An LSTM-Based Approach for Goal Recognition in Latent Space

Leonardo Amado*, **João Paulo Aires***, **Ramon Fraga Pereira***,
Maurício C. Magnaguagno*, **Roger Granada*** and **Felipe Meneguzzi**†
Pontifical Catholic University of Rio Grande do Sul (PUCRS), Brazil
Postgraduate Programme in Computer Science, School of Technology
*{leonardo.amado, joao.aires.001, ramon.pereira,
mauricio.magnaguagno, roger.granada}@acad.pucrs.br
†felipe.meneguzzi@pucrs.br

Abstract

Approaches to goal recognition have progressively relaxed the requirements about the amount of domain knowledge and available observations, yielding accurate and efficient algorithms capable of recognizing goals. However, to recognize goals in raw data, recent approaches require either human engineered domain knowledge, or samples of behavior that account for almost all actions being observed to infer possible goals. This is clearly too strong a requirement for real-world applications of goal recognition, and we develop an approach that leverages advances in recurrent neural networks to perform goal recognition as a classification task, using encoded plan traces for training. We empirically evaluate our approach against the state-of-the-art in goal recognition with image-based domains, and discuss under which conditions our approach is superior to previous ones.

1 Introduction

Goal recognition is the task of identifying the intended goal of an agent under observation by analyzing the agent behavior in an environment. Initial approaches on goal recognition were based on planning theories, which require a substantial amount of domain knowledge (Kautz and Allen 1986). Subsequent approaches have gradually relaxed such requirements using expressive planning and plan-library-based formalisms (Avrahami-Zilberbrand and Kaminka 2005; Geib and Steedman 2007) as well as achieving different levels of accuracy and amount of information available in observations required to recognize goals (Martín, Moreno, and Smith 2015; Sohrabi, Riabov, and Udrea 2016; Pereira and Meneguzzi 2016; Pereira, Oren, and Meneguzzi 2017). Recent work on goal recognition in latent space (Amado et al. 2018) overcomes the need of a domain expert tailored domain knowledge by building planning domain knowledge from raw data and using such domain knowledge on traditional goal recognition techniques (Pereira, Oren, and Meneguzzi 2017) to infer goals from image data. However, to build this domain knowledge, their approach requires a substantial amount of training data to create a complete PDDL domain. In this paper, we try to mitigate this problem by applying a recurrent neural network to solve the task of goal recognition directly rather than to use the training data to generate domain knowledge. Our main goal is to reduce the amount of training data necessary to correctly infer the intended goal of an agent by leveraging a Long short-term memory (LSTM) network. Long short-term networks (Hochreiter and Schmidhuber 1997) are capable of solving classification problems by receiving streams of data and returning a class based on the entirety of the data received. These streams of data can be used to model the actions of an agent under observation in goal recognition problems, where the class to be recognized by the LSTM network is the agent's goal.

Our main contributions are twofold. First, we develop an end-to-end machine learning technique for goal recognition based on training an LSTM network in Section 2.4. Second, we empirically compare the resulting approach with traditional goal recognition approaches (Ramírez and Geffner 2009; Pereira, Oren, and Meneguzzi 2017) in Section 4, discuss how our approach relates to the current state of the art in Section 5, and in Section 6, discuss the trade-offs between using machine learning exclusively or combining traditional techniques with machine learning.

2 Background

2.1 Goal Recognition

Goal recognition is the task of recognizing the intended goal that an agent (software or human) aims to achieve from observations of its acting in an environment (Sukthankar et al. 2014). Observations can be either a sequence of actions performed by the agent or the consequences of such actions, more specifically, properties as logical facts (*e.g.*, at home, resting). Furthermore, observations can be either seen as a full sequence of actions or a partial subsequence of actions performed by an agent in an environment. Plan recognition is a related task to goal recognition, however, the objective of this task is recognizing the plan (*i.e.*, sequence of actions) that an observed agent is executing to achieve a particular goal (Sukthankar et al. 2014). Goal and plan recognition in real-world data assume an underlying processing step that translates raw sensor data into some kind of symbolic representation (Sukthankar et al. 2014), as well as a model of the observed agent's behavior generation mechanism.

