Goal Recognition in Latent Space

Leonardo Amado*, Ramon Fraga Pereira*, João Aires*, Mauricio Magnaguagno*,
Roger Granada† and Felipe Meneguzzi†
Faculdade de Informática
Pontifícia Universidade Católica do Rio Grande do Sul
Av. Ipiranga, 6681, 90619-900, Porto Alegre, RS, Brazil
* Email: {first.last}@acad.pucrs.br
† Email: felipe.meneguzzi@pucrs.br

***Abstract*—Recent approaches to goal recognition have progressively relaxed the requirements about the amount of domain knowledge and available observations, yielding accurate and efficient algorithms. These approaches, however, assume that there is a domain expert capable of building complete and correct domain knowledge to successfully recognize an agent's goal. This is too strong for most real-world applications. We overcome these limitations by combining goal recognition techniques from automated planning, and deep autoencoders to carry out unsupervised learning to generate domain theories from data streams and use the resulting domain theories to deal with incomplete and noisy observations. We show the effectiveness of the technique in a number of domains and compare the recognition effectiveness of the autoencoded against hand-coded versions of these domains.**

I. Introduction

Goal and plan recognition refer to the tasks of identifying, respectively, the desired goal towards which an observed agent intends to achieve, and the specific plan to which the agent has committed to executing to achieve said goal. Although the first approaches to plan recognition based on planning theories required a substantial amount of domain knowledge [1], subsequent approaches have gradually relaxed such requirements either by using more expressive planning and plan-library based formalisms [2]–[5] as well as allowing for different levels of accuracy and amount of information available in observations required to recognize goals [6]–[9]. However, regardless of the type of domain model formalism describing the observed agent's behavior, all such approaches assume that a human domain engineer can provide an accurate and complete domain model for the plan recognition algorithm. Such dependence on a human domain engineer severely limits the applicability of modern plan and goal recognition algorithms to abstracted domains rather than real-world ones.

In this paper, we overcome the dependence on human domain engineers for goal recognition by automatically building planning domain knowledge from raw data and using the resulting model in an algorithm capable of recognizing an agent's goal from the same type of raw data. To automatically generate such domain knowledge, in Section III we employ a variational autoencoder (VAE) [10] to map from raw data (in this paper, images) into a latent space representing logical fluents, and, using such fluents, we derive a PDDL [11] action library over which we can reason using planning techniques [12]. Specifically, in Section IV we extend landmark-based goal recognition techniques [9] to infer goals from the encoded raw data and use the decoder part of the variational autoencoder to visualize the plan steps expected of the observed agent. Our main contribution, thus, is a novel goal recognition mechanism that combines deep-learning and heuristic planning techniques to obviate the need for accurate domain engineered planning domains. This allows modern goal recognition algorithms to work directly on real-world data, rather than rely on additional processing of such data into a symbolic representation. We evaluate our technique in Section V on a dataset consisting of domains from earlier work on planning in latent space [12] as well as images we generate automatically from domains from standard planning benchmarks. Our results show that our domain autoencoding scheme approximates the encoding of ground versions of hand-coded planning domains and allow recognition accuracies that, in the best case matches and in the worst case is within 33% of hand-coded goal recognition domains. Finally, we compare our contribution to recent work in Section VI and conclude the paper pointing towards further research in Section VII.

II. Background

A. Goal Recognition

Goal recognition is the task of recognizing the goal being pursued by a rational (software or human) agent from observations of its acting in the environment. The observations collected from the environment can be either a sequence of actions performed by the agent, or the consequences thereof — such sequences can be either seen in full or a partial subsequence of the actions performed by the agent. Plan recognition is a related task to goal recognition, but whose object is not only recognizing the goal of the agent being observed, but also inferring the upcoming actions the agent will take towards such goal [13,14]. Goal and plan recognition in real-world data assume an underlying processing step that translates raw sensor data into some kind of symbolic representation [15], as well as a model of the observed agent's behavior generation mechanism. Most goal and plan recognition approaches [2,16,17] employ plan libraries to represent agent behavior (*i.e.*, a library that describes all plans for achieving goals), and plan recognition techniques which use such libraries are analogous to parsing.

Recent work uses classical planning domain definitions to represent potential agent behavior bring goal and plan recognition closer to automated planning [6]–[9,18]–[20]. These approaches — which do not use plan libraries — show that automated planning techniques can be used to efficiently recognize goals and plans. In many domains where goal and plan recognition are important (*e.g.*, smart environments, user monitoring and crime detection), plan libraries may be unavailable, making this second class of approaches important.

Approaches that use STRIPS-style [21] domain encodings are often known as *plan recognition as planning* (PRAP) because they use planning domains to generate hypotheses of possible plans consistent with observations [18]. While most approaches for this kind of plan recognition have serious scalability issues, recent work on plan recognition as planning have solved this issue [9] by using landmark-based heuristics [22] to efficiently process observations without the need to call a planner multiple times. Planning landmarks are necessary facts or actions in plans that achieve a particular goal from an initial, and which can be used to discriminate observations from plans towards different goals. Our recent work has developed heuristics and efficient algorithms to use landmark information to rank goal hypotheses in time linear with the number of observations.

Formally, we model planning domains of the agents being observed as following a STRIPS [21] domain model $\mathcal{D} = \langle \mathcal{R}, \mathcal{O} \rangle$, where: $\mathcal{R}$ is a set of predicates with typed variables. Such predicates can be associated to objects in a concrete problem (i.e. grounded) representing binary facts. Grounded predicates represent logical values according to some interpretation as facts, which are divided into two types: positive and negated facts, as well as constants for truth ($\top$) and falsehood ($\perp$). The set $\mathcal{F}$ of positive facts induces the state-space of a planning problem, which consists of the power set $\mathbb{P}(\mathcal{F})$ of such facts, and the representation of individual states $S \in \mathbb{P}(\mathcal{F})$. $\mathcal{O}$ is a set of operators $op = \langle pre(op), \mathit{eff}(op) \rangle$, where $\mathit{eff}(op)$ can be divided into positive effects $\mathit{eff}^+(op)$ (the add list) and negative effects $\mathit{eff}^-(op)$ (the delete list). An operator op with all variables bound is called an action and the collection of all actions instantiated for a specific problem induces a state transition function $\gamma(S, a) \mapsto \mathbb{P}(\mathcal{F})$ that generates a new state from the application of an action to the current state. An action a instantiated from an operator op is applicable to a state S iff $S \models pre(a)$ and results in a new state S' such that $S' \leftarrow (S \cup \mathit{eff}^+(a))/\mathit{eff}^-(a)$.

