VI^{2}N: A Network for Planning Under Uncertainty based on Value of Information

Samantha N. Johnson
Allen Institute, Seattle, WA 98109
University of Chicago,
Neuroscience Institute, Chicago, IL 60637
`snjohnso@uchicago.edu`

Michael A. Buice
Allen Institute, Seattle, WA 98109
University of Washington,
Department of Applied Mathematics
Seattle, WA 98195
`michaelbu@alleninstitute.org`

Koosha Khalvati
Allen Institute, Seattle, WA 98109
`koosha.khalvati@alleninstitute.org`

Abstract

Despite of great success in the recent years, deep reinforcement learning architectures still face a tremendous challenge in many real-world scenarios due to perceptual ambiguity. Similarly, differentiable networks, known as value iteration networks, that performs well in novel situations by extracting the environment model from training setups, are mostly limited to fully observable tasks. In this paper, we propose a new architecture, the VI^{2}N (Value Iteration with Value of Information Network) that can learn to act in novel environments with high amount of uncertainty. Specifically, this architecture uses a heuristic that over-emphasizes on reducing the uncertainty before exploiting the reward. Our network outperforms the state of the art differentiable architecture for partially observable environments especially when long term planning is needed to resolve the uncertainty.

1 Introduction

Deep neural networks have provided a strong source of end-to-end solutions to Reinforcement Learning (RL) problems that map perception to action [2]. While one can approach this end-to-end learning in a classic supervised fashion (especially when provided an expert policy to imitate), incorporating fundamental mechanisms of reinforcement learning, such as the simulation of future events, experience replay, and the computation of belief over hidden variables, has been shown to improve the learning process significantly [13, 5]. For example, Value Iteration Networks (VINs) incorporate long term planning (the simulation of future events) by implementing the value iteration algorithm (i.e. a sequence of Bellman updates) via convolutional layers [1, 13, 10, 15, 4]. Trained either by reward or through imitation of an expert's actions, value iteration networks can learn to navigate in fully observable novel environments significantly better than fully connected and untied convolutional networks [13].

While VINs and deep reinforcement learning architectures in general have been very successful in many applications, they face a tremendous challenge in many real-world scenarios due to perceptual ambiguity. Perceptual ambiguity, often called *partial observability*, introduces uncertainty about the state of an agent's environment. This uncertainty must be accounted for in order to make correct decisions. It has been shown that even very simple networks with a probabilistic belief representation outperform networks with more sophisticated encoding and RL modules that perform well in challenging fully-observable environments [9, 6].

NeurIPS 2022 Deep Reinforcement Learning Workshop.

An important framework for decision making under uncertainty is Partially Observable Markov Decision Process (POMDP) [14]. Finding the optimal policy in a POMDP is NP-hard [12] and powerful sub-optimal approximations involves sampling and tree search techniques with no differentiable implementation, which additionally hinders our ability to use end-to-end neural network implementations. In fact, the state of the art network for decision making under partial observability is founded upon a very simple POMDP-solver, QMDP, which makes the assumption that all uncertainty disappears after the first step [5]. While this allows for a differential heuristic, this incorrect assumption causes the solver to fail in environments with high uncertainty.

Here we propose a new network architecture, the VI^{2}N (Value Iteration with Value of Information Network), that can learn to plan in unseen environments with a high amount of uncertainty. This network is based on a *Pairwise Heuristic*, which emphasizes information gathering and resolving uncertainty before maximizing the expected reward [7]. Importantly, since the Pairwise Heuristic is implemented by the Bellman equation, it can be implemented with a neural network, similar to the VIN. We demonstrate the power of our approach by testing it on navigation problems in the presence of uncertainty in different environments and compare performance to the current state-of-the-art network.

2 Overview

Markov Decision Process (MDP): A reinforcement learning problem is usually expressed as a Markov Decision Process (MDP). Formally, an MDP is (S, A, T, R, γ) where S is the set of states of the environment, A is the set of all available actions to the agent, the transition function $T : |S| \times |A| \times |S| \to [0, 1]$ defines $T(s, a, s') = P(s'|s, a)$, the probability of ending up in state s' by performing action a in state s, $R : |S| \times |A| \to \mathbb{R}$ is a bounded function determining the reward gained in state s, shown as $R(s)$, and $\gamma \in (0, 1]$ is the discount factor for the reward [14].

