Universal Value Iteration Networks: When Spatially-Invariant Is Not Universal

Li Zhang,[1] **Xin Li,**[*1] **Sen Chen,**[1] **Hongyu Zang,**[1] **Jie Huang,**[1] **Mingzhong Wang**[2]
[1]School of Computer Science, Beijing Institute of Technology, China
[2]USC Business School, University of the Sunshine Coast, Australia
{3120181069, xinli, 3220190781, zanghyu, 3220190813}@bit.edu.cn, mwang@usc.edu.au

Abstract

In this paper, we first formally define the problem set of spatially invariant Markov Decision Processes (MDPs), and show that Value Iteration Networks (VIN) and its extensions are computationally bounded to it due to the use of the convolution kernel. To generalize VIN to spatially variant MDPs, we propose Universal Value Iteration Networks (UVIN). In comparison with VIN, UVIN automatically learns a flexible but compact network structure to encode the transition dynamics of the problems and support the differentiable planning module. We evaluate UVIN with both spatially invariant and spatially variant tasks, including navigation in regular maze, chessboard maze, and Mars, and Minecraft item syntheses. Results show that UVIN can achieve similar performance as VIN and its extensions on spatially invariant tasks, and significantly outperforms other models on more general problems.

Introduction

Literature in psychology and neuroscience reports that behaviors are generally controlled by two systems, a rigid retrospective model-free system and a flexible prospective model-based system (Moran et al. 2019). Likewise, it is intuitive in deep reinforcement learning (DRL) research to combine the strength of both model-based (sample-efficiency) and model-free (accuracy) approaches to improve performance.

Despite the great advancement in DRL research, most existing approaches focus on leveraging the deep neural network structure to approximate the value function via a trial-and-error learning process (Mnih et al. 2015; Van Hasselt, Guez, and Silver 2016; Wang et al. 2015; Hessel et al. 2018; Lillicrap et al. 2015; Mnih et al. 2016; Schulman et al. 2015; 2017), but insufficiently address explicit planning computation as in the conventional model-based approaches, which is critical for long-term reasoning and inference.

Value Iteration Networks (VIN) (Tamar et al. 2016) proposes to encode the value iteration process with a convolutional neural network (CNN) to achieve learning and planning simultaneously. With the planning module, VIN generalizes well in conventional navigation domains, showing that the planning computation module is beneficial to applications requiring complex long-term reasoning.

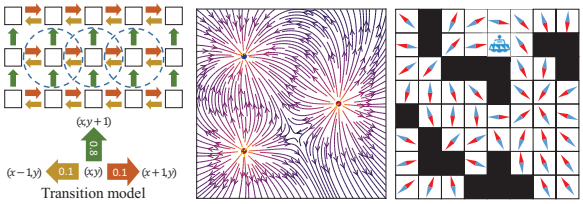


Figure 1: (a) Spatially invariant system with the dynamics in each state. (b) & (c) Toy navigation problem on Mars. (b) The magnetic fields on Mars. (c) A navigation map on Mars. The consequence of action "go north" varies in each grid on Mars, while remains the same on Earth.

To adapt to more complex navigation tasks, MACN (Khan et al. 2018) and AVINs (Schleich, Klamt, and Behnke 2019) extend VIN with a memory augmented controller and representations with multiple levels of abstraction.

The strength of VIN and its extensions (VINs) lie in the exploitation of the resemblance between the value iteration and the convolution operation. However, the use of the CNN structure also limits them to solving the "conventional" navigation problems, whose state space can be seen as regular lattices. To work with irregular spatial graphs, GVIN (Niu et al. 2018) employs a graph convolution operator to simulate the value iteration process.

Nevertheless, we argue that the use of convolution operator in VINs/GVIN makes them only applicable to the problem set of *spatially invariant* Markov Decision Processes (MDPs).

Definition 1. *An MDP is a quadruple* $M = \{S, A, R, T\}$ *where* S *is a set of states,* A *is a set of actions,* $R(s, a)$ *is a reward function for acting* $a \in A$ *at* $s \in S$*, and* $T : S \times A \times S \to [0, 1]$ *is the transition function which encodes the probability of the next state given the current state and the*

*Xin Li is the corresponding author. This work has been partially supported by National Key R&D Program of China under Grant No. 2017YFB0803300, NSFC under Grant No. 61772074.

chosen action.

Definition 2. *An MDP $M = \{S, A, R, T\}$ is spatially invariant if and only if there is a function $d : S \times S \to \mathbb{R}^n$, s.t.*

$$\begin{aligned} &\textit{if} \quad d(s_i, s_j) = d(s'_i, s'_j) \quad \textit{then} \\ &p(s_j|s_i, a) = p(s'_j|s'_i, a) \\ &\forall s_i, s_j, s'_i, s'_j \in S \quad \forall a \in A \quad \forall p \in T \end{aligned} \tag{1}$$

A "conventional" navigation problem is a spatially invariant MDP as it requires the same transition model for each grid unit in the map (Fig.1a). This can be proven by defining $d(s_{x_i,y_i}, s_{x_j,y_j}) = [x_i - x_j, y_i - y_j]$ in which $s_{x,y}$ denotes a state being in the grid (x, y). In fact, VINs implicitly exploit the property of spatially invariant MDPs with a relatively small convolution kernel to approximate such transition model.