A planning problem within $\mathcal{D}$ and a set of typed objects Z is defined as $\mathcal{P} = \langle \mathcal{F}, \mathcal{A}, \mathcal{I}, G \rangle$, where: $\mathcal{F}$ is a set of facts (instantiated predicates from $\mathcal{R}$ and Z); $\mathcal{A}$ is a set of instantiated actions from $\mathcal{O}$ and Z; $\mathcal{I}$ is the initial state ($\mathcal{I} \subseteq \mathcal{F}$); and G is a partially specified goal state, which represents a desired state to be achieved. A plan π for a planning problem $\mathcal{P}$ is a sequence of actions $\langle a_1, a_2, ..., a_n \rangle$ that modifies the initial state $\mathcal{I}$ into a state $S \models G$ in which the goal state G holds by the successive execution of actions in a plan π. Modern planners use the *Planning Domain Definition Language* (PDDL) as a standardized domain and problem representation medium [11], which encodes the formalism described thus far.

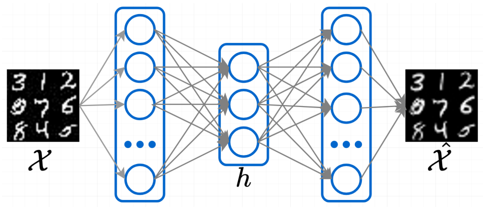

Fig. 1: Autoencoder represented as an input $\mathcal{X}$, output $\hat{\mathcal{X}}$ and a hidden layer (latent layer) h.

Bringing this all together, a goal recognition problem is a tuple $\mathcal{P}_{GR} = \langle \mathcal{D}, \mathcal{F}, \mathcal{I}, \mathcal{G}, O \rangle$, where $\mathcal{D}$ is a planning domain; $\mathcal{F}$ is the set of facts; $\mathcal{I} \subseteq \mathcal{F}$ is an initial state; $\mathcal{G}$ is the set of possible goals, which include a correct hidden goal G^* (*i.e.*, $G^* \in \mathcal{G}$); and $O = \langle o_1, o_2, ..., o_n \rangle$ is an observation sequence of executed actions, with each observation $o_i \in \mathcal{A}$, and the corresponding action being part of a valid plan π that sequentially transforms $\mathcal{I}$ into G^*. The solution for a goal recognition problem is the correct hidden goal $G \in \mathcal{G}$ that the observation sequence O of a plan execution achieves. An observation sequence O contains actions that represent an optimal or sub-optimal plan that achieves a correct hidden goal, and this observation sequence can be full or partial. A full observation sequence represents the whole plan that achieves the hidden goal, *i.e.*, 100% of the actions having been observed. A partial observation sequence represents a subsequence of the plan for the hidden goal, such that a certain percentage of the actions actually executed to achieve G^* could not be executed.

B. Neural Networks and Unsupervised Autoencoding

An autoencoder (AE) [23] is a neural network trained to encode an arbitrary input into an n-dimensional vector representation that can be decoded reproducing the exact input as the output of the entire network. In other words, training an autoencoder consists of learning from unlabeled data an approximation to the identity function, where the generated output $\hat{\mathcal{X}}$ is similar to the input $\mathcal{X}$ [24]. Internally, an autoencoder consists of two parts: an encoder function $h = f(x)$ that maps the input through multiple layers to a specific hidden layer h, that encodes a latent representation ($\mathcal{L}$) of the input, and a decoder that produces a reconstruction $r = g(h)$, as illustrated in Figure 1. Since autoencoders are designed to be unable to copy perfectly the input, they have to learn useful properties that resembles the training data. The purpose of an autoencoder is to constrain the latent layer h to a smaller dimension than the input $\mathcal{X}$, forcing the autoencoder to learn the most salient features of the training data. In fact, the autoencoder often learns a low-dimensional representation very similar to Principal Component Analysis (PCA). Unlike PCA, autoencoders that contain nonlinear encoder and decoder functions can learn more powerful nonlinear generalizations [23].

State Autoencoder (SAE) is a special type of autoencoder which learns bidirectional mapping between raw data and propositional states. In this work, the encode function maps images to propositional states, i.e., a symbolic representation as latent space vectors, and the decode function maps the propositional states back to images. In order to create a SAE as a Gumbel-Softmax Variational Autoencoder [12], a Gumbel-Softmax activation [25] is used in the latent layer of a Variational Autoencoder (VAE) [26]. VAE is a version of autoencoder that impose additional constraints on the encoded representations (latent layer). Instead of learning an arbitrary function to encode and decode the input data, VAE learns the parameters following a probability distribution of the data, such as Gaussian. Since the distribution has to be differentiable to the application of backpropagation, VAEs use a *reparametrization trick*. In SAE, Gumbel-Softmax performs the *reparametrization trick* for categorical distribution by approximating the Gumbel-Max [27]. Thus, a one-hot vector z is generated for each class as

$$z_i = \text{Softmax}\left(\frac{g_i + log\ \pi_i}{\tau}\right) \tag{1}$$

where g_i is independent and identically distributed samples drawn from Gumbel(0, 1) [28], π is the class probability vector and τ is the "temperature" that controls the magnitude of approximation, which is annealed to 0 by a certain schedule. Thus, the output of the Gumbel-Softmax converges to a discrete one-hot vector when $\tau \approx 0$.

III. Planning in Latent Space

Planning algorithms are based on the *factored* transition function $\gamma(S, a)$ that represents states as discrete facts. This transition function is traditionally encoded manually by a domain expert, and virtually all existing plan recognition approaches require varying degrees of domain knowledge in order to recognize observations [9]. Automatically generating such domain knowledge involves at least two processes: converting real-world data into a factored representation (i.e. the predicates in $\mathcal{R}$); and generating a transition function (i.e. the set of operators $\mathcal{O}$) from traces of the factored representation. Although a few approaches have tackled the challenge of applying learning to models of transition functions [29], almost no approaches have addressed the problem of generating domain models from real world data. A recent approach to planning generates domain models from images of the visualization of the state of simple games and problems, such as the sliding blocks puzzle or towers of Hanoi [12]. This approach uses an auto-encoder [30] neural network to automatically generate two functions with regard to an input image $\mathcal{X}$ and a latent representation $\mathcal{L}$: an encoder $\phi : \mathcal{X} \mapsto \mathcal{L}$ and a decoder $\psi : \mathcal{L} \mapsto \mathcal{X}$. In this specific case, the input is a d-dimensional image $\mathbb{R}^d$ and the output is an $n \times m$ matrix $\mathbb{R}^{n\times m}$ representing n categorical variables each of which with m categories. When m is two, the output of this autoencoder corresponds to binary variables that can be interpreted as propositional logic symbols comprising the $\mathcal{F}$ component of a planning domain (without the intermediary step of the generating the set $\mathcal{R}$ of predicates).