Starting from an initial state, s_0, the goal of an RL agent is to come up with a recipe for action selection, called a policy π, that maximizes the total discounted reward. Since the system is Markovian, the policy can be expressed as a mapping from states to actions, i.e. $\pi : |S| \times |A| \to [0, 1]$. The optimal policy π^* is $\pi^* = \arg\max_\pi \sum_{t=0}^{H} \gamma^t \mathbb{E}[R(s_t)|\pi, s_0]$ where the horizon H defines the length of this sequence. In deep reinforcement learning, this optimal policy/mapping is learned with a network with the state S (or a representation of it, $\phi(s)$) as the input and the action as the output of the network [2].

Value Iteration Network (VIN): Algorithms for finding the optimal policy of an MDP are generally divided into two categories: “model-free” and “model-based”. Model-based approaches use the structure of the environment, i.e transition and reward function to determine the optimal policy while model-free approaches try to learn the optimal policy directly from the accumulated obtained reward. Consequently, model-based approaches adapts faster upon changes in the environment as they only need to update their model, i.e. T and/or R. The value iteration algorithm is a model-based approach where the optimal value of each state, which is the expected gained reward in the future given the optimal policy, is computed through a series of Bellman updates, i.e. $V_t(s) = \max_a \left[R(s, a) + \gamma \sum_{s' \in S} T(s, a, s')V_{t-1}(s')\right]$ $(t \leq H)$ [1]. When the transition function is spatially invariant, a neural network can learn T and R by implementing the Bellman equation with convolutional layers (Roughly speaking, $T(s, a, s')$ is the kernel of these layers [13]). Such a network with integration of value iteration as an explicit planning module, generally known as a Value Iteration Network (VINs), significantly outperforms networks with similar computational power (e.g. layers) in learning to plan in unseen environments [13, 10, 15, 4].

Partially Observable Markov Decision Process (POMDP): Existence of uncertainty in the real world, especially in the form of perceptual ambiguity has made MDPs impractical in many situations [14]. Similarly, state of the art networks reaching extraordinary performance in very complicated yet fully observable tasks often fail to handle seemingly small amounts of ambiguity in the environment [9]. Partially Observable MDPs (POMDPs) represent the closest approach to MDPs that deals with uncertainty by adding an observation set and observation function to its framework. Formally, a POMDP is a tuple $(S, A, Z, T, O, R, \gamma)$ where S, A, T, R and γ are defined very similar to their definition in MDP. Z is the set of observations and $O : |S| \times |Z| \to [0, 1]$ is the observation function determining probability of observation z in state s, i.e. $O(s, z) = P(z|s)$. In a POMDP, the agent is not fully aware of its current state. Therefore, it has to maintain a probability distribution over states, often called its *belief* $b(s)$. Starting from a prior probability distribution over states of the environment, called the initial belief (b_0), the goal is to maximize the expected discounted reward [11]. For a POMDP, the optimal decision policy π^* can be expressed as a mapping from belief states (probability distributions over states) to distribution of actions that maximizes the total expected reward [12]:

$$\pi^* = \arg\max_{\pi} \sum_{t=0}^{H} \gamma^t E[R(s_t, a_t, z_{t+1})|b_t, \pi]. \tag{1}$$

The uncertainty about the state makes the agent navigate in the belief state space instead of the state space. At time step t, the belief state b_t is updated based on the previous belief state b_{t-1} after action a_{t-1} and observation z_t as follows:

$$b_t(s) \propto P(z_t|s, a_{t-1}) \sum_{s' \in S} P(s|s', a_{t-1}) b_{t-1}(s') \tag{2}$$

The uncertainty about the state also makes the problem of finding the optimal policy exponentially more complex than the MDP. While the optimal policy of an MDP can be found in polynomial time, finding the optimal policy of a POMDP is NP-hard [14]. As a result, the optimal policy can only be approximated by methods such as heuristics, sampling, and search trees [11].