2.2 Planning

We use planning domain theories to formalize agents' behavior and the environment description, following the STRIPS formalism proposed by Fikes and Nilsson (1971). A domain model is a tuple $\mathcal{D} = \langle \mathcal{R}, \mathcal{O} \rangle$, where: $\mathcal{R}$ is a set of predicates with typed variables. Predicates can be associated to objects in a concrete problem (*i.e.*, grounded) representing logical values. 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 ($\bot$). 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 $\mathit{eff}^+(op)$ (add list) and negative effects $\mathit{eff}^-(op)$ (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 instantiated action a 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. Most automated planners use the *Planning Domain Definition Language* (PDDL) as a standardized domain and problem representation medium (McDermott et al. 1998), which encodes the formalism described thus far.

We follow the definition from Ramírez and Geffner (2009; 2010) to formalize the problem of goal recognition as planning. A goal recognition problem as planning 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 percentage of the actions actually executed to achieve G^* could not be executed.

2.3 Planning in Latent Space

Most planning algorithms are based on the *factored* transition function $\gamma(S, a)$ that represents states as discrete facts. This transition function is usually encoded manually by a domain expert, and virtually all existing goal and plan recognition approaches require varying degrees of domain knowledge in order to recognize from observations. Automatically generating of such domain knowledge involves at least two processes: (1) converting real-world data into a factored representation (*i.e.*, the predicates in $\mathcal{R}$); and (2) 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 (Jiménez et al. 2012), almost no approaches have addressed the problem of generating domain models from real world data. Recently, Asai and Fukunaga (2017) developed an approach to planning that 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. This approach uses an autoencoder (Vincent et al. 2008) 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. Basically, the latent representation is a matrix of bits, and the latent space is every possible combination of bits that we can represent in this $n \times m$ matrix. When m is two, the output of this auto-encoder 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).

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 (Asai and Fukunaga 2017) 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 (Asai and Fukunaga 2017) shows that heuristics from the planning literature (Geffner and Bonet 2013, Chapter 3) are still applicable, however, given the propositional nature of the encoding, they are not so informative. Such lack of informativeness provides a challenge to the application of goal and plan recognition approaches in latent-space. As we see in Section 2.4, in order to successfully employ efficient goal recognition approaches, we need not only to learn a consistent latent representation of states, but also to use the propositional transition function induced from state pairs to generate STRIPS-style operators.

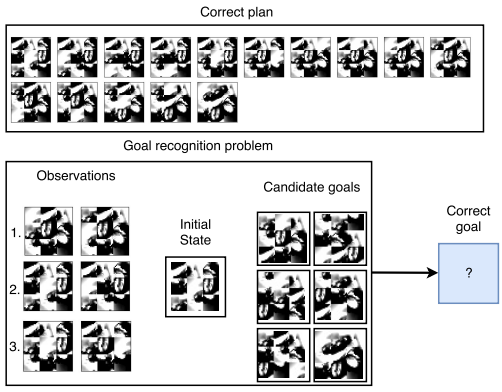


Figure 1: Image goal recognition problem.

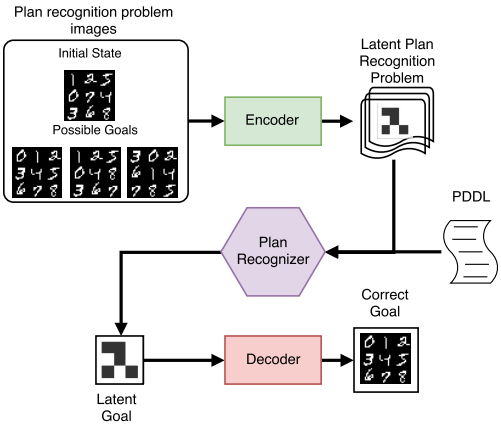


Figure 2: Image goal recognition process.

2.4 Goal Recognition in Latent Space

Goal recognition in Latent Space is a technique to apply classical goal recognition algorithms in raw data (such as images) by converting it into a latent representation (Amado et al. 2018). In Figure 1, we provide an example of the goal recognition problem in image domains. We want to infer what 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 in such task is not trivial, as the small number of observations provide little information.

To recognize goals in image based domains, Amado et al. (2018) proposed four steps. First, we must develop an autoencoder capable of creating a latent representation to a state of such image domain. Second, since classical goal recognition approaches require a PDDL domain, we need a technique capable of extracting a PDDL domain from the latent representation of the transition of the domain. 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 $\langle \mathcal{D}, \mathcal{F}, \mathcal{I}, \mathcal{G}, O \rangle$

The encoded representation can be achieved by using the an autoencoder similar to the one described by Asai and Fukunaga (2017) with the Gumbel softmax (Gumbel 1954) activation function. Each domain requires one autoencoder capable to converting an image state of the domain to a latent representation. We preprocessed these images by applying a grayscale filter and then binarizing the resulting image. With a trained autoencoder for each domain it is possible to output a PDDL domain using the Action Learner develop by Amado et al. (2018). This PDDL domain will have a compressed number of actions to improve the speed of the goal recognition process.

