Transfer Value Iteration Networks

Junyi Shen,[1,2] **Hankz Hankui Zhuo,**[1*] **Jin Xu,**[2] **Bin Zhong,**[2] **Sinno Jialin Pan**[3]
[1]School of Data and Computer Science, Sun Yat-Sen University, Guangzhou, China
[2]Data Quality Team, WeChat, Tencent Inc., China
[3]Nanyang Technological University, Singapore
{vichyshen, jinxxu, harryzhong}@tencent.com, zhuohank@mail.sysu.edu.cn, sinnopan@ntu.edu.sg

Abstract

Value iteration networks (VINs) have been demonstrated to have a good generalization ability for reinforcement learning tasks across similar domains. However, based on our experiments, a policy learned by VINs still fail to generalize well on the domain whose action space and feature space are not identical to those in the domain where it is trained. In this paper, we propose a transfer learning approach on top of VINs, termed Transfer VINs (TVINs), such that a learned policy from a source domain can be generalized to a target domain with only limited training data, even if the source domain and the target domain have domain-specific actions and features. We empirically verify that our proposed TVINs outperform VINs when the source and the target domains have similar but not identical action and feature spaces. Furthermore, we show that the performance improvement is consistent across different environments, maze sizes, dataset sizes as well as different values of hyperparameters such as number of iteration and kernel size.

Introduction

Convolutional neural networks (CNNs) have been applied to reinforcement learning (RL) tasks to learn policies, i.e., a mapping from observations of system states to actions (Mnih et al. 2015). As analyzed in (Tamar et al. 2016), reactive polices learned by conventional CNN-based architectures usually fail to generalize well to previously unseen RL domains even though most of the configurations remain the same to the training domain. To boost the generalization performance, *value iteration networks* (VINs) (Tamar et al. 2016) have been proposed to integrate a planning module into policy learning. VINs have been applied to various application tasks including path planning, e.g., visual navigation (Gupta et al. 2017) and the WebNav challenge (Nogueira and Cho 2016), which requires the agent to navigate the links of a website towards a goal web-page, specified by a short query. Despite the success of VINs, we observe that the generalizability of VINs is based on an implicit assumption that the feature space and the action space in the unseen domain are as the same as the ones in the seen domain for training the policy. To relax this assumption, in this paper, we propose a transfer learning framework to generate VIN-based policies across different domains even if their action spaces and feature spaces are not identical.

Intuitively, if the target domain has different feature space and action space to the source domain, the VIN-based policy learned from the source domain fails to be used in the target domain directly. A straight-forward solution is to learn a new policy from scratch from the target domain, which is time-consuming. Therefore, it is more desirable to transfer the learned knowledge captured in the source-domain VIN-based policy to the target domain, such that an optimal target policy can be learned with less training data and shorter training time. However, if the source domain and the target domain have totally different feature spaces or action spaces, it is extremely difficult to adapt a learned policy across domains effectively. Therefore, in this work, we assume that 1) the feature spaces between the source and the target domains can be different but are not heterogeneous (e.g., text v.s. images), 2) there is a common subset of actions between the source and the target domains.

We propose the Transfer VIN (`TVIN`) to transfer the a well-trained VIN-based policy from the source domain to the target domain with limited training data. Specifically, to address the difference between feature spaces and action spaces across domains, in `TVIN` , we develop two transfer learning modules with respect to the learned reward function and the learned transition function, respectively:

- We first encode state observations (with different features to the source domain) in the target domain to the same representation of the source domain, such that the reward function transferred from the pre-trained VIN can accurately produce reward images in the target domain.
- We then leverage the common subset of actions between the source and the target domains to transfer state-action transition information from the source domain to the target domain. Furthermore, we fine-tune the transferred transition function by introducing transfer weights to automatically learn to what degree the transferred actions resemble.

By leveraging knowledge transferred via the above transfer learning modules, we further design a new Value Iteration module (VI module) to generate a policy for the target

*corresponding author

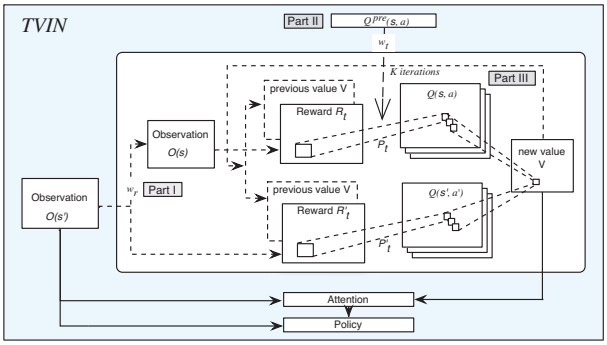

Figure 1: The Framework of a TVIN

domain. An optimal target-domain policy can be learned by back-propagating the gradient of loss through the whole TVIN in an end-to-end training manner.

To evaluate the effectiveness of our proposed TVIN , we conduct experiments to transfer knowledge between different 2D RL domains, including 2D mazes and Differential Drive (Lee et al. 2018). We evaluate the transfer performance of TVIN with varying environments, maze sizes, dataset sizes and hyperparameters, etc. Extensive experiments empirically show that our proposed TVIN is able to learn a target-domain policy significantly *faster* and reach a *higher* generalization performance, compared with the conventional VIN and another heuristic transfer learning method.

