Learning Visual Planning Models from Partially Observed Images

Zhanhao Xiao [1,2*], Kebing Jin [1,3*], Hankui Hankz Zhuo [1†] Hai Wan [1], Jiaran Cai [1],

[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 510006, P.R.China
[2]Guangdong Polytechnic Normal University, Guangzhou 510665, P.R.China
[3]State Key Laboratory of Public Big Data, Guizhou University, Guizhou 550025, P.R.China

Abstract

There has been increasing attention on planning model learning in classical planning. Most existing approaches, however, focus on learning planning models from structured data in symbolic representations. It is often difficult to obtain such structured data in real-world scenarios. Although a number of approaches have been developed for learning planning models from fully observed unstructured data (e.g., images), in many scenarios raw observations are often incomplete. In this paper, we provide a novel framework, `Recplan`, for learning a transition model from partially observed raw image traces. More specifically, by considering the preceding and subsequent images in a trace, we learn the latent state representations of raw observations and then build a transition model based on such representations. Additionally, we propose a neural network-based approach to learn a heuristic model that estimates the distance towards a given goal observation. Based on the learned transition model and heuristic model, we implement a classical planner for images. We exhibit empirically that our approach is more effective than a state-of-the-art approach to learning visual planning models in an environment with incomplete observations.

Introduction

Domain-independent classical planning has been applied in a growing number of real-world scenarios. It requires planning models to describe how the environment is changed by actions and in what conditions an action can be executed (Jin et al. 2022). Since creating planning models by hand is often time-consuming and resource-demanding, even for experts, automatically learning planning models from data has drawn increasing attention from researchers (Aineto, Celorrio, and Onaindia 2019). Most existing approaches focus on learning planning models from data in structured and symbolic representations (Yang, Wu, and Jiang 2007; Zhuo, Muñoz-Avila, and Yang 2014), such as Planning Domain Definition Language (PDDL), in order to take advantage of off-the-shelf efficient planners to compute plans. In many real-world scenarios, such as camera-based security monitoring, it is often difficult to acquire structured data for learning planning models, since the world states are described by images that are unstructured (i.e., represented by pixels). It is challenging to learn planning models from unstructured raw data and conduct planning based on unstructured initial states and goal states since we need to consider the large feature space of raw data (Asai and Muise 2020).

Recently, Asai and Fukunaga (2018) proposed a neural-symbolic approach, called `Latplan`, for domain-independent image-based classical planning. It learns latent (propositional or continuous) representations of images from a set of fully observed image transition pairs, and a transition function based on the learned representations. `Latplan` is applicable under the condition that each world state is fully observed in the form of an image, which in many real-world scenarios cannot be guaranteed. On the contrary, it is more common that the world states in the form of images are partially observed. For example, an object may be occluded by an obstacle from the view of the camera, which results in images with partial observation. Partial observation means that we need to handle the uncertainty of the world states, i.e., there exist different possibilities in the unobserved regions, which makes the model learning task even more challenging. While `Latplan` focuses on image transition pairs, it is sometimes impossible to entail the real information which is missing. Figure 1 shows an example in the 8-puzzle domain to illustrate the motivation. There are a number of possible cases satisfying the transition pair when the three tiles in the first row are missing, two of which are shown in the second and third levels in Figure 1. It is difficult to determine which of the two possible cases is true by only looking at the transition pair of the partially observed images, as `Latplan` does. Whereas, these possible cases may be implied by having some future image observation. It suggests to handle the learning task from the perspective of a whole trace rather than a transition pair. Furthermore, the data is neither symbolic nor structured, which implies that symbolic rules are hardly likely to be learned without human knowledge. That is, it cannot be conjectured that the missing region only includes three numbers: “0”, “4” and “7”. It is thus challenging to correctly recover the unobserved information and learn a planning model simultaneously.

In classical planning, it is satisfied that parts of world states unrelated to action execution will always remain unchanged. Taking the example in Figure 1, if we observed that “4” occurs in the top right corner of the third image observation

*Zhanhao Xiao and Kebing Jin contribute equally to this work.

†Corresponding author.

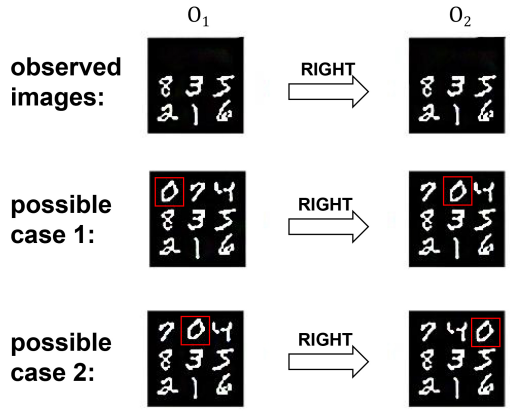

Figure 1: A transition pair of masked images in the 8-puzzle domain. In each configuration, only the tile "0" can be exchanged with one of its adjacent tiles. An image observation o_1 is transformed into another image observation o_2 by executing action "RIGHT". When the first row is masked, at least two possible cases satisfy these observations.

and the next action is "DOWN", we could conjecture that "4" also stays there in o_2, excluding the possible case 2. In other words, to rule out possibilities, we are supposed to leverage the information about the previous and subsequent image observations. It thus motivates us to explore the underlying relations due to actions to help reason about the missing parts by taking a whole trace of image observations into account.

In this paper, we propose a framework `Recplan` based on recurrent neural network (RNN), aiming to recover the missing information in the image observation traces and to learn a transition function simultaneously. More specifically, we first learn a prediction model to estimate the unchanged pixels and then complement the missing regions in the original image observations. Following that, we learn a bidirectional mapping between image observations and latent states via an RNN which takes a whole trace as input. Meanwhile, we train a transition model in terms of the latent states. During the training phase, the transition model in RNN takes not only the current state but also the history information as input, in order to reduce the possibilities due to the failure of observation. Furthermore, we present a neural-network-based approach to learning a heuristic model in terms of latent states in order to estimate the distance to the goal state. By conducting experiments on the three domains, we show that our approach is more effective in learning visual planning models in the environment with incomplete observations, compared with `Latplan`. We also show that our heuristic model is helpful in computing optimal plans.

This paper is organized as follows. We first review previous work related to our approach in the section and then formulate the planning problem. After that, we present our framework `Recplan` in detail and evaluate `Recplan` in three domains with a comparison to `Latplan`. Finally, we conclude our paper and address future work.