Following Section 2.2, we represent a goal recognition problem by the tuple $\mathcal{P}_{GR} = \langle \mathcal{D}, \mathcal{F}, \mathcal{I}, \mathcal{G}, O \rangle$. We extract the domain $\mathcal{D}$ using the Action Leaner, and the facts $\mathcal{F}$ represented by the 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 using the trained autoencoder. 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 (Ramírez and Geffner 2009; Ramírez and Geffner 2010; Sohrabi, Riabov, and Udrea 2016; Pereira, Oren, and Meneguzzi 2017). 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 2.

2.5 Long Short-Term Memory Networks

A Recurrent Neural Network (RNN) is a network that attempts to model a sequence of dependent events occurring through time such as financial time series (Akita et al. 2016), language modeling (Sundermeyer, Ney, and Schlüter 2015) and so on. The recurrence is performed by feeding the input layer of the network at time $t + 1$ with the output of the network layer at time t, keeping a "memory" of the past events. Unfortunately, RNNs suffer with well-known vanishing gradient problem (Bengio, Simard, and Frasconi 1994), *i.e.*, the gradients that are backpropagated thought the network during the training phase tend to decay or grow exponentially. Therefore, as dependencies in RNNs get longer, the gradient calculation becomes unstable, limiting the network to learn long-range dependencies.

In order to get rid of the vanishing gradient problem, Hochreiter and Schmidhuber (1997) propose an RNN architecture called Long Short-Term Memory (LSTM) network that modifies the original recurrent cell such that vanishing and exploding gradients are avoided, whereas the training algorithm is left unchanged. An LSTM cell contains mainly

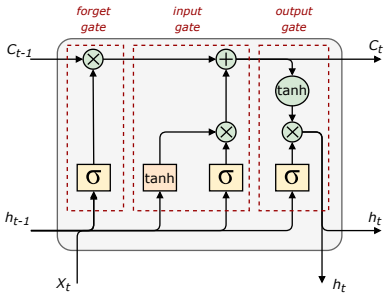


Figure 3: Internal structure of a LSTM cell.

four components called cell state, forget gate, input gate and output gate. The cell state (C) is responsible for passing the information through the cell to the next LSTM cell, while being changed by the gates. The forget gate decides what information should be forget from the previous cell state. This gate contains a *sigmoid* (σ) layer that outputs a number between 0 and 1, where 1 means "keep all information" and 0 means "get rid of this information". Input gate decides what information should be stored in the cell state by applying a *sigmoid* layer to decide what information to keep and a hyperbolic tangent (*tanh*) layer to select new candidates to the cell state, performing an update to the cell state. Finally, the output gate decides what information should be propagated forward by performing a pointwise multiplication of a *sigmoid* layer, which decides what part of the input should be forwarded, and a the cell state filtered by a *tanh* operation. Figure 3 illustrates the LSTM cell with its respective gates, where yellow boxes represent layers, elements in green represent pointwise operations ($\otimes$ pointwise multiplication, $\oplus$ pointwise addition and *tanh* pointwise hyperbolic tangent function), merging arrows represent the concatenation of elements and forking arrows represent the copy of the content to multiple points.

Therefore, an LSTM performs a classification problem by receiving a streamline of ordered data as input and returning a class based on the data sequence received. In this work, we use plan traces as input sequences and their corresponding goals as training class when training the network. Hence, the network learns the agent's goal based on the sequence of actions performed by the agent.

3 Goal Recognition in Latent Space using LSTM

Current approaches to recognize goals in latent space require enough data to build a complete PDDL domain (Amado et al. 2018). To avoid the need of such high amount of domain knowledge, we propose the usage of a machine learning model capable of recognizing goals using only plan traces as training data.

Our LSTM consists of three main layers. First, we use an embedding layer to convert our input sequence into a dense representation with a dimension of 1000 that will feed the LSTM units. Second, we use an LSTM layer containing 512 units. Finally, a fully connected layer receives the output from LSTM and generates the goal representation with 36 output neurons. We use sigmoid activation on the neurons from the output layer and a binary cross entropy loss using RMSprop as optimizer. Figure 4 illustrates our LSTM architecture.