Problem Definition

Let M denote the MDP of some domain, where an optimal policy π is expected to be learned. The states, actions, rewards, and transitions in M are denoted by $s \in \mathcal{S}$, $a \in \mathcal{A}$, $R(s,a)$ and $P(s'|s,a)$ respectively. Let $\phi(s)$ denote an observation for state s. R and P are dependent on the observations as $R = f_R(\phi(s))$ and $P = f_P(\phi(s))$. The functions f_R and f_P are learned jointly in the policy learning process. Given a pre-trained MDP in a source domain, we aim to transfer the learned knowledge including the learned reward function and transition function to the target domain, such that an optimal policy $\pi(a|\phi(s);\theta)$ for the target domain can be learned. Here θ denotes all the parameters of the TVIN .

Transfer Value Iteration Networks

In this section, we introduce our proposed TVIN in detail. The overall framework of TVIN is depicted in Figure 1. We develop an encoder to map observations of states in the target domain to the feature representation as the same as the source domain, which is indicated as "part I" in the figure. We then transfer the Q-network with respect to *the common subset of actions* from the source domain to the target domain, which is indicated as "part II" in the figure. After that, we enrich the Q-network by learning states transition for domain-specific actions of the target domain from scratch. By combining the above two Q-networks, we design a new Value Iteration module (VI module) for the target domain, which is indicated as "part III". The planning-integrated TVIN -based policy for the target domain can be trained in an end-to-end manner by back-propagating the gradient through the whole network.

Pre-trained VIN

For TVIN , we suppose that a well-trained source-domain VIN model is given in advance. Basically, a key idea behind many reinforcement learning algorithms is to estimate the action-value function (Tsitsiklis and Roy 2002), by using the Bellman equation as an iterative update, $Q_{i+1}(s,a) = \mathbb{E}_{s,a}\left[r + \gamma \max_{a'} Q_i(s',a')|s,a\right]$. The value-iteration algorithm is a popular algorithm for calculating the optimal value function V^* and deriving the correspondingly optimal policy π^*. In each iteration, $V_{n+1}(s) = \max_a Q_n(s,a) \quad \forall s$, where $Q_n(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a)V_n(s')$. The value function V_n converges to the optimal value function V^* when $n \to \infty$, from which an optimal policy is derived as $\pi^*(s) = \arg\max_a Q_\infty(s,a)$. In a VIN, a VI module implemented by a neural network is used to approximate the value iteration algorithm. Specifically, the VIN first produces a reward image R by $f_R(\phi(s);\theta)$ and inputs R of dimensions l, m, and n to the VI module. The reward is then fed into a convolutional Q-layer of $\mathcal{A}$ channels followed by a linear activation function: $Q_{a,i',j'} = \sum_{l,i,j} W^a_{l,i,j} R_{l,i'-i,j'-j}$. Each channel in this layer corresponds to $Q(s,a)$ for a particular action a. This layer is then max-pooled along the actions channel to produce the next-iteration value function layer, where $V_{i,j} = \max_a(Q(a,i,j))$. The next-iteration value function layer V is then stacked with the reward R, and fed back into the convolutional layer and max-pooling layer K times to perform K value iterations. By training the VIN end-to-end in the source domain, we obtain a source-domain VIN and its derived policy for knowledge transfer.

TVIN Algorithm

The overall algorithm is presented in Algorithm 1. Given the pre-trained VIN in the source domain, the pre-trained reward function f_R is first transferred to produce reward images for the observation s in the target domain (i.e., Step 3). After that the state transition values on the common subset of actions, f_p^{pre}, is transferred to the target domain with a learnable weight associated with each action to measure the similarity degree between domains. And the state transition values on new domain-specific actions, f_p^{new}, are learned from scratch. All of these state transition values reconstruct a transition function in the target domain, which is further used to compute the Q-function in each iteration for the target domain (i.e., Steps 6 and 7). An attention vector is fed as an input to generate the target policy π_T (i.e., Step 11). Finally, the back-propagation algorithm is used to update the parameters of the whole network to learn an optimal target-domain policy (i.e., Step 13). The implementation details of transferring the reward function and the transition function are described in the following sections.

Reward function transferring In the source VIN, $f_R(s)$ maps observations of input states to reward images, and pass the reward images to the VI module. For example in the gird-world domain (Tamar et al. 2016), f_R can map an observation to a high reward at the goal, and negative reward

Algorithm 1 Transfer Value Iteration Algorithm

Initialize value function $V(s)$ with zeros
for epoch = 1, M **do**
  Set reward $R(s,a) = f_R(\phi(s), a; \theta)$
  for n = 1, K **do**
    Construct transition functions for each of the states:
    $P(s'|s,a) = \begin{cases} f_P^{new}(\phi(s), a; \theta), & a \in \mathcal{A}_{new} \\ \theta_t f_P^{pre}(\phi(s), a), & a \in \mathcal{A}_{transfer} \end{cases}$
    $Q_n(s,a;\theta) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V_n(s')$
    $V_{n+1}(s;\theta) = \max_a Q_n(s,a;\theta)$
  end for
  Construct optimal Q with $Q^*(s,a) = R(s,a) + \gamma \sum_{s'} P(s'|s,a) V^*(s')$
  Add attention vector $\psi(s;\theta)$ to the final policy $\pi_T(a|\psi(s);\theta)$
  Compute TVIN policy $\pi_T(a|s;\theta)$ with $\pi^*(s) = \arg\max_a Q^*(s,a)$
  Update θ by back-propagating the gradient according to (4)
end for

near an obstacle. If we directly adopt the pre-trained f_R from the source domain to the target domain, the reward function may be constrained to the task-specific features due to the diversity of pixel-level inputs. Therefore, for the target domain where the feature space is different from that of the source domain, we propose a feature mapping component to map states from different domains onto the same representation. Specifically, we encode the state observations in the target domain into the same representation as in the source domain by using an autoencoder (Zhuang et al. 2015). In this way, the reward function transferred from the pre-trained VIN is able to accurately produce a reward image for the target domain before being passed to the new VI module. In particular, we reuse the learned parameters of pre-trained reward function in source domain, and retrain an additional fully-connected layer acting as the feature encoder to output a shared representation for the input states s in the target domain. This feature encoder is trained in an end-to-end manner with the whole TVIN . The new reward function is denoted by $R(s,a) = f_R(s,a;\theta)$, where θ denotes all the parameters of the whole TVIN .