Related Work

Visual Planning

In order to handle goal-directed planning problems with raw observations, there have been efforts in visual planning recently. Some researchers focus on applying learning-based approaches to robotic manipulation directly from raw observations, while some pay attention to learning planning models and state representation from image traces.

Many approaches have borrowed deep learning methods to tackle complex visual planning problems, which aim at generating an action sequence to reach a given goal observation from a given initial observation. For example, (Nair et al. 2017) proposed to learn a model to manipulate deformable objects. Recently, Causal InfoGAN (Kurutach et al. 2018) was proposed to learn deformable objects from sequential data and generate goal-directed trajectories. In 2019, Hafner et al. gave a planning framework PlaNet that learns the environment dynamics from images and chooses actions through predicting the rewards ahead for multiple time steps. To compute goal-directed plans, SPTM (Liu et al. 2020) was proposed. It builds a connectivity graph where each observation is considered a node. Besides, HVF (Nair and Finn 2020) was presented to compute plans by decomposing a visual goal into a sequence of subgoals. However, the above approaches suppose a fully observable environment where each observation is completely obtained, which in fact is a demanding requirement.

Researchers also show their interest in representation learning from image sequences. First of all, Asai and Fukunaga proposed `Latplan` (Asai and Fukunaga 2018) and opened the door to image-based domain-independent classical planning, which bridges the gap between deep learning perceptual systems and symbolic classical planners. Later (Asai 2019) offered an approach to obtain first-order logic representation from images, which is plannable with human knowledge. Recently, based on `Latplan`, (Asai and Muise 2020) proposed to learn planning models that are neural networks restricted to cube-like graphs, which increases significantly planning accuracy. Compared with the series of works about `Latplan`, we in this paper assume that images are partially observed. In consequence, we require that action labels should be given in the training data. If we followed their unlabeled assumption, the learning task would become intractable as there is little information to reason. Another difference lies in that we take the whole image traces as training data while they adopt a set of image transition pairs.

Inpainting

Our paper is also related to the work of image inpainting and video inpainting. Image inpainting is rooted in (Bertalmío et al. 2000), which aims to fill corrupted regions of images with fine-detailed contents. Existing inpainting methods can be classified into two categories: classical approaches and deep learning-based approaches. The classical approaches either replace incomplete regions with surrounding textures and apply a diffusive process (Roth and Black 2005), or fill holes by searching similar patches from the same image or external image databases (He and Sun 2012). However, such kinds of approaches fail to capture semantical content and only work on the images with simple and repeated textures. The recent success of deep learning approaches has irradiated such a conventional image process task and has inspired a series of deep learning-based approaches. For example, the

first attempt is Context Encoder (Pathak et al. 2016) which uses a deep convolution encoder-decoder. It has inspired a series of works that extend Context Encoder in different ways, such as (Yan et al. 2018), etc. Recently, researchers have paid more attention to exploiting image structure knowledge for inpainting (Xiong et al. 2019; Yang, Qi, and Shi 2020). However, existing image inpainting approaches only focus on an image itself without analyzing images on a transition pair or a trace. It leads that these approaches still suffer the issue in Figure 1, failing to eliminate possible cases.

Planning Model Learning

Planning model learning has attracted a lot of attention and there exist a number of approaches (Arora et al. 2018). The seminal learning approach from partially observed plan traces should be ARMS (Yang, Wu, and Jiang 2007), which invokes a MAX-SAT solver to obtain planning models. Besides, there is a large body of work from partially observed plan traces (LAWS (Zhuo et al. 2011), TRAMP (Zhuo and Yang 2014), NLOCM (Gregory and Lindsay 2016), DUP (**?**)), PELA (Martínez et al. 2016)), from state pairs (FAMA (Aineto, Celorrio, and Onaindia 2019)) and from labeled graphs (Bonet and Geffner 2020). Whereas, the above approaches require to structured symbolic training data, which leads a difficult and time-consuming task.

Heuristics Learning in Classical Planning

Our paper also relates to the work of learning heuristics in classical planning via machine learning techniques (Arfaee, Zilles, and Holte 2010; Shen, Trevizan, and Thiébaux 2020; Trunda and Barták 2020). It is also related to learning control knowledge (Yoon, Fern, and Givan 2008) and learning search policies (Gomoluch et al. 2020) in classical planning. These approaches are applicable to descriptive planning models. While we learn a transition model on the latent state of images, these approaches become inapplicable. On the other hand, Asai and Muise (2020) recently presented to learn descriptive planning models from images, which allows applying state-of-the-art heuristics planners. However, the approach is still only applicable to fully observed images.

Transition Model Learning

In this section, we first formulate the problem of transition model learning and then present our framework `Recplan` that includes: (i) an inpainting model to complete masked image observations, (ii) a State Autoencoder (SAE) to embed a completed image into a latent propositional state, (iii) a transition model to update a latent state by an action.

Problem Formulation

Let $O^i = \langle o_0^i, o_1^i, ..., o_n^i \rangle$ be a sequence of image observations, $A^i = \langle a_1^i, a_2^i, ..., a_n^i \rangle$ be a sequence of actions and $M^i = \langle m_0^i, m_1^i, ..., m_n^i \rangle$ be a sequence of mask matrices with the same size of image observations. The superscript i is an identifier of a sequence and the subscript indexes elements in a sequence. Intuitively, the image observation o_j^i results from executing the action a_j^i on the image observation o_{j-1}^i, according to an underlying transition function. In every mask matrix, value "1" denotes an observed pixel while value "0" denotes a masked one.

We define the image transition model learning problem as a tuple $(\{O^i\}, \{M^i\}, \{A^i\})$, which aims to learn a transition model $\hat{\gamma}$ to approximate the ground-truth transition function γ. Formally, given any unmasked image o and an applicable action a, $\hat{\gamma}(o, a) \approx \gamma(o, a)$.

remark 1 *Given the success of occlusion detection approaches (Suresh, Chitra, and Deepak 2013; Askar et al. 2020), which can be used to detect the masks in the given images, we believe it is reasonable to incorporate mask matrices as an input of the learning problem. In other words, it is not difficult to obtain mask matrices of raw observations from a realistic perspective.*

Observation Inpainting

Considering there exist strong relations between each two continuous image observations, i.e., the latter image observation is the result of applying an action to the former image observation, we aim to exploit those relations to help "complement" the missing parts of the image observations. In classical planning, only direct effects are considered: each pixel in the image remains unchanged until it is affected by actions. Based on such an assumption, we propose to learn a prediction function ρ that takes two neighbor image observations o_{j-1}^i, o_j^i in a sequence and the connecting action a_j^i as input, and outputs a matrix $\hat{p}_j^i$ with the same size of the image observation. Every element in $\hat{p}_j^i$ is a real number between -1 and 1, which means to be a confidence on whether the corresponding pixel in o_j^i changes or not. When it is closer to 1, the corresponding pixel will be more likely to be unchanged in the original image of the observation o_j^i.