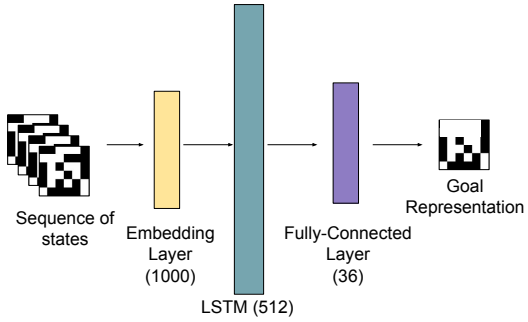


Figure 4: LSTM Architecture

In order to create a model to recognize goals, we train an LSTM that receives a sequence of encoded states and predicts an encoded goal. To perform a fair comparison to the state-of-the-art, we use as input encoded states generated by the encoder module from the autoencoder created by Asai and Fukunaga (2017). Thus, we convert each image-state into a latent representation (a 6x6 binary matrix). Figure 5 illustrates the process of training and testing our LSTM model, we highlight three main steps of such process. First, given a set of image-states representing a sequence of states and the goal of a certain plan, we use the encoder to generate the latent representation for each image. Second, using the representations, we train the LSTM to predict the goal given the states. The output is a representation of this goal. Finally, we use the decoder from Asai and Fukunaga autoencoder to convert the produced representation into an image.

To train the LSTM network, we require data extracted from plans for each domain. We use plan traces generated by Amado et al. (2018), observing the states that were reached in each plan. Each trace generated a list of states, and then we included the goal of each trace as a class to the

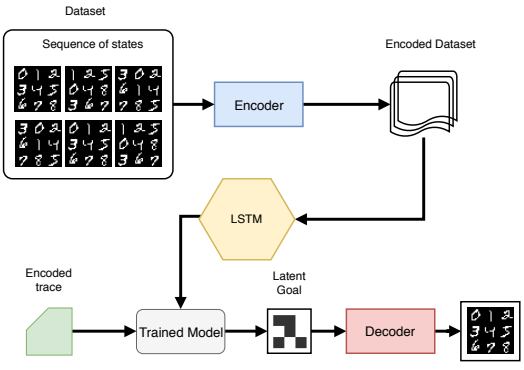


Figure 5: Goal recognition using LSTMs

LSTM. To improve accuracy in low observability scenarios, we included partially observable traces (which means some states were removed from the plan trace), including 10%, 30%, 50%, 70% of observability. During the training phase, we use early stopping to avoid overfitting and set a limit of 10,000 epochs. Early stopping monitors validation loss ensuring training will stop when loss stops decreasing.

We manipulate LSTM inputs by converting the latent representations into a specific encoding. In our specific case, we turn each state into an integer number, thus, we differentiate them simplifying the input. An entry example of such model is: *22, 23, 33, 48, 12*, where each number is a specific state from the state-space in its domain and the sequence is an entire plan. The output layer is 36 binary neurons, which we use to rebuild the latent representation by reshaping it into a 6x6 matrix.

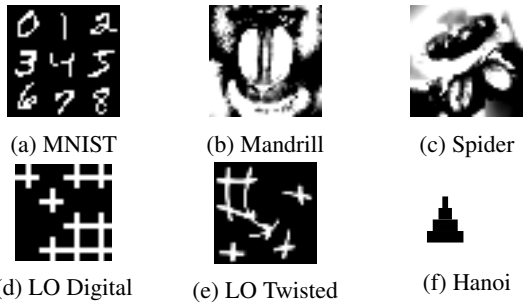
4 Experiments

4.1 Datasets

In order to evaluate our LSTM approach, we generated a number of image-based datasets based on existing goal recognition problems (Pereira and Meneguzzi 2017; Asai and Fukunaga 2017; Amado et al. 2018). We have two main experimental objectives: first, we want to compare the performance of our LSTM approach with the existing approaches to goal recognition in latent space, using problems were the goal of each problem is contained in the dataset; second, we want to evaluate the performance of our LSTM approach when dealing with problems were the goal is not contained as a class in our dataset. Our evaluation dataset thus are two distinct datasets. The first, a dataset containing the exact problems used in (Amado et al. 2018) to evaluate their latent goal recognition approach. The second, a dataset containing new goal recognition problems, with distinct goals where these goals do not appear as a class in the training dataset for the LSTM. In order to generate such traces, we use a standard PDDL planner (Helmert 2006) 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 domains:

- Three variations of the 8-Puzzle, whose goal to order a set of pieces when you can only move the blank space: (1) 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; (2) Mandrill 8-puzzle uses the image of a Mandrill, shown in Figure 6b—we use the mandrill's right eye as the blank space; (3) 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 (Fleischer and Yu 2013), 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; (1) lights-out digital (LO Digital) is a standard lights out representation using crosses to represent when a light is on, illustrated in Figure 6d; (2) 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
- Tower of Hanoi puzzle, which 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 the puzzle with three disks and four stakes illustrated in Figure 6f.