Transition function transferring To transfer the transition function across domains, we design a new VI module, which performs value iteration by approximating the Bellman-update through a CNN in the target domain. Specifically, the CNN used in the VI module is comprised of stacked convolution and max-pooling layers. The input to each convolution layer is a 3-dimensional signal X, typically, an image with l channels and $m \times n$ pixels. Its output h is a l'-channel convolution of the image with different kernels: $h_{l',i',j'} = \sigma\Big(\sum_{l,i,j} W^{l'}_{l,i,j} X_{l,i'-i,j'-j}\Big)$, where σ is an activation function. A max-pooling layer then down-samples the image by selecting the maximum value among some dimension. In this sense, each iteration in our new VI module can be approximately regarded as passing the reward R as well as the previous value function V_n through a convolution layer and max-pooling layer. As mentioned in (Tamar et al. 2016), each channel in the convolution layer corresponds to the Q-function for a specific action, and convolution kernel weights correspond to the discounted transition probabilities. Thus, we leverage the state transition values regarding the common subset of actions from the pre-trained model as a bridge of transition functions between the source domain and the target domain. At the high level, the new VI module divides the channels in the convolution layer into two parts: one corresponds to the Q-function for the common subset of actions and the other is for the new actions in the target domain.

For common actions between domains, some of them perform more similarly on both domains, while others may perform less similarly. To model the degree of similarity of actions between domains, we propose to add a learnable weight θ_t for each common action. The fine-tuned transition function in the target domain is defined by $P(s'|s,a) = \theta_t f_P^{pre}(\phi(s), a)$, if $a \in \mathcal{A}_{transfer}$, and $P(s'|s,a) = f_P^{new}(\phi(s), a)$, if $a \in \mathcal{A}_{new}$, where $\mathcal{A}_{transfer}$ is the common subset of actions, f_P^{pre} is adopted from the pre-trained transition function, $\mathcal{A}_{new}$ is the subset of domain-specific actions in the target domain, and f_P^{new} is learned from scratch with target domain data. When back-propagating the gradient through the TVIN in the target domain, we fix f_P^{pre} and only learn θ_t and f_P^{new}. To sum up, convolution kernel weights corresponding to the discounted transition probabilities in TVIN are computed based on two different cases:

$$W^a = \begin{cases} \theta_t W^a_{pre} & a \in \mathcal{A}_{transfer}; \\ W^a_{new} & a \in \mathcal{A}_{new}, \end{cases} \tag{1}$$

where W^a_{pre} stands for the pre-trained convolution kernel parameters corresponding to the discounted transition probabilities for the transferred (common) actions, θ_t stands for the transfer weight, and W^a_{new} stands for the new convolution kernel parameters corresponding to the discounted transition probabilities for new actions in target domain. The value function V is stacked with the reward R, and they are fed back into the convolutional layer, where each channel corresponds to the Q-function for a specific action. The convolution operation in new VI module is formulated as

$$Q_{a,i',j'} = \sum_{l,i,j} W^a_{l,i,j} R_{l,i'-i,j'-j} + \sum_{i,j} W^a_{l+1,i,j} V_{i'-i,j'-j}, \tag{2}$$

where i, j stand for the input state $s = (i,j)$, R is the reward and V is the value function in each iteration. These convolutional channels in both two parts are then max-pooled along all channels to produce the next-iteration value function layer V with $V_{n+1}(s;\theta) = \max_a Q_n(s,a;\theta)$. Performing value iteration for K times in this form, the new VI module outputs the approximate optimal value function $V^* = V_K$. The value iteration module in TVIN has an effective depth of K, which is larger than the depth of the well-known Deep Q-Network (Mnih et al. 2015). To reduce parameters for training process, we share the weights in the K recurrent layers in the TVIN .

After learning the internal transfer VI module which is independent to observations, we generate a pol-

icy for the input state s according to $\pi^*(s) = \arg\max_a R(s,a) + \gamma \sum_{s'} P(s'|s,a)V^*(s')$. Note that the transition $\sum_{s'} P(s'|s,a)V^*(s')$ only depends on a subset of the optimal value function V^*, if the states have a topology with local transition dynamics such as the grid-world application. Thus, we suppose that a local subset of s is sufficient for extracting information about the optimal TVIN plan.

Motivated by the wide use of attention mechanism (Xu et al. 2015) to improve learning performance by reducing the effective number of network parameters during training, in TVIN , we introduce an attention moduel to select the value of the current state after K iterations of value iteration. Intuitively, for a given label prediction (action), only a subset of the input features (value function) is relevant. The attention module can be represented by a parameterized function to output an attention modulated vector $\psi(s;\theta)$ for the input state s. And this vector is added as additional features to the TVIN to generate the final policy $\pi_T(\psi(s);\theta)$. By back-propagating through the whole network in an end-to-end manner, we update the joint parameters θ and learn the planning-based TVIN policy for the target domain.

