Using Self-Attention LSTMs to Enhance Observations in Goal Recognition

Leonardo Amado*, Gabriel Paludo Licks*, Matheus Marcon*, Ramon Fraga Pereira, and Felipe Meneguzzi
School of Technology
Pontifical Catholic University of Rio Grande do Sul
Porto Alegre, Brazil

***Abstract*—Goal recognition is the task of identifying the goal an observed agent is pursuing. The quality of its results depends on the quality of the observed information. In most goal recognition approaches, the accuracy significantly decreases in settings with missing observations. To mitigate this issue, we develop a learning model based on LSTMs, leveraging attention mechanisms, to enhance observed traces by predicting missing observations in goal recognition problems. We experiment using a dataset of goal recognition problems and apply the model to enhance the observation traces where missing. We evaluate the technique using a state-of-the-art goal recognizer in four different domains to compare the accuracy between the standard and the enhanced observation traces. Results show that recurrent neural networks with self-attention mechanisms increase on average 60% the accuracy metrics of state-of-the-art goal recognition techniques.**

I. Introduction

Goal recognition is the task of discerning the intended goal of an agent by observing its interactions in an environment. Initial approaches to goal and plan recognition are based on plan-libraries, which require a substantial amount of domain knowledge to represent the agents' behavior [1]–[3]. Subsequent approaches have gradually relaxed such requirements by using planning domain models [4], [5]. However, the accuracy of most recognition approaches is directly related to the amount of observations (e.g., sequences of states) available in order to recognize correctly the intended goal [6]–[9]. Such limitation appears in most goal recognition techniques, including the state-of-the-art in goal recognition as planning [10], yielding poor results in some domain models when dealing with fewer observations.

To overcome this issue, we introduce a novel method for enhancing the observations in goal recognition problems. To do so, we use a learning model using Recurrent Neural Networks (RNNs) and attention mechanisms to predict missing states in between the observations provided in the goal recognition problem. We train four models, one to each of the domains we experiment, and apply the models to a dataset of goal recognition problems. We evaluate the method in two types of goal recognition problems: (1) classical goal recognition problems, where domains are formalized in Planning Domain Definition Language (PDDL) [11], a well-known domain language used in AI Planning; (2) goal recognition problems in latent space, where the domains are learned using convolutional neural networks and autoencoders [12]. For the classical goal recognition problems, we train two domain-specific models based on two Artificial Intelligence Planning System (AIPS) domains, i.e., blocks-world and logistics [13], two well known classical domain models in the planning literature. For the goal recognition problems in latent space, which uses domains with features extracted in latent space using autoencoders [12], [14], we train two domain-specific models to two domains: the MNIST 8-puzzle, where numbers are images extracted from the MNIST dataset; and the Lights-Out (LO Digital) game. Experiments and evaluation show that our novel method is able to improve the accuracy of state-of-the-art techniques in goal recognition as planning [10] in approximately 60%.

This paper is structured as follows. First, in Section II, we provide a background on Goal Recognition, Long Short-Term Memory Networks (LSTM), attention mechanisms, and Goal Recognition in Latent Space. Second, in Section III, we describe the method we develop to improve goal recognition and how we use a learning model to predict observations. Third, in Section V, we describe the experiments we performed and discuss the results on the domains we use for goal recognition. Finally, in Section VII, we conclude our work stating what we achieved with our approach and what we are currently investigating for future work.

II. Background

In this section, we detail the necessary background to explain our work. First, due to the multi-disciplinary nature of our work, we explain the problem of goal recognition in planning. Second, we describe long short-term memory (LSTMs) networks. Third, we detail attention mechanisms in neural networks, and how they can improve such models. Finally, we provide a brief summary of recent work in goal recognition in latent space, to clarify some of experiments in this work.

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

*These authors have contributed equally to this paper.

sequence of the actions performed by the agent. Goal 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, often using STRIPS-style [16] descriptions.

Most goal recognition approaches [2], [17], [18] 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. While most approaches of this kind have serious scalability issues, recent work on plan recognition as planning have solved this issue [9] by using landmark-based heuristics [19] to efficiently process observations without the need to call a planner multiple times.

We describe goal recognition problems following a STRIPS [16] model. Formally, we model planning domains of the agents being observed as $\mathcal{D} = \langle \mathcal{R}, \mathcal{O} \rangle$, 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. 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 if $S \models pre(a)$ 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 [20], which encodes the formalism described thus far.

Bringing this all together, a goal recognition problem is a tuple $\mathcal{P}_{GR} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G}, O \rangle$, where $\mathcal{D}$ is a planning domain including the facts $\mathcal{F}$; $\mathcal{I}$ 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.

