tamar et al., 2016)	$N = 20$	99.15 ± 0.20	90.50 ± 0.59	53.31 ± 2.28	$N = 30$	93.94 ± 0.33	38.32 ± 3.64	2.25 ± 0.35
	$N = 40$	98.54 ± 0.13	86.71 ± 0.56	69.49 ± 2.77	$N = 60$	88.64 ± 2.81	33.74 ± 3.27	6.16 ± 2.38
	$N = 60$	98.29 ± 0.23	81.58 ± 2.57	61.12 ± 6.42	$N = 90$	0.20 ± 0.09	0.00 ± 0.00	0.00 ± 0.00
	$N = 80$	0.61 ± 0.16	0.00 ± 0.00	0.00 ± 0.00	$N = 120$	0.20 ± 0.09	0.00 ± 0.00	0.00 ± 0.00
	$N = 100$	0.72 ± 0.06	0.00 ± 0.00	0.00 ± 0.00	$N = 150$	0.21 ± 0.09	0.00 ± 0.00	0.00 ± 0.00
	$N = 120$	0.72 ± 0.06	0.00 ± 0.00	0.00 ± 0.00	$N = 180$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	$N = 160$	0.58 ± 0.04	0.00 ± 0.00	0.00 ± 0.00	$N = 240$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
	$N = 200$	0.56 ± 0.00	0.00 ± 0.00	0.00 ± 0.00	$N = 300$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
GPPN (Lee et al., 2018)	$N = 20$	99.35 ± 0.12	83.42 ± 2.24	46.97 ± 5.81	$N = 30$	96.33 ± 0.33	19.94 ± 4.66	2.46 ± 1.50
	$N = 40$	$\mathbf{99.64 \pm 0.16}$	90.47 ± 6.65	62.09 ± 12.1	$N = 60$	$\mathbf{96.53 \pm 0.65}$	21.20 ± 1.05	1.01 ± 0.58
	$N = 60$	99.36 ± 0.18	82.08 ± 4.75	49.27 ± 7.18	$N = 90$	94.78 ± 0.21	17.96 ± 2.25	0.66 ± 0.66
	$N = 80$	99.41 ± 0.24	88.89 ± 7.41	61.21 ± 13.3	$N = 120$	95.44 ± 0.38	18.97 ± 3.67	1.22 ± 0.89
	$N = 100$	99.27 ± 0.27	84.47 ± 5.51	55.82 ± 12.6	$N = 150$	95.05 ± 0.66	15.52 ± 2.88	0.70 ± 0.65
	$N = 120$	99.48 ± 0.12	88.22 ± 4.97	64.14 ± 14.6	$N = 180$	82.70 ± 11.6	5.40 ± 6.49	0.40 ± 0.56
	$N = 160$	99.26 ± 0.19	85.70 ± 3.63	60.75 ± 11.1	$N = 240$	87.76 ± 3.64	5.82 ± 3.55	0.12 ± 0.08
	$N = 200$	99.38 ± 0.08	84.65 ± 2.02	47.10 ± 3.90	$N = 300$	88.50 ± 4.61	4.10 ± 2.93	0.02 ± 0.03
Highway network (Srivastava et al., 2015b)	$N = 40$	98.98 ± 0.12	89.37 ± 0.73	71.27 ± 9.95	$N = 60$	92.57 ± 1.68	46.95 ± 6.95	13.72 ± 4.40
	$N = 60$	98.85 ± 0.06	85.92 ± 0.31	59.57 ± 6.76	$N = 90$	85.25 ± 3.39	35.28 ± 9.05	10.88 ± 4.04
	$N = 80$	98.62 ± 0.19	83.36 ± 0.17	56.04 ± 4.09	$N = 120$	83.27 ± 0.19	26.71 ± 2.02	2.48 ± 0.57
	$N = 100$	97.88 ± 0.24	80.06 ± 3.08	55.21 ± 9.75	$N = 150$	71.57 ± 5.70	7.47 ± 2.58	0.10 ± 0.10
	$N = 120$	97.89 ± 0.33	78.04 ± 2.05	46.05 ± 4.48	$N = 180$	0.23 ± 0.01	0.00 ± 0.00	0.00 ± 0.00
	$N = 160$	97.88 ± 0.42	78.57 ± 0.95	54.11 ± 3.72	$N = 240$	0.25 ± 0.04	0.00 ± 0.00	0.00 ± 0.00
	$N = 200$	0.65 ± 0.09	0.00 ± 0.00	0.00 ± 0.00	$N = 300$	0.24 ± 0.00	0.00 ± 0.00	0.00 ± 0.00
Highway VIN (ours)	$N = 40$	98.78 ± 0.04	$\mathbf{92.81 \pm 0.74}$	85.49 ± 2.83	$N = 60$	93.47 ± 0.67	62.70 ± 7.87	17.15 ± 8.31
	$N = 60$	98.47 ± 0.12	91.46 ± 1.55	88.67 ± 3.33	$N = 90$	89.72 ± 2.16	63.87 ± 8.26	27.84 ± 5.12
	$N = 80$	98.62 ± 0.23	91.29 ± 0.50	90.99 ± 0.41	$N = 120$	90.41 ± 1.76	64.20 ± 0.79	49.63 ± 2.57
	$N = 100$	98.43 ± 0.05	90.67 ± 0.51	$\mathbf{94.64 \pm 1.61}$	$N = 150$	92.00 ± 0.58	71.91 ± 2.37	64.70 ± 9.88
	$N = 120$	98.37 ± 0.16	90.24 ± 1.31	93.16 ± 3.63	$N = 180$	90.65 ± 1.93	66.42 ± 1.83	66.25 ± 2.94
	$N = 160$	98.30 ± 0.11	89.15 ± 1.30	92.00 ± 0.44	$N = 240$	91.32 ± 2.24	70.78 ± 4.28	71.09 ± 8.04
	$N = 200$	98.26 ± 0.10	89.33 ± 0.92	92.76 ± 2.08	$N = 300$	93.36 ± 1.85	$\mathbf{73.35 \pm 4.15}$	$\mathbf{81.08 \pm 2.87}$