Updating Parameters

By specifying the forms of the reward function f_R, the transition function f_P, and the attention function, and denoting the parameters of the whole TVIN by θ, we define the policy objective over the TVIN as the cross-entropy loss function between the expert policy and the current policy derived by TVIN . The TVIN can be trained by minimizing the loss function $\mathcal{L}(\theta)$,

$$\mathcal{L}(\theta) = \sum_{a \in \mathcal{A}} \pi_E(a|s) \log \pi_T(a|s;\theta), \tag{3}$$

where $\pi_T(a|s;\theta)$ is the TVIN policy parameterized by θ, and $\pi_E(a|s)$ is the expert policy for training data. To acquire training data, we can sample the expert to generate the trajectories used in the loss. In contrast to the deep reinforcement learning objective (Mnih et al. 2015) which recursively relies on itself as a target value, we use imitation learning (IL) (Giusti et al. 2016), which uses a stable training signal generated by an expert to guide the transfer network. Learning the TVIN policy then becomes an instance of supervised learning.

We consider the updates that optimize the policy parameter θ of the state representation, the reward function, and the new VI model. We update the θ towards the expert outcome. The gradient of the loss function with respect to the weights can be computed via

$$\nabla_\theta \mathcal{L}(\theta) = \sum_{a \in \mathcal{A}} \frac{\pi_E(a|s)}{\pi_T(a|s;\theta)} \nabla_\theta \pi_T(a|s;\theta). \tag{4}$$

We use the above gradient to update parameters by stochastic gradient descent (SGD) (Boyd and Vandenberghe 2004). In summary, the joint parameters θ of are updated to make the planning-integrated TVIN -based policy π_T more close to the expert policy π_E.

Experiments

Datasets and Criteria

Dataset The RL task domains for our experiments are synthetic 2D maps with randomly placed obstacles, in which observations include positions of agents, goal positions and the map configurations. Specifically, we use three different 2D environments similar to the GPPN experiments in (Lee et al. 2018) : the NEWS, the Moore and the Differential Drive. In NEWS, the agent can move {**East, West, North, South**}; in Differential Drive, the agent can move forward along its current orientation, or turn left/right by 90 degrees. The action space is {**Move forward, Turn left, Turn right**}; in Moore, the agent can move to any of the eight cells in its neighborhood. The action space of Moore is {**East, West, North, South, Northeast, Northwest, Southeast, Southwest**}. When considering knowledge transfer in the following experiments, we give the pairs of possible similar actions between different domains. Between NEWS and Differential Drive, the similar pairs are {(North, Move forward), (East, Turn left), (West, Turn right)}. Between NEWS and Moore, the similar pairs are {(East, East), (West, West), (North, North), (South, South)}.

In the experiments on the above three domains, the state vectors given as input to the models consist of the maps and the goal location. In NEWS and Moore, the target is an x-y coordinate. Similar to the experimental setup in (Tamar et al. 2016), we produce a $(2 \times m \times m)$-sized observation image for each state $s = (i, j)$ in each trajectory, where m is the maze size. The first channel of the image encodes the obstacle configuration (1 for obstacle, 0 otherwise), while the second channel encodes the goal position (1 at the goal, 0 otherwise). The full state observation vector consists of the observation image and the state $s = (i, j)$. While in Differential Drive, the goal location contains an orientation along with the x-y coordinate. Consequently, the dimension of the goal map given as input to the models is $4 * m * m$ in Differential Drive. In addition, for each state, we produce a ground-truth label encoding the action that an optimal shortest-path policy would take in that state. Experimentally, our ground-truth label is created with a maze generation process that uses depth-first search with the recursive back-tracker algorithm (Cormen et al. 2009).

Criteria In the following experiments, we empirically compare TVIN and VIN using two metrics referred to (Lee et al. 2018) : %Optimal (%Opt) is the percentage of states whose predicted paths under the policy estimated by the model has optimal length. %Opt is denoted by:

$$\%Opt = \frac{Num(a_i = a_i^*)}{S_{test}},$$

where S_{test} represents the total number of states in test set, a_i^* represents the optimal action for state s_i, and a_i is the action prediction generated by models for s_i. The second metric %Success (%Suc) is the percentage of states whose predicted paths under the policy estimated by the model reach the goal state. A trajectory is said to succeed if it reached the goal without hitting obstacles. Let N_{test} denote the total number

Source		NEWS-9						NEWS-15						NEWS-28					
Target		Moore-9		Moore-15		Moore-28		Moore-9		Moore-15		Moore-28		Moore-9		Moore-15		Moore-28	
N	Model	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc
1k	VIN	84.2	87.7	77.3	81.7	56.2	65.8	84.2	87.7	77.3	81.7	56.2	65.8	84.2	87.7	77.3	81.7	56.2	65.8
1k	TVIN	**89.8**	**94.2**	**88.3**	**91.0**	**66.7**	**74.7**	**94.6**	**96.6**	**90.1**	**92.8**	**66.1**	**75.3**	**94.3**	**95.8**	**86.4**	**89.1**	**62.1**	**71.1**
5k	VIN	90.5	92.5	86.7	88.7	64.3	72.9	90.5	92.5	86.7	88.7	64.3	72.9	90.5	92.5	86.7	88.7	64.3	72.9
5k	TVIN	**97.0**	**98.0**	**93.8**	**94.9**	**80.4**	**86.3**	**97.1**	**97.2**	**95.2**	**96.0**	**73.4**	**84.3**	**97.8**	**98.2**	**91.1**	**92.6**	**76.2**	**84.3**
10k	VIN	86.2	88.0	91.1	92.3	60.8	68.0	86.2	88.0	91.1	92.3	60.8	68.0	86.2	88.0	91.1	92.3	60.8	68.0
10k	TVIN	**97.6**	**97.8**	**95.4**	**96.2**	**83.1**	**88.3**	**97.4**	**97.5**	**96.2**	**96.7**	**87.8**	**91.8**	**96.6**	**96.8**	**92.5**	**93.7**	**78.7**	**84.0**

Table 1: Transfer from NEWS to Moore with varying dataset sizes N and maze sizes M.