(a) MNIST (b) Mandrill (c) Spider

(d) LO Digital (e) LO Twisted (f) Hanoi

Figure 6: Sample state for each domain.

Table 1 describes our dataset specifications, such as domains, number of traces for each domain, and training time. As we can see, most domains have more than 1000 traces with training times smaller than 5 minutes. As outliers, LO digital and twisted have small number of traces, it occurs because their plans are relatively smaller when compared to the other domains, which limits the variety of traces.

Domain	**# of Traces**	**Training Time (seconds)**
Hanoi	1552	22.57
LO digital	230	294.58
LO twisted	224	64.33
Mandrill	2520	14.87
MNIST	1427	211.42
Spider	2216	333.25

Table 1: Dataset specifications.

4.2 Goal Recognition

To compare our approach with the existing approaches of goal recognition for latent space, we use the exact same dataset used in (Amado et al. 2018). This dataset consists of 6 distinct problems for each domain, where each problem has at least 4 distinct candidate goals. The candidate goals are not necessary for the LSTM. 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.

				POM (h_{gc})			LSTM			RG		
Domain	$\|\mathcal{G}\|$	(%) **Obs**	$\|O\|$	**Time (s))** θ (0 / 10)	**Accuracy** % θ (0 / 10)	**Spread in** $\mathcal{G}$ θ (0 / 10)	**Time (s)**	**Accuracy** %	**Spread in** $\mathcal{G}$	**Time (s)**	**Accuracy** %	**Spread in** $\mathcal{G}$
MNIST	6.0	10	1.2	0.591 / 0.603	33.3% / 83.3%	1.6 / 4.0	0.346	16.6%	1.0	21.25	100.0%	6.0
		30	3.0	0.612 / 0.625	33.3% / 83.3%	1.4 / 2.8	0.335	100.0%	1.0	22.26	100.0%	4.8
		50	4.0	0.673 / 0.677	60.0% / 100.0%	2.2 / 3.0	0.326	100.0%	1.0	22.48	100.0%	4.8
		70	5.8	0.698 / 0.703	100.0% / 100.0%	2.4 / 3.0	0.394	100.0%	1.0	23.53	100.0%	3.2
		100	7.8	0.724 / 0.730	100.0% / 100.0%	2.4 / 3.0	0.357	100.0%	1.0	26.34	100.0%	3.4
Mandrill	6.0	10	1.8	0.013 / 0.014	16.6% / 83.3%	1.0 / 3.8	0.335	50%	1.0	1.02	83.3%	5.6
		30	4.8	0.015 / 0.017	16.6% / 100.0%	1.0 / 4.8	0.366	100.0%	1.0	1.38	83.3%	3.8
		50	6.0	0.018 / 0.018	33.3% / 83.3%	1.1 / 4.8	0.389	100.0%	1.0	1.44	83.3%	4.1
		70	8.1	0.020 / 0.021	50.0% / 83.3%	1.3 / 4.3	0.353	100.0%	1.0	1.68	66.6%	1.8
		100	11.3	0.022 / 0.023	66.6% / 100.0%	1.8 / 5.16	0.347	100.0%	1.0	1.71	66.6%	1.8
Spider	6.0	10	1.5	0.166 / 0.178	33.3% / 66.6%	2.3 / 4.8	0.375	83.3%	1.0	1.35	83.3%	4.1
		30	4.0	0.181 / 0.190	66.6% / 66.6%	4.1 / 5.1	0.423	83.3%	1.0	1.57	83.3%	3.0
		50	5.6	0.193 / 0.199	50.0% / 83.3%	3.5 / 5.5	0.431	100.0%	1.0	1.66	83.3%	2.8
		70	7.5	0.201 / 0.205	83.3% / 83.3%	4.6 / 5.5	0.384	100.0%	1.0	1.79	66.6%	2.3
		100	10.5	0.208 / 0.217	100.0% / 100.0%	5.5 / 6.0	0.368	100.0%	1.0	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	0.315	83.3%	1.0	42.52	100.0%	6.0
		30	1.6	0.884 / 1.09	33.3% / 66.6%	1.5 / 4.3	0.317	100.0%	1.0	43.07	100.0%	5.5
		50	2.5	0.915 / 1.13	33.3% / 83.3%	1.5 / 4.5	0.336	100.0%	1.0	43.41	83.3%	5.1
		70	3.6	0.970 / 1.19	83.3% / 100.0%	3.6 / 4.5	0.371	83.3%	1.0	43.78	100.0%	4.8
		100	4.3	1.12 / 1.24	100.0% / 100.0%	2.6 / 4.3	0.330	83.3%	1.0	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	0.376	66.6%	1.0	121.97	100.0%	5.8
		30	1.6	1.25 / 1.39	16.6% / 50.0%	1.0 / 3.8	0.320	100.0%	1.0	123.92	100.0%	5.0
		50	2.1	1.33 / 1.46	16.6% / 50.0%	1.0 / 4.5	0.339	100.0%	1.0	124.42	100.0%	5.6
		70	3.3	1.48 / 1.50	16.6% / 83.3%	1.0 / 3.3	0.312	100.0%	1.0	127.22	100.0%	5.5
		100	4.3	1.57 / 1.62	100.0% / 100.0%	2.3 / 5.0	0.327	100.0%	1.0	129.99	100.0%	5.5
Hanoi	4.0	10	1.0	0.304 / 0.318	33.3% / 66.6%	1.0 / 2.3	0.334	66.6%	1.0	6.08	100.0%	4.0
		30	3.0	0.316 / 0.320	100.0% / 100.0%	4.0 / 4.0	0.365	100.0%	1.0	6.21	100.0%	4.0
		50	4.3	0.322 / 0.337	100.0% / 100.0%	4.0 / 4.0	0.371	100.0%	1.0	7.01	66.6%	3.3
		70	6.0	0.345 / 0.354	100.0% / 100.0%	4.0 / 4.0	0.372	66.6%	1.0	7.26	100.0%	4.0
		100	8.3	0.354 / 0.362	100.0% / 100.0%	4.0 / 4.0	0.329	66.6%	1.0	8.19	100.0%	4.0

