Paper Review

Value Iteration Networks - NIPS 2016

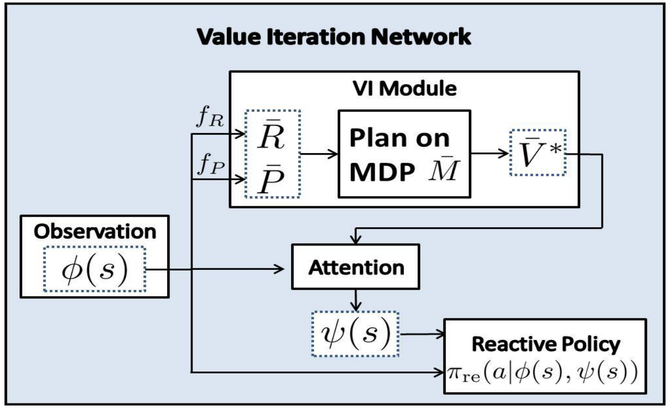
Paper Authors
Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, and Pieter Abbeel
Dept. of Electrical Engineering and Computer Sciences, UC Berkeley

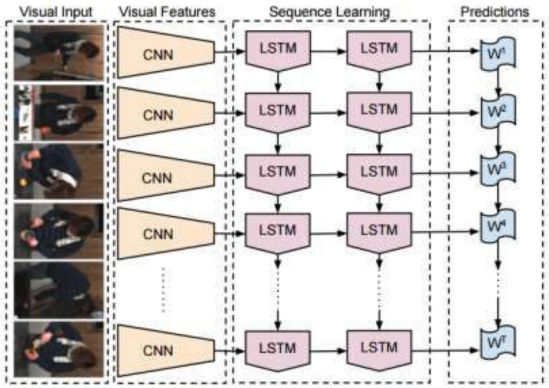
Submitted by :
Balagopal Unnikrishnan (A0178398E)
Kenneth Rithvik (A0178448M)
Madan Kumar Manjunath (A0178237W)
Ria Vijay Nagpal (A0178257R)

What are VINs and why design them?

Value Iteration Network (VIN)

- A neural network that can ***learn to plan***
- Fully differentiable and end to end trainable with backpropagation[1]
- Produces a differentiable approximation of the Value Iteration Algorithm used for planning
- Works for discrete and continuous path planning domains
- Generalizes well to unseen domains





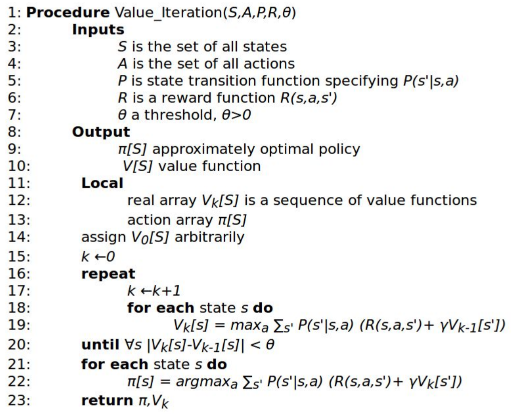
Motivation

- Goal of authors is to use this network to in autonomous robotics[2]
- ***Until now, CNN's were mostly use as feature extractors*** *(as seen in models in the left).*[3]
- Reinforcement learning has achieved breakthroughs for many policy and planning problems in the past.
- However, Reinforcement Learning does not generalize well - because they are ***reactive*** in nature.
- So plan first and provide these plans to the reactive policy as additional inputs.
- Hence, **design a neural network with an embedded planning module**

Background - Keys to the puzzle

Value Iteration Algorithm

- Usually, planning problems are modelled using a Markov Decision Process (MDP) which gives policies which specify which action to take so as to maximize long term rewards for the modelled problem.