Source		Moore-9						Moore-15						Moore-28					
Target		NEWS-9		NEWS-15		NEWS-28		NEWS-9		NEWS-15		NEWS-28		NEWS-9		NEWS-15		NEWS-28	
N	Model	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc
1k	VIN	77.8	81.0	69.3	71.1	45.6	51.9	77.8	81.0	69.3	71.1	45.6	51.9	77.8	81.0	69.3	71.1	45.6	51.9
1k	TVIN	**94.7**	**94.8**	**85.5**	**86.8**	**69.1**	**71.6**	**94.8**	**94.9**	**96.3**	**96.4**	**89.2**	**89.4**	**82.0**	**84.0**	**73.1**	**75.0**	**64.0**	**67.7**
5k	VIN	79.8	81.9	70.7	73.5	57.8	60.9	79.8	81.9	70.7	73.5	57.8	60.9	79.8	81.9	70.7	73.5	57.8	60.9
5k	TVIN	**95.0**	**95.0**	**88.6**	**89.4**	**75.1**	**77.8**	**97.1**	**97.1**	**96.5**	**96.6**	**93.0**	**93.1**	**85.1**	**86.7**	**77.3**	**80.3**	**65.1**	**68.1**
10k	VIN	87.1	88.4	88.1	88.4	58.4	61.5	87.1	88.4	88.1	88.4	58.4	61.5	87.1	88.4	88.1	88.4	58.4	61.5
10k	TVIN	**96.6**	**96.6**	**89.3**	**90.0**	**80.1**	**82.2**	**97.4**	**97.4**	**97.0**	**96.9**	**94.4**	**94.5**	**91.7**	**92.5**	**88.7**	**89.6**	**68.4**	**72.9**

Table 2: Transfer from Moore to NEWS with varying dataset sizes N and maze sizes M.

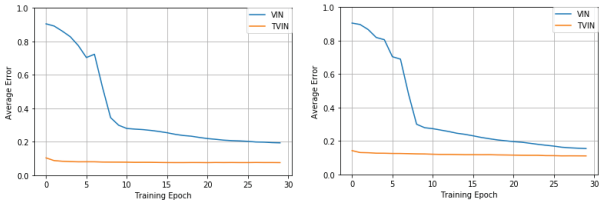

Figure 2: Training process on Moore-15 with 1k training data transferred from NEWS-9 compared with VIN. Left: domains of 30% obstacles. Right: domains of 50% obstacles.

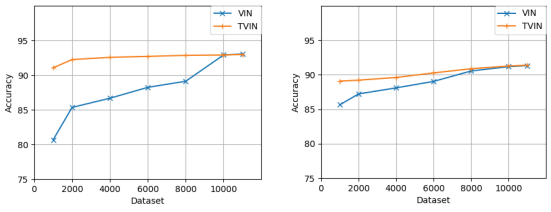

Figure 3: Prediction accuracy on Moore-15 with varying dataset sizes transferred from NEWS-9 compared with VIN. Left: domains of 30% obstacles. Right: domains of 50% obstacles.

of test trajectories, $\mathcal{T}_{goal}$ represents the goal state of trajectory $\mathcal{T}$ and $\mathcal{T}_{end}$ represents the end state of the trajectory predicted by the models. Then %Suc can be denoted by:

$$\%Suc = \frac{Num(\mathcal{T}_{end} = \mathcal{T}_{goal})}{N_{test}}.$$

Experimental Results

Our experiments attempt to transfer policies between 2D domains with different environments and maze sizes. We evaluate our TVIN approach in the following aspects:

1. We first evaluate TVIN between different domains, including transfer from NEWS to Moore, transfer from Moore to NEWS and transfer from Differential Drive to NEWS. Additionally we vary the maze sizes in each domains, dataset sizes in the target domains, etc., to see the performance of TVIN when only limited training data is available.
2. We then evaluate TVIN approach on hyperparameter sensitivity, including the iteration count K and the kernel size F. Experiments show that TVIN can indeed perform better than single VIN and does not rely on the setting of these hyperparameters.
3. We finally evaluate TVIN by varying the amount of pre-trained knowledge transferred from the source domain, which is characterized by the number of transferable actions between source and target domains. We aim to see the impact of the amount of transferred knowledge.

In 2D domains, an optimal policy can be calculated by exact value iteration algorithm. And the pre-trained VIN represented by a neural network has been proved to learn planning results. However, for these different tasks of similar complexity and sharing similar actions, TVIN can greatly accelerate training process as well as improving the performance of training by leveraging learned knowledge and by reducing the learning expense of parameters.