$$Q(s_{x,y}, a) = \sum_{i,j} W^a_{i,j}(\gamma V(s_{x-i,y-j}) + r) \tag{2}$$

Eq.(2)[1] denotes VIN's planning computation via convolution operation where W^a denotes the convolution kernel for action a, which encodes the transition model for navigation problems. It is apparent that the transition probability ($W^a_{i,j}$) of taking a at (x, y) needs to be the same for any pair of states (x', y') whose coordinates satisfy $x' - x = i$ and $y' - y = j$. Thus, VIN greatly reduces the parameters to be learnt to embed the MDP's transition dynamics on spatially invariant problems. More specifically, the size of transition function is reduced from $|S| \times |A| \times |S|$ to $|A| \times |K|$ where $|K|$ is the size of the convolution kernel which is usually far less than $|S|$, which in turn improves the sample efficiency.

In fact, the use of convolution kernel presumes the existence of the universal transition model of the problems. When this is not true, or when the problem is not spatially invariant, VINs will fail because a small kernel cannot precisely summarize the transition dynamics. Our experiments later show that VINs fail to deal with cases in Fig.1b and 1c, which depict a navigation problem on Mars where a global intrinsic magnetic field does not exist, e.g., an agent performing an action of going west at a grid does not necessarily land in its left grid, which contradicts to the spatially invariant MDP case as on Earth.

There are two major directions to enable VINs on general spatially variant problems: 1) Keep the CNN module during the planning computation, but determine the proper coordinates of states in 2-D space, thus allowing the convolution kernel to capture the dynamics as sufficiently as possible; 2) Replace the CNN module with more flexible network structures to capture the dynamics.

In this paper, we propose Universal Value Iteration Networks (UVIN) to explore the second path. The key of UVIN is to automatically learn a compact module/structure for the planning computation which can sufficiently summarize the transition dynamics and save computational cost of value iteration. UVIN can be trained via imitation learning or reinforcement learning. In imitation learning, the planning module is supervised by the expert trajectories. In contrast, in reinforcement learning, the planning module is updated along with agent's interactions with the environment during the learning process, and the planning module is optimized periodically and synchronously.

We first evaluate UVIN in "conventional" maze navigation problems and demonstrate that it achieves the same good performance in comparison with other models (VIN, GVIN and GPPN). We then evaluate these models on general spatially variant problems, including chessboard maze navigation, navigation on Mars, and deterministic/stochastic Minecraft tasks. Rainbow (Hessel et al. 2018), the state-of-the-art model-free sequential decision making framework, is also included in the comparison. The experimental results demonstrate our UVIN significantly outperforms all other models in spatially variant problems.

The main contributions of this paper include:

- We provide a formal definition of spatially invariant MDP and show that VINs are computationally bounded to it due to the use of the convolution kernel.
- We propose UVIN to deal with spatially variant MDPs. UVIN automatically learns a flexible but compact network structure to encode the transition dynamics of the problems and support the differentiable planning module.
- We evaluate UVIN with both spatially variant and spatially invariant tasks. The results demonstrate that our method significantly outperforms other state-of-the-art approaches in spatially variant tasks while achieving the same level of performance in conventional spatially invariant tasks.

Related Work

Value Iteration. VI (Bellman 1957) was introduced to compute an optimal solution to an MDP (Def.1). It formalizes the underlying model as interactions between the agent and the environment (Sutton and Barto 1998). At time step t, the agent receives $state\ s_t \in S$, the representations of the environment, and takes an $action\ a_t \in A$. Then, it receives a scalar $reward\ r_t = R(s_t, a_t)$, and lands in the next state s_{t+1}. The goal is to find a $policy\ \pi(a_t \mid s_t)$ that maximizes the cumulative rewards:

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}. \tag{3}$$

Following a given policy π, let $V^\pi(s)$ be the expected long-term reward from state s, and $Q^\pi(s, a)$ be the expected reward of taking action a in state s.

$$\begin{aligned} V^\pi(s) &= E[R_t \mid s_t = s] \\ Q^\pi(s, a) &= E[R_t \mid s_t = s, a_t = a] \end{aligned} \tag{4}$$

The optimal policy is $\pi^* = \operatorname{argmax}_\pi V^\pi$, the optimal state-value function is $V^* = \max_\pi V^\pi$, and the optimal action-value function is $Q^* = \max_\pi Q^\pi$. To compute π^*, Bellman equation is solved iteratively as:

$$\begin{aligned} Q(s, a) &\leftarrow R(s, a) + \gamma \sum_{s'} T(s, a, s') V(s') \\ V(s) &\leftarrow \max_a Q(s, a) \end{aligned} \tag{5}$$

[1]Please refer to (Tamar et al. 2016) for more details.