Table 2: Experimental results on Goal Recognition in Latent Space.

				POM (h_{gc})			RG		
Domain	$\|\mathcal{G}\|$	(%) **Obs**	$\|O\|$	**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	0.075	33.3%	1.3
		30	4.0	0.011 / 0.012	66.6% / 100.0%	1.0 / 1.3	0.080	100.0%	2.3
		50	6.3	0.012 / 0.013	66.6% / 100.0%	1.0 / 1.6	0.085	100.0%	1.3
		70	8.6	0.013 / 0.013	100.0% / 100.0%	1.3 / 1.3	0.091	100.0%	1.3
		100	11.6	0.013 / 0.013	100.0% / 100.0%	1.6 / 2.0	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	0.179	100.0%	4.8
		30	3.0	0.109 / 0.120	66.6% / 100.0%	1.1 / 2.3	0.188	100.0%	1.3
		50	4.0	0.117 / 0.129	66.6% / 100.0%	1.0 / 2.0	0.191	100.0%	1.3
		70	5.3	0.121 / 0.135	100.0% / 100.0%	1.0 / 1.8	0.210	100.0%	1.0
		100	7.3	0.133 / 0.141	100.0% / 100.0%	1.0 / 1.1	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	5.76	100.0%	5.6
		30	1.6	0.721 / 0.780	50.0% / 83.3%	1.6 / 4.5	5.79	100.0%	5.3
		50	2.6	0.788 / 0.811	33.3% / 100.0%	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	5.90	100.0%	5.3
		100	4.3	0.875 / 0.956	100.0% / 100.0%	4.6 / 6.0	5.93	100.0%	4.8

Table 3: Experimental results on Goal Recognition using handmade domains.

We set five different percentages of observability: 100%, 70%, 50%, 30% and 10%.

The observations from the Dataset described in Section 4.1 are pruned so that only the specified fraction of the original observations are left. We use two goal recognition approaches to compare with our LSTM approach (LSTM in Table 2): the landmark-based heuristics h_{gc} (Goal Completion Heuristic) developed by Pereira, Oren, and Meneguzzi (POM in in Table 2) in (Pereira, Oren, and Meneguzzi 2017), and the most accurate approach developed by Ramírez and Geffner (2009) (RG in in Table 2). We choose this approach instead of the most recent one (Ramírez and Geffner 2010), since the approach from 2009 is faster and has higher accuracy. These two approaches are the current state-of-the-art in goal and plan recognition in terms of time and accuracy, respectively.