Deep networks for solving POMDPs: The belief update can be easily implemented in a network especially if the observation function is spatially invariant [5]. Given the policy (belief to action) module, a POMDP solver architecture is a recurrent network. However, as mentioned before, designing a good policy module is very challenging due to the differentiability requirement. As a result, current networks for solving POMDPs have a very simple policy module, e.g. a model free RL module [9] and QMDP [5].

3 Model

Our architecture is founded upon a “Pairwise Heuristic”. Originated from Bayesian active learning in which the heuristic is used to find the correct hypothesis with a set of noisy tests [3], Pairwise Heuristic has been also used in robot localization [8] and a general-purpose POMDP-solver[7]. Here we present the POMDP-solver version, slightly modified for our framework.

In an MDP, the only goal is maximizing the reward. But when we deal with uncertainty, information gathering and ambiguity resolution must be considered as well. A POMDP solver implicitly does information gathering and reward maximization simultaneously. In most problems, a good policy not only obtains rewards but also tries to decrease uncertainty. This does not mean decreasing uncertainty is a separate task. In fact, in most problems this decrease helps the agent to get higher reward in the future. As a result, we can say that uncertainty decreases in general while performing a good policy. In one of the steps of the policy, uncertainty gets low enough that we can say we approximately know the current state of the agent. This is similar to *robot localization*. In robotics, “localization" refers to finding the robot’s current state [14], where the state is the robot’s position. Similarly, in our case localization means finding the actual current state. But this time, the state is more general than the agent’s position. At this point we can say that the agent is localized and we can treat the problem as an MDP after this point. So our whole problem can be seen as maximizing the total reward in the localization phase plus the reward of the resultant MDP after localization.

We change the POMDP problem to something that seems easier to solve, but even this problem is very difficult if the agent is uncertain about being in many states (i.e. localization itself is NP-hard). But if the uncertainty is limited to only two states, the problem becomes much easier. We can solve the problem for every pair of states and then use these solutions to solve the main problem.

3.1 The Pairwise Heuristic

Our heuristic needs an optimal sequence of actions for each pair of states (s, s'). We call the total discounted reward of this optimal sequence the value function of the pair, $V(s, s')$. To explain how to find these sequences, we assume that at first we only want to do the localization task for each pair. How can we find an optimal sequence for localization for each pair? Some pairs do not need any action for localization. These pairs are “distinguishable". For example, for the pair (s, s'), if all possible observations in s have zero probability in s' and vice versa, there is no need for localization.

For other pairs of states, i.e. indistinguishable ones, we can carry out the localization by going to distinguishable pairs. After the localization, the current state is determined and to fulfill the reward maximization goal we just need to solve the problem in the MDP framework. One could argue that as the states are partially observable and the actions are not deterministic, the uncertainty about the state may arise again. In this situation we do the localization task again. Two states are distinguishable if there is a high probability that different observations are recorded in the two states. Formally, s and s' are distinguishable if and only if:

$$o^* = argmax_o p(o|s)$$
$$o'^* = argmax_o p(o|s')$$
$$\sum_{s',s'} [p(o^*|s)(1 - p(o^*|s')) + p(o'^*|s')(1 - p(o'^*|s)] \geq 2\lambda \quad (3)$$

λ is a constant that is specified by a domain expert. If it is 1, the observations should be completely different. But, as in the localization problem where a robot is usually considered localized if the probability of a state is more than a threshold like 0.95, we can set this threshold to a value less than 1 in noisy environments. As shown in the formula the observations that are considered are the most probable observations of the posterior states. If there is more than one observation with maximum probability, one would be chosen arbitrarily.

The value function of a distinguishable pair is set to:

$$V(s, s') = 0.5[V(s) + V(s')] \quad (4)$$

$V(s)$ and $V(s')$ are the value functions of s and s' in the underlying MDP.