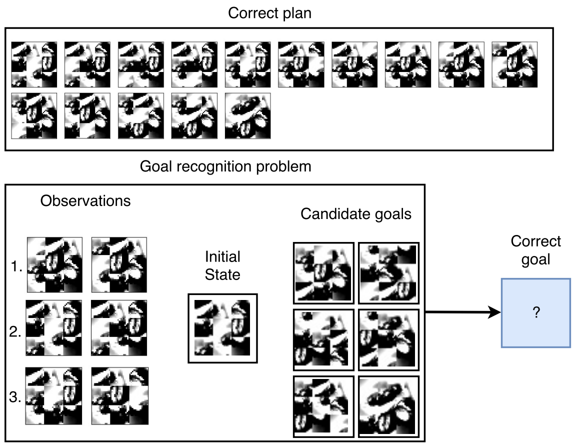

Fig. 2: Image goal recognition problem.

The resulting representation in latent space is amenable to automatically inducing a transition function γ from pairs of states under the assumption that state transitions correspond exactly to pairs of consecutive images in the observed traces. Under this assumption, they generate a large number of propositional actions representing changes between these images as add and delete effects of STRIPS-style actions. The resulting domain representation encodes in latent-space the propositional features from the images. LatPlanα is a heuristic-based forward-search planner [12] that uses this representation to plan solutions for problems derived from images of the initial and target state using the encoded domains. Preliminary experimentation with LatPlanα [12] shows that heuristics from the planning literature [31, Chapter 3] are still applicable, however, given the propositional nature of the encoding, they are not as informative. Such lack of informativeness provides a challenge to the application of heuristics for goal and plan recognition [8,9,32], especially those based on landmarks. As we see in Section IV, in order to successfully employ efficient goal recognition approaches, we need to not only learn a consistent latent representation of states, but also use the propositional transition function induced from state pairs to generate STRIPS-style operators.

IV. Goal Recognition in Latent Space

In this section we propose an approach capable of applying different goal recognition techniques in image domains. Figure 2, illustrates the problem of goal recognition using images, in this case we want to infer which is the correct image configuration that the agent is trying to achieve from the set of candidate goals using only observations consisting of intermediate image configurations. As we can see, inferring the correct goal from such domain is not a trivial task when a small number observations are provided.

To recognize goals in image based domains, there are 4 milestones we must achieve. First, we must train an autoen-

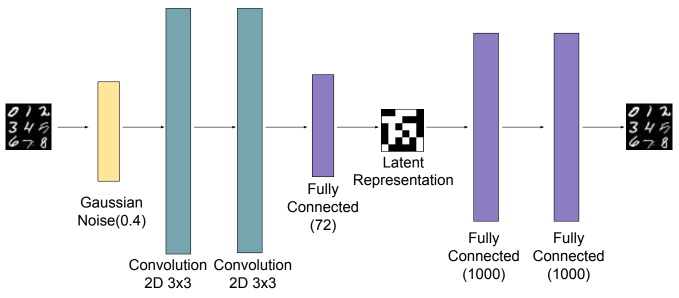

Fig. 3: Autoencoder architecture.

coder capable of creating a latent representation to a state of such image domain. Second, we derive a PDDL domain using Algorithm 1, by extracting the transitions of such domain when encoded in latent space, obtaining a domain $\mathcal{D}$. Third, we must convert to a latent representation a set of images representing, the initial state $\mathcal{I}$, the set of facts $\mathcal{F}$ and a set of possible goals $\mathcal{G}$, where the hidden goal G^* is included. Finally, we can apply goal recognition techniques using the computed tuple $\langle \mathcal{D}, \mathcal{F}, \mathcal{I}, \mathcal{G}, O \rangle$

We create the encoded representation of the image states using an autoencoder adapted from Asai and Fukunaga [12], and which has the architecture illustrated in Figure 3. The input to the network are 42x42 single channel (black and white) images, which correspond to the visual representation of problems in our experimental domains. The encoder part consists of three layers: two 2D convolution layers [33] using a 3x3 filter in both of them, followed by a fully connected layer with 72 nodes, using rectified linear unit (ReLU) activation [34]. Prior to the first convolutional layer, we add 0.4 Gaussian noise, as well as a 0.4 dropout rate after the two convolutional layers. The fully connected layer connects to a 6x6 (36 nodes) latent layer, using Gumbel Softmax [25] activation. This layer generates the latent representation of the image, which we use to infer the planning domain. We use the latent layer with a size of 36 bits in order to represent the entire state space of all problems in our experimental domains. The decoder part of the network consists of two fully connected layers using ReLU activation, connected to the output of the latent layer. We also add a 0.4 dropout rate after each fully connected layer of the decoder. Finally, the decoder reconstructs the input image using an output layer with the same size as the input layer.

Our approach requires us to create one distinct autoencoder for each domain and train each of them with pre-processed images sampled from the domain. We trained the autoencoder with 20000 distinct states as images from each domain. We pre-processed the images before training by applying a grayscale filter and then binarizing the resulting images with a threshold of 0.4. Training took 150 epochs, with a 0.1 learning rate and a batch size of 1000 samples.