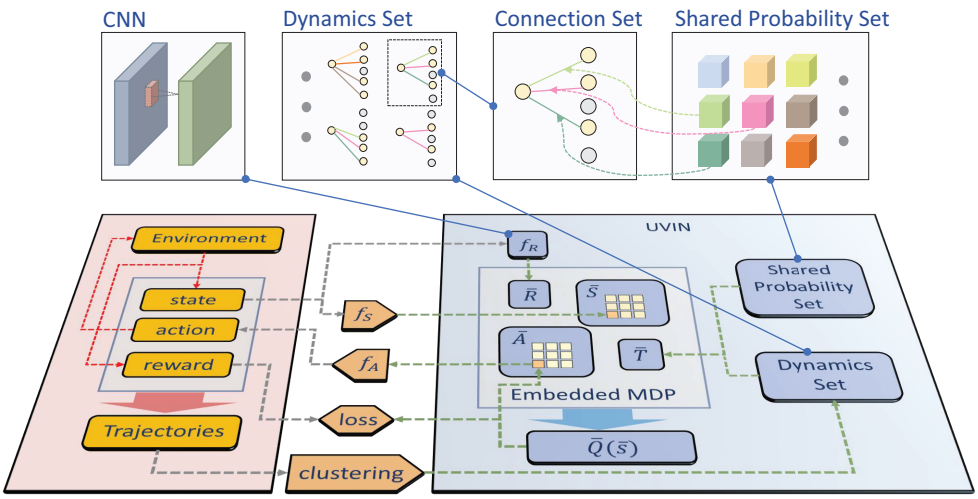


Figure 2: The UVIN Framework. It performs value iteration on the embedded MDP $\bar{M}$. The embedded transition function $\bar{T}$ is represented by both weights and the structure of the network. Dynamics Set defines the structure, which is learnt from the trajectories. Shared Probability Set contains trainable variables which define the weights of the network, and it is trained by BP.

Differentiable planning module. VIN (Tamar et al. 2016) proposes to embed the differentiable planning module via a CNN. Memory Augmented Control Network (MACN) (Khan et al. 2018) extends VIN model with a memory augmented controller, empowering VIN to backtrack through the history of previous trajectories. AVIN (Schleich, Klamt, and Behnke 2019) extends VIN with the representations supporting multiple levels of abstraction, allowing VIN to deal with more complex tasks, e.g., planning omnidirectional driving for a search-and-rescue robot in cluttered terrain. Cognitive Mapper and Planner (CMP) (Gupta et al. 2017) proposes to plan actions in first person view. It combines a neural network, which processes first person images to build up a latent map representation of the environment, with a hierarchical planning module based on VIN, which plans on multiple spatial scales. GVIN (Niu et al. 2018) substitutes the CNN module with a graph convolution operator to enable the application of VIN in irregular spatial graphs. Recently, Value Propagation Network (VPN)(Nardelli et al. 2019) was proposed to employ the "Value Propagation" and "state-dependent discount factor" to generalize VIN for better sample complexity, however the convolutional kernels are still kept in VPN hidering the adaption of the spatially-variant MDP problems.

However, VIN and its extensions are only applicable to spatially invariant MDP problems. (Lee et al. 2018a) commented on the issue, but did not provide the formal or clear identification of the restriction regarding the "spatially invariant" problem set. GPPN (Lee et al. 2018b) somewhat alleviates this issue by reframing VIN as a recurrent-convolutional network and replacing the recurrent update with LSTM, which enable it to filter out some inappropriate values caused by the inaccurate transition model. Although GPPN can capture simple patterns, it fails to deal with more complex transition dynamics in spatially variant problems.

Universal Value Iteration Networks

This section introduces the UVIN framework which learns a compact structure to embed the transition dynamics together with the explicit planning computation. The training of UVIN via imitation learning (IL) and reinforcement learning (RL) is also discussed respectively.

Let $M = \{S, A, R, T\}$ be the underlying MDP of the task environment. Let $\bar{M} = \{\bar{S}, \bar{A}, \bar{R}, \bar{T}\}$ be the embedded MDP of UVIN. The goal of UVIN is to learn $\bar{M}$ which approximates M in terms of the optimal policy, and then solve $\bar{M}$, thus obtaining the optimized solution for the original task M. With $\bar{M}$ initialized, Eq.(5) is applied to compute the Q-function $\bar{Q}(\bar{s}, \bar{a})$ of the embedded MDP. The objective is to minimize the loss between $\bar{Q}(\bar{s}, \bar{a})$ of $\bar{M}$ and $Q(s, a)$ of M. As the VI algorithm does not have any non-differentiable operator, the parameters of $\bar{M}$ can be iteratively updated with supervisory signals (optimal actions in IL setting and rewards in RL setting) and backpropagation.

To construct $\bar{M}$, we need to first define $\bar{S}$ and $\bar{A}$. A general approach is to find the optimal projection function $f_S : S \to \bar{S}$ and $f_A : A \to \bar{A}$. In this paper, we simply let $\bar{M}$ and M share the same action space ($f_A(a) = a$), and let f_S select a subset of s. For example, if $s \in S$ is $\{map, goal, current\_position\}$, $f_S(s)$ could be $\{current\_position\}$.

To encode the reward signals, we leverage a CNN to learn the function $f_R : S \to \{\bar{R}\}$ to convert $s \in S$ to $\bar{R}$. It should be noted that the use of CNN for the embedded reward representation does not compromise UVIN's capability of dealing with spatially variant MDPs because Def.2 only sets the constraints on the transition dynamics.

Then, the challenging part is to encode the transition function $\bar{T}$. Apparently, $|S| \times |A| \times |S|$ variables are sufficient, but not efficient, to encode the transition dynamics. In fact, it is not necessary to maintain such massive number of variables, as many of them share the same dynamics.