To find the value function and optimal action for indistinguishable pairs, we use a value iteration algorithm in an MDP where the states are pairs of states of our original problem. The transition function is determined as follows:

$$p((s'', s''')|(s, s'), a) = p(s''|s, a)p(s'''|s', a) \quad (5)$$

Also the reward, $R(s, s')$ is equal to:

$$R(s, s') = 0.5[R(s) + R(s')] \quad (6)$$

where $R(s)$ and $R(s')$ belong to the original problem. We run the value iteration only for indistinguishable pairs. The initial value function for these pairs is set to the minimum reward in the main problem. Actions are the same as actions in the original problem. In addition, the discount factor of the new MDP is the same as the discount factor of the original problem. This algorithm is shown as Algorithm 1.

As shown above, we use the value of 0.5 in all of the equations for value functions which gives the unweighted average. This means we assume equal probability for the two states in computing their optimal action and value function. The states may not have equal probability in the original problem, but the pairwise value function is used as a heuristic and does not need to be exact. By using this simplification, obtaining value functions and optimal actions does not depend on the initial belief state and would be an offline calculation. So for each domain, we only need to run this algorithm once.

3.2 The greedy strategy

To solve the POMDP, we only need a one step greedy strategy that uses the value functions of the pairs. In each step, the selected action should maximize the expected total value function of the pairs. However, the expected instant reward of the actions should be considered as well. As a result the selected action is:

$$a_k^* = \arg\max_a \sum_{(s,s')} Q((s^*, s'^*), a)b_k(s)b_k(s')] \quad (7)$$

If in one of the steps of the planning the probabilities of all states, except the most likely one, become less than the threshold, the selected action would be the optimal action of the underlying MDP for that most likely state in that step.

4 VI^{2}N Architecture

Our architecture implements a neural network with a built-in value iteration algorithm equipped to handle a pairwise set of states. The algorithm begins with a localization module to help with long-term information gathering. After localization, the network alternates between a Bayesian filter for belief update and a planner module, where the policy is updated based on maximum information gain.

Localization Module The localization module consists of two parts, the Value Iteration (VI) Module and the VI^2 Module for pairwise states. Following the algorithm from the Pairwise Heuristic Algorithm 1 , we implement the Bellman equation as follows using a convolutional layer and a max pooling layer:

$$V_{k+1}(s) = max_a R(s) + \gamma \Sigma T(s,a) V_k(s) \tag{8}$$

From this point, the objective becomes converting elements of the environment to a pair-space representation to allow for the VI^2 implementation. Specifically, we must convert $T(s,a)$ and $R(s)$ into $T((s,s'),a)$ and $R(s,s')$ for all $s, s' \in S \times S$. We can convert $R(s)$ using an averaging layer across all s. Transition T is used as the kernel in the VI Module, and must be transformed into a transition kernel for the state space, which involves increasing the size of the kernel from $(3,3)$ to $(2(\sqrt{s}+1)+1, 2(\sqrt{s}+1)+1)$ to allow for row and column transitions between pairs. This kernel is constructed using the learned transition probabilities from the VI Module and has a number of channels equal to the number of actions available in the environment. Finally, we must determine which set of pairs (s,s') are distinguishable by observations O. We implement this through a convolutional layer of possible observations in each state and threshold at a confidence level.

The second step of the localization module is the VI^2 Module, where the objective is to obtain the values of the pair states. Following the Pairwise Heuristic Algorithm 1(Lines 14-23), values for pairs are calculated differently based on whether or not the pair is distinguishable. Distinguishable pairs are calculated through an averaging layer of $V(s)$ and $V(S')$, while indistinguishable pairs require the Bellman equation in a pair-space through a convolutional layer and a max pooling layer as follows:

$$V_{k+1}(s,s') = max_a R(s,s') + \gamma \Sigma T((s,s'),a) V_k(s,s') \tag{9}$$

Once all the V and Q are calculated, the Localization Module is complete and long term planning through distinguishable information gain has been accomplished.

