Research on path planning algorithm based on deep reinforcement learning

VIN Introduction

柴士佳_Shijia CHAI

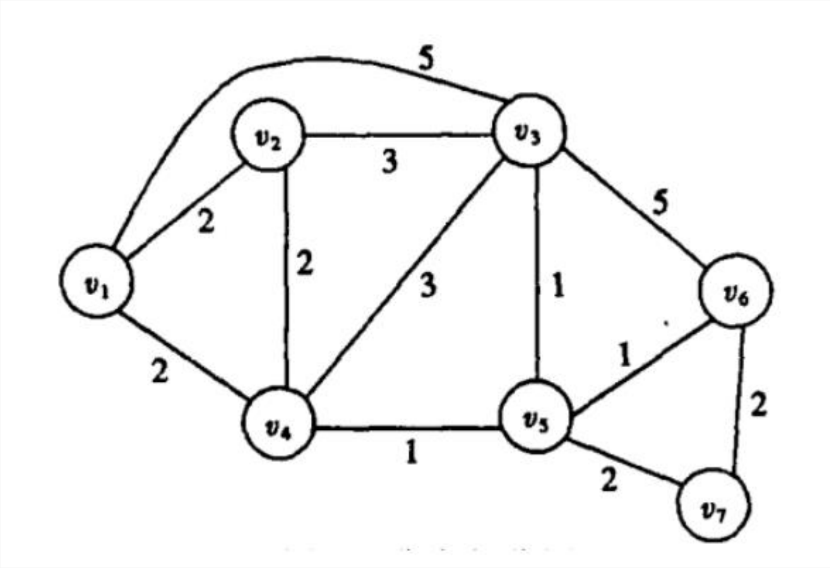
BACKGROUND

BACKGROUND

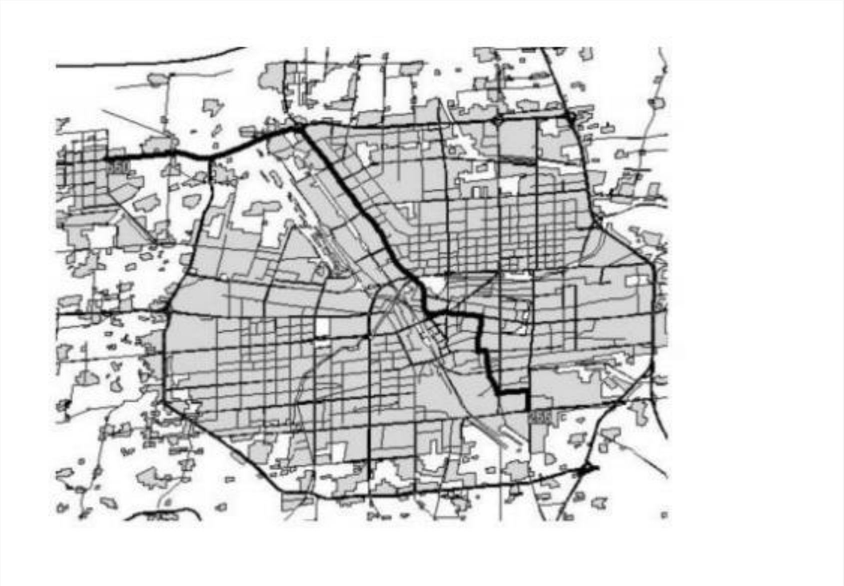
Path planning is a primary task of autonomous control for agents

And we have some traditional methods.

- Dijkstra algorithm



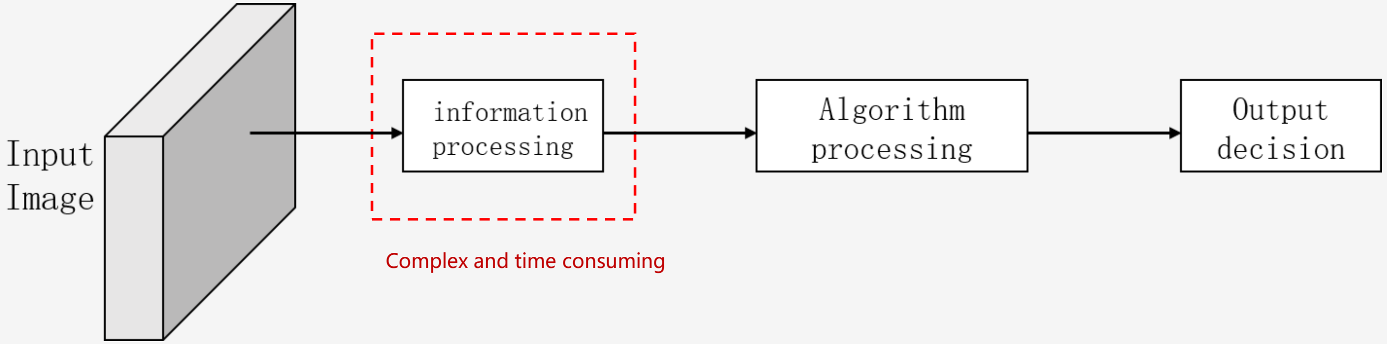
A* algorithm



BACKGROUND

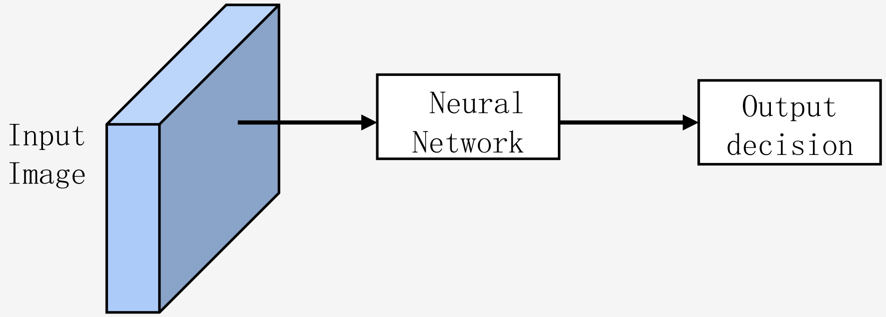
But those methods have some problems.

- Not an end-to-end model
- No learning ability, no generalization ability
- No intelligent understanding of the problem



BACKGROUND

Therefore, the intelligent algorithm based on deep reinforcement learning has been performed.



- An end-to-end training and prediction model
- The model has the ability of learning and generalization
- Intelligent understanding of the problem

A MODEL FOR POLICIES THAT PLAN

A PLANNING-BASED POLICY MODEL

Planning in MDP(Markov Decision Process)

- States $s \in S$, actions $a \in A$
- Reward $R(s, a)$
- Transitions $P(s'|s, a)$
- Policy $\pi(a|s)$
- Value function $V^{\pi}(s) = E^{\pi}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)|s_0 = s]$
- Value iteration(VI)

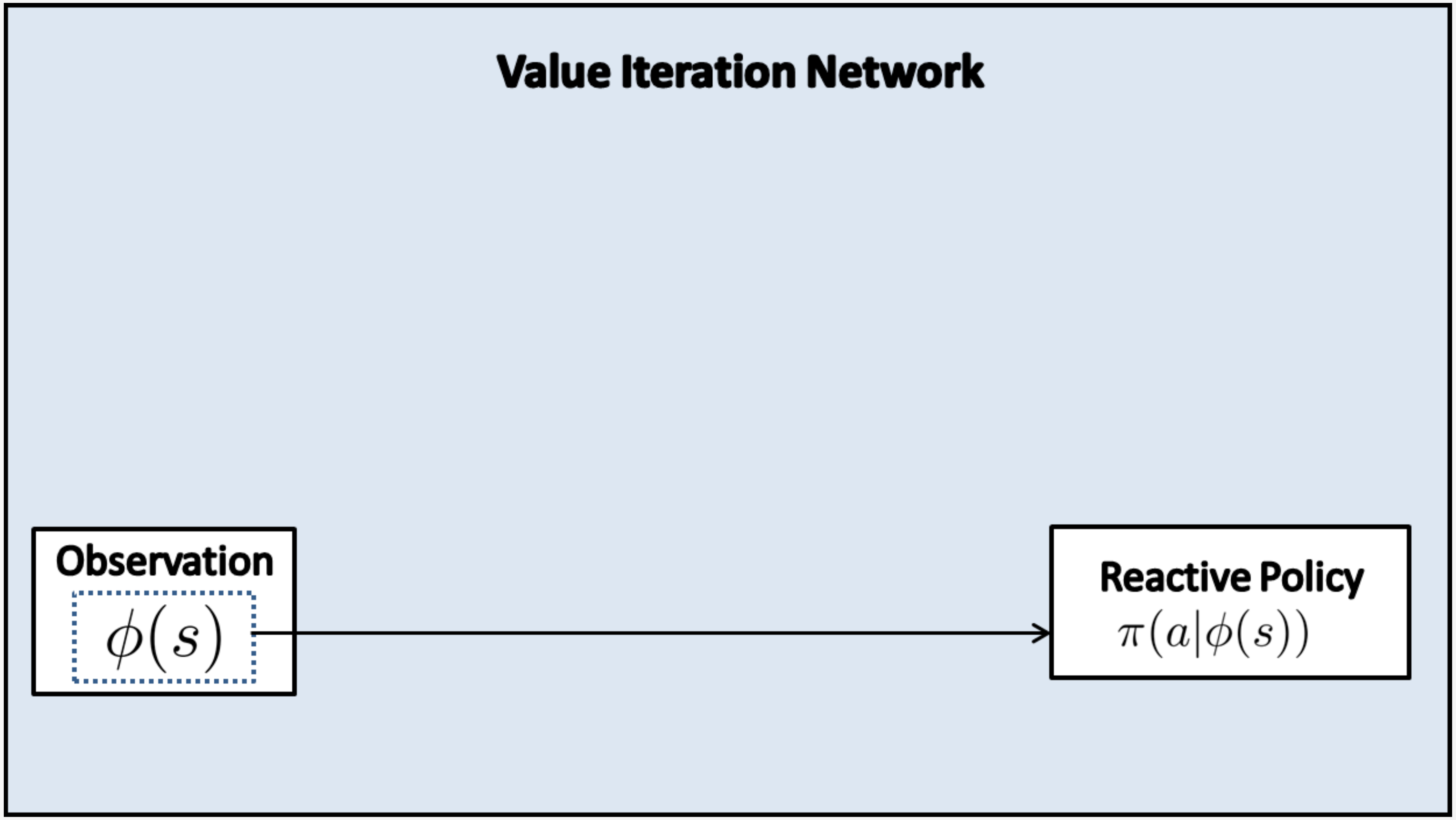
$$V_{n+1}(s) = \max_a Q_n(s, a) \qquad \forall s,$$

$$Q_n(s, a) = R(s, a) + \gamma \sum_{s'} P(s'|s, a)\, V_n(s')$$

- Converge to $V^* = \max_{\pi} V^{\pi}$
- Optimal policy $\pi^*(a|s) = \arg\max_a Q^*(s, a)$