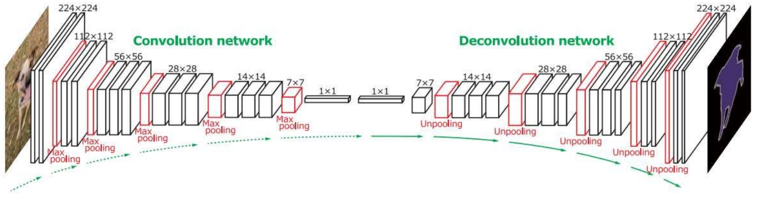


```
**Procedure** Value_Iteration($S,A,P,R,\theta$)
    **Inputs**
        $S$ is the set of all states
        $A$ is the set of all actions
        $P$ is state transition function specifying $P(s'|s,a)$
        $R$ is a reward function $R(s,a,s')$
        $\theta$ a threshold, $\theta>0$
    **Output**
        $\pi[S]$ approximately optimal policy
        $V[S]$ value function
    **Local**
        real array $V_k[S]$ is a sequence of value functions
        action array $\pi[S]$
    assign $V_0[S]$ arbitrarily
    $k \leftarrow 0$
    **repeat**
        $k \leftarrow k+1$
        **for each** state $s$ **do**
            $V_k[s] = \max_a \sum_{s'} P(s'|s,a) (R(s,a,s')+ \gamma V_{k-1}[s'])$
    **until** $\forall s\ |V_k[s]-V_{k-1}[s]| < \theta$
    **for each** state $s$ **do**
        $\pi[s] = \operatorname{argmax}_a \sum_{s'} P(s'|s,a) (R(s,a,s')+ \gamma V_k[s'])$
    **return** $\pi,V_k$
```

- ***Value Iteration Algorithm*** [4]*(given above)* is just one algorithm that helps achieve this.
- It is interesting to note that this algorithm has existed since 1957[5]

Convolutional Neural Networks

- Stacked layers of convolution and max pooling layers
- Trainable by backpropagation



Reinforcement Learning

- Teaching policies by Trial and Error
- Usually done when the search space of MDP is very large
- Done by maximizing the reward

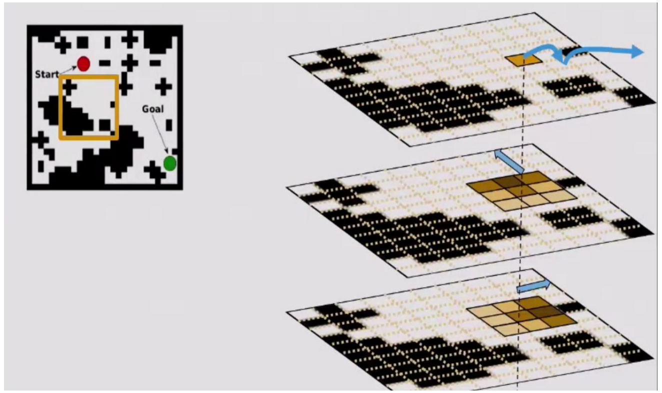
Imitation Learning

- Teaching policies by expert supervision
- Reduces the problem to a supervised learning problem

Novelty - Observing the parellels

Representing the Value Iteration Algorithm as a CNN

- One of the key observations that the authors had was realizing that ***the classic VI Algorithm could be represented as a CNN***[6]
- While the classic VI Algorithm relies on a MDP, a CNN by virtue of it being ***differentiable*** can be used to approximate the policy provided by the classic algorithm by simple backpropagation.



Parallels in two dimensional operations

- While modelling a grid-world problem, it is quite easy to observe that calculating the value of stepping into the neighbouring cells is very similar to a classic convolution followed by max pooling operation along the direction of maximum rewards.[2]

Converting the regular mathematical formula of the classic VI Algorithm to a differentiable CNN

Model as max pooling layers

Model as convolution layers

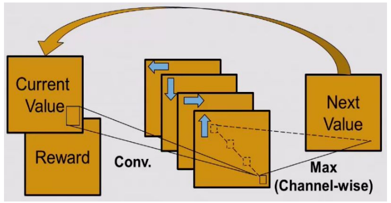
$$V_{n+1}(s) = \boxed{\max_a} Q_n(s,a) \quad \forall s, \quad \text{where} \quad Q_n(s,a) = R(s,a) + \boxed{\gamma \sum_{s'} P(s'|s,a)} V_n(s').$$

V-Value | R-Reward | S-States | γ-discount factor | P-transition kernel

Creating the Value Iteration Network

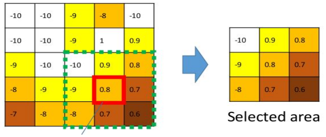
How to Plan?