Having trained an autoencoder for each domain, we must derive a PDDL domain representation. A PDDL domain consists of multiple actions that can be performed in the domain at specific situations encoded in their pre-conditions, and transition the environment by changing the state of binary variables representing such environment. To extract such actions, we create a list containing all possible transitions of a domain. These transitions map which binary variables change as the observed agent executes actions in the domain environment. We encode these transitions using the latent representation containing 72 bits divided into two blocks of 36 bits representing, respectively, the previous state and the state resulting from applying the action. We generated this latent representation of the transitions by seeding two images (the state s before the action and the next state s', after the action) and encoding both using the respective autoencoder. Using this set of encoded transitions we derive a set of PDDL actions by performing a bit-wise comparison on both states of a transition to compute the changes between state s and s' using Algorithm 1. We perform a modified XOR operation on both states to compute the effect of each transition, and call this operation XOR_E standing for *Effect XOR* in Line 5. The difference of this operation is that the output is 1 for *positive effects* (i.e. when a bit has 0 in s and 1 in s') and -1 for *negative effects* (i.e. when a bit has 1 in s and 0 in s') to distinguish between these two types of effect. XOR_E allows to group transitions that change the same set of bits into a set of candidate actions, which we further differentiate by inferring their pre-conditions. To compute the precondition of candidate actions, we use a variation of the XNOR operator, which we call $XNOR_P$ standing for *Precondition XNOR*, in Line 10. The $XNOR_P$ operation aims to distinguish, from the grouped action candidates, which ones share the *exact same* bit configuration in s, i.e. which bits in *do not change* between the preconditions of a set of candidate actions. The idea is that if a bit has the same value through all states s of every

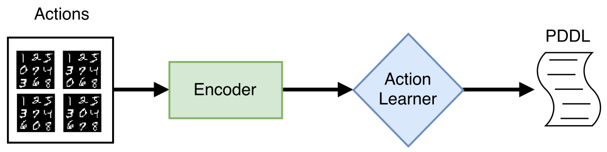

Fig. 4: PDDL domain generation.

Algorithm 1 Learn actions of a PDDL domain

```
Require: Set of transitions $T$
 1: function ACTION-LEARNER($T$)
 2:     $E \leftarrow \langle\rangle$                        ▷ Map of candidate actions
 3:     $A \leftarrow \langle\rangle$                        ▷ Set of generated actions
 4:     for all $(s, s') \in T$ do
 5:         $eff \leftarrow XOR_E(s, s')$
 6:         $E(eff) \leftarrow E(eff) \cup s$
 7:     for all $eff \in E$ do
 8:         $pre \leftarrow \emptyset$                    ▷ Derived pre-condition
 9:         for all $s \in E(eff)$ do
10:             $pre \leftarrow XNOR_P(pre, s)$
11:         $A \leftarrow A \cup \langle pre, eff \rangle$
12:     return $A$
```

TABLE I: Effect XOR and Precondition XNOR operators.

A	B	XOR_E	$XNOR_P$
0	0	0	-1
0	1	1	0
1	0	-1	0
1	1	0	1

transition in this group of transitions with the same effect, it must be a necessary predicate to execute this action (Lines 9–11). Similar to our XOR_E operation $XNOR_P$ uses 1 to represent *positive preconditions* and -1 to represent negative preconditions. The behavior of both XOR_E and $XNOR_P$ is summarized in Table I. With this process, we compute all the elements of a PDDL action, and call this process Action Learner, as illustrated in Figure 4. Using the Action Learner we can output a PDDL domain with a compressed number of actions. Finally, to be able to plan using this computed domain, we must have a planning problem. We compute a planning problem by seeding an image that is the initial state and an image that is the desired state (goal).

Having computed a planning problem and derived a PDDL domain, we must now setup a goal recognition problem. Following Section II-A, we represent a goal recognition problem by the tuple $\mathcal{P}_{GR} = \langle \mathcal{D}, \mathcal{F}, \mathcal{I}, \mathcal{G}, O \rangle$. We already computed the domain $\mathcal{D}$ in Algorithm 1, and the facts $\mathcal{F}$ represented by the 36 bits in latent space representation. We compute the initial state $\mathcal{I}$, a set of candidate goals $\mathcal{G}$, and finally a set of observations O. To compute $\mathcal{I}$ and the set of goals $\mathcal{G}$, we use the image representations of these states and convert them to latent representation. To derive the observations O, we take pairs of images representing of the environment. These images are encoded to the latent representation, and then by using the PDDL domain we extracted, we compute which action from the PDDL domain was responsible for such state transition. After building a goal recognition problem, we can now apply off-the-shelf goal recognition techniques, such as [7,9,18,19]. The output of such techniques is the goal with highest probability of being the correct one, in the latent space representation. We then decode the inferred goal, obtaining its image representation using the decoder. This process is illustrated in Figure 5.

V. Experiments

A. Datasets

In order to evaluate our approach to goal recognition, we generated a number of image-based datasets based on existing goal recognition problems [12,35]. We have two main experimental objectives: first, we want to compare the performance of goal recognition approaches using domain knowledge built by hand with that of automatically-learned domain knowledge; second, we want to evaluate the performance of various approaches to goal recognition using the automatically-learned domain knowledge. Our evaluation dataset thus consists of a number of goal recognition problems generated by taking the generated PDDL domains and an image that serves as both the initial state of the problem $\mathcal{I}$, and the starting point from

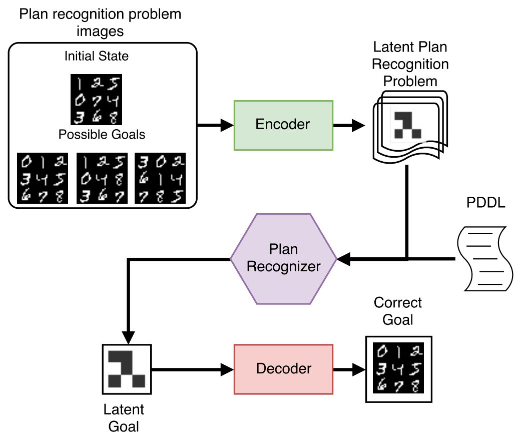

Fig. 5: Plan recognition process.

which we generate a trace of images that correspond to the steps of a plan to achieve a goal. In order to generate such traces, we use a standard PDDL planner [36] to search for a plan for a set of randomly generated goals.From the resulting traces, we can generate the observations at various levels of observability by omitting the states resulting from a percentage of the actions generated by the planner.