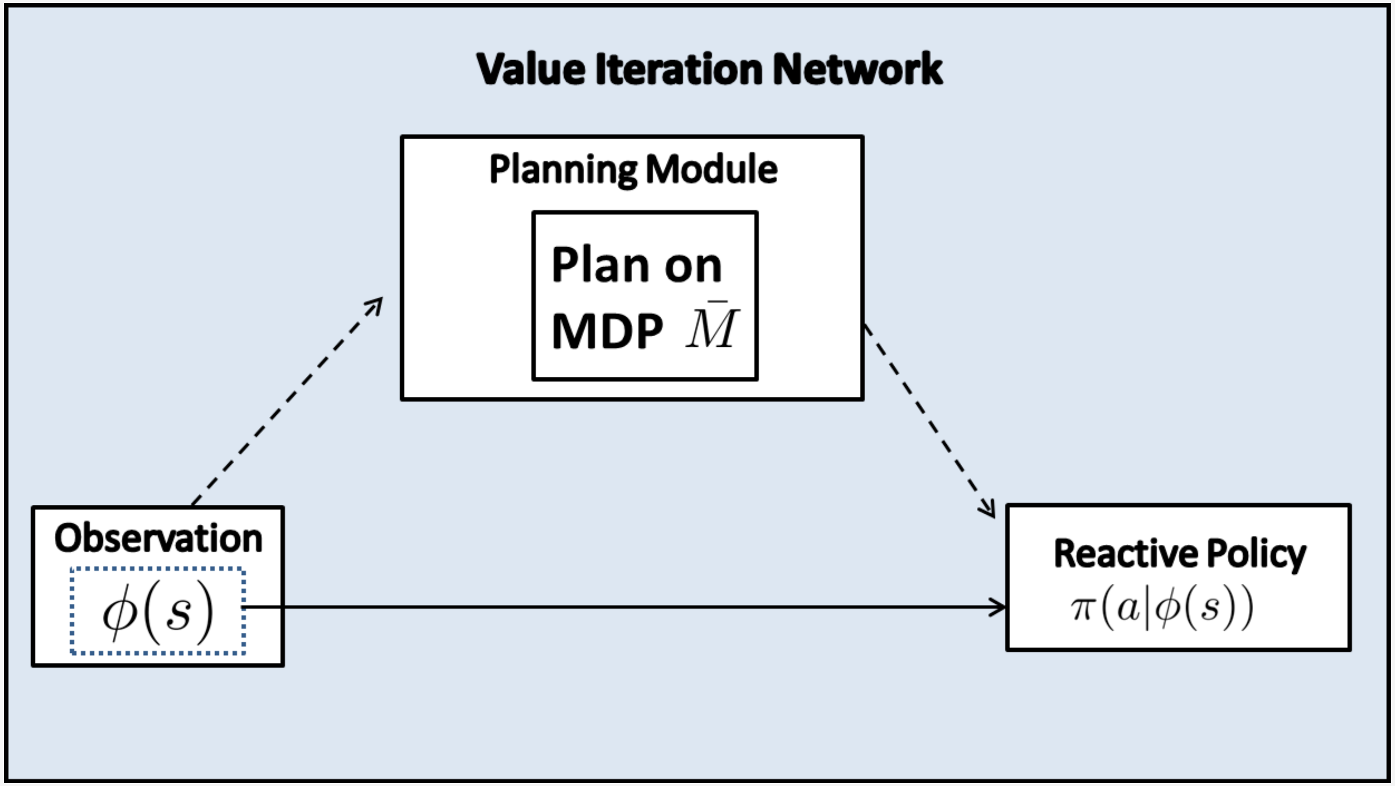
A PLANNING-BASED POLICY MODEL

- Start from a reactive policy;



A PLANNING-BASED POLICY MODEL

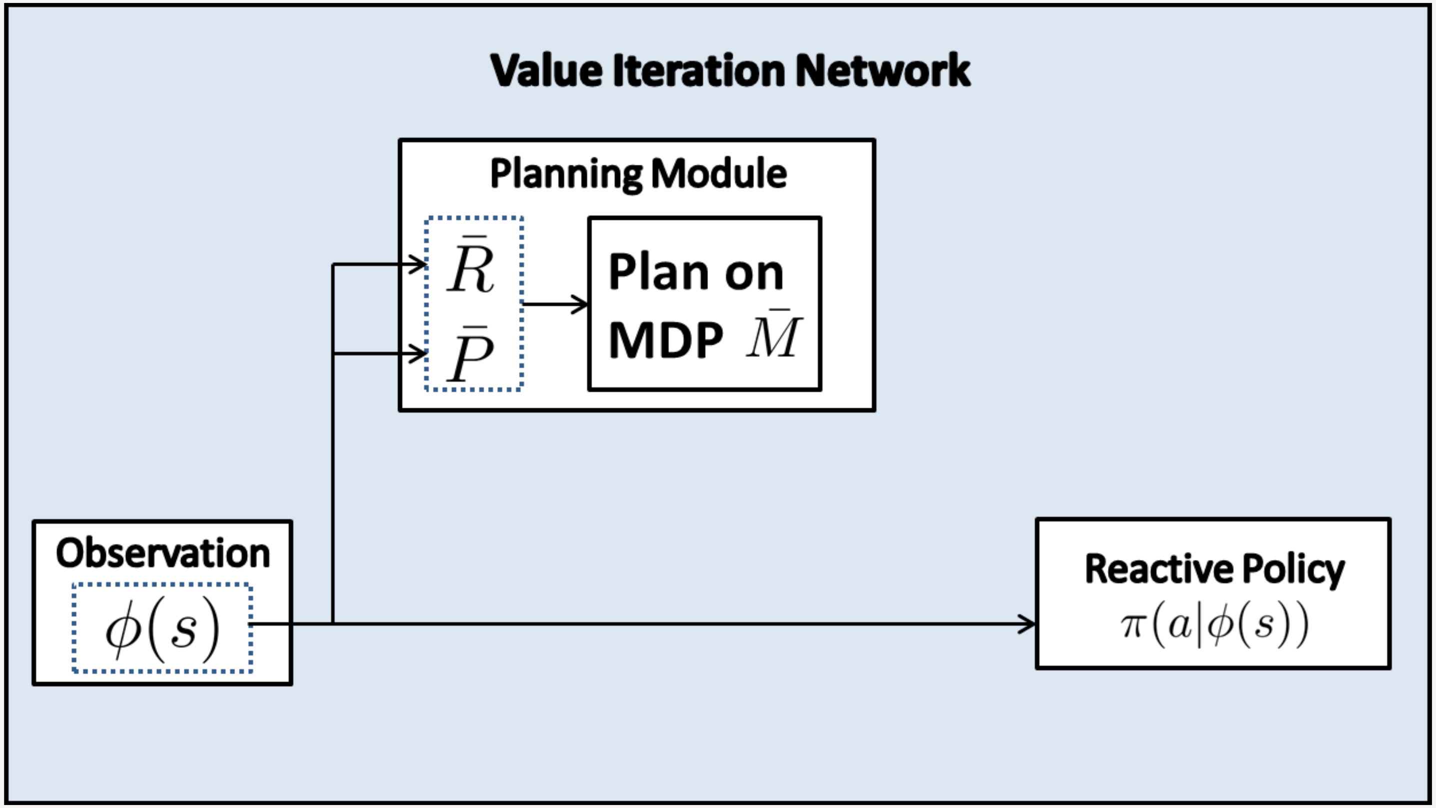
- Add an explicit planning computation
- Map the observation to planning MDP $\bar{M}$



- Key point: observation will be mapped to a useful but **unknown** new map and we plan on the **new** map.