Algorithm 1: Finding the value functions and optimal actions for the pairs

```
Data: $(S, A, O, T, Z, R, \gamma)$
Result: $V(s,s')$ and $u(s,s')$ for all pairs
1  Calculate value functions, $V(S)$ of $MDP(S, A, T, R, \gamma)$
2  foreach pair $(s,s')$ do
3      $V(s,s') = R_{min}$
4  end
5  foreach pair $(s,s')$ do
6      $R((s,s')) = 0.5[R(s) + R(s')]$
7  end
8  foreach pair $(s,s')$ do
9      $p((s'',s''')|(s,s'),a) = p(s''|s,a)p(s'''|s',a)$
10 end
11 foreach distinguishable pair $(s,s')$ do
12     $V(s,s') = 0.5[R(s,a) + R(s',a) + \gamma(V(s) + V(s'))]$
13 end
14 repeat
15     foreach indistinguishable pair $(s,s')$ do
16         $V_k(s,s') = max_a[R((s,s')) + \gamma \sum_{s'',s'''}) V(s'',s''') p((s'',s''')|(s,s'),a)$
17     end
18 until convergence
```

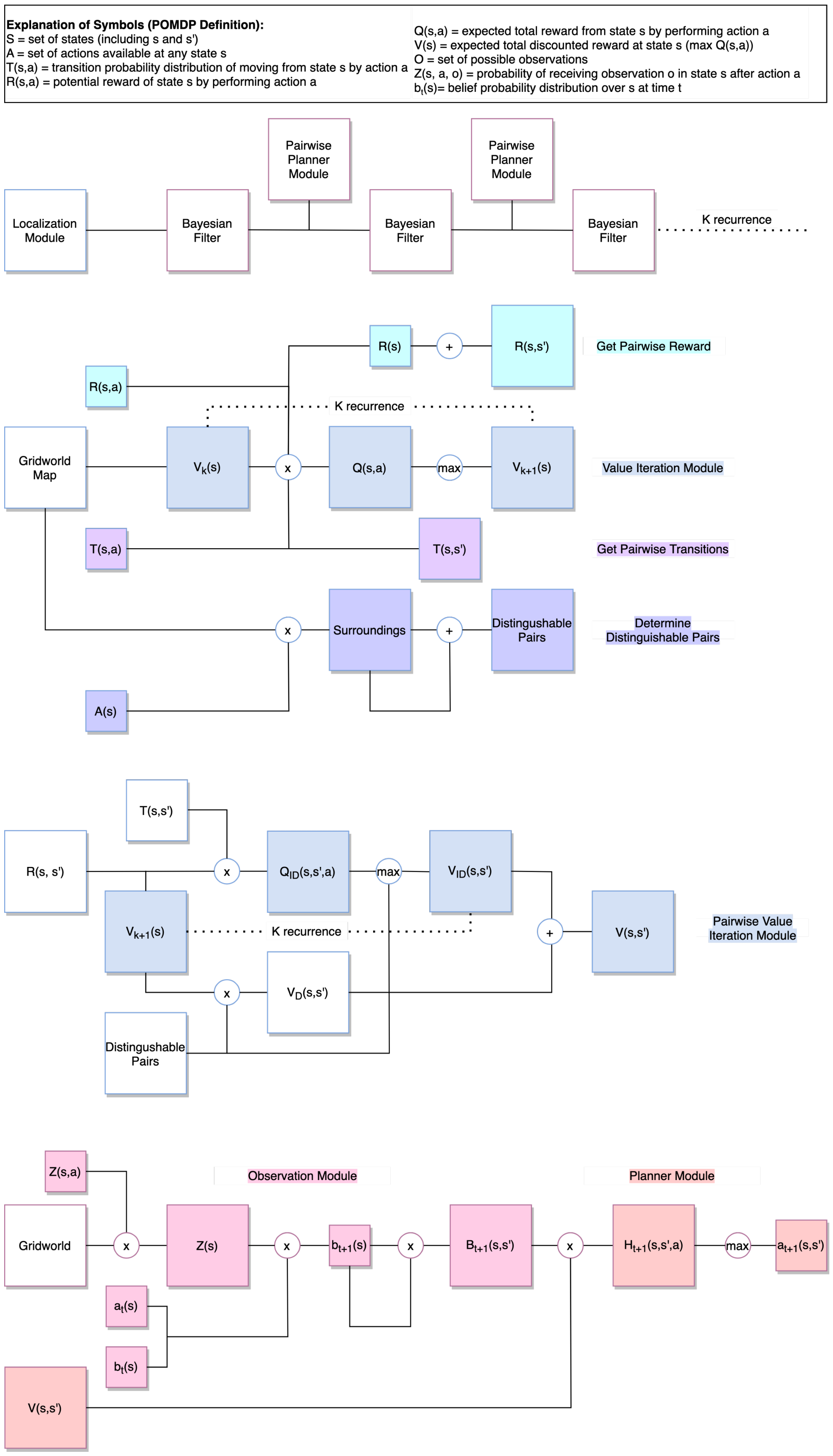
Explanation of Symbols (POMDP Definition):
S = set of states (including s and s')
A = set of actions available at any state s
T(s,a) = transition probability distribution of moving from state s by action a
R(s,a) = potential reward of state s by performing action a
Q(s,a) = expected total reward from state s by performing action a
V(s) = expected total discounted reward at state s (max Q(s,a))
O = set of possible observations
Z(s, a, o) = probability of receiving observation o in state s after action a
b_t(s)= belief probability distribution over s at time t