Source		Drive-9						Drive-15						Drive-28					
Target		NEWS-9		NEWS-15		NEWS-28		NEWS-9		NEWS-15		NEWS-28		NEWS-9		NEWS-15		NEWS-28	
N	Model	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc
1k	VIN	77.8	81.0	69.3	71.1	45.6	51.9	77.8	81.0	69.3	71.1	45.6	51.9	77.8	81.0	69.3	71.1	45.6	51.9
1k	TVIN	**86.7**	**88.1**	**70.2**	**72.2**	**49.2**	**52.6**	**80.0**	**81.4**	**83.4**	**84.6**	**63.9**	**68.7**	**78.3**	**81.3**	**72.8**	**74.8**	**57.5**	**59.8**
5k	VIN	79.8	81.9	70.7	73.5	57.8	60.9	79.8	81.9	70.7	73.5	57.8	60.9	79.8	81.9	70.7	73.5	57.8	60.9
5k	TVIN	**88.0**	**88.8**	**83.7**	**86.0**	**84.1**	**84.9**	**86.0**	**86.8**	**93.4**	**93.6**	**91.9**	**92.1**	**81.9**	**84.4**	**85.2**	**85.1**	**78.8**	**80.5**
10k	VIN	87.1	88.4	88.1	88.4	58.4	61.5	87.1	88.4	88.1	88.4	58.4	61.5	87.1	88.4	88.1	88.4	58.4	61.5
10k	TVIN	**92.8**	**93.3**	**92.9**	**93.1**	**91.2**	**91.5**	**90.9**	**91.7**	**94.2**	**94.3**	**93.3**	**93.4**	**89.5**	**90.7**	**95.5**	**95.5**	**92.5**	**92.5**

Table 3: Transfer from Drive to NEWS with varying dataset sizes N and maze sizes M.

Accuracy w.r.t. domains Based on these guidelines, we evaluate several instances of knowledge transfer, i.e., from NEWS to Moore, from Moore to NEWS and from Differential Drive to NEWS. For each transfer, we compare TVIN policy to the VIN reactive policy. Additionally we vary the maze sizes in each domains and dataset sizes in the target domains. Note that, K is required to be chosen in proportion to the maze size. In the implementation, we refer to (Lee et al. 2018) and set the default recurrence K relative to the maze sizes: $K = 20$ for 9×9 mazes, $K = 30$ for 15×15 mazes and $K = 56$ for 28×28 mazes. Results are respectively reported in Table 1, Table 2, and Table 3, showing that our transfer learning approach TVIN provides a definite increase in accuracy when we have limited data in the target domain. Even compared to the standard reactive networks DQN of the success rate 74.2% on Moore-28 with full dataset which is shown in (Tamar et al. 2016), TVIN can reach the success rate of 84.3% (Table 1), outperforming DQN only with 5k training data in the same case. Additionally, training process of the TVIN and VIN on 1k training data of Moore-15 is depicted in Figure 2. It also shows that knowledge transfer by TVIN speeds up learning process and reaches a higher generalization.

Accuracy w.r.t. transfer methods As shown in Table 4, we make comparison with a simple transfer method denoted by VIN_i. VIN_i is a heuristic transfer method (Parisotto, Ba, and Salakhutdinov 2016) by directly leveraging pre-trained weights of f_R and part of f_p (with respect to similar actions) as the initialization for training in the target domain. Taking the experiments between NEWS-15 to MOORE for example, the results show that heuristic transfer by VIN_i give useful pre-trained information, compared to training from scratch. Moreover, the TVIN policy learned in target domain performs much better than heuristic transfer VIN_i, which shows that our transfer strategies are effective and applicable.

Accuracy w.r.t. planning complexity The complexity of planning in the 2-D maze domains generally depends on the number of obstacles and their distribution on the grid map. We thus synthesize domains based on different number of obstacles and different size of the grid map. In this experiments, We compare two complexity, which are 30 percent and 50 percent. It means 30 percent or 50 percent of the map is randomly placed with obstacles. Although we evaluate our approach on these 2-D domains, we should note that many real-world application domains, such as *navigations*, *warehouse scheduling*, etc. can be matched to 2-D maze domains with different complexity, and thus such evaluation in these domains should be convincing.

Source		NEWS-15					
Target		Moore-9		Moore-15		Moore-28	
N	Model	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc
1k	VIN	84.2	87.7	77.3	81.7	56.2	65.8
1k	VIN_i	92.8	94.9	88.6	91.2	65.2	74.6
1k	TVIN	**94.6**	**96.6**	**90.1**	**92.8**	**66.1**	**75.3**
5k	VIN	90.5	92.5	86.7	88.7	64.3	72.9
5k	VIN_i	96.2	96.1	94.2	95.4	71.9	80.9
5k	TVIN	**97.1**	**97.2**	**95.2**	**96.0**	**73.4**	**84.3**
10k	VIN	86.2	88.0	91.1	92.3	60.8	68.0
10k	VIN_i	96.1	96.3	95.0	95.5	84.6	90.4
10k	TVIN	**97.4**	**97.5**	**96.2**	**96.7**	**87.8**	**91.8**

Table 4: Policy performance compared with simple transferred VIN_i and TVIN

In this experiment, we view 9×9 NEWS as source domains, and transfer pre-trained knowledge to 15×15 Moore. We investigate the transfer performance with respect to different complexity. The results are show in Figure 2 and Figure 3, where the left one shows the transfer between domains of 30 percent obstacles, and the right one is the transfer between domains of 30 percent obstacles. In both cases, adjusting weights of the transferred knowledge in TVIN can indeed outperform the mechanism of randomly initializing VIN. It illustrates that TVIN planning policies, by our transfer strategies, are technically effective either in simple environment or complex environment. The performance gap between transfer learning policy TVIN and original VIN policy is more significant in low complexity domain, whereas in high complexity domains the gap between TVIN and VIN is comparatively slight. Furthermore, the difference in the performance gap shows that it is more challenging for TVIN to leverage the pre-trained knowledge when the complexity of planning is much higher.