A PLANNING-BASED POLICY MODEL

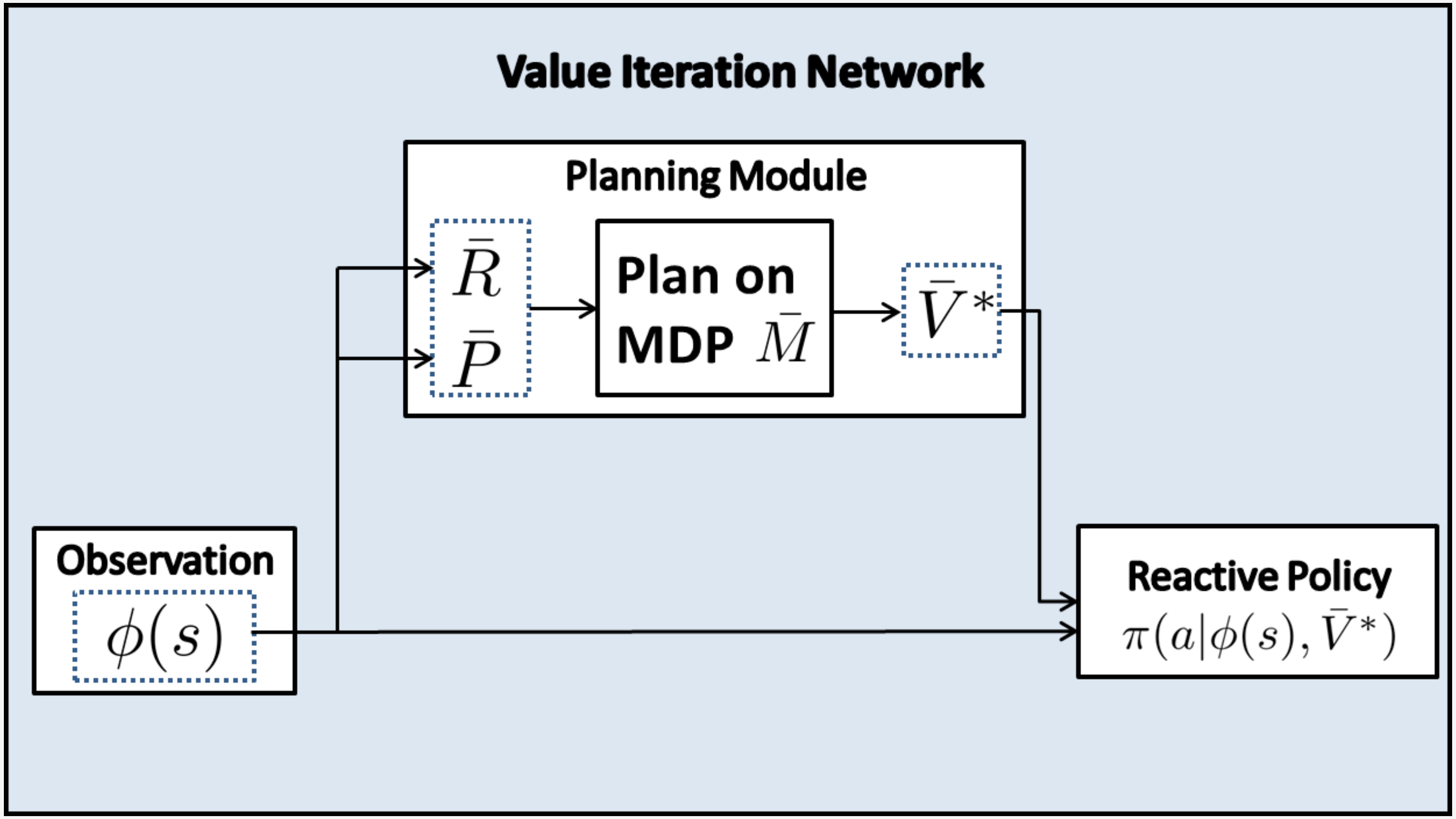
- Neural Networks map observation to reward and transitions
- Later- learn these



- Both of them($\bar{R}$ *and* $\bar{P}$) are **trainable**.

A PLANNING-BASED POLICY MODEL

- Value function = sufficient information about plan

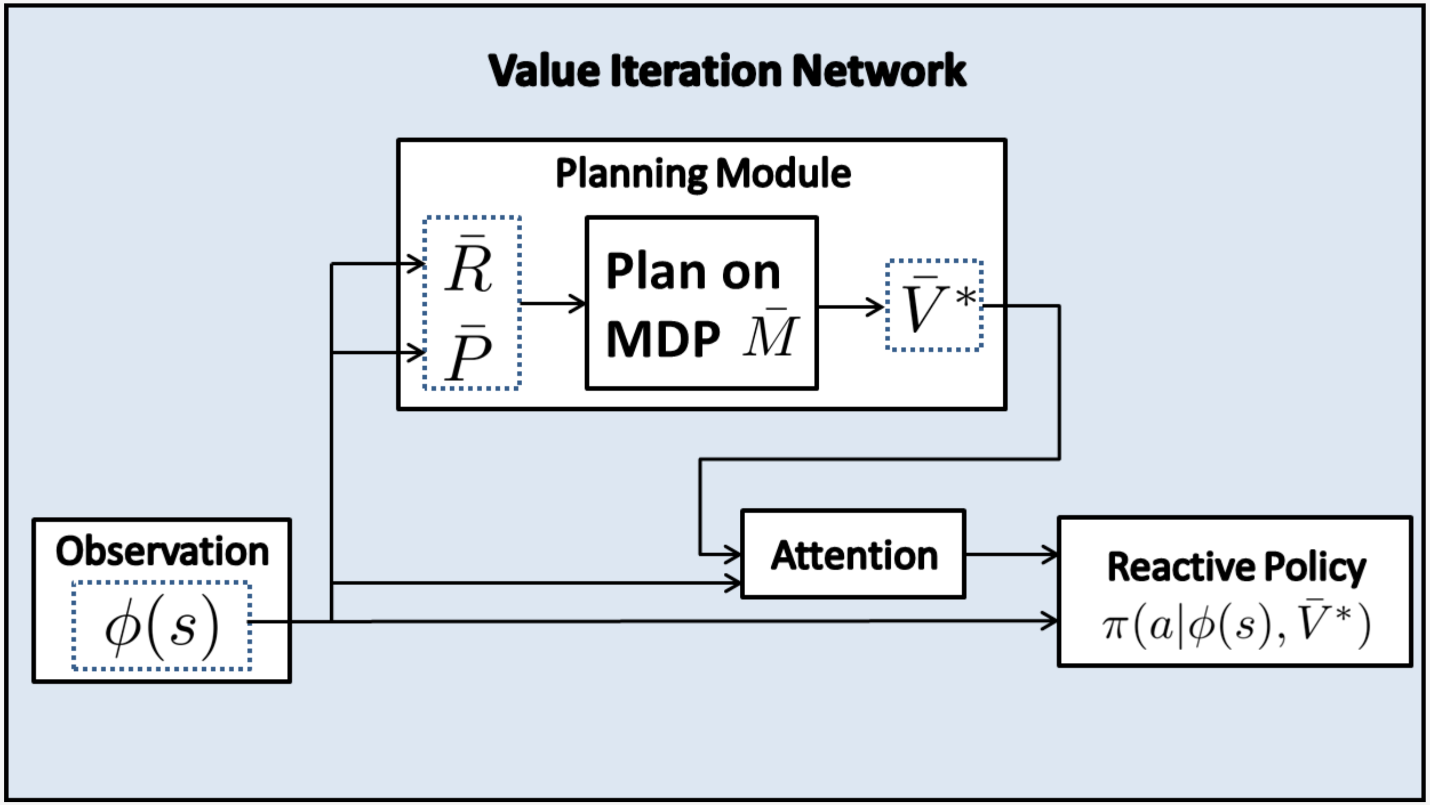


A PLANNING-BASED POLICY MODEL

- Fact is: action prediction can require only subset of $\overline{V^*}$

$$\pi^*(a|s) = \arg\max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a)V^*(s')$$

- Use attention models in the networks



VALUE ITERATION NETWORKS

VALUE ITERATION = CONVNET

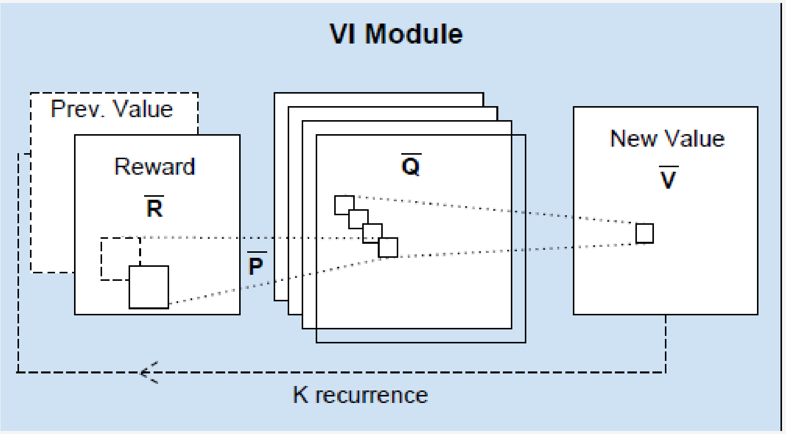
- Value Iteration:

K iterations of:

$$Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a)\, V_n(s')$$

$$V_{n+1}(s) = \max_a Q_n(s,a) \qquad \forall s,$$

- Convnet :