Figure 1: VI^{2}N Overview. a) Network begins with a single pass through a Localization Module then alternates between a Bayesian Filter and Pairwise Planner until horizon is reached. b) The Localization Module begins with a VI Module and a reformatting of R(s) and T(s,a) into pair space. c) The Localization Module ends with a VI^2 module to calculate $V(s, s')$ for all pairs in the environment. d) The Bayesian Belief Update and Planner Modules alternate in an online planner to update the belief at each step allowing for policy determination.

Algorithm 2: Choosing the optimal action in step k

```
Data: $(S, A, O, T, Z, b_k, R, \gamma)$
Result: Optimal action, $a_k^*$
1 if $|\{p(s) > 0\}| = 1$ then
2     $a_k^*$ = optimal action of $S'$ in the MDP
3 end
4 else
5     $a_k^* = \arg\max_a \sum_{(s,s')} Q((s^*, s'^*), a) b_k(s) b_k(s')]$
6 end
```

Bayesian Filter The Bayesian Filter is responsible for the belief update in every step of the solver (and is thus considered an online planner). The Bayesian Filter that maps current belief, action, and observation to a subsequent belief was first implemented in the QMDP-net [5]. In this implementation, the fundamental calculations remain the same while accounting for a shift to a pair-space belief. As in previous work, the Bayesian Filter implements the belief update through two equations, first accounting for the agent's action, then computing belief based on subsequent observation.

$$b'_t(s) = \Sigma_{s'} T(s, a_t, s') b_t(s) \tag{10}$$

$$b_{t+1}(s) = \eta Z(s, o_t) b'_t(s) \tag{11}$$

However, the belief is converted to a pairwise belief by an assumption of independence via $B(s) * B(s') = B(s, s')$.

Pairwise Planner Module The purpose of the planner module is to select the action at each step. This module is primarily structured after the Pairwise Heuristic planner algorithm shown in Algorithm 2. After obtaining the new belief based on previous actions and observations, the Planner Module takes in the previously calculated V(s,s') from the localization module and the new belief B(s,s') to output the optimal action at each step.

$$a_t = max_a \Sigma_{s,s'} R(s, s') + \gamma V(s, s') B(s', s) \tag{12}$$

This module is implemented through two layers: a multiplication layer to combine the convolutional product of Q(s,s') with B(s,s'), followed by a max pooling layer to select the optimal action.