UVIN represents the network structure as a "Dynamics Set" $D = \{C_{11}, C_{12}, \cdots, C_{21}, C_{22}, \cdots, C_{|\bar{S}||\bar{A}|}\}$, in which $C_{ij} = \{(\bar{s}_{k_1}, p_{k_1}), \cdots, (\bar{s}_{k_{cn}}, p_{k_{cn}})\}$ is a "Connection Set" recording the local connectivity for state $\bar{s}_i \in \bar{S}$. C_{ij} delimits all states that could be reached and their corresponding probability p when the agent at state $\bar{s}_i$ takes action $\bar{a}_j \in \bar{A}$.

To boost the sample efficiency, UVIN uses a "Shared Probability Set" $P = \{p_1, p_2, \cdots, p_{p_n}\}$, which is trained during the learning process, for possible values of transition probabilities.

Computing the optimal D can be treated as the estimation of transition probabilities for random variables. Let $\hat{p}(\bar{s}'|\bar{s}, \bar{a})$ be the estimated value of $p(\bar{s}'|\bar{s}, \bar{a})$. In addition to the conventional estimation problem, we have an additional constraint: $\forall \hat{p}(\exists p_i \in P \wedge \hat{p} = p_i)$. That is, we must select a value from P rather than use an arbitrary value. The objective is to minimize the mean distance between p and $\hat{p}$:

$$\begin{aligned} &\min \frac{1}{|\bar{S}|^2|\bar{A}|} \sum_{\bar{s},\bar{a},\bar{s}'} |p(\bar{s}'|\bar{s}, \bar{a}) - \hat{p}(\bar{s}'|\bar{s}, \bar{a})| \\ &\hat{p}(\bar{s}'|\bar{s}, \bar{a}) = \begin{cases} p_k & (\bar{s}', p_k) \in C_{\bar{s}\bar{a}} \\ 0 & (\bar{s}', *) \notin C_{\bar{s}\bar{a}} \end{cases} \\ &\forall C_{\bar{s}\bar{a}} \in D \end{aligned} \tag{6}$$

To compute P, a straightforward solution is to do clustering on all $p(\bar{s}'|\bar{s}, \bar{a})$ with the number of clusters as $|P|$. After the clustering process is converged, let $\hat{p}(\bar{s}'|\bar{s}, \bar{a})$ equal to its nearest cluster center. Thereafter, the optimal D can be computed. It should be noted that both P and D jointly define the network. In UVIN, after D is determined, P will be further updated via backpropagation running through the planning computation (VI) as below:

$$\begin{aligned} \bar{Q} &= \bar{r} + \gamma \sum p_i \bar{V} \\ \bar{V} &= \max \bar{Q} \end{aligned} \tag{7}$$

Fig.2 illustrate the overall structure of the UVIN framework. The rest of this section discusses the differences in training UVIN with IL and RL.

Imitation Learning In IL, expert trajectories, but not the interactions with the environment, are available to the agent. Thus, we could utilize the frequencies of the transition tuples $\{(s_1, a_1, s_1'), \cdots, (s_n, a_n, s_n')\}, \forall s_i, s_i' \in S, \forall a_i \in A$ to estimate $p(\bar{s}'|\bar{s}, \bar{a})$, and compute P and D.

With D known, UVIN outputs the embedded $\bar{Q}$ for a mini batch sample via Eq.(7). By computing the loss between $\bar{Q}$ and the optimal action (Ross, Gordon, and Bagnell 2011), f_R and P can then be updated by backpropagation.

Reinforcement Learning In RL, the frequencies of embedded dynamics $f(\bar{s}'|\bar{s}, \bar{a})$ can only be counted during the learning process, and the "Dynamics Set" D will be updated periodically by a clustering program, indicating that the network structure is dynamically changing during the weight optimization process.

In UVIN, the structure is guided by the frequency $f(\bar{s}'|\bar{s}, \bar{a})$, which is independent of the weights and will converge to the probability $p(\bar{s}'|\bar{s}, \bar{a})$ according to the law of large numbers.

Algorithm 1 UVIN training via reinforcement learning

Input:
a state projection function $f_S : S \to \bar{S}$
an action projection function $f_A : A \to \bar{A}$

initialize parameterized reward projection function f_R^θ
initialize Shared Probability Set $P = \{p_0, p_1, \cdots, p_{p_n}\}$
initialize Dynamics Set D
for $episode = 1$ to T **do**
  reset the environment, get initial state s
  repeat
    $\bar{R} = f_R^\theta(s)$
    update $\bar{Q}$ iteratively with Eq.(7)
    $\bar{a} = \text{argmax}_a \bar{Q}(f_S(s), a)$
    $s', r = \text{step}(f_A^{-1}(\bar{a}))$
    add $(s, f_A^{-1}(\bar{a}), r)$ to buffer
    update the frequency $f(\bar{s}'|\bar{s}, \bar{a})$
  until episode ends
  $R = 0$
  for (s, a, r) in reversed(buffer) **do**
    $R = r + \gamma R$
    update $\bar{Q}$ iteratively with Eq.(7)
    $L = \text{MSE}(\bar{Q}(f_S(s), f_A(a)), R)$
    update f_R^θ and P by $\frac{\partial L}{\partial \theta}$ and $\frac{\partial L}{\partial p_i}$
  end for
  clear the buffer
  update D by clustering on $f(\bar{s}'|\bar{s}, \bar{a})$
end for

We apply episodic Q-learning similar to GVIN. When an episode ends, calculate the cumulative discounted reward for each state and its MSE loss with the Q-value output via NN to update the corresponding parameters. The details of training UVIN via RL is summarized in Algorithm 1.