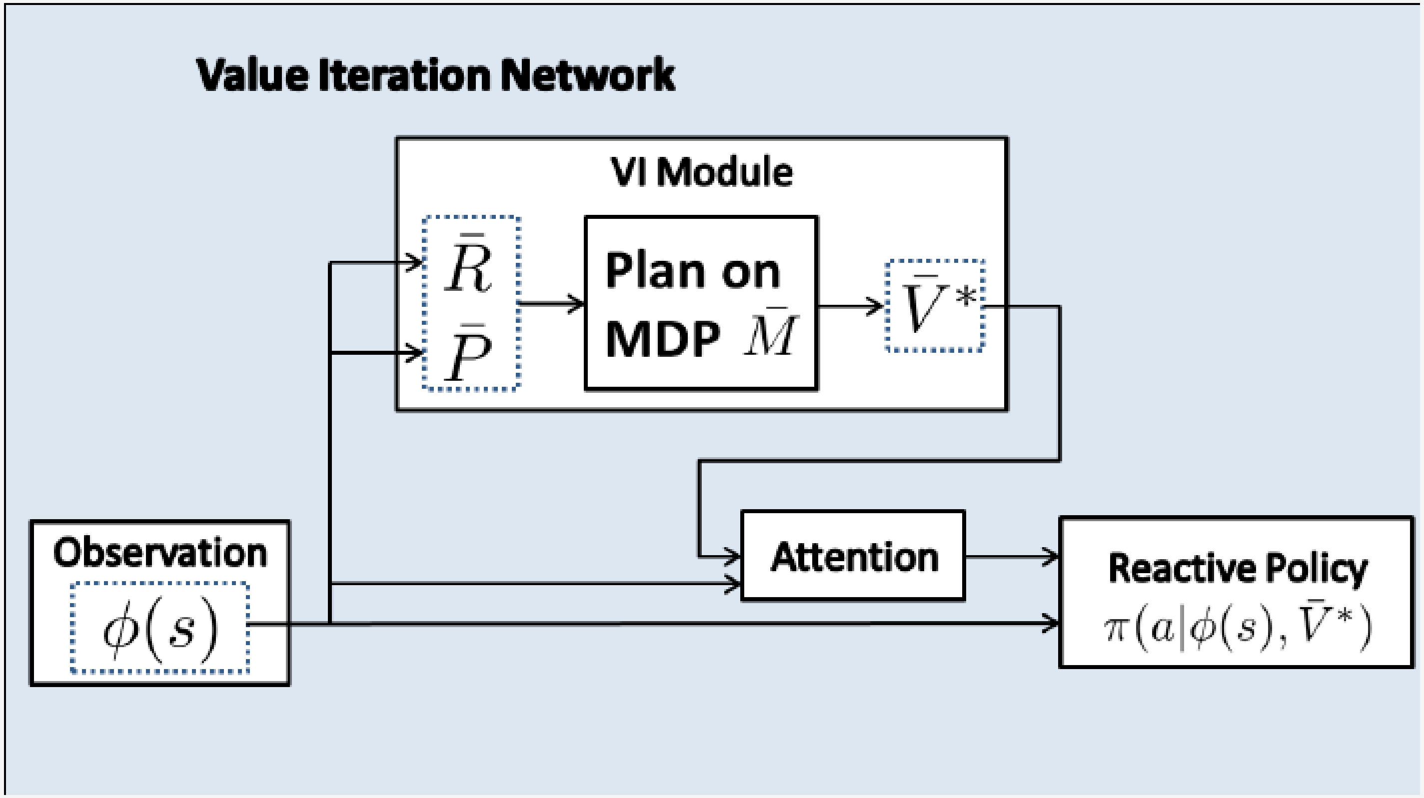


· $\bar{A}$ channels in $\bar{Q}$ layers
· Linear filters $\Leftrightarrow \gamma\overline{P}$
· Channel-wise max-pooling
· Tied weights

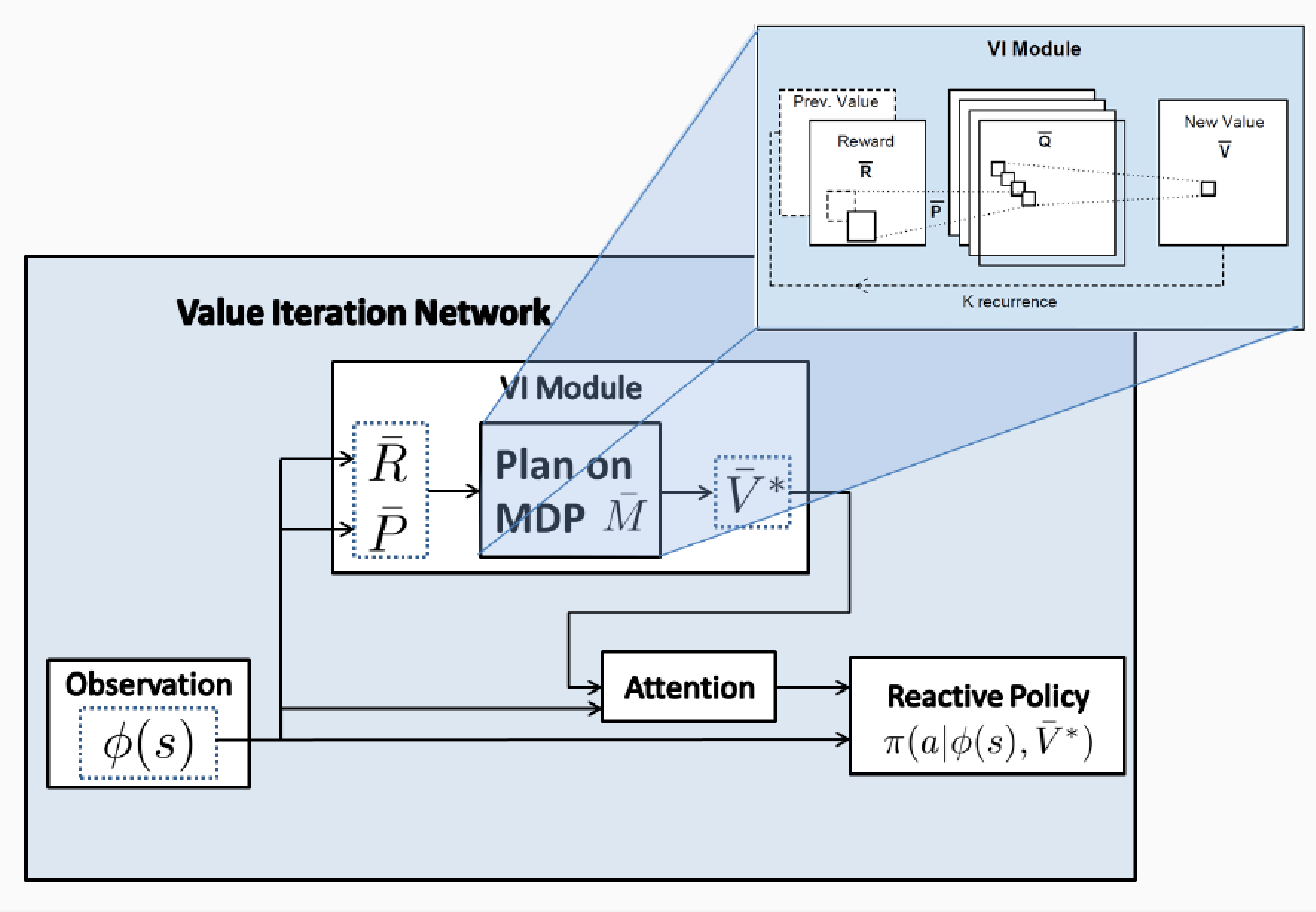
·Best for locally connected problems(grids, graphs)

VALUE ITERATION NETWORK

- Use VI model for planning:

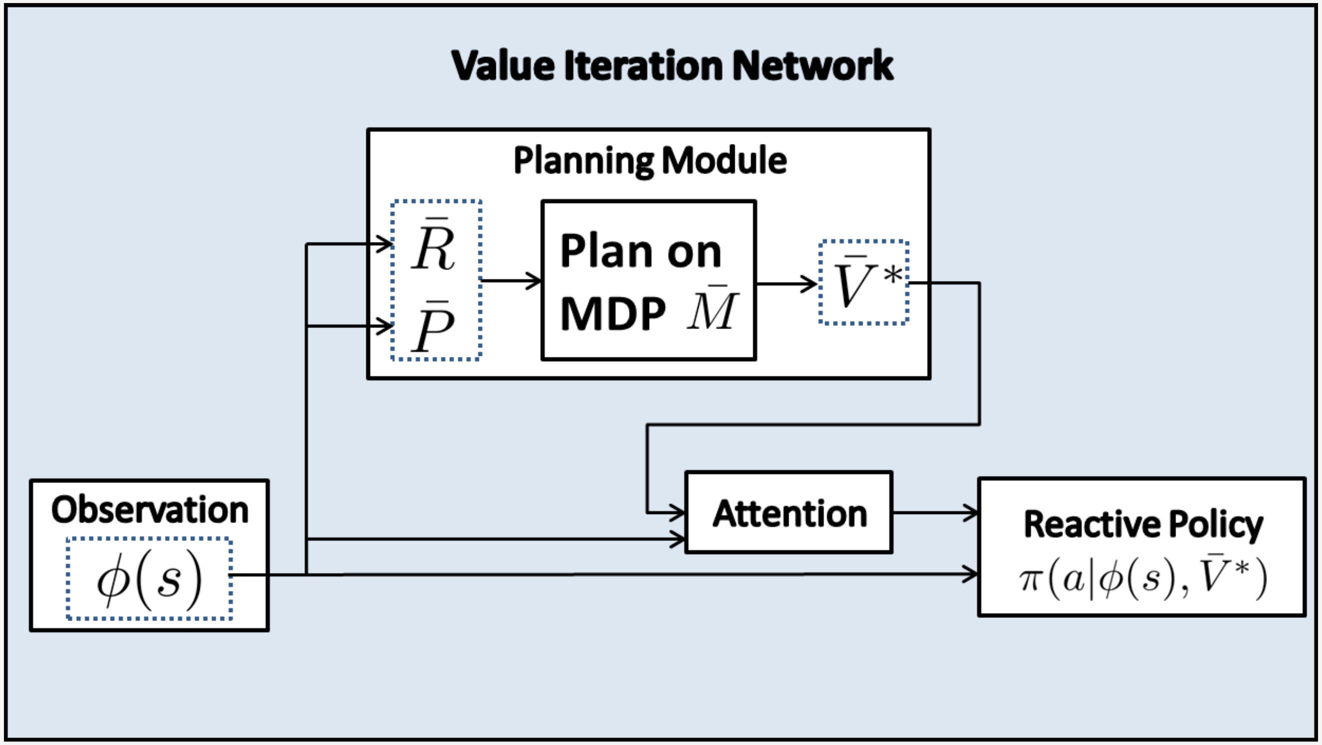


- Value iteration Network(VIN)