Value iteration is done in the VI module by treating the problem as a convolution problem as mentioned in the previous slide



How to use the Plan?

The reactive policy requires as input the rewards to reach its goal. Since only rewards of the neighbours is needed for the next step, the concept of ***attention*** for neural networks is used.[7]

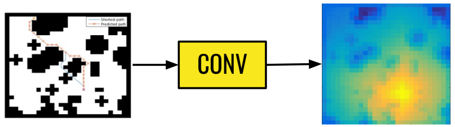


-10	-10	-9	-8	-10
-10	-10	-9	1	0.9
-9	-10	-10	0.9	0.8
-8	-9	-9	0.8	0.7
-7	-8	-8	0.7	0.6

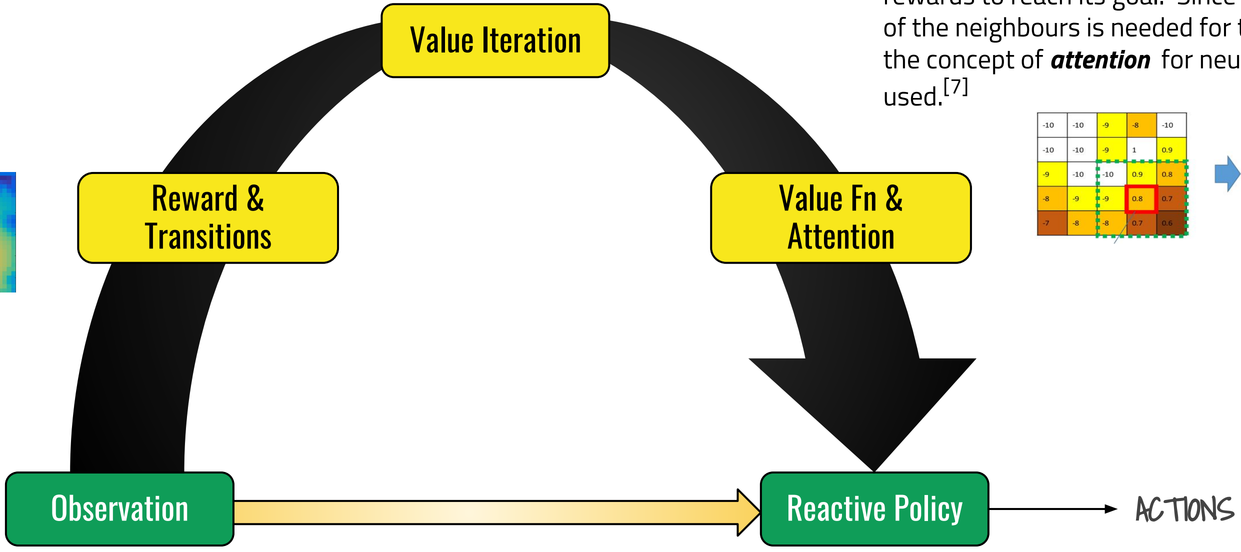
-10	0.9	0.8
-9	0.8	0.7
-8	0.7	0.6

Selected area

What to Plan?