Indubitably, it is a semi-supervised task: the pixels in each image observation that are observed to be unchanged are labeled as unchanged; those unobserved pixels are unknown. Then we construct the training data as a set of label matrix sequences $\{\langle p_1^i, p_2^i, ..., p_n^i \rangle\}$. Formally, for an element (x, y) in a label matrix p_j^i, denoted by $p_j^i(x, y)$, its value is defined as follows:

- If $o_{j-1}^i(x, y) = o_j^i(x, y),\ m_j^i(x, y) = m_{j-1}^i(x, y) = 1$, then $p_j^i(x, y) = 1$;
- if $o_{j-1}^i(x, y) \neq o_j^i(x, y),\ m_j^i(x, y) = m_{j-1}^i(x, y) = 1$, then $p_j^i(x, y) = -1$;
- otherwise $p_j^i(x, y) = 0$.

Intuitively, when a pixel (x, y) in o_j^i is explicitly observed to be unchanged, it is labeled as "1" in the label matrix p_j^i and is labeled as "-1" when it is definitely observed to be changed. When it is masked in the either current or the next observation, it is labeled as "0", with the meaning of unknown.

The objective of the training phase is to learn a prediction function that exploits the relations between image observations caused by actions. When it converges, for each image observation o_j^i, we can obtain a matrix that consists of the prediction result on each pixel and we use $\hat{p}_j^i$ to denote it.

Next, we complement the image observations based on the prediction matrices. Given an image observation sequence

O^i, a mask matrix sequence M^i and a prediction matrix sequence $\hat{P}^i$, we define the complemented image sequence $\overline{O}^i = \langle \overline{o}^i_1, \overline{o}^i_2, ..., \overline{o}^i_n \rangle$ as follows: for every pixel (x, y),

$$\overline{o}^i_{j-1}(x,y) = \begin{cases} o^i_j(x,y) & \text{if } m^i_{j-1}(x,y)=0, m^i_j(x,y)=1, \\ & \hat{p}^i_j(x,y) > \lambda \\ o^i_{j-1}(x,y) & \text{otherwise} \end{cases}$$

where λ is a predefined positive threshold.

Intuitively, when the confidence for a masked pixel is higher than the threshold λ, it is predicted as unchanged. It can be substituted with an unmasked pixel in the latter image observation which is predicted as unchanged. Meanwhile, we update the mask matrices for the complemented pixels, denoted by $\overline{M}^i = \langle \overline{m}^i_1, \overline{m}^i_2, ..., \overline{m}^i_n \rangle$. Formally, if $o^i_j(x,y) \neq \overline{o}^i_j(x,y)$, we set $\overline{m}^i_j(x,y) = 1$.

remark 2 *Notably, the above inpainting approach adopts a complementing procedure from back to front. One may argue that by considering the complementing procedure in dual directions, more masked regions could be complemented. However, the neural network-based prediction model cannot guarantee perfect correctness and such a bidirectional procedure may result in a contradiction. It possibly happens that a masked pixel is predicted as unchanged in two neighbor images but it would be complemented as different pixels.*

Representation Learning

Inspired by `Latplan`, we introduce the State Autoencoder (SAE) of our framework `Recplan`. SAE is a variational autoencoder (VAE) neural network architecture (Kingma et al. 2014) with a Gumbel-Softmax activation (Jang, Gu, and Poole 2017) which aims to learn a bidirectional mapping from image observations to latent states. SAE contains two neural networks: one is called Encoder, translating an image observation into a latent state, and the other one is called Decoder, reconstructing an image from a latent state.

The whole procedure of SAE is shown in the Figure 2. First, an image observation o is translated into a $k \times 3$-dimension matrix $\mathbf{z}$ via Encoder. We further translate $\mathbf{z}$ into a $k \times 3$-dimension matrix $\mathbf{z}'$ via a Gumbel-Softmax activation function. The Gumbel-Softmax activation function is a reparametrization trick that connects the continuous space and the discrete binary space. Every row in $\mathbf{z}'$ is a one-hot representation of three exclusive cases: "True", "False", and "Unknown". More specifically, $(1,0,0)$ indicates a bit is "True", $(0,1,0)$ indicates "False", and $(0,0,1)$ indicates "Unknown". We call the output matrix $\mathbf{z}'$ a latent state and divide it into two parts: a matrix composed of "True" and "False" columns, denoted by $\mathbf{s}$, and the "Unknown" column, denoted by $\mathbf{u}$. Intuitively, each bit of a latent state potentially represents some region of an image, which is implemented via Decoder. Actually, the "True/False" matrix itself is sufficient to represent a latent state because "(0,0)" means to be "Unknown". So we sometimes simply call the matrix a latent state as well.

The objective of training Encoder and Decoder is to make the original image observation o and the reconstructed image $\hat{o}$ become identical as possible in those observed regions. Then we employ the Mean Square Error (MSE) as the reconstruction loss function:

$$L_{recon} = \Sigma_i \Sigma_j ||o^i_j \odot m^i_j - \hat{o}^i_j \odot m^i_j||^2$$

where $\odot$ is element-wise multiplication.