VALUE ITERATION NETWORK

- Use Neural Network to achieve Value Iteration
- The Networks can learn to plan
- Train like any other Networks
- Backprop – just like a convent
- Implementation – Use TensorFlow or Pytorch



EXPERIMENTS

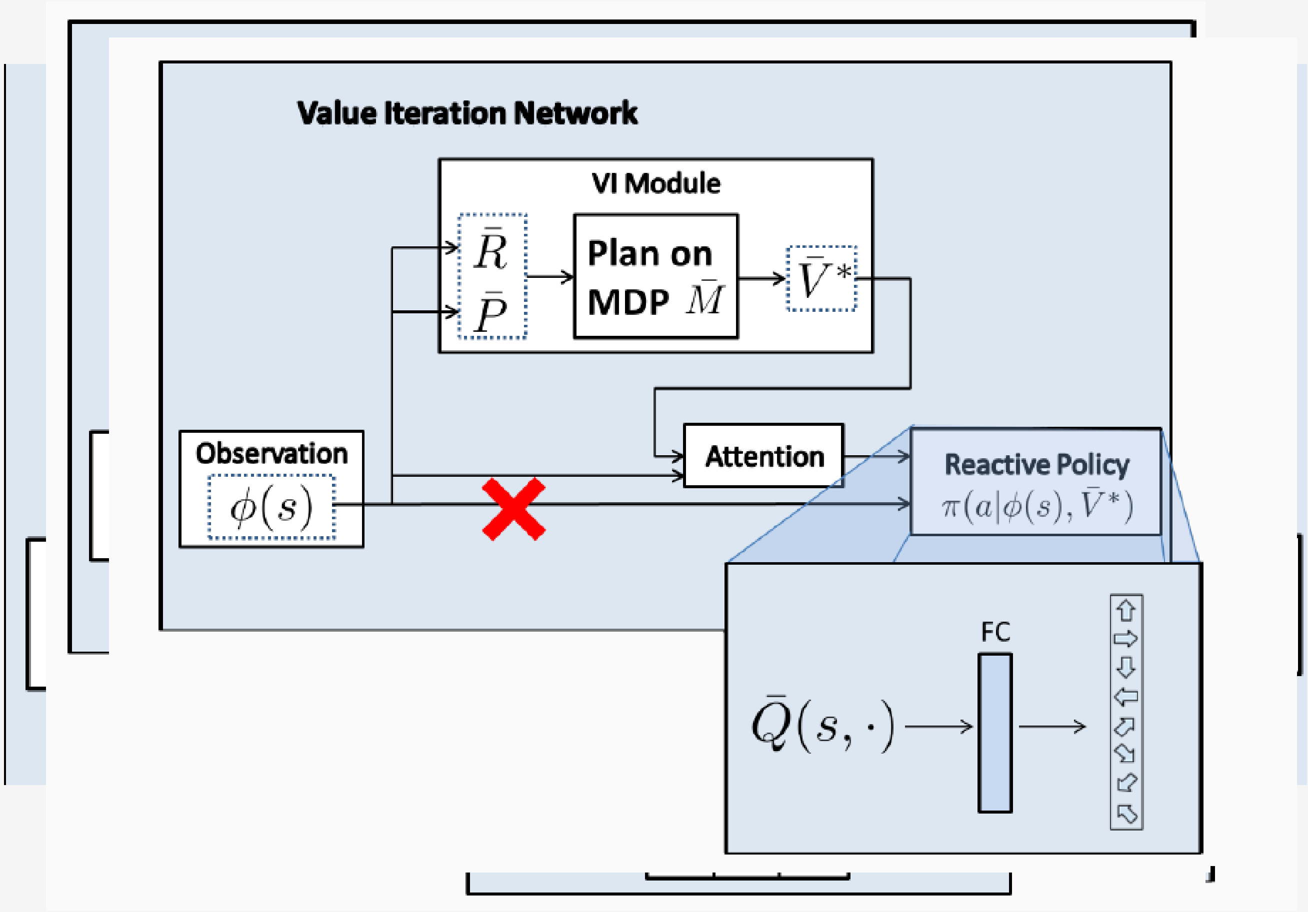
GRID-WORLD DOMAIN

- Supervised learning from expert(shortest path)
- Observation: image of obstacles + goal, current state
- Compare VINs with reactive policies

GRID-WORLD DOMAIN

- VI State space: grid-world
- VI Reward map: Convent
- VI Transitions: 3×3 kernels

- Attentions: choose Q values for current state
- Reactive Policy: FC, softmax



GRID-WORLD DOMAIN

Compare with:

- CNN inspired by DQN architecture[1]
  - 5 layers
  - Current state as additional input channel
- Fully convolutional net (FCN) [2]
  - Similar to our attention mechanism
  - 3 layers
  - Pixel-wise semantic segmentation (labels=actions)

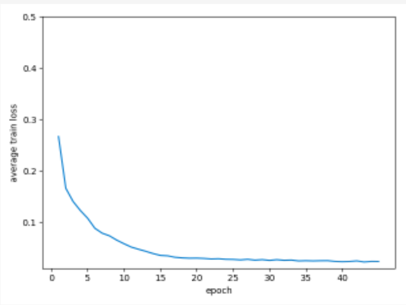
Training:

- 5000 random maps, 7 trajectories in each
- Supervised learning from shortest path

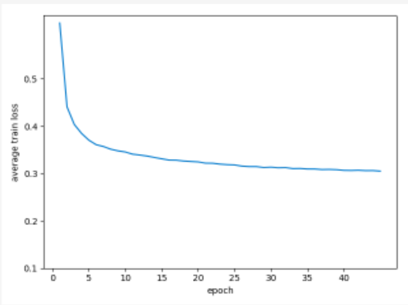
[1] Mnih et al. Nature 2015
[2] Long et al. CVPR 2015

Results:

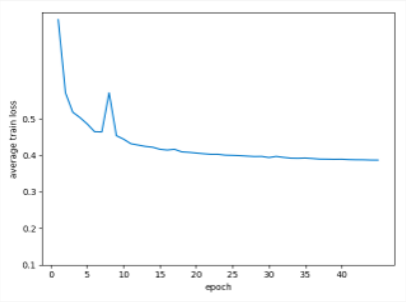
Average training loss:



(a)

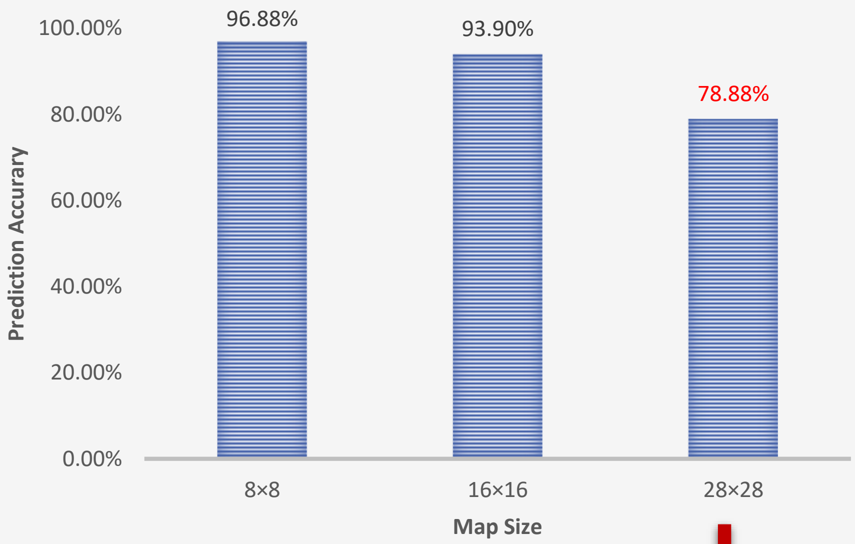


(b)



(c)

Success rate on test set:



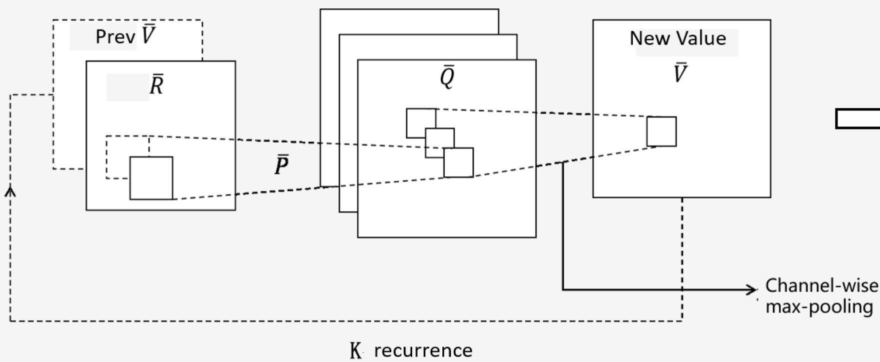
Need further improvement!