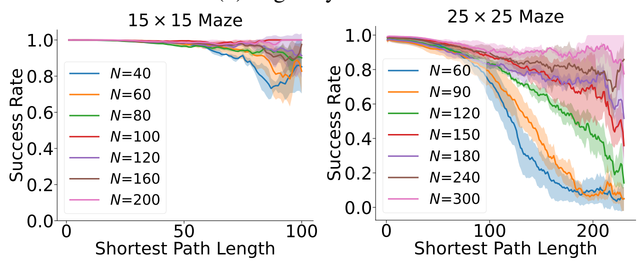
(a) VIN



(b) Highway network

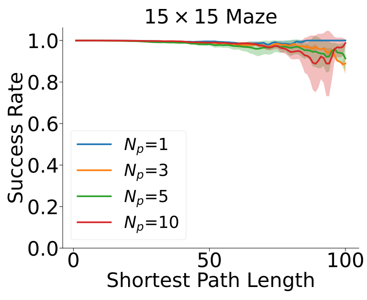


(c) GPPN

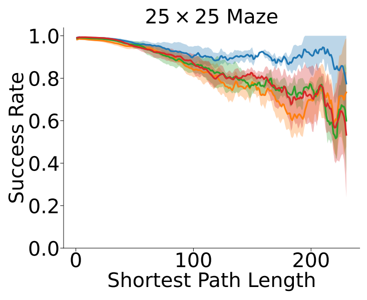


(d) Highway VIN (our)

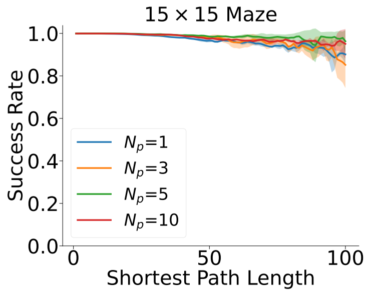
Figure 9: Success rates of each algorithm as a function of varying shortest path lengths on 2D maze navigation tasks.