Table 2 summarizes goal recognition performance of each approach using the latent representation and learned PDDL encoding provided in (Amado et al. 2018), for all domains in the dataset and three different goal recognition approaches. In the LSTM approach, the learned PDDL is not needed to perform goal recognition, only the encoded traces. In this comparison, every hidden goal was included in the LSTM training set at least once. We guarantee that the traces used in this comparison were not included in the training set. 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. For the LSTM is always one, as it always returns one goal. As we can see, the LSTM achieved overall good accuracy across all domains and observability scenarios. The execution time was between 0.3 and 0.5 seconds. While the RG approach has a good 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. The POM approach also struggled with high spread in some domains, such as the Hanoi domain, but was much faster than RG in all scenarios. Overall the LSTM achieved better results, considering it returns always one goal and the other approaches struggled with high spread. Thus, for recognizing goals that are contained in the training set, the LSTM is a promising approach that does well in both speed and accuracy.

For comparison, Table 3 shows the results of solving these problems with hand made PDDL domains. Since there is no learning inaccuracies in the PDDL of 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.

In Table 4 we display the results when dealing with goals that are not contained in the the training set. The test set consists of 6 distinct problems with distinct goals, where each

problem generates 5 traces using different observability (10, 30, 50, 70, 100%). The LSTM was unable to recognize any of the goals that are not contained in the dataset. We present the reconstruction accuracy, that estimates how close was the LSTM to reconstruct the correct goal. There is no direct comparison to other goal recognition approaches, as there is a training data is used in the other approaches. As we can see, our approach needs the goal to be contained in the training set, as the LSTM network is unable to reconstruct a goal that it has not seen. In such scenarios enumerating every possible goal is not recommended, as the number of possible states (and so goals) in a 8-Puzzle problem is 362,880. Thus our approach by encoding classes for classification is very promising, as long as the training set contains many goals (and thus classes), as it removes the burden of enumerating every classes.

Domain	**Reconstruction Accuracy** (%)	**# Problems**	**Correct Predictions**
MNIST	48,6%	30	0%
Mandrill	53,6%	30	0%
Spider	58,4%	30	0%
LO-Digital	53%	30	0%
LO-Twisted	51,2%	30	0%

Table 4: LSTM results with unknown goals.

5 Related Work

Min et al. (2014) propose a deep LSTM network approach capable of recognizing goals of a player in an educational game scenario. The dataset used for training the deep LSTM is a player behavior corpus consisting of distinctive player actions. The challenge comes from recognizing goals when handling uncertainty from noise input and non-optimal player behavior. The LSTM is able to do standard metric-based goal recognition and online goal recognition, as information is fed. Although this work is very similar to ours, the main difference is that the entirety of the goals are already known in the work proposed by Min et al., while in our work, we try to reconstruct the goal from the observation traces. This is a significant difference, because our approach tries to recognize goals without any domain knowledge from a domain expert, making our approach completely domain independent. The results will vary depending how many times the goal appears in the training data.

Granada et al. (Granada et al. 2017) develop a hybrid approach that combines activity and plan recognition for video streams. This approach uses deep learning to analyze video data (frames) in order to identify individual actions in a scene, and based on this set of identified actions, a plan recognition algorithm then uses a plan library describing possible overarching activities for recognizing the ultimate goal of the subject in a video.

Asai and Fukunaga (2017) 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.

Amado et al. (2018) develop an approach to recognize goals in image domains, by converting images to a latent representation, deriving a PDDL from domain knowledge converted to a latent representation and then applying off-the-shelf recognition algorithms. Our work extends this approach by using an LSTM as a recognizer. The difference is a trade-off between domain knowledge, since the LSTM does not require a PDDL domain, and training dataset using plan traces that is necessary to train the LSTM network.

Ramírez and Geffner (2009) 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 modified heuristics, 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. Recently, Pereira, Oren, and Meneguzzi (2017) develop goal recognition approaches that rely on landmarks, i.e., 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.