Accuracy w.r.t. dataset sizes To evaluate the objective on transfer learning, we compare the performance of TVIN model by using different size of dataset. As is illustrated

Model	K = 10		K = 20		K = 30	
	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc
VIN	70.3	78.3	67.7	77.0	64.7	74.5
TVIN	**78.0**	**85.8**	**81.6**	**90.8**	**80.1**	**91.8**

Table 5: Test performance on Moore-15 transferred from NEWS-9 with varying iteration counts K.

Model	F = 3		F = 5		F = 7	
	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc
VIN	64.7	74.5	77.3	81.7	77.8	83.1
TVIN	**80.1**	**91.8**	**88.3**	**91.0**	**85.3**	**88.9**

Table 6: Test performance on Moore-15 transferred from NEWS-9 with varying kernel sizes F.

in Table 1, Table 2 and Table 3, the size of training data on target domain influences the performance of TVIN . Prediction accuracy with varying training data in target domain is also depicted in Figure 4. It shows that, in each case, TVIN can indeed outperform the mechanism of randomly initializing VIN. Although the performance gap decreases gradually with the dataset size increasing, the performance of TVIN turns out to be significantly greater than VIN when there is limited data in the target domain. This shows that if there is already sufficient data for a novel domain to learn optimal policies, information transferred from the source domain would not help improve the performance a lot. Rather, our transfer strategies focus more on generating planning-based TVIN policies for a target domain with limited dataset.

Accuracy w.r.t. hyperparameters Following the above results that TVIN performs better or equals to VIN, we further evaluate the effect of varying both iteration count K and kernel size F on the TVIN models. Table 5 and Table 6 show $\%Opt$ and $\%Suc$ results of TVIN and VIN on Moore-15 for different values of F and K, and we use NEWS-9 as the source domain. This shows that TVIN outperforms VIN even when hyperparameters such as iteration count K and kernel size F are set differently in the target domains. Although in VINs, larger mazes require larger kernel sizes and iteration counts, the performance gap between TVIN and single VIN do not rely on a specific choice of hyperparameters.

Accuracy w.r.t. transferred knowledge Finally, we evaluate the influence of the number of transferable actions between source and target domains in TVIN . The more actions are transferred, the more knowledge is leveraged in target domain. Table 7 shows results for different numbers of transferable actions between the source domain (NEWS-9) and the target domain (Moore-15) with 1k training data. It is illustrated that the more similar actions to transfer, the better performance for target TVIN to gain.

Actions	Num = 1		Num = 2		Num = 3		Num = 4	
	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc	%Opt	%Suc
VIN	77.3	81.7	77.3	81.7	77.3	81.7	77.3	81.7
TVIN	82.0	86.1	82.2	86.5	86.2	90.9	**88.3**	**91.0**

Table 7: Test performance on target domain Moore-15 transferred from the source domain NEWS-9 with varying number of transferred actions.

Related Work

In Reinforcement Learning (RL), the agent act in the world and learn a policy from trial and error. RL algorithms in (Sutton and Barto 2005; Schulman et al. 2015; Levine et al. 2016) use these observations to improve the value of the policy. Recent works investigate policy architectures that are specifically tailored for planning under uncertainty. VINs (Tamar et al. 2016) take a step in this direction by exploring better generalizing policy representations. The Predictron (Silver et al. 2017), Value Prediction Network (Oh, Singh, and Lee 2017) also learn value functions end-to-end using an internal model, implemented with recurrent neural networks (RNNs) (Mikolov et al. 2010) acting as the transition functions over abstract states. However, none of these abstract planning-based models have been considered for transfer. Our work investigates the generalization properties of the pre-trained policy and proposes the TVIN model for knowledge transfer.

A wide variety of methods have also been studied in the context of RL transfer learning(Taylor and Stone 2009). Policy distillation (Hinton, Vinyals, and Dean 2015; Chen et al. 2017) aims to compress the capacity of a deep network via efficient knowledge transfer . It has been successfully applied to deep reinforcement learning problems (Rusu et al. 2016). Recently, successor features and generalised policy improvement, has been introduced as a principled way of transferring skills (Barreto et al. 2018). Also (Abel et al. 2018) considers value-function-based transfer in RL. However the key to our approach is that the Q-functions for specific actions learned from the source domain can be transferred to the corresponding VI module in the target domain. we also build a mapping between feature spaces in the source and target domains, transfer Q-networks related to *similar actions* from the source to the target domain and build policy networks for *dissimilar* actions which are learned from scratch.

Conclusions

We propose a novel transfer learning approach TVIN to learn a planning-based policy for the target domain with different feature spaces and action spaces by leveraging pre-trained knowledge from source domains. In addition, we exhibit that such a transfer network TVIN leads to better performance when the training data is limited in the target domain. In this paper we assume the pairs of possible similar actions is provided beforehand. In the future, it would be interesting to exactly learn the action similarities based on Web search (Zhuo et al. 2011; Zhuo and Yang 2014) or language model learning (Tian, Zhuo, and Kambhampati 2016;

Feng, Zhuo, and Kambhampati 2018) before employing the transfer method.

Acknowledgement

Hankz H. Zhuo thanks the support of the National Natural Science Foundation of China (U1611262), Guangdong Natural Science Funds for Distinguished Young Scholar (2017A030306028), Guangdong special branch plans young talent with scientific and technological innovation, Pearl River Science and Technology New Star of Guangzhou, Key Laboratory of Machine Intelligence and Advanced Computing (Sun Yat-Sen University) of Ministry of Education of China, and Guangdong Province Key Laboratory of Big Data Analysis and Processing for the support of this research. Sinno J. Pan thanks the support from NTU Nanyang Assistant Professorship (NAP) grant M4081532.020.