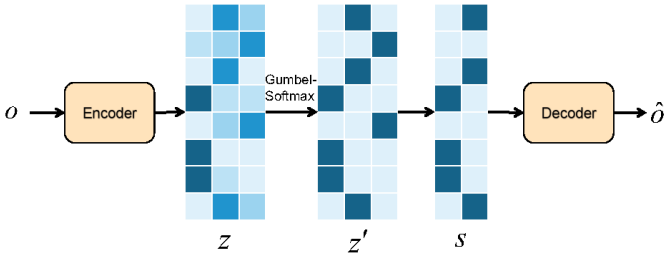

Figure 2: The procedure of the State Autoencoder

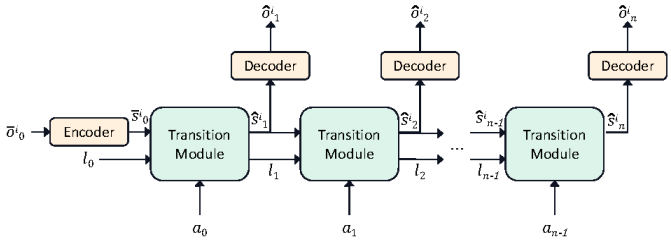

Figure 3: The transition model learning procedure of `Recplan`

To obtain a more accurate latent state, we design a loss function based on the relation between the image observation and its complemented image. As the complemented image $\overline{o}$ differs from the image observation o only on the masked regions, except for the "Unknown" rows, the latent state $\overline{\mathbf{s}}$ of $\overline{o}$ should coincide with $\mathbf{s}$ of o. On the other hand, the image observation has more masks than the complemented image. Thus, the "Unknown" bits of $\mathbf{u}$ should include those of $\overline{\mathbf{u}}$. Then we have the following loss function for latent states:

$$L_{state} = \Sigma_i \Sigma_j (||\overline{\mathbf{s}}^i_j \odot \mathbf{s}^i_j - \mathbf{s}^i_j||^2 + ||\overline{\mathbf{u}}^i_j \odot \mathbf{u}^i_j - \mathbf{u}^i_j||^2).$$

Finally, we define the loss function of SAE as:

$$L = L_{recon} + L_{state}.$$

The Learning Procedure

Next, we learn a transition model to simulate the underlying transition function that propagates an image into another image according to some action. Different from `Latplan` which only takes image transition pairs without action labels as input, we propose to learn a transition model via a recurrent neural network (RNN) in order to capture the unobserved information from a successive image observation trace. Figure 3 shows the learning procedure of the transition model. In the beginning, the transition module takes as input the latent state $\overline{\mathbf{s}}^i_0$ of the initial complemented image $\overline{o}^i_0$, the

first action a_0^i and the initial history information l_0. At every following step, the transition module takes the latent state $\mathbf{s}_j^i$ and the history information l_j obtained from the previous step and an executed action a_j^i. Then the transition module outputs a latent state $\mathbf{s}_{j+1}^i$ which can be decoded as an image $\hat{o}_{j+1}^i$ and a history information l_{j+1} to the next step. Here actions are denoted by vectors in the one-hot representation.

Intuitively, the next latent state computed by the transition module is restricted by the reconstruction loss not only from the current image but also from the previous images.

To approximate the underlying transition function, we require the image $\hat{o}_j^i$ decoded from $\mathbf{s}_j^i$ to coincide with the complemented image $\overline{o}_j^i$ on the observed regions. We thus use MSE as the loss function:

$$L_\gamma = \Sigma_i \Sigma_j ||\overline{o}_j^i - \hat{o}_j^i \odot \overline{m}_j^i||^2$$

When the loss function converges, every image observation is recovered to a complete image and the initial history information l_0 is fixed. More specifically, given any action a and any image o, we first translate o into a latent state $\mathbf{s}$ via Encoder, and compute its next latent state $\mathbf{s}'$ via the transition module with $\mathbf{s}$ and a as input, and then apply Decoder to reconstruct an image o' for $\mathbf{s}'$. That is, we obtain a transition model $\hat{\gamma}$ such that $o' = \hat{\gamma}(o, a)$, which recovers the masked images as possible in the input image observation traces.

Visual Planning and Heuristic Model Learning

In this section, we show how to learn a heuristic function based on latent states and propose a goal-oriented planning approach for images.

Heuristic Model Learning

Essentially, latent states represent images in a high-dimensional space. It actually provides a way to exploit relations among images and actions from another perspective. For that, we propose to learn a neural network-based heuristic function based on latent states. Basically, we consider the heuristic learning problem as a regression problem that aims to find a value that describes the distance from the current image to the goal image in an image sequence.

First, we show how to construct the training data for this learning task. We define the objective heuristic value for the latent state $\hat{\mathbf{s}}_{j+1}^i$ as the number of actions to obtain the goal image o_n^i from the current image o_j^i. Formally, the heuristic value of selecting the action a_{j+1}^i on the latent state $\hat{\mathbf{s}}_j^i$ is $n - j - 1$. In order to select the smallest heuristic value at each step, we hope that the heuristic value of the appropriate action is smaller than that of other actions. Thus, we simply set the heuristic value as $n - j + 1$ when selecting other actions rather than a_{j+1}^i on the latent state $\hat{\mathbf{s}}_j^i$. Formally, for an image observation sequence O^i with its goal image o_n^i and every action a occurring in all sequences, we define the training data about heuristic values as follows: for $0 \leq j < n$,

$$h(\hat{\mathbf{s}}_j^i, \overline{s}_n^i, a) = \begin{cases} n - j - 1, & \text{if } a = a_{j+1}^i \\ n - j + 1, & \text{otherwise} \end{cases}$$

where $\hat{\mathbf{s}}_j^i$ is the latent state of o_j^i computed by `Recplan` and $\overline{s}_n^i$ is the latent state of o_n^i computed by Encoder.

remark 3 *Notably, the reason why we use $\overline{s}_n^i$ rather than the latent state of the complete image computed by `Recplan`, i.e., $\hat{\mathbf{s}}_n^i$, is because we hope to learn a heuristic model that is applicable to a goal image with masks. It allows the approach to conduct planning in a partially observable environment.*

The learning task is to obtain a heuristic model $\hat{h}$ to simulate the objective heuristic function h. Finally, we take MSE as the loss function:

$$L_h = \Sigma_i \Sigma_j ||h(\hat{\mathbf{s}}_j^i, \overline{s}_n^i, a) - \hat{h}(\hat{\mathbf{s}}_j^i, \overline{s}_n^i, a)||^2.$$

Visual Planning

The visual planning task aims to compute an action sequence leading a given initial image to propagate to be a given goal image and generate the intermediate images between the two given images.