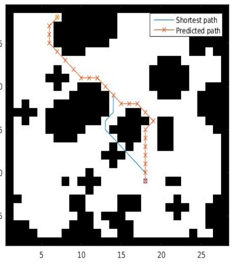


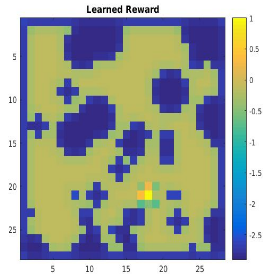
Extract the rewards from the input observations using a convnet



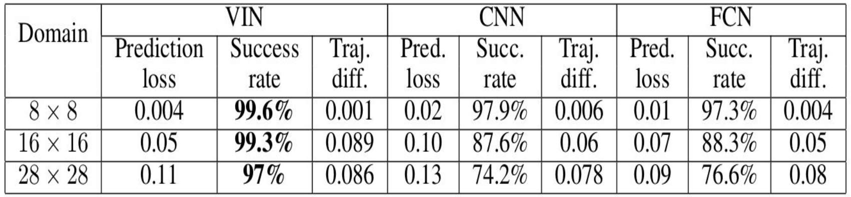
Experiments

Grid world Search



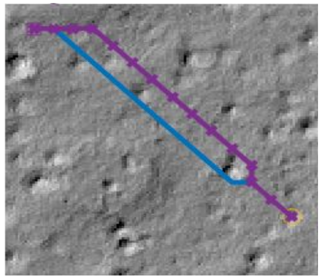


- The goal is to traverse from start point to end point without crossing the obstacles. The VIN plans first and provides this planning attributes to the reactive policy - unlike regular reactive policies with CNNs or Fully Connected Networks(FCN)
- Hence, VINs are found to outperform CNN and FCN in all sizes of grids and generalize better. ***Even though the prediction loss of CNNs and FCNs is smaller than VINs, the overall success rate is higher for VINs***
- This also goes to prove that this is not a case of overfitting or underfitting to reactive policies. The visualization of the rewards show that the VINs are learning the right rewards as well.



Domain	VIN			CNN			FCN		
	Prediction loss	Success rate	Traj. diff.	Pred. loss	Succ. rate	Traj. diff.	Pred. loss	Succ. rate	Traj. diff.
8×8	0.004	**99.6%**	0.001	0.02	97.9%	0.006	0.01	97.3%	0.004
16×16	0.05	**99.3%**	0.089	0.10	87.6%	0.06	0.07	88.3%	0.05
28×28	0.11	**97%**	0.086	0.13	74.2%	0.078	0.09	76.6%	0.08

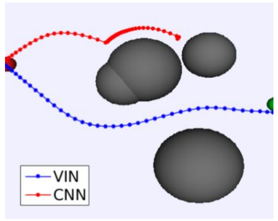
Mars Rover Navigation

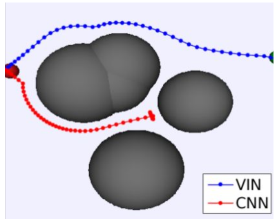


- The task for the network is to traverse from start point to end point by avoiding points greater than 10 degree elevation.
- The input is a 128 x 128 image patch which defines a 16 x 16 grid world. Random start and end positions were given to predict the path.
- It is interesting to note here that VIN made the planned path without access to the elevation data - meaning that raw images were used as input.
- An 84.8% success rate was achieved
- The Markov Decision Process(MDP) is similar to that of the Grid world search

This experiments prove that VINs can learn to plan from natural input images. Hence these can be used to in real world planning applications.

Experiments





Continuous Control

- A 2D path planning problems - with continuous states and continuous actions.
- The VIN does a high level plan with a discrete grid-world representation.
- On contact, the obstacles show elastic force and friction.
- This model has to plan both the trajector and also the control of the particle- which has characteristics like mass and inertia.
- The VIN outperforms the CNNs by a huge margin in cases previously unseen as see by the large reduction in test error as we progress from CNN to VIN

	Train Error	**Test Error**
CNN	0.39	0.59
VIN	0.30	0.35

WebNav Challenge

- The WebNav challenge requires navigation from a start page to a target web page - specified by a 4 sentence query
- It is a ***language based graph search task***
- The next state is selected based on the embedding features of the state and the next possible states
- The dataset used is ‘Wikipedia for Schools’ with over 6000 pages and 292 links per page
- The VINs are made to plan ahead on1st and 2nd level categories and their word-embedding features.
- The reward mechanism was provided ***similarity score*** between query features and features of subset nodes selected.