References

Abel, D.; Jinnai, Y.; Guo, S. Y.; Konidaris, G.; and Littman, M. L. 2018. Policy and value transfer in lifelong reinforcement learning. In *ICML*, 20–29.

Barreto, A.; Borsa, D.; Quan, J.; Schaul, T.; Silver, D.; Hessel, M.; Mankowitz, D. J.; Zídek, A.; and Munos, R. 2018. Transfer in deep reinforcement learning using successor features and generalised policy improvement. In *ICML*, 510–519.

Boyd, S., and Vandenberghe, L. 2004. *Convex Optimization*. New York, NY, USA: Cambridge University Press.

Chen, G.; Choi, W.; Yu, X.; Han, T. X.; and Chandraker, M. 2017. Learning efficient object detection models with knowledge distillation. In *NIPS*, 742–751.

Cormen, T. H.; Leiserson, C. E.; Rivest, R. L.; and Stein, C. 2009. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition.

Feng, W.; Zhuo, H. H.; and Kambhampati, S. 2018. Extracting action sequences from texts based on deep reinforcement learning. In *IJCAI*, 4064–4070.

Giusti, A.; Guzzi, J.; Ciresan, D. C.; He, F.; Rodriguez, J. P.; Fontana, F.; Faessler, M.; Forster, C.; Schmidhuber, J.; Caro, G. D.; Scaramuzza, D.; and Gambardella, L. M. 2016. A machine learning approach to visual perception of forest trails for mobile robots. *IEEE Robotics and Automation Letters* 1(2):661–667.

Gupta, S.; Davidson, J.; Levine, S.; Sukthankar, R.; and Malik, J. 2017. Cognitive mapping and planning for visual navigation. In *CVPR*.

Hinton, G.; Vinyals, O.; and Dean, J. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531.

Lee, L.; Parisotto, E.; Chaplot, D. S.; Xing, E.; and Salakhutdinov, R. 2018. Gated path planning networks. In *ICML*.

Levine, S.; Finn, C.; Darrell, T.; and Abbeel, P. 2016. End-to-end training of deep visuomotor policies. *Journal of Machine Learning Research* 17(1):1334–1373.

Mikolov, T.; Karafiát, M.; Burget, L.; Cernocký, J.; and Khudanpur, S. 2010. Recurrent neural network based language model. In *INTERSPEECH*, 1045–1048.

Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A. A.; Veness, J.; Bellemare, M. G.; Graves, A.; Riedmiller, M. A.; Fidjeland, A.; Ostrovski, G.; Petersen, S.; Beattie, C.; Sadik, A.; Antonoglou, I.; King, H.; Kumaran, D.; Wierstra, D.; Legg, S.; and Hassabis, D. 2015. Human-level control through deep reinforcement learning. *Nature* 518(7540):529–533.

Nogueira, R., and Cho, K. 2016. End-to-end goal-driven web navigation. In *NIPS*, 1903–1911.

Oh, J.; Singh, S.; and Lee, H. 2017. Value prediction network. In *NIPS*, 6120–6130.

Parisotto, E.; Ba, J. L.; and Salakhutdinov, R. 2016. Actor-mimic: Deep multitask and transfer reinforcement learning. In *ICLR*.

Rusu, A. A.; Colmenarejo, S. G.; Gulcehre, C.; Desjardins, G.; Kirkpatrick, J.; Pascanu, R.; Mnih, V.; Kavukcuoglu, K.; and Hadsell, R. 2016. Policy distillation. In *ICLR*.

Schulman, J.; Levine, S.; Moritz, P.; Jordan, M. I.; and Abbeel, P. 2015. Trust region policy optimization. In *ICML*, 1889–1897.

Silver, D.; van Hasselt, H.; Hessel, M.; Schaul, T.; Guez, A.; Harley, T.; Dulac-Arnold, G.; Reichert, D. P.; Rabinowitz, N. C.; Barreto, A.; and Degris, T. 2017. The predictron: End-to-end learning and planning. In *ICML*, 3191–3199.

Sutton, R. S., and Barto, A. G. 2005. Reinforcement learning: An introduction. *Machine Learning* 16(1):285–286.

Tamar, A.; Levine, S.; Abbeel, P.; Wu, Y.; and Thomas, G. 2016. Value iteration networks. In *NIPS*, 2146–2154.

Taylor, M. E., and Stone, P. 2009. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research* 10(10):1633–1685.

Tian, X.; Zhuo, H. H.; and Kambhampati, S. 2016. Discovering underlying plans based on distributed representations of actions. In *AAMAS*, 1135–1143.

Tsitsiklis, J., and Roy, B. V. 2002. An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control* 42(5):674–690.

Xu, K.; Ba, J.; Kiros, R.; Cho, K.; Courville, A. C.; Salakhutdinov, R.; Zemel, R. S.; and Bengio, Y. 2015. Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, 2048–2057.

Zhuang, F.; Cheng, X.; Luo, P.; Pan, S. J.; and He, Q. 2015. Supervised representation learning: Transfer learning with deep autoencoders. In *IJCAI*, 4119–4125.

Zhuo, H. H., and Yang, Q. 2014. Action-model acquisition for planning via transfer learning. *Artificial Intelligence* 212:80–103.

Zhuo, H. H.; Yang, Q.; Pan, R.; and Li, L. 2011. Cross-domain action-model acquisition for planning via web search. In *ICAPS*.