Using this method to produce experimental datasets, we generated PDDL domains and images for six different datasets:

- three variations of the 8-Puzzle, whose goal to order a set of pieces when you can only move the blank space:
  - the MNIST 8-puzzle uses the handwritten digits from the MNIST dataset as the pieces of the puzzle, with the number 0 representing the blank space, as illustrated in Figure 6a—every image of the dataset uses the same handwritten digit for every repeating number;
  - the Mandrill 8-puzzle uses the image of a Mandrill, shown in Figure 6b—we use the mandrill's right eye as the blank space;
  - the Spider 8-puzzle uses the image of a Spider, shown in Figure 6c—like the mandrill data set, we use the spider's right eye as the blank space;
- two variations of the Lights-out puzzle game [37], which consists of a 4 by 4 grid of lights that can be turned on and off, and which starts with a random number of lights initially on—toggling any of the lights also toggles every adjacent light—the objective is to turn every light off;
  - lights-out digital (LO Digital) is a standard lights out representation using crosses to represent when a light is on, illustrated in Figure 6d;
  - lights-out twisted (LO Twisted) is a variation of the digital version of lights out such that the image representation undergoes a distortion filter, twisting the exact position of each light, as seen in Figure 6e; and
- the Tower of Hanoi puzzle consists of stacked disks of different sizes and stakes—the objective is to move every disk to a different stack, and we we use a version of

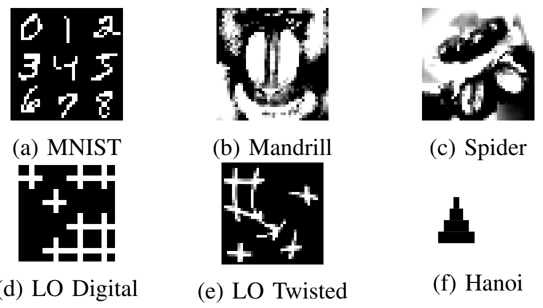
(a) MNIST (b) Mandrill (c) Spider (d) LO Digital (e) LO Twisted (f) Hanoi

Fig. 6: Sample state for each domain.

the puzzle with three disks and four stakes illustrated in Figure 6f.

B. Domain Encoding

Table II shows the performance of our approach in learning the latent representation and inferring PDDL actions for each domain. We measure this performance in terms of the accuracy with which the autoencoder distinguishes real transitions in the underlying domain (SAE Accuracy %) and the degree of redundancy in the inferred PDDL actions (PDDL Redundancy). SAE accuracy measures the percentage of the number of transitions from the original domain (total transitions) that was captured by the encoded domain (encoded transitions), whereas PDDL redundancy measures the ratio between inferred actions (Computed actions) and ground actions in the original domain (Total actions). The closer to the real number of transitions the better, since it means the autoencoder is capable of completely distinguishing all possible transitions of a domain. When a domain includes too many redundant state representations, some of the bits in the latent representation tend to become meaningless, and thus constitute noise for all algorithms that rely on that representation. Similarly, when we generate redundant transitions and actions, these become noise for the goal recognition algorithms that need to deal with the computed domain model. The MNIST, Mandrill and spider datasets all represent the same problem, the N-Puzzle, however the redundancy in the generated PDDL is largely different. The MNIST generated domain is much more redundant, As we can see in Table II, the state autoencoder generated a distinct representation for each of the transitions in most of the domains, with the exception of the MNIST 8-puzzle domain, where $0,4\%$ of the transitions were duplicates. The MNIST, Mandrill and spider datasets all represent the same N-Puzzle problem, however the redundancy in the generated PDDL differs widely. The MNIST generated domain is much more redundant, meaning there are more overlapping actions that could be pruned than the other domains, likely caused by the inability of the autoencoder to generate a distinct representation for each transition.

C. Goal Recognition

To test the ability of our approach to recognize goals using only images, we created a dataset of goal recognition problem with only images. This dataset consists of 6 distinct problems for each domain, where each problem has at least 4 distinct candidate goals. From each of these problems (i.e. the initial states and candidate goals), we generate 5 different conditions for the goal recognition algorithm, by altering the level of observability available to the algorithm. We set five different percentages of observability: 100%, 70%, 50%, 30% and 10%. The observations from the Dataset described in Section V-A are pruned so that only the specified fraction of the original observations are left. We use three goal recognition approaches to evaluate our approach: the landmark-based heuristics h_{gc} (Goal Completion Heuristic) and h_{uniq} (Uniqueness Heuristic) developed by Pereira, Oren, and Meneguzzi (POM in the table) in [9], and the most accurate approach developed by Ramírez and Geffner in [18] (RG in the table). These three approaches are the current state-of-the-art in goal and plan recognition in terms of time and accuracy, respectively.

Table IV summarizes goal recognition performance using our latent representation and learned PDDL encoding for all domains in the dataset and three different goal recognition approaches. Each row of this table shows averages for the number of candidate goals $|\mathcal{G}|$; the percentage of the plan that is actually observed (&) Obs; the average number of observations per problem $|O|$; and, for each goal recognition approach, the time in seconds to recognize the goal given the observations; the Accuracy $\%$ with which the approaches correctly infer the hidden goal; and Spread in $\mathcal{G}$, representing the average number of returned goals. As we can see, the approaches differ widely in accuracy and time elapsed. While the RG approach has better accuracy, it does so with a large spread and long execution times. This trade-off is highlighted in the most complex domains, such as Lights out digital and lights out twisted. For comparison, Table III shows the results of solving these problems with hand made PDDL domains. Since there is no learning inaccuracies in such domains, the results are often superior than the learned models. However, in the lights out model, we can see that the approaches also struggle with a high amount of spread.

VI. Related Work

Asai and Fukunaga [12] develop a planning architecture capable of planning using only pairs of images (representing, respectively, the initial and goal states) from the domain by converting the images into a latent space representation. Their architecture consists of a variational autoencoder (VAE) followed by an off-the-shelf planning algorithm. The architecture convert images into discrete latent vectors using the VAE, and uses the information in such latent vectors to plan over the images and find a sequence of actions that transforms the state into one matching the goal image.

Although our work is based on the one proposed by Asai and Fukunaga, there are two main differences between them. First, they do not create a PDDL domain file, instead they train a neural network to act as an action discriminator. This action discriminator is responsible for defining which actions can be performed in each state, and validating them.

TABLE II: PDDL generation performance for each domain.

Domain	Total Transitions	Encoded Transitions	SAE Accuracy %	Computed Actions	Ground Actions	PDDL Redundancy
MNIST	967680	963795	99.6%	4946	192	25.76
Mandrill	967680	967680	100.0%	495	192	2.578
Spider	967680	967680	100.0%	763	192	3.974
LO Digital	1048576	1048576	100.0%	5940	1392	4.267
LO Twisted	1048576	1048576	100.0%	12669	1392	9.101
Hanoi	237	237	100.0%	211	38	5.552

TABLE III: Experimental results on Goal Recognition using handmade domains.