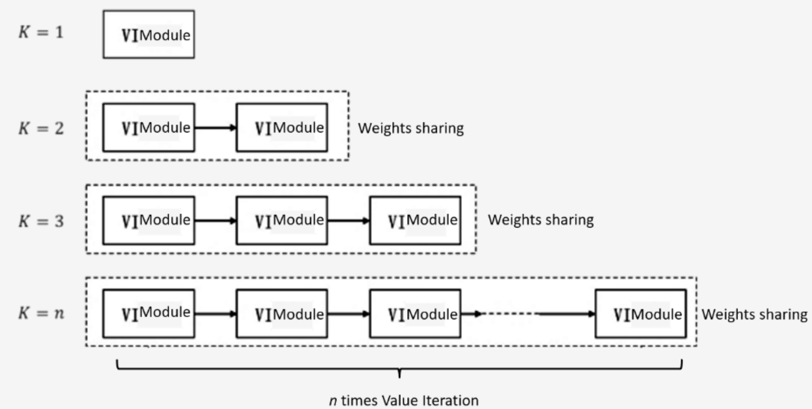
- Success rate – reach target without hitting obstacles

IMPROVEMENT

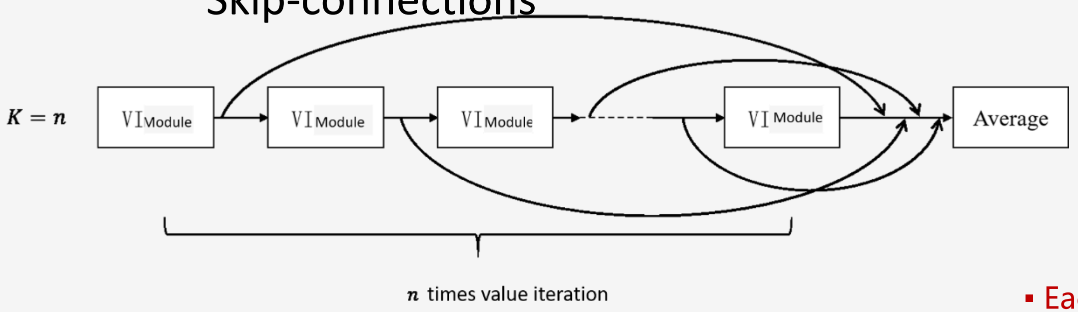
- Use deeper value iteration network:

Value iteration module





Skip-connections

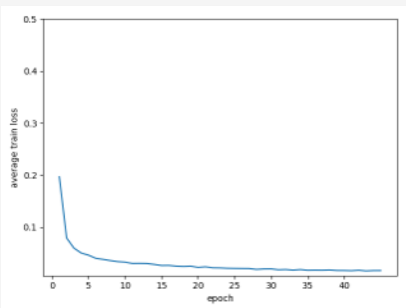


- Each value iteration uses different parameters
- Improve data flow with jump connections
- Greatly enhance the depth of the network

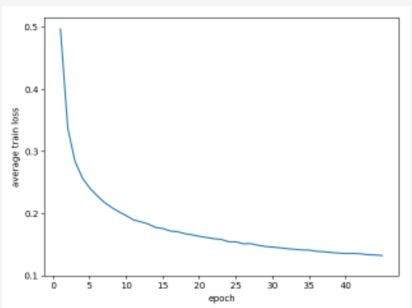
IMPROVEMENT

- Use deeper value iteration network:

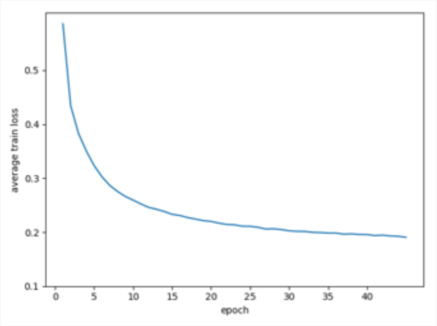
Average training loss:



(a)

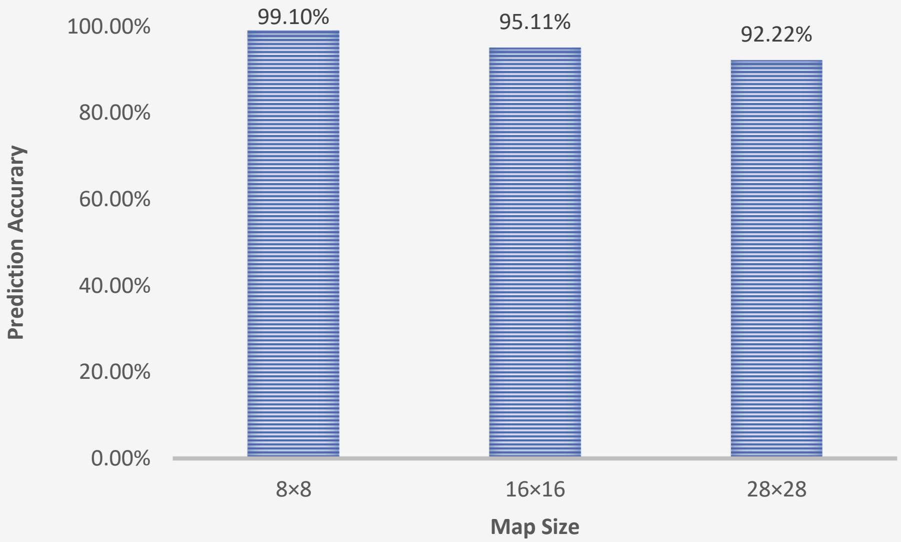


(b)



(c)

Prediction accuracy



- The average training loss of the network is obviously reduced, it performs well on the training set

- The accuracy is over 92% on the big map of the test set, which is significantly improved

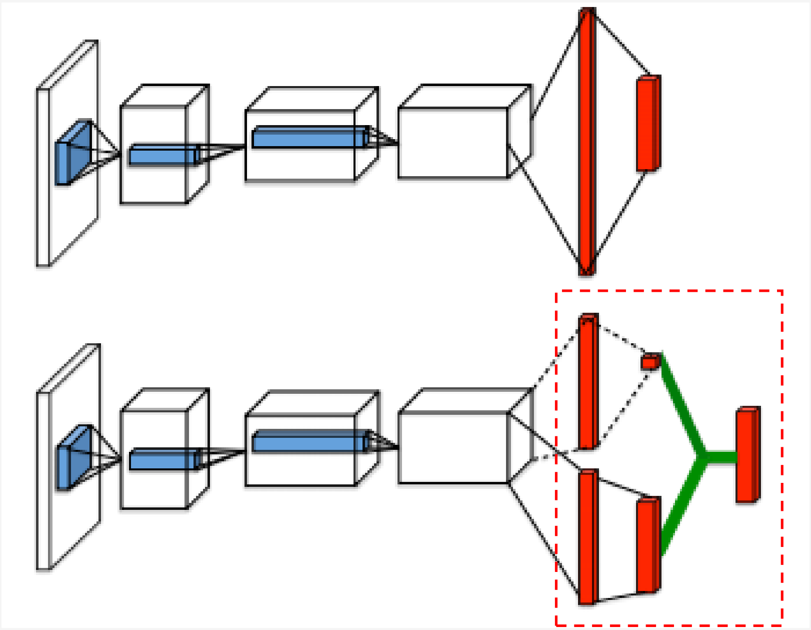
IMPROVEMENT

- Use Dueling Architecture(ICML2016)

-We notice that the Q function can be written into two parts:

$$Q_\pi(s,a) = V_\pi(s) + A_\pi(s,a)$$

-We only need to change the last layers(FC layers, softmax)



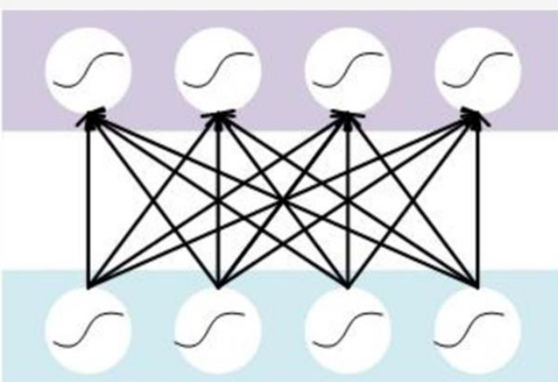
▪Competition structure
▪Two-channel output

IMPROVEMENT

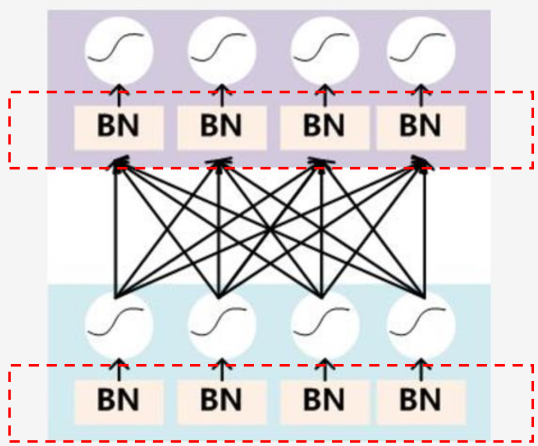
- Use Batch Normalization(BN_Layers)