UVIN utilizes a flexible structure to find the compact representations for each problem. In comparison, VIN uses the fixed structure (CNN) and works only for spatially invariant problems as stated earlier. Moreover, the structure found by UVIN could be more compact than CNN, allowing UVIN to use less parameters to capture the transition model. In other words, with the same number of parameters, UVIN could represent more complicated dynamics[2].

Experiments

We evaluated UVIN with both spatially invariant and spatially variant tasks, covering: 1) Maze navigation, which consists of randomly generated mazes, and 2) Penalized Minecraft synthesis task, which consists of goal and undesired inventory states.

[2]VIN needs $|A| \times |K|$ parameters to capture the transition model, UVIN only uses $|K|$ parameters to achieve similar performance as VIN. In experiments, we set $|K| = 9$ as suggested in (Tamar et al. 2016).

Table 1: Performance on Maze Navigation. All models used the same training epoch, learning rate, and the number of iterations. Bold text highlights the best result passing t-test[3].

	Regular			Chessboard			Mars		
	%Acc	%Suc	Reward	%Acc	%Suc	Reward	%Acc	%Suc	Reward
UVIN	99.36	99.01	$0.679(\pm 0.27)$	98.43	99.04	$\mathbf{0.673(\pm 0.27)}$	82.27	95.52	$\mathbf{0.615(\pm 0.47)}$
GPPN	98.01	93.97	$0.598(\pm 0.49)$	98.32	95.24	$0.623(\pm 0.44)$	57.82	25.25	$-0.779(\pm 0.94)$
GVIN	94.43	98.05	$0.660(\pm 0.35)$	30.69	8.16	$-1.241(\pm 0.69)$	13.29	1.45	$-1.309(\pm 0.38)$
VIN	95.10	99.04	$\mathbf{0.683(\pm 0.27)}$	46.64	5.84	$-1.346(\pm 0.62)$	18.56	2.69	$-1.288(\pm 0.46)$
VI	n/a	99.17	$0.684(\pm 0.27)$	n/a	99.07	$0.677(\pm 0.28)$	n/a	98.33	$0.659(\pm 0.34)$

Compared Methods

We compare our proposed UVIN with the following methods:

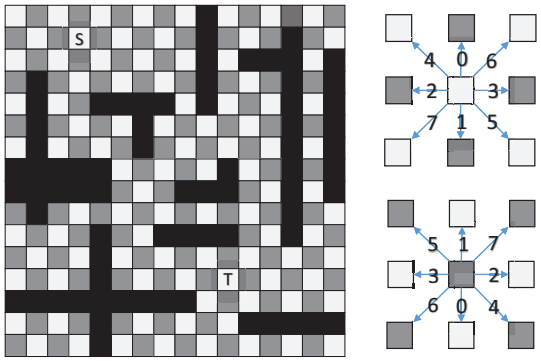
- **VIN:** Value Iteration Networks was proposed to embed the differentiable planning module via a CNN (Tamar et al. 2016).
- **GVIN:** Generalized Value Iteration Networks (Niu et al. 2018) substitutes the CNN module with a graph convolution operator to enable the application of VIN in irregular spatial graphs.
- **GPPN:** Gated Path Planning Networks (Lee et al. 2018b) reframes VIN as a recurrent-convolutional network and replaces the recurrent updates with LSTM.
- **Rainbow:** A state-of-the-art model-free method (Hessel et al. 2018) which ensembles several improvements to DQN, showing great advantages in Atari games.

Maze Navigation via Imitation Learning

In maze navigation problems, each grid is denoted as 1 for being an obstacle or 0 for a free-zone. A state includes the information of the map, the agent's position and the goal. Eight actions are available for the agent, each represents a move along with a specific direction, such as east and northeast. We applied three types of mazes with different transition models, including regular maze, chessboard maze, and maze on Mars. Regular maze navigation belongs to the typical spatially invariant MDP problems. The other two represent spatially variant problems.

For each experiment, we generated $20,000$ maps of size 16×16 with obstacles and the goal randomly set[4]. We provided 10 positions with the corresponding optimal actions to the goal as the supervised labels for each map (given by *shortest path algorithm*). We used 80% of them as the training set, the remaining 20% as the test set.

In comparison with the shortest path problem, the agent does not have complete knowledge about the consequence of each action. Therefore, VINs use the convolution operator to introduce a prior to ensure an action leads to its neighbor position, and the consequence of the actions is same for diverse states (spatially invariant). However, UVIN does not require any prior knowledge about the actions. In stead, it learns everything from the imitation data.



(a) (b)

Figure 3: Chessboard Maze. The obstacles, starting position, and the goal are denoted as black grids, S, and T respectively. It has two types of free grids (white and grey), which have opposite consequences for an action. (b) shows the transition models for each type of free grids.

Three metrics are used to evaluate the performance:

- *%Acc*, the accuracy of predicting labels in test set
- *%Suc*, the success rate that the agent reaches the goal state
- *Reward*, the mean and standard deviation of the cumulative discounted rewards

Table 1 shows the results of UVIN in comparison with VIN, GVIN, GPPN, and the conventional VI.