				POM (h_{gc})			POM (h_{uniq})			RG		
Domain	$\|\mathcal{G}\|$	(%) **Obs**	$\|O\|$	**Time (s))** θ (0 / 10)	**Accuracy** % θ (0 / 10)	**Spread in** $\mathcal{G}$ θ (0 / 10)	**Time (s)** θ (0 / 10)	**Accuracy** % θ (0 / 10)	**Spread in** $\mathcal{G}$ θ (0 / 10)	**Time (s)**	**Accuracy** %	**Spread in** $\mathcal{G}$
Hanoi	4.0	10	1.6	0.010 / 0.012	66.6% / 100.0%	1.6 / 2.3	0.008 / 0.008	66.6% / 66.6%	1.6 / 2.0	0.075	33.3%	1.3
		30	4.0	0.011 / 0.012	66.6% / 100.0%	1.0 / 1.3	0.009 / 0.009	100.0% / 100.0%	1.0 / 1.6	0.080	100.0%	2.3
		50	6.3	0.012 / 0.013	66.6% / 100.0%	1.0 / 1.6	0.009 / 0.010	66.6% / 66.6%	1.0 / 2.0	0.085	100.0%	1.3
		70	8.6	0.013 / 0.013	100.0% / 100.0%	1.3 / 1.3	0.010 / 0.010	66.6% / 66.6%	1.3 / 1.6	0.091	100.0%	1.3
		100	11.6	0.013 / 0.013	100.0% / 100.0%	1.6 / 2.0	0.011 / 0.011	100.0% / 100.0%	1.3 / 1.6	0.098	100.0%	1.3
8-Puzzle	6.0	10	1.0	0.098 / 0.111	16.6% / 33.3%	1.0 / 2.6	0.074 / 0.080	33.3% / 33.3%	2.6 / 2.6	0.179	100.0%	4.8
		30	3.0	0.109 / 0.120	66.6% / 100.0%	1.1 / 2.3	0.079 / 0.085	83.3% / 83.3%	1.0 / 2.5	0.188	100.0%	1.3
		50	4.0	0.117 / 0.129	66.6% / 100.0%	1.0 / 2.0	0.088 / 0.091	100.0% / 100.0%	1.1 / 1.6	0.191	100.0%	1.3
		70	5.3	0.121 / 0.135	100.0% / 100.0%	1.0 / 1.8	0.092 / 0.100	100.0% / 100.0%	1.0 / 1.0	0.210	100.0%	1.0
		100	7.3	0.133 / 0.141	100.0% / 100.0%	1.0 / 1.1	0.108 / 0.110	100.0% / 100.0%	1.0 / 1.0	0.246	83.3%	1.1
Light-Out	6.0	10	1.0	0.689 / 0.766	33.3% / 66.6%	1.3 / 3.8	0.571 / 0.602	33.3% / 66.6%	1.3 / 4.1	5.76	100.0%	5.6
		30	1.6	0.721 / 0.780	50.0% / 83.3%	1.6 / 4.5	0.590 / 0.682	50.0% / 83.3%	1.3 / 5.0	5.79	100.0%	5.3
		50	2.6	0.788 / 0.811	33.3% / 100.0%	2.6 / 5.3	0.622 / 0.704	33.3% / 83.3%	2.6 / 5.3	5.82	100.0%	5.4
		70	3.6	0.804 / 0.849	66.6% / 100.0%	3.8 / 5.0	0.669 / 0.742	66.6% / 83.3%	3.8 / 5.0	5.90	100.0%	5.3
		100	4.3	0.875 / 0.956	100.0% / 100.0%	4.6 / 6.0	0.798 / 0.833	100.0% / 100.0%	4.6 / 5.3	5.93	100.0%	4.8

TABLE IV: Experimental results on Goal Recognition in Latent Space.