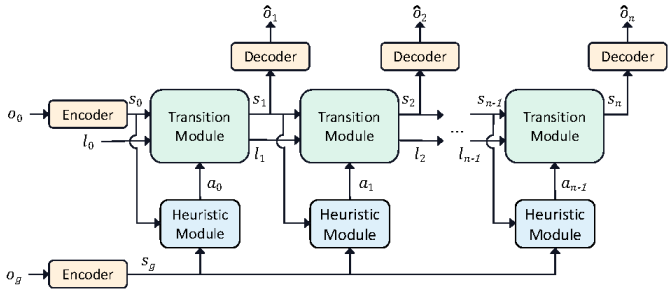

Figure 4: The planning procedure of `Recplan`

Indeed, we do not learn a declarative PDDL-like planning model, which makes it impossible to invoke an off-the-shelf PDDL-based planner. Similar to (Asai and Fukunaga 2018), we implemented an A^* algorithm for visual planning based on the learned transition model. But different from `Latplan` which trivially takes the number of the unachieved goals as the heuristic value, we apply the learned heuristic model.

Figure 4 illustrates the planning procedure of our framework `Recplan`. Different from the training phase where a whole image observation sequence is considered as input, in the planning phase, `Recplan` takes as input a fully observed initial image o_0 and a possibly partially observed goal image o_g. Finally, it outputs a plan with a sequence of complete images connecting the initial image o_0 and a complete image that coincides with o_g. At every step, the heuristic module takes the current latent state $\mathbf{s}_j$ and the latent state $\mathbf{s}_g$ of the goal image as input. It outputs an action a_j with the smallest heuristic value via traversing all actions. Meanwhile, the transition module has four inputs: the latent state $\mathbf{s}_j$ which is computed from the previous step, and the action a_j obtained from the heuristic module and the history information l_j. It outputs a latent state $\mathbf{s}_{j+1}$ which will be decoded into an image $\hat{o}_{j+1}$ and will be propagated to the next step.

When the decoded image $\hat{o}_j$ for some step is close to the goal image despite masks, i.e., their difference is smaller than a quite small threshold, the planning algorithm terminates.

The sequence of the actions output by the heuristic module is a solution plan and the sequence of decoded images describes how the initial image evolves to the goal image.

Experiments

Domains and Datasets

Next, we evaluate our approach Recplan on three domains: 8-puzzle (MNIST) and 8-puzzle (Mandrill, Spider).

8-puzzle (MNIST). Every configuration in an 8-puzzle domain is composed of 9 digits (0-8) which are arranged in a 3×3 tile matrix. The tile "0" is considered as blank. Only actions of swapping the blank tile with one of its adjacent tiles are valid. We use "UP", "DOWN", "RIGHT" and "LEFT" to denote valid actions. For any configuration, we generate a 42×42 image obtained by replacing each digit tile with the corresponding digit image from the MNIST dataset (LeCun et al. 1998).

8-puzzle (Mandrill and Spider). For these two domains, we divide two pictures Mandrill and Spider into nine sub-images by a 3×3 matrix, respectively. Each sub-image is labeled by nine digits (0-8). The other settings are similar to the 8-puzzle (MNIST) domain. Then each state yields a 42×42 image composed of nine muddled sub-images.

It is remarkable that our heuristic learning approach does not require the training sequences to be optimal. If valid plans rather than optimal plans are used as training data, the learned heuristic model computes plans that are not optimal but valid. However, to assess approaches in terms of plan quality, we set out to create optimal plans as a dataset. First, for each domain, we randomly generate initial states.

Following (Asai and Fukunaga 2018), we use a breadth-first search algorithm to enumerate action sequences with a length of 7 to construct a tree in which each node represents a state and each edge represents an action. We build the dataset by selecting those plans whose final state never appears in the tree with a level less than 8, which thus are assured to be optimal. Then, for each domain, we generate 10100 distinct state-action sequences. We take 8500 sequences for training, 1500 for validation, and 100 for testing.

After creating an image observation for each state as we mentioned above, we randomly generate mask matrix sequences with different mask sizes and numbers. By element-wise multiplication, we then obtain a set of image observation sequences.

Experiment Details

In this paper, we need to train five models. The following are the details of the models:

- The prediction model is a neural network composed of two 3×3 convolutional layers, two 400-dimension fully connected layers and a tanh activation function.
- We construct the SAE Encoder as a neural network with two 3×3 convolutional layers with tanh as activation function, a 1000-dimension fully connected layer and a 72×2-dimension output layer.
- The SAE Decoder consists of two 1000-dimension fully connected layer, with ReLU as an activation function, and outputs a 42×42 image.
- The transition model as a 400-dimension RNN cell, a 1000-dimension fully connected layer for the 8-puzzles domains, a ReLU activation function, and a 72-dimension output layer.
- The heuristic model by three 76-dimension fully connected layers and a 4-dimension output layer, with LeakyReLU as the activation function.

We train all models with batch size 500 and learning rate 10^{-3}. Specifically, we train (1) the prediction model for 500 epochs; (2) the SAE Encoder and Decoder for 1000 epochs with Gumbel-Softmax temperature: $5.0 \rightarrow 0.7$, droptout factor 0.4 and Batch Normalization (Ioffe and Szegedy 2015); (3) the transition model and the heuristic model for 1500 epochs. All experiments are conducted on a machine with a 12GB GeForce GTX 1080 Ti.

Baseline. In this paper, we take the seminal image-based classical planning approach Latplan (Asai and Fukunaga 2018) as the baseline approach and we use Lat to denote it. As mentioned above, Latplan is proposed for the fully observed setting, so we add the inpainting approach to predict the missing parts before training the networks of Latplan. Then we use $\texttt{Lat}^+$ to denote it. On the other hand, we use Rec to denote our approach Recplan. Similarly, for comparison, we also equip our inpainting approach into Recplan and use $\texttt{Rec}^+$ to denote it.

Metrics

We evaluate the approaches on the following aspects:

- **Accuracy of transition models.** Taking a fully observed initial image and its corresponding action sequence as input, we compare the difference in pixels between the image sequence calculated by the approaches and the ground truth image sequence. The MSE metric on pixels is used to test the accuracy of the learned transition models.
- **Planning validity and optimality.** Taking a fully observed initial image and a partially observed goal image, we invoke these approaches to compute plans and test their planning performances in terms of validity and optimality. We define the metric of planning validity rate as the proportion of valid plans on all testing instances. A plan is valid if each of its actions is valid and finally leads to the goal state under the ground-truth transition function. Similarly, the planning optimality rate is defined as the proportion of valid plans with a length of 7 on all testing instances. The planning cutoff time is set to 180 seconds.
- **Average node expansions.** To evaluate the performance of the learned heuristic model, we also count the average node expansions when computing valid plans.