5 Results

Preliminary testing was completed to compare performance of QMDP-net against the VI^{2}N on various navigation environments. We generated two varieties of environments, termed "random" and "symmetric". In the random environment, obstacles are randomly placed within the environment at both 5% and 10% density/sparsity levels (Figure 2, left), sometimes with the constraint of minimal continuity in each axis, which leads to produce squares (here with the side size of 1 and 4). These environment were generated to model environments where obstacles are randomly placed as independent clusters, such as desert landscapes. In the symmetric environment, four copies of a smaller random environment are placed in each corner of a larger grid-world (Figure 2, right). The density of each of the four "rooms" (small environment block) was $5\%, 10\%$, or 15%. This environment requires more long term planning and localization, as it has more indistinguishable states that could lead to incorrect assumptions about belief in simpler updates. The symmetric grid-world models environments where obstacles are closer together in a maze-like orientation with lots of repetition, such as trails through the forest or identical floors in a building.

For each type of environment, labels of "correct" policies were generated using two types of expert solvers: the QMDP and Pairwise solvers. This allowed us to explore reinforcement learning, which necessitates the training expert to be the same as the planner embedded in the networks. For the QMDP-net, QMDP was used as the expert solver, and the VI^{2}N, which uses the Pairwise Heuristic planner, uses the Pairwise Heuristic as the expert. Models were trained on each environment type separately, using policies of expert solvers as labels. It is valuable to note that not every expert policy label is a success, as some environments are too difficult for either the QMDP or Pairwise Heuristic to solve. Similar to the original QMDP-Net, networks were trained on just successful trials [5] (less than

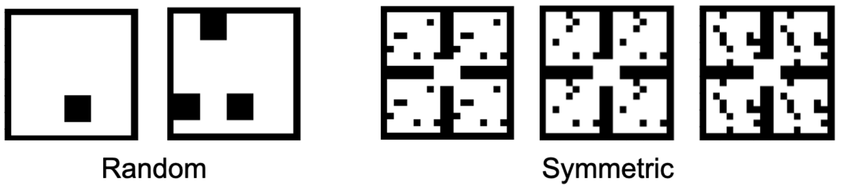

Figure 2: Example testing environments. Two types, random and symmetric, each of which with various density rates.

Table 1: Success Rate of network solvers over various environments

Environment (Sparcity)	VI^{2}N	QMDP-Net
Random (5%)	91%	88%
Random (10%)	91%	89%
Symmetric (5%)	76%	61%
Symmetric (10%)	74%	51%
Symmetric (15%)	65%	41%

50 steps needed to reach the goal). The training data set consisted of 12000 environments, each of which with two or three different start state. For each type, density rates were distributed uniformly, e.g 6000 environments with 5% density for the "random".

Only one action were picked from the trajectory of (initial start state, environment) pairs, making the training data-set around 30000 training instances. Since only actions from successful trajectories were picked, each network were trained on less than 30000 instances. Training performance were evaluated through $95\% - 5\%$ train-validation process. Test success rate was then calculated by running the generated model on 1,000 novel environments of the same type, each of which with 20 different start states, for each density. The initial belief state for the tests were always uniform among all possible states (starting from an obstacle or goal was not possible).

The results of the network comparison are shown in Table 5. In the random environment, the VI^{2}N outperformed the QMDP-Net test success rate, although QMDP-Net still performed well. The random environment is a less complex map that does not require as much localization or long term planning, and thus allows for high performance by a variety of solvers. However, an important difference between the networks is the symmetric environment, which highlights the need for long term planning and localization. In this environment, the VI^{2}N drastically outperforms the QMDP-Net, which demonstrates its ability to use long-term planning to generate effective policies. We can also observe that the VI^{2}N is more robust to changes in sparsity, whereas the QMDP-Net performance drops at a higher rate as environments increase density.

We hypothesize that the reason for the above trends in test success rate is due to the relative strengths and weaknesses of the model. A widely accepted strength of the QMDP solver is its simplicity. Thus, the QMDP-Net can capitalize on that simplicity to demonstrate high performance in environments with decreased uncertainty, such as the random environment. However, the Pairwise Heuristic focuses on long-term planning to resolve uncertainty, making the VI^{2}N a more robust model for environments of higher uncertainty, such as the symmetric environment. This would explain we see VI^{2}N only slightly outperform the QMDP-Net in the less complex random environment and VI^{2}N outperform QMDP-Net in the more complex symmetric environment. In future work, we could test this hypothesis by exploring network performance in higher uncertainty environments than tested here, expecting VI^{2}N to excel while QMDP-Net performance suffers as complexity increases.