				POM (h_{gc})			POM (h_{uniq})			RG		
Domain	$\|\mathcal{G}\|$	(%) **Obs**	$\|O\|$	**Time (s))** θ (0 / 10)	**Accuracy** % θ (0 / 10)	**Spread in** $\mathcal{G}$ θ (0 / 10)	**Time (s)** θ (0 / 10)	**Accuracy** % θ (0 / 10)	**Spread in** $\mathcal{G}$ θ (0 / 10)	**Time (s)**	**Accuracy** %	**Spread in** $\mathcal{G}$
MNIST	6.0	10	1.2	0.591 / 0.603	40.0% / 80.0%	1.6 / 4.0	0.555 / 0.562	40.0% / 60.0%	1.6 / 3.2	21.25	100.0%	6.0
		30	3.0	0.612 / 0.625	40.0% / 80.0%	1.4 / 2.8	0.587 / 0.599	20.0% / 80.0%	1.4 / 3.0	22.26	100.0%	4.8
		50	4.0	0.673 / 0.677	60.0% / 100.0%	2.2 / 3.0	0.609 / 0.628	60.0% / 80.0%	2.2 / 2.8	22.48	100.0%	4.8
		70	5.8	0.698 / 0.703	100.0% / 100.0%	2.4 / 3.0	0.631 / 0.654	60.0% / 100.0%	2.4 / 3.6	23.53	100.0%	3.2
		100	7.8	0.724 / 0.730	100.0% / 100.0%	2.4 / 3.0	0.676 / 0.681	80.0% / 100.0%	2.4 / 3.0	26.34	100.0%	3.4
Mandrill	6.0	10	1.8	0.013 / 0.014	16.6% / 83.3%	1.0 / 3.8	0.011 / 0.012	16.6% / 33.3%	1.1 / 2.8	1.02	83.3%	5.6
		30	4.8	0.015 / 0.017	16.6% / 100.0%	1.0 / 4.8	0.013 / 0.014	20.0% / 83.3%	1.1 / 4.0	1.38	83.3%	3.8
		50	6.0	0.018 / 0.018	33.3% / 83.3%	1.1 / 4.8	0.015 / 0.016	16.6% / 83.3%	1.1 / 4.8	1.44	83.3%	4.1
		70	8.1	0.020 / 0.021	50.0% / 83.3%	1.3 / 4.3	0.016 / 0.018	33.3% / 83.3%	1.3 / 4.0	1.68	66.6%	1.8
		100	11.3	0.022 / 0.023	66.6% / 100.0%	1.8 / 5.16	0.019 / 0.020	33.3% / 100.0%	2.1 / 4.5	1.71	66.6%	1.8
Spider	6.0	10	1.5	0.166 / 0.178	33.3% / 66.6%	2.3 / 4.8	0.151 / 0.154	33.3% / 66.6%	2.3 / 4.5	1.35	83.3%	4.1
		30	4.0	0.181 / 0.190	66.6% / 66.6%	4.1 / 5.1	0.159 / 0.162	66.6% / 66.6%	5.3 / 5.3	1.57	83.3%	3.0
		50	5.6	0.193 / 0.199	50.0% / 83.3%	3.5 / 5.5	0.167 / 0.175	50.0% / 66.6%	4.8 / 4.8	1.66	83.3%	2.8
		70	7.5	0.201 / 0.205	83.3% / 83.3%	4.6 / 5.5	0.016 / 0.018	83.3% / 83.3%	4.6 / 5.3	1.79	66.6%	2.3
		100	10.5	0.208 / 0.217	100.0% / 100.0%	5.5 / 6.0	0.019 / 0.020	100.0% / 100.0%	5.8 / 5.8	2.04	66.6%	1.1
LO Digital	6.0	10	1.0	0.831 / 0.902	33.3% / 33.3%	1.5 / 3.0	0.809 / 0.823	16.6% / 50.0%	1.5 / 3.6	42.52	100.0%	6.0
		30	1.6	0.884 / 1.09	33.3% / 66.6%	1.5 / 4.3	0.835 / 0.840	16.6% / 83.3%	1.5 / 4.5	43.07	100.0%	5.5
		50	2.5	0.915 / 1.13	33.3% / 83.3%	1.5 / 4.5	0.848 / 0.854	16.6% / 83.3%	1.6 / 5.0	43.41	83.3%	5.1
		70	3.6	0.970 / 1.19	83.3% / 100.0%	3.6 / 4.5	0.891 / 0.913	83.3% / 100.0%	3.6 / 4.5	43.78	100.0%	4.8
		100	4.3	1.12 / 1.24	100.0% / 100.0%	2.6 / 4.3	0.913 / 0.938	100.0% / 100.0%	2.6 / 3.3	43.91	100.0%	4.8
LO Twisted	6.0	10	1.0	1.16 / 1.21	16.6% / 16.6%	1.0 / 3.0	1.04 / 1.10	16.6% / 33.3%	1.5 / 4.1	121.97	100.0%	5.8
		30	1.6	1.25 / 1.39	16.6% / 50.0%	1.0 / 3.8	1.11 / 1.18	33.3% / 66.6%	1.3 / 5.1	123.92	100.0%	5.0
		50	2.1	1.33 / 1.46	16.6% / 50.0%	1.0 / 4.5	1.26 / 1.29	16.6% / 66.6%	1.5 / 5.0	124.42	100.0%	5.6
		70	3.3	1.48 / 1.50	16.6% / 83.3%	1.0 / 3.3	1.31 / 1.35	16.6% / 100.0%	1.5 / 5.3	127.22	100.0%	5.5
		100	4.3	1.57 / 1.62	100.0% / 100.0%	2.3 / 5.0	1.40 / 1.44	100.0% / 100.0%	2.3 / 5.5	129.99	100.0%	5.5
Hanoi	4.0	10	1.0	0.304 / 0.318	33.3% / 66.6%	1.0 / 2.3	0.293 / 0.299	33.3% / 100.0%	1.0 / 4.0	6.08	100.0%	4.0
		30	3.0	0.316 / 0.320	100.0% / 100.0%	4.0 / 4.0	0.298 / 0.303	100.0% / 100.0%	4.0 / 4.0	6.21	100.0%	4.0
		50	4.3	0.322 / 0.337	100.0% / 100.0%	4.0 / 4.0	0.306 / 0.311	100.0% / 100.0%	4.0 / 4.0	7.01	66.6%	3.3
		70	6.0	0.345 / 0.354	100.0% / 100.0%	4.0 / 4.0	0.310 / 0.319	100.0% / 100.0%	4.0 / 4.0	7.26	100.0%	4.0
		100	8.3	0.354 / 0.362	100.0% / 100.0%	4.0 / 4.0	0.327 / 0.331	100.0% / 100.0%	4.0 / 4.0	8.19	100.0%	4.0

Second, their goal is to plan over the converted images from a domain, whereas we perform a goal recognition over the generated PDDL domain from latent vectors. Thus, we have extended their architecture by allowing both planning and plan recognition tasks over the latent vectors.

Ramírez and Geffner [18] propose planning approaches for goal and plan recognition, and instead of using plan-libraries, they model the problem as a planning domain theory with respect to a known set of candidate goals. This work uses a modified heuristic, an optimal and modified sub-optimal planner to determine the distance to every goal in a set of candidate goals given a sequence of observations. More recently, Pereira, Oren, and Meneguzzi [9] develop landmark-based approaches for goal recognition, more specifically, they develop a two fast and accurate heuristics for goal recognition. Their first approach, called Goal Completion Heuristic, computes the ratio between the number of achieved landmarks and the total number of landmarks for a given candidate goal. The second approach, called Uniqueness Heuristic, uses the concept of landmark uniqueness value, representing the information value of the landmark for a particular candidate goal when compared to landmarks for all candidate goals. Thus, the heuristic estimative provided by this heuristic is the ratio between the sum of the uniqueness value of the achieved landmarks and the sum of the uniqueness value of all landmarks of a candidate goal.

VII. CONCLUSIONS

We developed an approach for goal recognition using image data as evidence, obviating the need for human engineering to create a task for goal recognition. We compared three state-of-the-art approaches of goal recognition to evaluate the domain we derived from image evidence. Empirical evaluation on multiple datasets shows that while we can solve some problems with the same or higher accuracy than hand-coded problems, many other come to within 33% of the accuracy of recognizing such problems. Regardless, our approach allows breakthroughs in goal recognition techniques using planning domains to be used for goal recognition without human domain engineering and using only images as input. Our current approach has two main limitations. First, we need all possible transitions of the domain in order to infer its encoding. Without every single transition of the domain, we can not ensure that the actions generalize enough to every possible state. We plan on solving this using degrees of uncertainty in each predicate when generating the domain. Second, we currently use relatively small images as is input, limiting its applicability to inputs such as video feeds [38].

As future work, we aim to improve pruning of redundant actions in the domain inference process and encode domains with incomplete information to account for the noise and inconsistencies generated by the autoencoder. As an example, the encoded lights out domain has many more actions than its hand-coded counterpart. As the problems become more complex, we expect the number of actions to increase. We would like to study ways to detect common patterns in the computed actions and merge actions that fit in these patterns. The main benefit of such patterns would be to either derive a state-variable (SAS^+) representation or generalize the actions to lifted versions of themselves. Finally, we would like to develop plan recognition algorithms for incomplete domain models in order to cope with the inaccuracies of the PDDL inference algorithm, and to be able to just sample the space of possible transitions in a domain, rather than having to generate all of them.