Experimental Results

Accuracy of transition models We train the approaches on the datasets with different settings of masks and test the learned transition models on the initial image and action sequence of the testing instances. Table 1 shows the results on the transition model accuracy of the approaches. Compared with Lat and its variant, Rec and its variant $\texttt{Rec}^+$ perform better. In all domains, Rec obtains the best performance

Table 1: Accuracies of the learned transition models.

Mask setting	MNIST				Mandrill				Spider			
	Rec	Rec$^+$	Lat	Lat$^+$	Rec	Rec$^+$	Lat	Lat$^+$	Rec	Rec$^+$	Lat	Lat$^+$
0	**0.05**	**0.05**	0.11	0.11	**5.79**	**5.79**	7.60	7.60	**7.13**	**7.13**	7.71	7.71
10, 3×3	**0.00**	0.12	0.42	0.09	**6.02**	6.93	7.37	7.55	**5.57**	6.66	8.18	6.73
20, 3×3	**0.00**	**0.00**	0.04	0.30	**5.77**	6.78	6.90	6.76	**5.15**	6.06	7.40	6.83
30, 3×3	**0.00**	**0.00**	0.01	0.01	**5.87**	6.86	6.96	6.86	**5.16**	5.88	7.55	7.88
40, 3×3	**0.00**	0.01	0.23	0.20	**5.94**	6.52	6.91	6.80	**5.31**	6.57	7.39	7.67
50, 3×3	**0.00**	0.01	0.03	0.12	**5.60**	6.86	6.97	7.13	6.67	**6.01**	6.74	7.17
5, 3×3	**0.00**	0.01	0.47	0.19	**5.62**	6.69	7.20	7.11	**5.59**	6.13	7.81	7.29
5, 6×6	**0.00**	0.07	0.28	0.38	**5.54**	6.76	6.77	7.06	**5.96**	6.79	9.18	8.44
5, 9×9	**0.01**	**0.01**	0.07	0.32	**6.27**	6.72	6.81	6.95	**6.59**	8.06	8.08	6.51
5, 12×12	**0.01**	0.88	20.27	4.30	**7.21**	7.22	7.47	7.52	**7.95**	8.26	9.97	8.55
mean	**0.01**	0.12	2.19	0.54	**5.96**	6.71	7.10	7.13	**6.11**	6.76	8.00	7.48

For space limitation, three decimal places are missing each value should be multiplied by 10^{-3}. The number of the masks and the size are separated by a comma. For example, "10, 3×3 " means each matrix contains ten pieces of 3×3 .

while in the MNIST domain, it almost recovers the whole image sequences. On average, Rec performs 16% better than Lat in Mandrill and 24% better in Spider. Despite MNIST, the difference between Rec and Lat in the accuracy of the transition model is statistically significant with a p-value of 8.1e-5 by a two-tailed t-test. As the p-value is far less than 0.05, it shows that Rec improves Lat significantly. Furthermore, Rec outperforms Lat$^+$ by 16% in Mandrill and by 18% in Spider. The difference between Rec and Lat$^+$ in the accuracy is statistically significant with a p-value of 7.6e-4 by a two-tailed t-test, demonstrating the significant advance of Rec.

On the other hand, on average Rec$^+$ beats Lat by 39% and Lat$^+$ by 31%. By two-tailed t-tests, Rec$^+$ differs significantly from Lat in the transition model accuracy with p-value 0.024 and from Lat$^+$ with p-value 0.039. The reason why the Rec approaches outperform the Lat approaches is that the recurrent framework of Rec takes a whole sequence into account, which makes up for the unknown information caused by the masks.

Surprisingly, it is counter-intuitive that the inpainting approach has a detrimental influence on the performance of Rec. It is because the learned model predicts incorrectly when complementing some masked regions, which leads wrong inpainting. Such incorrect information finally makes Rec learn a less accurate transition model. On the other hand, it turns out that the inpainting approach improves Lat via making up for the masks. Because Lat is not designed for the partially observed environment, the inpainting approach mitigates the consequence of the masks. It is notable that for Lat, the prediction model takes effect more signally on the environments with fewer masks.

Planning validity with different numbers and sizes of masks To investigate how the account of small masks and the size of masks influence the planning performance, we test the validity rates on different settings of masks. Table 2 shows the results of the validity rates. Note that the goal image observation is also masked identically as the training sequences. That is, if a transition model is learned in a setting of masks then it would be tested on an instance with a goal image observation with the same setting of masks.

Unsurprisingly, our approaches Rec and Rec$^+$ have much better performance and solve most of the testing instances. The difference between Rec and Lat in the validity rate is statistically significant with p-value 5.5e-5 by a two-tailed t-test. Also, Rec$^+$ differs significantly from Lat$^+$ in statistics with p-value 2.1e-7 by a two-tailed t-test. In most of the failure cases, the Lat approaches cannot output an action sequence within the cutoff time. It is because the accuracy of its transition model is insufficient and it cannot terminate for finding an image consistent with the goal image observation.

Essentially, the results of their validity rates conform to the results of their transition accuracies. The validity rates of Rec and Rec$^+$ are comparable. While the validity of Lat$^+$ is higher than that of Lat. In some settings, though our inpainting approach does not improve Lat in the transition model accuracy but improves its performance in planning because it makes the latent state of the image observation closer to that of the ground-truth image and it is easier to find an appropriate action towards to the goal.

Planning optimality with different sizes of masks We also evaluate the influence of masks on the planning optimality performance of the approaches. Table 2 shows the results of the optimality rates under different numbers and sizes of masks. It is not difficult to find that Rec and Rec$^+$ are superior to Lat and Lat$^+$ in the optimality rate. On average, Rec outperforms Lat by up to 47% and Lat$^+$ by up to 38%. For statistical significance analysis, the difference between Rec and Lat has a p-value of 4.4e-5 by a two-tailed t-test. With our inpainting approach, Rec$^+$ is able to optimally solve 38% more instances than Lat$^+$. The difference between Rec$^+$ and Lat$^+$ is statistically significant with a p-value of 7.1e-8 by a two-tailed t-test.