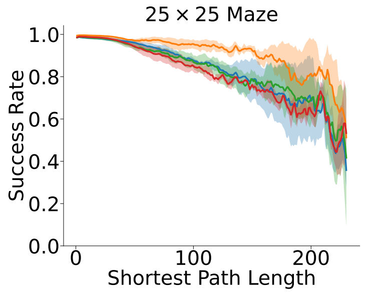
(a) Depth $N = 200$



(b) Depth $N = 300$

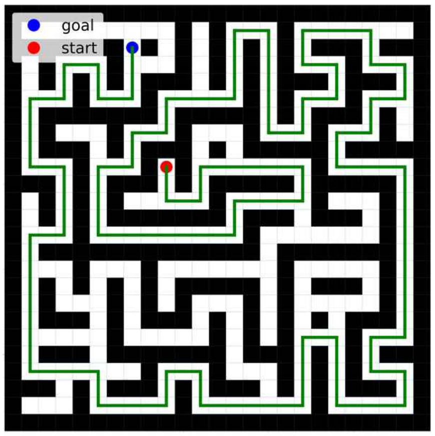


(c) Depth $N = 100$

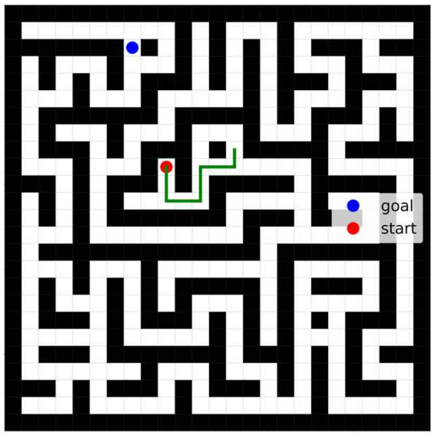


(d) Depth $N = 150$

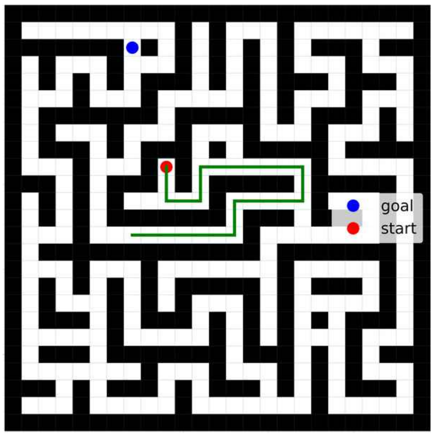
Figure 10: Success rates of highway VINs with varying numbers of parallel VE modules N_p under varying depths N of the network.



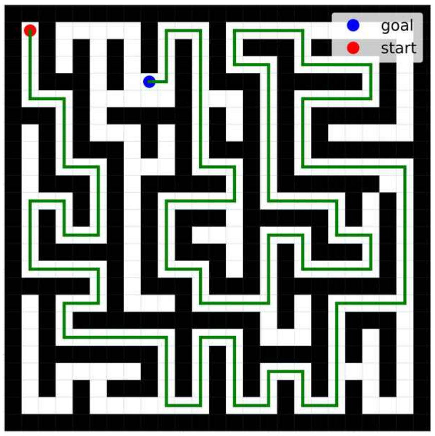
(a) Highway VIN



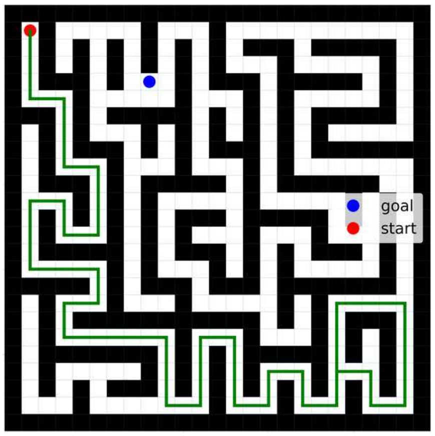
(b) VIN



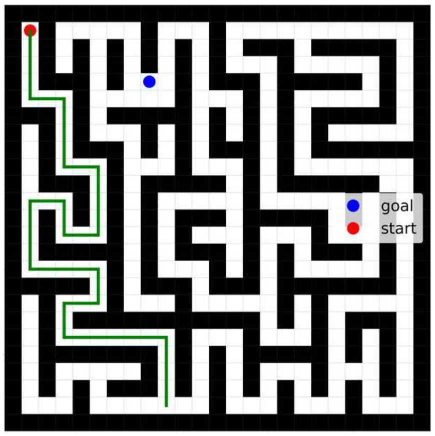
(c) GPPN



(d) Highway VIN



(e) VIN



(f) GPPN

Figure 11: Examples of 2D maze navigation tasks where highway VIN succeeds, but other methods fail.