6 Discussion

We have introduced the VI^{2}N as a deep learning architecture for decision making under uncertainty, modeled after the fully differentiable Pairwise Heuristic. The VI^{2}N architecture demonstrates the ability for long-term planning for resolving the uncertainty which exceeds the capacity of previously proposed network architectures seen in the VIN and the QMDP-Net. This study shows the VI^{2}N outperforms the state-of-the-art model on the tested environments, warranting subsequent exploration. This claim could be further investigated by expanding this long-term planning to other complex applications, including more complex navigation, foraging and object manipulation tasks.

7 Acknowledgments

We wish to thank the Allen Institute founder, Paul G. Allen, for their vision, encouragement and support.

References

[1] Richard Bellman. A Markovian Decision Process. *Journal of Mathematics and Mechanics*, 6(5):679–684, 1957. Publisher: Indiana University Mathematics Department.

[2] Vincent François-Lavet, Peter Henderson, Riashat Islam, Marc G. Bellemare, and Joelle Pineau. An Introduction to Deep Reinforcement Learning. *Foundations and Trends® in Machine Learning*, 11(3-4):219–354, December 2018. Publisher: Now Publishers, Inc.

[3] Daniel Golovin, Andreas Krause, and Debajyoti Ray. Near-optimal Bayesian active learning with noisy observations. In *Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1*, NIPS'10, pages 766–774, Red Hook, NY, USA, December 2010. Curran Associates Inc.

[4] Shu Ishida and João F. Henriques. Towards real-world navigation with deep differentiable planners. In *2022 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2022.

[5] Peter Karkus, David Hsu, and Wee Sun Lee. QMDP-Net: Deep Learning for Planning under Partial Observability. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[6] Peter Karkus, David Hsu, and Wee Sun Lee. QMDP-Net: Deep Learning for Planning under Partial Observability. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[7] Koosha Khalvati and Alan Mackworth. A Fast Pairwise Heuristic for Planning under Uncertainty. *Proceedings of the AAAI Conference on Artificial Intelligence*, 27(1):503–509, June 2013.

[8] Koosha Khalvati and Alan K. Mackworth. Active robot localization with macro actions. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 187–193, October 2012. ISSN: 2153-0866.

[9] Tianwei Ni, Benjamin Eysenbach, and Ruslan Salakhutdinov. Recurrent Model-Free RL Can Be a Strong Baseline for Many POMDPs, June 2022. Number: arXiv:2110.05038 arXiv:2110.05038 [cs].

[10] Sufeng Niu, Siheng Chen, Hanyu Guo, Colin Targonski, Melissa Smith, and Jelena Kovačević. Generalized Value Iteration Networks:Life Beyond Lattices. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1), April 2018. Number: 1.

[11] S. Ross, J. Pineau, S. Paquet, and B. Chaib-draa. Online Planning Algorithms for POMDPs. *Journal of Artificial Intelligence Research*, 32:663–704, July 2008.

[12] Edward J. Sondik. The Optimal Control of Partially Observable Markov Processes over the Infinite Horizon: Discounted Costs. *Operations Research*, 26(2):282–304, 1978. Publisher: INFORMS.

[13] Aviv Tamar, YI WU, Garrett Thomas, Sergey Levine, and Pieter Abbeel. Value Iteration Networks. In *Advances in Neural Information Processing Systems*, volume 29. Curran Associates, Inc., 2016.

[14] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. Probabilistic Robotics (Intelligent Robotics and Autonomous Agents), 2005.

[15] Li Zhang, Xin Li, Sen Chen, Hongyu Zang, Jie Huang, and Mingzhong Wang. Universal Value Iteration Networks: When Spatially-Invariant Is Not Universal. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(04):6778–6785, April 2020. Number: 04.