Regular Maze As a typical spatially invariant problem, Table 1 shows that VIN has the best performance which is very close to that of the classic VI which works directly on the ground-truth model, which verifies the effectiveness of using convolution operator on such type of problems. UVIN performs slightly worse than VIN but better than GVIN and GPPN, proving that UVIN can effectively learn the transition model without any prior knowledge of the structure.

It should be noted that VIN and GVIN require $|A|$ channels to capture the transition model for each $a \in A$. In fact, the models they learnt can be further abstracted/compressed simply by the rotation of coordinate axes. In comparison, UVIN directly explores the transition model to achieve more

[3]Use Benjamini-Hochberg procedure with the significance level $p = 0.05$.

[4]The numbers of obstacles were randomly selected following the uniform distribution on the interval $[1, |\bar{S}|/2]$. If there was no path to the goal state, the generated problem was discarded.

compact NN structure to learn the dynamics, thus achieving better sample-efficiency.

Chessboard Maze In a chessboard maze, there are two types of free grids, which are labeled as grey and white in Fig.3. They have opposite consequences for an action. Such setting makes the navigation task no longer spatially invariant as in regular maze problems.

Table 1 shows that UVIN has the best performance which is very close to VI. It is evident that VIN and GVIN do not work well on such spatially variant problems. GPPN shares the same problem as VIN and GVIN, but it performs much better than VIN and GVIN. This may be attributed to its use of LSTM to replace the max operator in Eq.(5). Therefore, GPPN can filter out some noise caused by the inaccurate transition model.

Figure 4 visualize the navigation map and the corresponding value function for UVIN, VIN, and GVIN. The shape of the value function of UVIN resembles the original navigation map the most truthfully, proving that UVIN has learnt the reward function and transition function close to the original task.

As the value functions of VIN and GVIN show a relatively accurate approximation near the goal state, we can conclude that the reward functions learnt by them are relatively good. Thus, their poor performance (Table 1) should be the result of the inaccurate learning of the transition model (the convolution kernel). For spatially variant problems, the enforced use of the convolution kernel tends to average the weights inside the kernel for different consequences of an action to minimize the loss, thus making the value function more "smooth".

Maze on Mars As Mars does not have a global intrinsic magnetic field, an agent performing an action of going west at a grid does not necessarily land in the grid on its left, which turns the navigation on Mars no longer spatially invariant as on Earth. In this paper, we tested the navigation in a local area on Mars with a hypothetical magnetic field. The landing pattern of taking a specific action at a state varies grid by grid, making Mars navigation much more complex than the chessboard navigation.

Table 1 shows that UVIN again has the best performance, which is close to that of the classic VI, due to its capability of capturing complex transition models. In comparison, all other models, including GPPN, fail to deal with navigation problems at this level of complexity.

Penalized Minecraft Synthesis Task (PMST)

Minecraft is a sandbox video game that allows players to explore, gather, and craft in a 3D world. In Minecraft, most items can not be obtained directly, but need to be synthesized. And the synthesized items could be used as ingredients to synthesize more advanced items. To collect the desired items in the inventory, players need to plan whether to search for or synthesize a new item, and how. Thus, Minecraft is a typical problem requiring long-term reasoning.

To evaluate UVIN in more realistic domains, we designed synthesis tasks in the Minecraft environment. In addition to getting the desired inventory, if the agent's decisions result in an unwanted inventory, it will be penalized for them.

Table 2: Performance Comparison on Deterministic PMSTs

	%Acc	%Suc	Reward
Imitation Learning			
UVIN	96.18	100.0	$\mathbf{0.955(\pm 0.02)}$
GPPN	87.01	51.00	$0.023(\pm 0.96)$
GVIN	25.47	0.00	$0.000(\pm 0.00)$
VIN	25.65	0.00	$0.000(\pm 0.00)$
Reinforcement Learning			
UVIN	-	100.0	$\mathbf{0.951(\pm 0.02)}$
GPPN	-	1.00	$0.007(\pm 0.11)$
GVIN	-	1.00	$-0.036(\pm 0.04)$
VIN	-	3.00	$0.03(\pm 0.17)$
Rainbow	-	5.00	$0.046(\pm 0.22)$

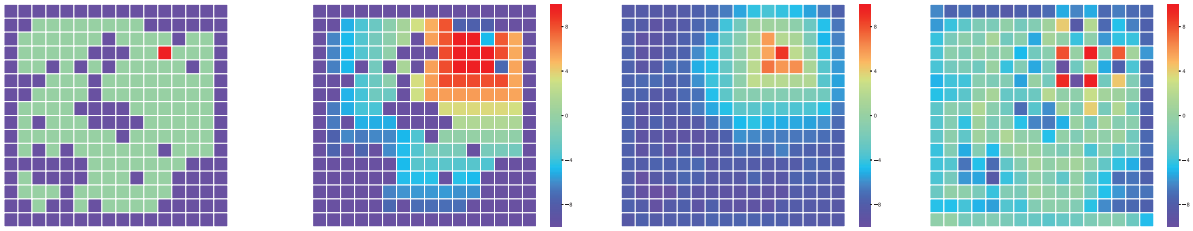
PMSTs can be formalized as being hierarchical. The input of PMSTs contains the frames of images and its associated inventory storage. UVIN works at the top level to determine a sub-policy to be used to enable the inventory state transition and eventually complete the execution.