Acknowledgments

Felipe thanks CNPq for partial financial support under its PQ fellowship, grant number 305969/2016-1.

References

[1] H. A. Kautz and J. F. Allen, "Generalized Plan Recognition." in *AAAI*, 1986, pp. 32–37.

[2] D. Avrahami-Zilberbrand and G. A. Kaminka, "Fast and complete symbolic plan recognition," in *Proceedings of the 19th international joint conference on Artificial intelligence*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2005, pp. 653–658.

[3] C. W. Geib and M. Steedman, "On natural language processing and plan recognition," in *Proceedings of the 20th IJCAI*, 2007, pp. 1612–1617.

[4] F. Meneguzzi and J. Oh, "Proactive Assistant Agents: Papers from the AAAI Fall Symposium," Arlington, Virginia, Tech. Rep., 2010.

[5] M. Fagundes, F. Meneguzzi, R. H. Bordini, and R. Vieira, "Dealing with ambiguity in plan recognition under time constraints," in *Proceedings of the Thirteenth International Conference on Autonomous Agents and Multiagent Systems*, 2014, pp. 389–396.

[6] Y. E. Martín, M. D. R. Moreno, and D. E. Smith, "A Fast Goal Recognition Technique Based on Interaction Estimates." *IJCAI*, 2015.

[7] S. Sohrabi, A. V. Riabov, and O. Udrea, "Plan Recognition as Planning Revisited." in *International Joint Conference on Artificial Intelligence*, 2016, pp. 3258–3264.

[8] R. F. Pereira and F. Meneguzzi, "Landmark-based Plan Recognition," in *European Conference on Artificial Intelligence*, 2016, pp. 1706–1707.

[9] R. F. Pereira, N. Oren, and F. Meneguzzi, "Landmark-Based Heuristics for Goal Recognition," in *Proceedings of the 32st AAAI Conference on Artificial Intelligence*, 2017.

[10] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised Learning with Deep Generative Models," in *Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589.

[11] M. Fox and D. Long, "PDDL2. 1: An extension to PDDL for expressing temporal planning domains." *Journal of Artificial Intelligence Research*, 2003.

[12] M. Asai and A. Fukunaga, "Classical Planning in Deep Latent Space: From Unlabeled Images to PDDL (and back)," in *Workshop on Knowledge Engineering for Planning and Scheduling*, 2017, pp. 27–35.

[13] J. Oh, F. Meneguzzi, K. P. Sycara, and T. J. Norman, "An Agent Architecture for Prognostic Reasoning Assistance," in *IJCAI*, 2011, pp. 2513–2518.

[14] J. Oh, F. Meneguzzi, and K. Sycara, "Probabilistic plan recognition for intelligent information agents: Towards proactive software assistant agents," in *Proceedings of the Third International Conference on Agents and Artificial Intelligence*, 2011, pp. 281–287.

[15] G. Sukthankar, R. P. Goldman, C. Geib, D. V. Pynadath, and H. H. Bui, *Plan, Activity, and Intent Recognition: Theory and Practice*. Elsevier, 2014.

[16] C. W. Geib and R. P. Goldman, "A probabilistic plan recognition algorithm based on plan tree grammars." *Artif. Intell.*, vol. 173, no. 11, pp. 1101–1132, 2009.

[17] R. Mirsky, R. Stern, Y. K. Gal, and M. Kalech, "Sequential Plan Recognition." in *International Joint Conference on Artificial Intelligence*, 2016.

[18] M. Ramírez and H. Geffner, "Plan recognition as planning," in *International Joint Conference on Artificial Intelligence*, 2009, pp. 1778–1783.

[19] M. Ramírez and H. Geffner, "Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners," in *AAAI*, 2010, pp. 1121–1126.

[20] S. Keren, A. Gal, and E. Karpas, "Goal Recognition Design." in *International Conference on Automated Planning and Scheduling*, 2014, pp. 154–162.

[21] R. Fikes and N. Nilsson, "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving," *Artificial Intelligence*, vol. 2, no. 3-4, pp. 189–208, 1971.

[22] J. Hoffmann, J. Porteous, and L. Sebastia, "Ordered Landmarks in Planning," *Journal of Artificial Intelligence Research*, vol. 22, no. 1, pp. 215–278, Apr. 2004.

[23] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016.

[24] G. Hinton and R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, 2006.

[25] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," 2017.

[26] D. P. Kingma and M. Welling, "Auto-Encoding Variational Bayes," *arXiv.org*, Dec. 2013.

[27] C. J. Maddison, D. Tarlow, and T. Minka, "A
ast sampling," in *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2014, pp. 3086–3094.

[28] E. J. Gumbel, *Statistical theory of extreme values and some practical applications: a series of lectures*, ser. Applied mathematics series. U. S. Govt. Print. Office, 1954.

[29] S. Jiménez, T. de la Rosa, S. Fernández, F. Fernández, and D. Borrajo, "A review of machine learning for automated planning." *Knowledge Engineering Review*, vol. 27, no. 4, pp. 433–467, 2012.

[30] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders." in *25th International Confer- ence on Machine Learning*. New York, New York, USA: ACM Press, 2008, pp. 1096–1103.

[31] H. Geffner and B. Bonet, *A Concise Introduction to Models and Methods for Automated Planning*. A Concise Introduction to Models and Methods for Automated Planning, 2013, vol. 7.

[32] R. F. Pereira, N. Oren, and F. Meneguzzi, "Monitoring plan optimality using landmarks and domain-independent heuristics," in *The AAAI 2017 Workshop on Plan, Activity, and Intent Recognition (PAIR)*, 2017.

[33] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 1097–1105.

[34] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, G. Gordon, D. Dunson, and M. Dudk, Eds., vol. 15. Fort Lauderdale, FL, USA: PMLR, 11–13 Apr 2011, pp. 315–323.

[35] R. F. Pereira and F. Meneguzzi, "Goal and plan recognition datasets using classical planning domains," Tech. Rep., Jul. 2017.

[36] M. Helmert, "The fast downward planning system," *Journal of Artificial Intelligence Research*, vol. 26, pp. 191–246, 2006.

[37] R. Fleischer and J. Yu, *A Survey of the Game "Lights Out!"*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 176–198.

[38] R. Granada, R. F. Pereira, J. Monteiro, R. Barros, D. Ruiz, and F. Meneguzzi, "Hybrid Activity and Plan Recognition for Video Streams," in *The AAAI 2017 Workshop on Plan, Activity, and Intent Recognition*, 2017.