6 Conclusions and Discussion

We developed an approach for goal recognition in latent space using an LSTM network, obviating the need for human engineering to create a task for goal recognition. Other approaches require either human engineered domains (Pereira, Oren, and Meneguzzi 2017), or an extensive amount of domain knowledge to build a PDDL domain (Amado et al. 2018). Empirical evaluation on multiple datasets shows that while we can solve some problems with the same or higher accuracy than hand-coded problems, our LSTM approach does not easily generalize for goals outside the training dataset. Nevertheless, our approach provides a meaningful initial step towards goal recognition without human domain engineering and minimal amount of training data. In summary the advantages of using our LSTM approach to recognize goals are: high accuracy and fast prediction time when dealing with known goals; no false positive predictions, given that it only predicts a single goal; no need of a PDDL domain, which requires extensive domain knowledge. However, our approach has the following disadvantages: like most pure machine learning approaches, performance is tied to the robustness of the training dataset;

requires training, which is unnecessary for classical goal recognition approaches; very limited generalizability with small datasets.

As future work, we aim to develop a dataset to test the ability of the different goal recognition approaches in latent space when dealing with noisy observations. In the LSTM case, a noisy observation would be a state in the encoded trace that is not relevant to achieving the desired goal (*i.e.*, an unnecessary step performed by the agent being observed). Since Amado et al. (2018) compute every transition of the domain to generate a complete PDDL domain, we would like to investigate ways to use such information to improve the LSTM performance when dealing with goals that are not contained in the training set. Furthermore, we would like to study ways to improve generalization in our approach. We envision using a percentage of the encoded transitions as a pre-training mechanism for the network, forcing the network to reconstruct many of the goals of the domain.

References

Akita, R.; Yoshihara, A.; Matsubara, T.; and Uehara, K. 2016. Deep learning for stock prediction using numerical and textual information. In *Proceedings of the IEEE/ACIS 15th International Conference on Computer and Information Science*, 1–6.

Amado, L.; Pereira, R. F.; Aires, J. a. P.; Magnaguagno, M.; Granada, R.; and Meneguzzi, F. 2018. Goal recognition in latent space. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*.

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

Avrahami-Zilberbrand, D., and Kaminka, G. A. 2005. Fast and complete symbolic plan recognition. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 653–658.

Bengio, Y.; Simard, P.; and Frasconi, P. 1994. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks* 5(2):157–166.

Fikes, R., and Nilsson, N. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2(3-4):189–208.

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

Geffner, H., and Bonet, B. 2013. *A Concise Introduction to Models and Methods for Automated Planning*, volume 7. A Concise Introduction to Models and Methods for Automated Planning.

Geib, C. W., and Steedman, M. 2007. On natural language processing and plan recognition. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Granada, R.; Pereira, R. F.; Monteiro, J.; Barros, R.; Ruiz, D.; and Meneguzzi, F. 2017. Hybrid Activity and Plan Recognition for Video Streams. In *The AAAI 2017 Workshop on Plan, Activity, and Intent Recognition*.

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

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26:191–246.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780.

Jiménez, S.; de la Rosa, T.; Fernández, S.; Fernández, F.; and Borrajo, D. 2012. A review of machine learning for automated planning. *Knowledge Engineering Review* 27(4):433–467.

Kautz, H. A., and Allen, J. F. 1986. Generalized Plan Recognition. In *AAAI*, 32–37.

Martín, Y. E.; Moreno, M. D. R.; and Smith, D. E. 2015. A Fast Goal Recognition Technique Based on Interaction Estimates. *International Joint Conference on Artificial Intelligence (IJCAI)*.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL − The Planning Domain Definition Language.

Min, W.; Ha, E.; Rowe, J. P.; Mott, B. W.; and Lester, J. C. 2014. Deep learning-based goal recognition in open-ended digital games. In *AIIDE*.

Pereira, R. F., and Meneguzzi, F. 2016. Landmark-based Plan Recognition. In *European Conference on Artificial Intelligence (ECAI)*.

Pereira, R. F., and Meneguzzi, F. 2017. Goal and Plan Recognition Datasets using Classical Planning Domains. Zenodo.

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

Ramírez, M., and Geffner, H. 2009. Plan recognition as planning. In *International Joint Conference on Artificial Intelligence (IJCAI)*.

Ramírez, M., and Geffner, H. 2010. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *AAAI*, 1121–1126.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan Recognition as Planning Revisited. In *International Joint Conference on Artificial Intelligence*, 3258–3264.

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

Sundermeyer, M.; Ney, H.; and Schlüter, R. 2015. From feedforward to recurrent lstm neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 23(3):517–529.

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