Surprisingly, most of the valid plans computed by the Rec approaches are optimal. For only one instance in Mandrill and one in Spider, Rec cannot compute a valid and non-optimal plan that contains repeated actions. That is, the ratio of its optimality rate to its validity rate average is extremely close to 100%. Whereas, the optimality rates of the Lat approaches are lower than their validity rates. Considering

Table 2: The optimality/validity rates trained with images with different mask settings

Mask	MNIST				Mandrill				Spider			
setting	Rec	Rec$^+$	Lat	Lat$^+$	Rec	Rec$^+$	Lat	Lat$^+$	Rec	Rec$^+$	Lat	Lat$^+$
0	**98 /98**	**98/98**	68/70	68/70	**95/95**	**95/95**	70/71	7071	**9898**	**98/98**	57/62	57/62
10, 3×3	**98/98**	**98/98**	60/64	74/76	**98/98**	**98/98**	72/73	79/83	95/95	**96/96**	66/67	66/66
20, 3×3	**99/ 99**	**99/99**	76/80	7476	**98/99**	98/98	65/69	75/76	95/95	**98/99**	68/69	77/78
30, 3×3	**100/100**	**100/100**	74/76	66/68	**97/97**	95/95	68/72	68/68	96/96	**100/100**	73/75	64/67
40, 3×3	**98/ 98**	**98/98**	64/66	70/72	95/95	**9696**	77/80	73/74	97/97	**99/99**	75/77	68/74
50, 3×3	**98/98**	94/94	73/75	68/72	**99/99**	95/95	70/72	73/77	**96/96**	95/95	69/72	68/72
5, 3×3	**97/97**	96/96	73/78	73/75	**96/96**	**96/96**	71/74	78/81	98/98	**99/99**	59/66	70/73
5, 6×6	**97/97**	94/94	65/66	73/75	**97/97**	**97/97**	70/72	69/75	**97/97**	95/95	50/58	68/69
5, 9×9	**100/100**	98/98	56/56	66/67	**98/98**	95/95	72/77	66/71	97/**98**	**98/98**	78/78	79/80
5, 12×12	**95/ 95**	91/91	12/19	60/64	93/93	**96/96**	75/77	77/81	**99/99**	98/98	65/74	71/82
mean	98/98	96.6/96.6	62.1/65	69.2/71.5	96.6/96.7	96.1/96.1	71/73.7	72.8/75.7	96.8/96.9	97.6/97.7	66/69.8	68.8/72.3

The percentage symbol "%" is not shown. The optimality and validity are separated by "/".

Table 3: Average node expansions of finding valid plans

Mask	MNIST				Mandrill				Spider			
setting	Rec	Rec$^+$	Lat	Lat$^+$	Rec	Rec$^+$	Lat	Lat$^+$	Rec	Rec$^+$	Lat	Lat$^+$
0	42.57	42.57	3848.46	3848.46	31.49	31.49	4271.66	4271.66	48.94	48.94	2891.35	2891.35
10, 3×3	40.49	44.90	4280.08	4407.42	34.45	38.45	5459.95	3212.90	43.34	39.63	4067.67	2235.79
20, 3×3	40.69	36.69	2633.68	5513.58	33.66	34.00	4679.3	2302.59	40.38	35.92	3513.22	2126.11
30, 3×3	39.40	35.68	2949.11	2277.23	37.57	40.84	5140.56	1907.06	36.83	36.56	3746.44	3242.48
40, 3×3	40.61	33.55	2032.91	1861.8	38.06	47.92	3034.20	2777.97	39.42	34.02	3352.30	1784.29
50, 3×3	34.45	35.28	2073.00	2119.77	44.65	41.77	4124.56	2404.91	37.46	33.73	4741.21	2190.90
5, 3×3	34.31	34.54	3096.56	3921.97	35.54	39.38	3104.75	4302.73	42.24	39.27	6868.38	2385.33
5, 6×6	35.92	33.85	4734.08	3889.55	38.31	41.36	2671.25	2734.22	35.42	33.68	4518.10	2975.04
5, 9×9	41.16	41.47	1492.07	2465.31	34.90	35.20	1612.52	3125.46	40.24	40.57	1735.28	1059.40
5, 12×12	54.02	43.52	3805.05	5427.38	40.26	36.38	2850.86	1929.88	39.60	43.84	3896.27	3513.61
mean	38.38	37.00	2911.44	3307.08	37.14	39.87	3728.39	2845.98	39.42	36.67	4067.83	2249.92

the Lat approaches use a goal-distance heuristics function, the superior results of the Rec approaches benefit from the excellent guiding performance of the learned heuristic model on selecting actions.

Planning efficiency We also evaluate the searching ability of the approaches by counting the average node expansions for computing valid plans. The results are depicted in Table 3. To find a valid plan, the Lat approaches need to expand two orders of magnitude more nodes than the Rec approaches. By a two-tailed t-test, the difference between Rec and Lat is statistically significant with a p-value of 6.9e-6. Likewise, the difference between Rec$^+$ and Lat$^+$ is statistically significant with p-value 6.8e-6.

Considering that most of the valid plans by Rec and Rec$^+$ are optimal, each of which contains seven steps, the searching procedure based on the learned heuristic model expands about five nodes averagely at each step. It also shows that the learned heuristic model is able to return an appropriate action nearly at every step.

Conclusion

A growing number of people are interested in learning techniques, especially those based on unstructured data, due to its availability and commonality. In this paper, we focus on planning model learning on images, which can be easily obtained from cameras. Compared to the assumption that every image is fully observed, we tackle a more realistic scenario where every image is partially observed. More specifically, we present a novel visual planning model learning framework that is applicable to an environment with incomplete observations. By conducting experiments on the three domains, we show the superiority of our approach on transition model learning and planning performance.

In this paper, we do not learn the condition where an action is valid, which is generally a part of a planning model. It is in fact difficult in a partially observable environment. If the learned precondition is too strong, it probably happens that there would be no action to be applicable; if it is too weak, an invalid action may be chosen. Generally, learning a sound and complete model of action precondition requires a highly accurate prediction on the masked regions. We take the task of learning precondition as one of our future works.

Acknowledgement

This research was funded by the National Natural Science Foundation of China (No. 62076263, 61906216, 61976232).

References

Aineto, D.; Celorrio, S. J.; and Onaindia, E. 2019. Learning action models with minimal observability. *Artif. Intell.*, 275: 104–137.