The state of PMST is defined as $s_i = [z_1, z_2, ..., z_{|IT|}]$, $s \in S \subseteq \mathbb{N}^{|IT|}$, where $z_j \in \mathbb{N}$ and $|IT|$ represents the number of item types. Each sub-policy needs to work on a specific z_j, such as collecting or synthesizing enough copies of i-th type of items (z_j). Thus, each sub-policy is in fact a high-level action $a_i \in A$ leading the state transitions and computing the rewards in UVIN. To speed up the training process, we developed a simulator, which skips the gameplay, to mimic the state transition.

We set up both deterministic and stochastic environments to evaluate UVIN's performance on PMSTs. The state transition probability can only be 0 or 1 ($p(s'|s,a) \in \{0,1\}$) in the deterministic environment, while it can be $p(s'|s,a) \in [0,1]$ in the stochastic environment. The results of experiments demonstrate that UVIN is capable of solving all tasks successfully via IL/RL, while other methods fail.

Deterministic PMST (DPMST) In DPMST, the request for each item type is binary ($z_i \in \{0,1\}$). That is, either a player needs it. Therefore, if we have $|IT|$ types of items to work with, there will be $2^{|IT|}$ states in the DPMST. Similar to the maze experiments, we generated $20,000$ DPMSTs with the penalized states and goal state randomly set in the experiments. During the execution, an agent gets $+1$ reward for reaching the goal state, or gets -1 and terminates the episode for reaching any the penalized state. If the agent cannot arrive at the goal state within 250 steps, the episode also terminates. The state space was converted into a 2-D grid map to run VIN, GVIN, and GPPN as the comparison models. To minimize the impact of grid-map reshaping on their performance, we experimented grid-maps with varied shapes for fairer comparison.

DPMST via Imitation Learning DPMST is far more complex than maze problems. Table 2 shows that UVIN is



(a) map (b) UVIN (c) VIN (d) GVIN

Figure 4: Visualization of the map and value function. (a) The map: purple, red, and green denote the obstacles, goal, and free grids respectively. (b) The value function of UVIN. (c) The value function of VIN. (d) The value function of GVIN.

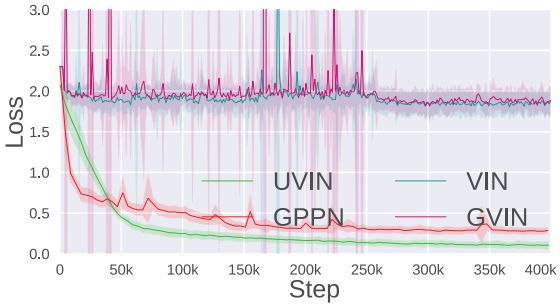
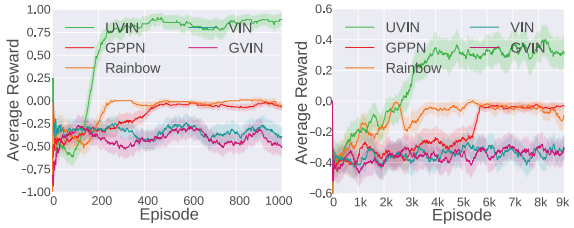


Figure 5: Training loss in Deterministic PMST (IL)



(a) Deterministic PMST (b) Stochastic PMST

Figure 6: Training performance in PMST (RL)

the only model that is capable of dealing with DPMSTs via IL. The results prove that the flexible compact module designed for UVIN can learn the model dynamics much more effectively than the convolution operators in VINs.

GPPN has relatively high accuracy in predicting labels. However, its success rate is only half of the UVIN's which indicates the difficulties of such problems and the importance of long-term reasoning.

IL is in general a supervised learning. Figure 5 illustrates the loss in the training process of each model. UVIN stably converges along with learning steps while VIN and GVIN oscillate acutely at the beginning. This might be related to their enforced use of the convolution kernel which has inconsistent gradient directions when the affects/dynamics of an state-action vary over the state space.

DPMST via Reinforcement Learning. In this setting, the agent interacted with our simulator for 2 million steps. UVIN, VIN, GVIN, and GPPN were all trained by episodic Q-learning (Niu et al. 2018). Moreover, Rainbow (Hessel et al. 2018) was included in the comparison. Rainbow is a state-of-the-art model-free method which ensembles six improvements to DQN, showing great advantages in Atari games. We used the same hyper parameters as its paper except the first convolution layer was adjusted to adapt to the input size.

Table 2 shows that UVIN is the only model that is capable of dealing with DPMSTs via RL. VIN and GVIN perform slightly better than their results in IL. The reason is that RL cares more about the consequences of action history while IL aims at accurate label prediction. Therefore, even though VIN and GVIN cannot model the actual dynamics, they still achieve better reward.

Table 3: Performance on Stochastic PMST (RL)

	%Suc	Reward
UVIN	53.00	**0.419(±0.49)**
GPPN	0.00	0.000(±0.00)
GVIN	0.00	−0.001(±0.00)
VIN	0.00	−0.005(±0.05)
Rainbow	0.00	−0.103(±0.30)

The results demonstrate that VIN, GVIN, and GPPN do not have the ability to learn the correct dynamics to perform the planning computation. Although Rainbow might be a more advanced model-free approach, its performance is still far worse than UVIN which properly incorporates the planning computation.