-We have

$$y = \frac{x - E[x]}{\sqrt{Var[x] + \varepsilon}} * \gamma + \beta$$



Without BNLayers



With BNLayers

-Converge faster

-Use a larger learning rate

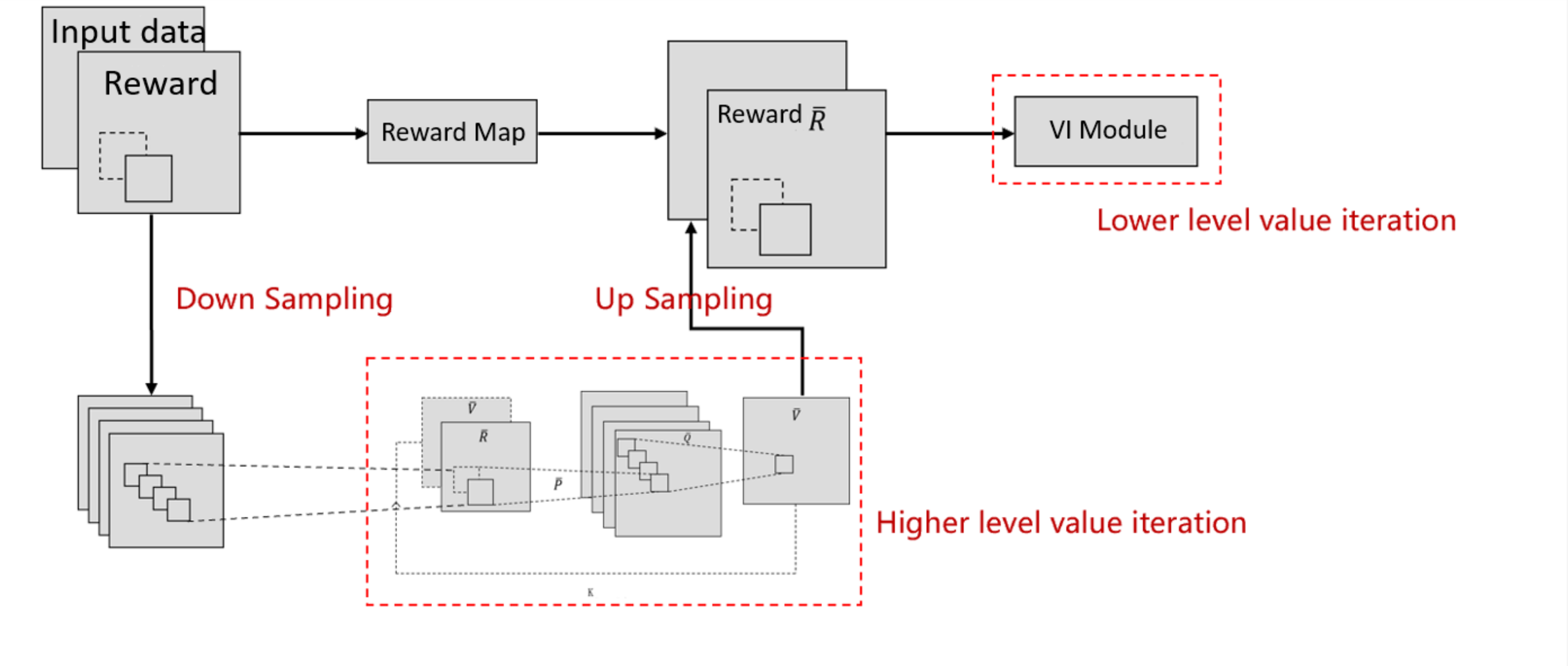
-Reduce loss function

-Avoid gradient explosion

IMPROVEMENT

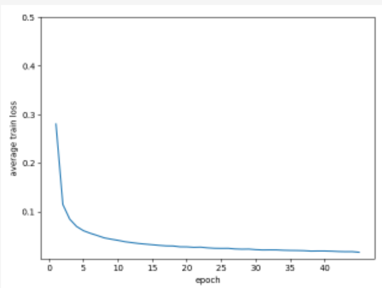
- Hierarchical Structure VIN:

To convey reward information faster in VI, and reduce the effective K, we use multiple levels of resolution.

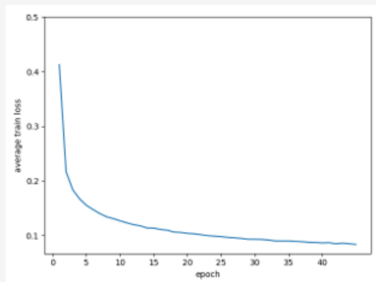


Use all the tricks

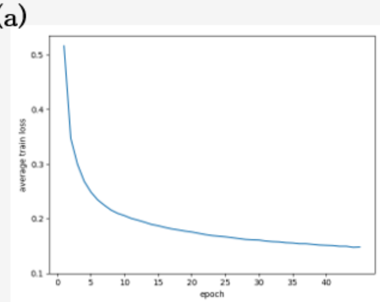
Average training loss:



(a)

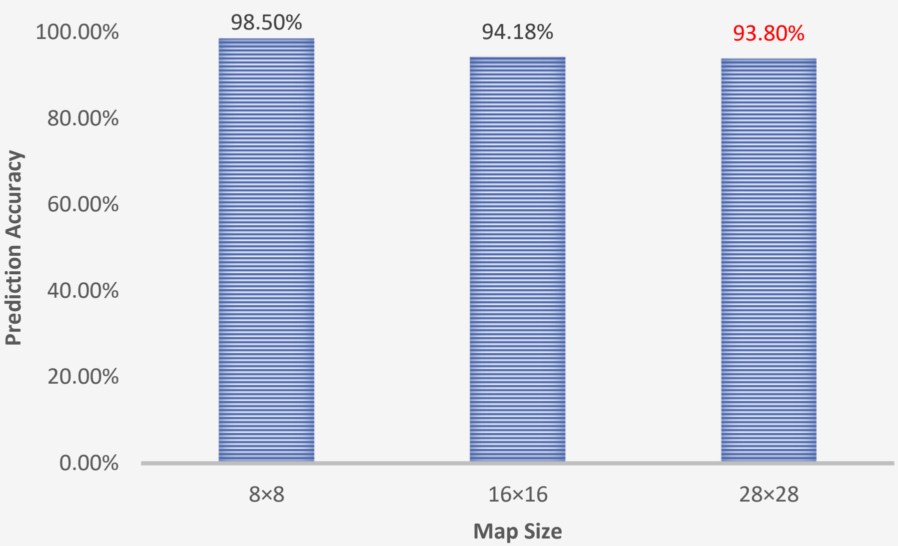


(b)



(c)

Prediction accuracy



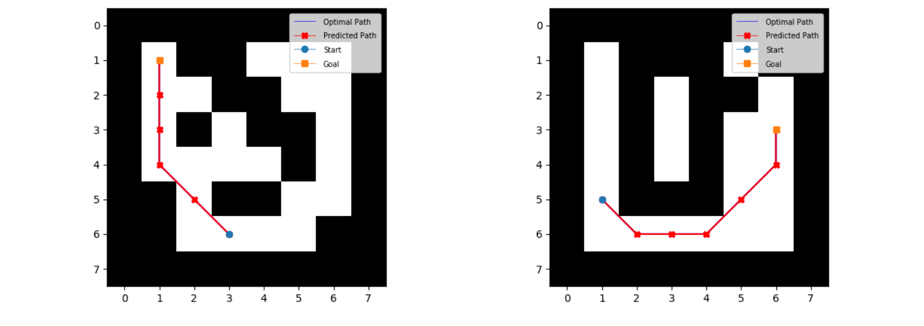
- The convergence speed of the network is accelerated and the average training loss is further reduced
- The model has the best performance with 93.7989% accuracy on the test set.

SUMMARY

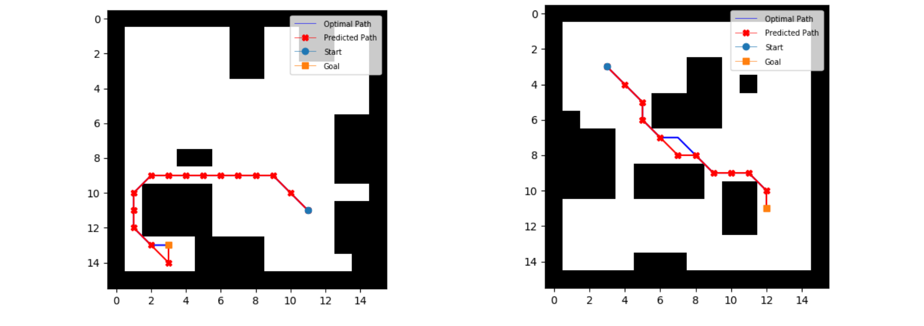
SUMMARY

☐ Path planning results

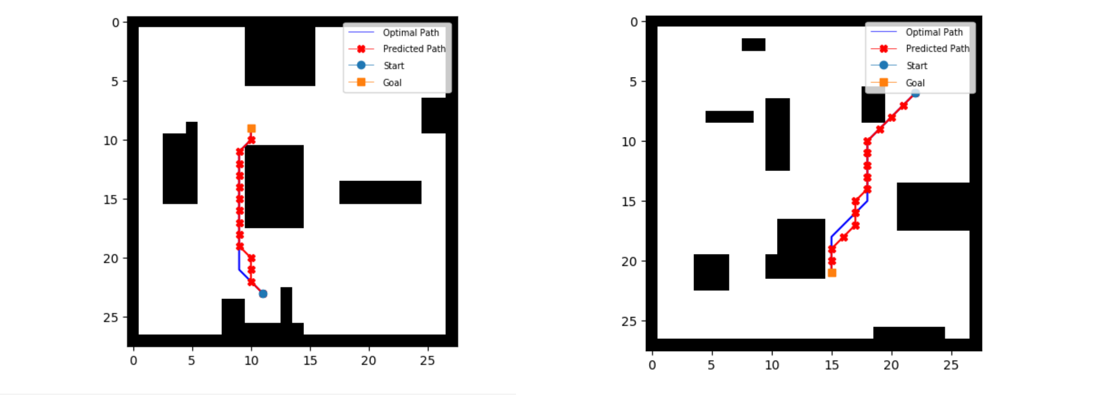
8×8:



16×16:

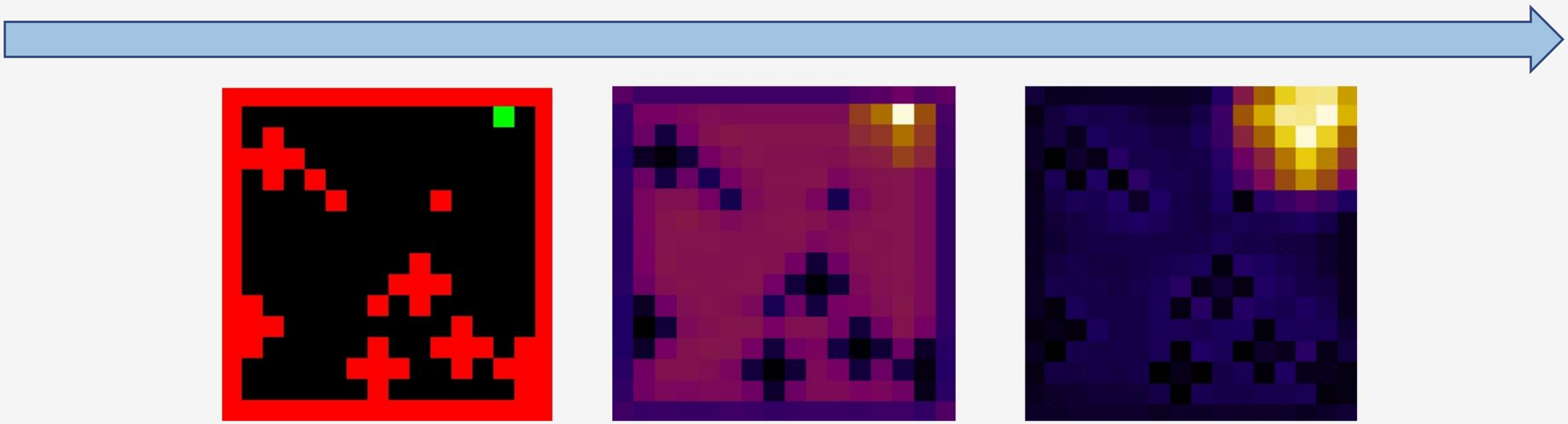


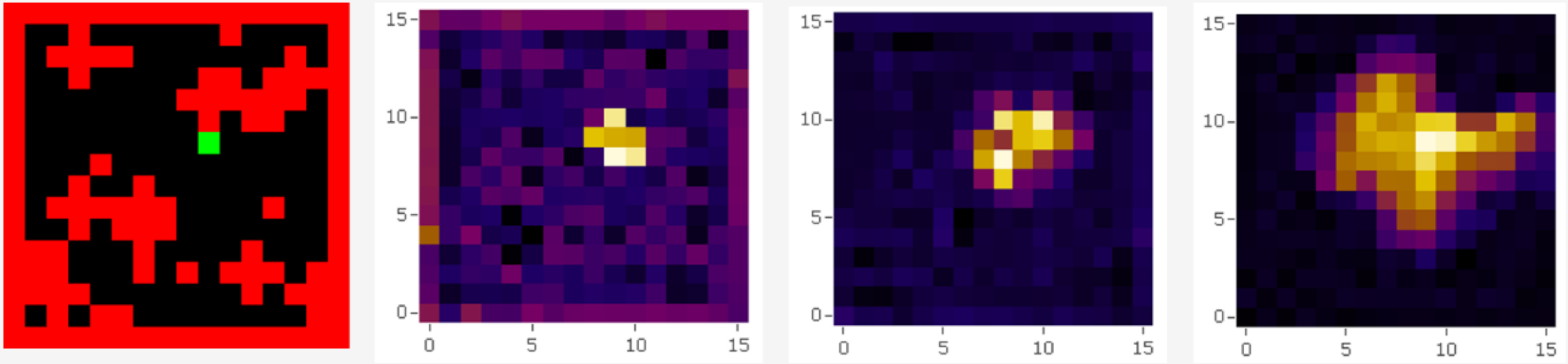
28×28:



- Value iteration process

With VI Iteration!





SUMMARY

- The performance of the models is summarized as follows

Network Structure	8×8	16×16	28×28
Original VIN	96. 8750%	93. 9019%	**78. 8834%**
Deeper VIN	99. 1027%	95. 1141%	92. 2152%
Dueling + BN_Layer	98. 0924%	95. 0924%	92. 6279%
Hierarchical VIN	98. 5027%	94. 1778%	**93. 7989%**

- Finally, the accuracy of the improved network is higher than 93% on different size maps
- The performance of the network on the big map has improved significantly, from less than 79% to more than 93%
- It shows that a series of improvements are effective and the network generalization ability is improved

SUMMARY

Compared with CNN and FCN

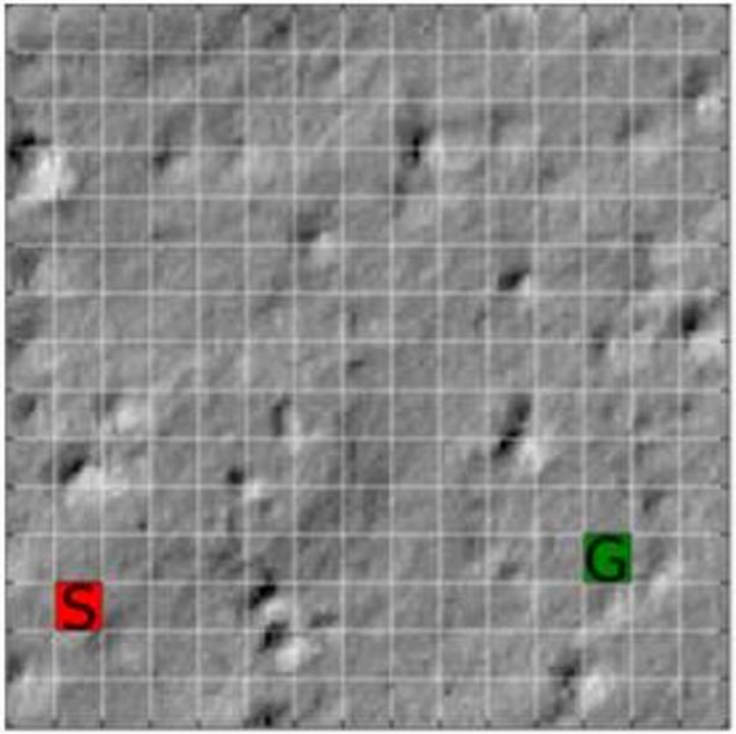
Map Size	Convolutional Neural Networks	Fully Convolutional Network	Hierarchical VIN	Accuracy Improved(Comp-ared with CNN)
8×8	97.9%	97.3%	98.5%	+ 0.6%
16×16	87.6%	88.3%	94.2%	+ 6.6%
28×28	74.2%	76.6%	93.8%	+ 19.6%

- Our improved networks get best performance on any size of map
- The accuracy of the network is higher than 93% on different maps
- Especially on the large map, the accuracy is significantly improved, and it is increased by nearly 20%

MARS-NAVIGATION

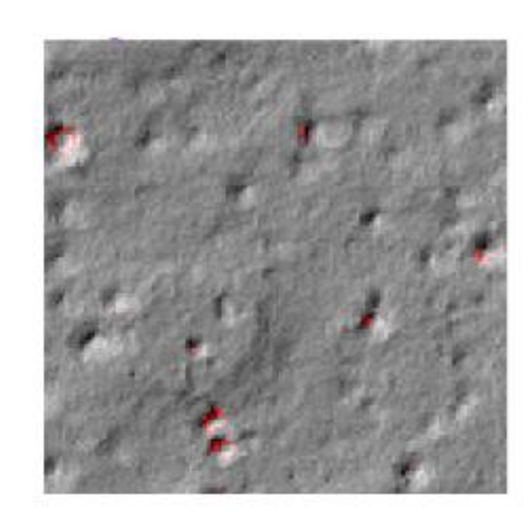
MARS-NAVIGATION

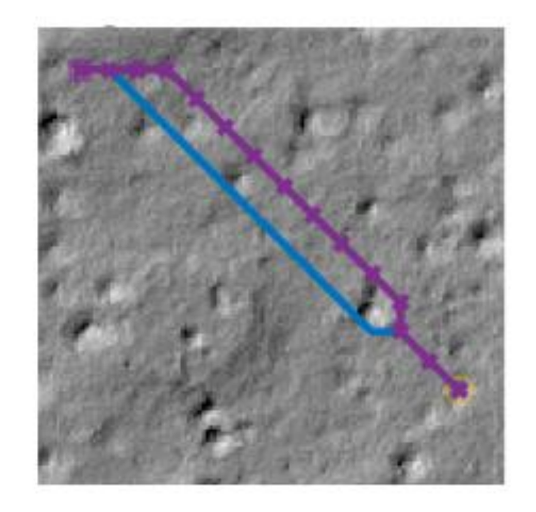
- Grid-world with natural image observations
- Overhead images of Mars terrain
- Obstacle = slope of 10∘ or more
- Elevation data not part of input



Results:

	Pred. loss	Succ. rate
VIN	0.089	84.8%
Best achievable	-	90.3%





THANK YOU!

REFERENCE

1. 章永龙. Dijkstra最短路径算法优化[J]. 南昌工程学院学报, 2006, 25(3):30-33.
2. 史辉, 曹闻, 朱述龙, et al. A*算法的改进及其在路径规划中的应用[J]. 测绘与空间地理信息, 2009, 32(6):208-211.
3. Volodymyr Mnih, et al. Human-level control through deep reinforcement learning[J]. Nature, 2015, 518: 529-533
4. Aviv Tamar, Yi Wu, Garrett Thomas, et al. Value iteration networks[C]. Advances in Neural Information Processing Systems, 2016: 2154-2162
5. Ren S., He K., Girshick R., et al. Faster r-cnn: Towards real-time object detection with region proposal networks[C]. Advances in neural information processing systems, 2015: 91-99
6. Ziyu Wang, et al. Dueling Network Architectures for Deep Reinforcement Learning[J]. arXiv preprint arXiv:1511.06581, 2016
7. Richard S.Sutton, et al. Reinforcement Learning: An Introduction 2nd ed[M]. London: The MIT Press, 2017: 59-69
8. Cormen, et al. Introduction to Algorithms 2nd ed[M]. The MIT Press & McGraw-Hill, 2001: 344
9. Tieleman, Tijmen, and Geoffrey Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. COURSERA: Neural networks for machine learning 4.2, 2012: 26-31
10. Gao Huang, Zhuang Liu, et al. Densely Connected Convolutional Networks[C]. Computer Vision and Pattern Recognition, 2017: 4700-4708