Arfaee, S. J.; Zilles, S.; and Holte, R. C. 2010. Bootstrap Learning of Heuristic Functions. In *SOCS-10*. AAAI Press.

Arora, A.; Fiorino, H.; Pellier, D.; Métivier, M.; and Pesty, S. 2018. A review of learning planning action models. *Knowledge Eng. Review*, 33: 1–25.

Asai, M. 2019. Unsupervised Grounding of Plannable First-Order Logic Representation from Images. In *ICAPS-18*, 583–591. AAAI Press.

Asai, M.; and Fukunaga, A. 2018. Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary. In *Proceedings of the 32nd AAAI Conference (AAAI-18)*, 6094–6101.

Asai, M.; and Muise, C. 2020. Learning Neural-Symbolic Descriptive Planning Models via Cube-Space Priors: The Voyage Home (to STRIPS). In *IJCAI-20*, 2676–2682.

Askar, W. A.; Elmowafy, O.; Ralescu, A.; Youssif, A. A.; and Elnashar, G. A. 2020. Occlusion detection and processing using optical flow and particle filter. *Int. J. Adv. Intell. Paradigms*, 15(1): 63–76.

Bertalmío, M.; Sapiro, G.; Caselles, V.; and Ballester, C. 2000. Image inpainting. In *SIGGRAPH-00*, 417–424. ACM.

Bonet, B.; and Geffner, H. 2020. Learning First-Order Symbolic Representations for Planning from the Structure of the State Space. In *ECAI-20*, volume 325, 2322–2329. IOS Press.

Gomoluch, P.; Alrajeh, D.; Russo, A.; and Bucchiarone, A. 2020. Learning Neural Search Policies for Classical Planning. In *ICAPS-20*, 522–530. AAAI Press.

Gregory, P.; and Lindsay, A. 2016. Domain Model Acquisition in Domains with Action Costs. In *ICAPS-16*, 149–157.

Hafner, D.; Lillicrap, T. P.; Fischer, I.; Villegas, R.; Ha, D.; Lee, H.; and Davidson, J. 2019. Learning Latent Dynamics for Planning from Pixels. In *ICML-19*, volume 97, 2555–2565. PMLR.

He, K.; and Sun, J. 2012. Statistics of Patch Offsets for Image Completion. In *ECCV-12*, volume 7573, 16–29. Springer.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *ICML-15*, volume 37, 448–456. JMLR.org.

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical Reparameterization with Gumbel-Softmax. In *ICLR-17*. OpenReview.net.

Jin, K.; Zhuo, H. H.; Xiao, Z.; Wan, H.; and Kambhampati, S. 2022. Gradient-based mixed planning with symbolic and numeric action parameters. *Artif. Intell.*, 313: 103789.

Kingma, D. P.; Mohamed, S.; Rezende, D. J.; and Welling, M. 2014. Semi-supervised Learning with Deep Generative Models. In *NeurIPS-14*, 3581–3589.

Kurutach, T.; Tamar, A.; Yang, G.; Russell, S. J.; and Abbeel, P. 2018. Learning Plannable Representations with Causal InfoGAN. In *NeurIPS-18*, 8747–8758.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Liu, K.; Kurutach, T.; Tung, C.; Abbeel, P.; and Tamar, A. 2020. Hallucinative Topological Memory for Zero-Shot Visual Planning. In *ICML-20*, volume 119, 6259–6270. PMLR.

Martínez, D.; Alenyà, G.; Torras, C.; Ribeiro, T.; and Inoue, K. 2016. Learning Relational Dynamics of Stochastic Domains for Planning. In *ICAPS-16*, 235–243.

Nair, A.; Chen, D.; Agrawal, P.; Isola, P.; Abbeel, P.; Malik, J.; and Levine, S. 2017. Combining self-supervised learning and imitation for vision-based rope manipulation. In *ICRA-17*, 2146–2153. IEEE.

Nair, S.; and Finn, C. 2020. Hierarchical Foresight: Self-Supervised Learning of Long-Horizon Tasks via Visual Subgoal Generation. In *ICLR-20*. OpenReview.net.

Pathak, D.; Krähenbühl, P.; Donahue, J.; Darrell, T.; and Efros, A. A. 2016. Context Encoders: Feature Learning by Inpainting. In *CVPR-16*, 2536–2544.

Roth, S.; and Black, M. J. 2005. Fields of Experts: A Framework for Learning Image Priors. In *CVPR-05*, 860–867. IEEE Computer Society.

Shen, W.; Trevizan, F. W.; and Thiébaux, S. 2020. Learning Domain-Independent Planning Heuristics with Hypergraph Networks. In *ICAPS-20*, 574–584. AAAI Press.

Suresh, S.; Chitra, K.; and Deepak, P. 2013. A Survey On Occlusion Detection. *International journal of engineering research and technology*, Volume 2.

Trunda, O.; and Barták, R. 2020. Deep Learning of Heuristics for Domain-independent Planning. In *ICAART-20*, 79–88.

Xiong, W.; Yu, J.; Lin, Z.; Yang, J.; Lu, X.; Barnes, C.; and Luo, J. 2019. Foreground-Aware Image Inpainting. In *CVPR-19*, 5840–5848. Computer Vision Foundation / IEEE.

Yan, Z.; Li, X.; Li, M.; Zuo, W.; and Shan, S. 2018. Shift-Net: Image Inpainting via Deep Feature Rearrangement. In *ECCV-18*, volume 11218, 3–19. Springer.

Yang, J.; Qi, Z.; and Shi, Y. 2020. Learning to Incorporate Structure Knowledge for Image Inpainting. In *AAAI-20*, 12605–12612. AAAI Press.

Yang, Q.; Wu, K.; and Jiang, Y. 2007. Learning action models from plan examples using weighted MAX-SAT. *Artif. Intell.*, 171(2-3): 107–143.

Yoon, S. W.; Fern, A.; and Givan, R. 2008. Learning Control Knowledge for Forward Search Planning. *J. Mach. Learn. Res.*, 9: 683–718.

Zhuo, H. H.; Muñoz-Avila, H.; and Yang, Q. 2014. Learning hierarchical task network domains from partially observed plan traces. *Artif. Intell.*, 212: 134–157.

Zhuo, H. H.; and Yang, Q. 2014. Action-model acquisition for planning via transfer learning. *Artif. Intell.*, 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 *Proceedings of the 21st International Conference on Automated Planning and Scheduling, (ICAPS-11)*.