Stochastic PMST (SPMST) SPMST is more difficult than DPMST.

In SPMST, the request for each item type is an integer ($z_i \in \mathbb{N}$), resulting in much larger number of states than that in the deterministic setting. As the transition probability is $p(s'|s,a) \in [0,1]$ in the stochastic environment, SPMST has more complex dynamics to capture. Therefore, SPMST requires the learning model to be compact to perform the task efficiently and effectively.

Experiments for SPMST shared the same setting as DPMST. Table 3 shows that UVIN is the only model that is capable of dealing with SPMSTs via RL. Fig. 6a and Fig. 6b illustrate RL's training process of each model in deterministic setting and stochastic setting, respectively.

In general, SPMST needs more episodes to converge than DPMST due to its more complex dynamics.

We validated UVIN in MineRL (Guss et al. 2019) to play Minecraft games without extra training. The sub-policies were pre-trained via Rainbow. Please refer to https://github.com/bit1029public/UVIN for the source codes and the videos of game playing with the policy computed.

Conclusion

This paper investigated the strength behind VIN and its extensions, and formally identified that they are confined to spatially invariant MDPs. To generalize VIN to spatially variant problems, we proposed Universal Value Iteration Networks which automatically explore a compact module to capture transition dynamics and incorporate it into the differentiable planning computation. Experiment results demonstrated that UVIN can achieve similar performance as VINs on spatially invariant problems, and significantly outperforms other models on more complex problems.

References

Bellman, R. 1957. A markovian decision process. *Journal of mathematics and mechanics* 679–684.

Gupta, S.; Davidson, J.; Levine, S.; Sukthankar, R.; and Malik, J. 2017. Cognitive mapping and planning for visual navigation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, 7272–7281.

Guss, W. H.; Codel, C.; Hofmann, K.; Houghton, B.; Kuno, N.; Milani, S.; Mohanty, S.; Liebana, D. P.; Salakhutdinov, R.; Topin, N.; et al. 2019. The minerl competition on sample efficient reinforcement learning using human priors. *arXiv preprint arXiv:1904.10079*.

Hessel, M.; Modayil, J.; van Hasselt, H.; Schaul, T.; Ostrovski, G.; Dabney, W.; Horgan, D.; Piot, B.; Azar, M. G.; and Silver, D. 2018. Rainbow: Combining improvements in deep reinforcement learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 3215–3222.

Khan, A.; Zhang, C.; Atanasov, N.; Karydis, K.; Kumar, V.; and Lee, D. D. 2018. Memory augmented control networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Lee, L.; Parisotto, E.; Chaplot, D. S.; and Salakhutdinov, R. 2018a. LSTM iteration networks: An exploration of differentiable path finding. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*.

Lee, L.; Parisotto, E.; Chaplot, D. S.; Xing, E.; and Salakhutdinov, R. 2018b. Gated path planning networks. *arXiv preprint arXiv:1806.06408*.

Lillicrap, T. P.; Hunt, J. J.; Pritzel, A.; Heess, N.; Erez, T.; Tassa, Y.; Silver, D.; and Wierstra, D. 2015. Continuous control with deep reinforcement learning. *CoRR* abs/1509.02971.

Mnih, V.; Koray, K.; David, S.; A, R. A.; Joel, V.; G, B. M.; Alex, G.; Martin, R.; K, F. A.; Georg, O.; et al. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Mnih, V.; Badia, A. P.; Mirza, M.; Graves, A.; Lillicrap, T.; Harley, T.; Silver, D.; and Kavukcuoglu, K. 2016. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, 1928–1937.

Moran, R.; Keramati, M.; Dayan, P.; and Dolan, R. J. 2019. Retrospective model-based inference guides model-free credit assignment. *Nature communications* 10(1):750.

Nardelli, N.; Synnaeve, G.; Lin, Z.; Kohli, P.; Torr, P. H. S.; and Usunier, N. 2019. Value propagation networks. In *International Conference on Learning Representations*.

Niu, S.; Chen, S.; Guo, H.; Targonski, C.; Smith, M. C.; and Kovacevic, J. 2018. Generalized value iteration networks: Life beyond lattices. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, 6246–6253.

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

Schleich, D.; Klamt, T.; and Behnke, S. 2019. Value iteration networks on multiple levels of abstraction. *CoRR* abs/1905.11068.

Schulman, J.; Levine, S.; Abbeel, P.; Jordan, M.; and Moritz, P. 2015. Trust region policy optimization. In *International conference on machine learning*, 1889–1897.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Sutton, R. S., and Barto, A. G. 1998. *Reinforcement learning - an introduction*. Adaptive computation and machine learning. MIT Press.

Tamar, A.; Levine, S.; Abbeel, P.; Wu, Y.; and Thomas, G. 2016. Value iteration networks. In *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*, 2146–2154.

Van Hasselt, H.; Guez, A.; and Silver, D. 2016. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI conference on artificial intelligence*.

Wang, Z.; Schaul, T.; Hessel, M.; Van Hasselt, H.; Lanctot, M.; and De Freitas, N. 2015. Dueling network architectures for deep reinforcement learning. *arXiv preprint arXiv:1511.06581*.