	Successful runs from root node	**Successful runs from random node**
Baseline	1025	304
VIN	1030	346

Applications

Robotics

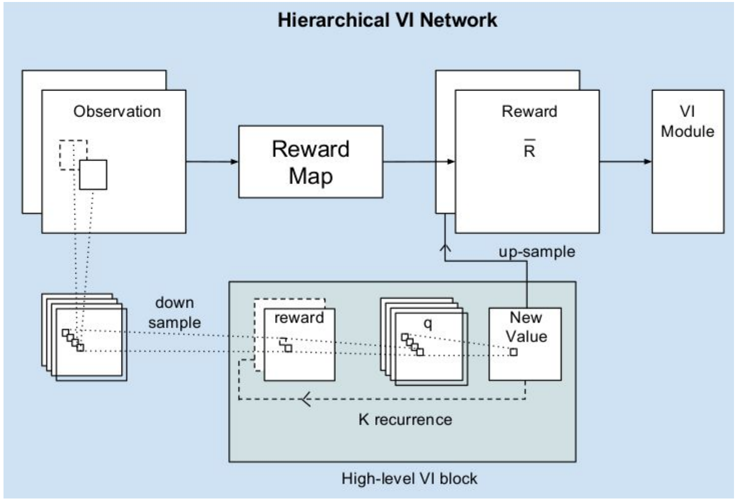
- The original intention of the authors while making Value Iteration Networks was to use in autonomous robots.
- These can be used for tasks such and path planning for navigation and transfer robots.
- VINs have shown that they generalize to cases not within the test domain.
- This is extremely helpful as Reinforcement Learning - a common method in teaching actions to robots cannot work outside the cases provided during training.
- They have been shown to work for both discrete and continuous control cases

Hierarchical Context

- The model has performed better than baseline models for the WebNav challenge.
- Hence, this kind of generalization can be used on any kind of dataset which has a tree like structure ie. categories, subcategories, sub-sub-categories etc.

Hierarchical VINs

- Downsampling by a factor of ***d*** offeres ***d*** times speedup at the cost of accuracy. To convey reward information faster and reduce iterations, by feeding in coarser information from upper levels to the lower levels.
- This has shown to achieve a better success rate over 28x28 (larger) grid-world problems.



Issues and Improvements

Comparison with Strategic Attentive Writer for Learning Macro-Actions (STRAW)

- STRAWS[8] is a paper by Google DeepMind which also creates long term implicit plans to maximize reward.
- There is no comparison with this contemporary paper which was also released during the same time.
- Only once these two methods are evaluated side by side can there be a true comparison.
- This comparison is also very relevant because of underlying similarities between the methods proposed in VIN and for action matrix in STRAW

Computing Cost

- The cost of computation of the CNNs and policies have not been mentioned in the paper. Without benchmarks on performance over standard datasets, it is not clear as to the computational efficiency of VINs

Usage in Language Models

- Even though the model has outperformed regular models in the WebNav task, there has only been 346 successful runs out of 4000 test queries. There is a scope for improvement in the area of language models.
- This can be done by better word embeddings and improved attention mechanisms

Learning To Plan

- Reviews[9] of the paper have pointed out that if the VIN model actually "learns to plan" by itself as the paper claims - it has to learn the process of value iteration by itself - not from the set existing model

Requirement of additional background parameters to create initial model

- Lots of background knowledge is needed to create a VIN - state space, goal state, attention model to be used.
- This might not be possible in a typical Reinforcement/Imitation Learning problem

References

1. Value Iteration Network : Aviv Tamar, Yi Wu, Garrett Thomas, Sergey Levine, Pieter Abbeel
2. Video Presentation by Aviv Tamar on Channel 9
3. Paper on Long-term Recurrent Convolutional Networks for Visual Recognition and Description
4. Value Iteration - AI : Foundation of Computational Agents
5. Article on The Morning Paper about value iteration networks
6. Presentation by Sungoon Choi on value iteration networks
7. Presentation by **Fujimoto Keisuke on** value iteration networks
8. Paper on Strategic Attentive Writer for Learning Macro-Actions
9. Proceedings of NIPS 2016