B. Long Short-Term Memory Networks

A Recurrent Neural Network (RNN) is a type of network that attempts to model a sequence of dependent events occurring through time, e.g., a financial time series [21], or language modeling [22]. 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 the well-known vanishing gradient problem [23], i.e., the gradients that are backpropagated through 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 eliminate the vanishing gradient problem, [24] 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 four components: the cell state, the forget gate, the input gate and the 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 forgotten 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 "forget this information". The input gate computes 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 computes what information should be propagated forward by performing a pointwise multiplication of a *sigmoid* layer, which computes

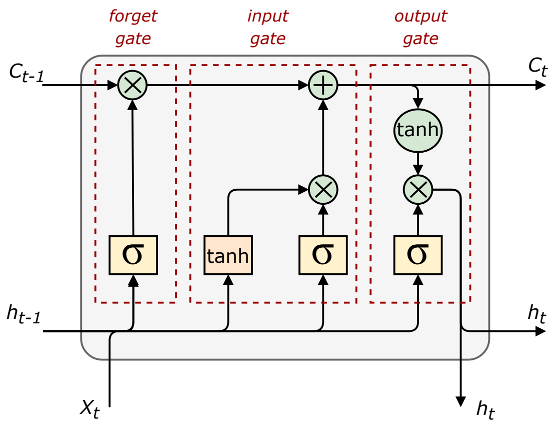


Fig. 1: Internal structure of the LSTM cell.

what part of the input is forwarded, and a the cell state filtered by a *tanh* operation. Figure 1 illustrates an 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.

C. Self-Attention networks

Despite its advantages, LSTM networks still present limitations due to its fixed-length internal representations. Such limitation creates issues for the processing of longer sequences of data. Attention mechanisms [25] have become an essential component of modern sequence processing models. Attention was initially employed in Neural Machine Translation (NMT) tasks, which consist of two RNNs serving as encoder and decoder. A context vector attributes different weights to the elements of the whole input sequence, and mediates information exchange between both networks. Thus, it allows the decoder to focus on the most relevant input elements independently of their position in the sequence. Self-attention [26], also called intra-attention, is a mechanism that encodes relationships among elements in different positions of the same sequence, allowing the representation of its composition.

An alternative attention layer for regular LSTMs is proposed in [27], where an attention matrix A is used to capture similarities among input elements. The relationship between elements x_t and $x_{t'}$ of hidden states h_t and $h_{t'}$ at steps t and t' is stored in the similarity element $a_{t,t'} \in A$. Similarity is computed by applying a *hyperbolic tangent* layer to the weight matrices associated with hidden states h_t and $h_{t'}$ followed by a *sigmoid* layer, both with added bias. The weighted summation of $h_{t'}$ elements and their similarities $a_{t,t'}$ to elements of h_t compose the attention hidden state l_t, which contains information on the relevance of a given element at any step in proportion to other elements in its sequence.

D. Goal recognition in Latent Space

Planning algorithms are based on the *factored* transition function 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. Automatically generating such domain knowledge involves at least two processes: converting real-world data into a factored representation (i.e., predicates for the planning process); and generating a transition function (i.e., the set of possible actions in the planning domain) from traces of the factored representation. A recent approach from Asai and Fukunaga [12] uses an auto-encoder [28] neural network to automatically generate domain models from images of simple games and problems. The neural network uses an encoder to convert an input image into a discretized representation.

The encoder receives 42x42 black and white images and outputs a 6x6 latent representation activated by Gumbel-Softmax [29]. The decoder reconstructs the input image from the discretized representation. Leveraging from such auto-encoder, in [14] the authors developed an approach (LatRec) to recognize goals in latent space. By comparing images of sequential states, the authors are able to infer an approximated PDDL domain. Thus, LatRec is able to recognize goals in real-world data, by combining unsupervised learning and state-of-the-art goal recognition techniques.

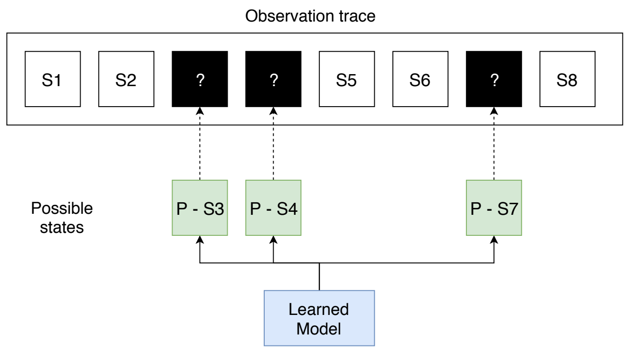


Fig. 2: Predicting missing observations.

III. Approach

State-of-the-art goal recognition techniques compare their approaches with varying degrees of observability, to proper measure the efficiency of these techniques in different situations. As the degree of observability decreases, which means that less observations are given to the recognizer, it becomes harder to recognize goals, so the accuracy decreases and the spread increases. To improve goal recognition approaches, we propose a learning model capable of predicting missing observations, enhancing the information used in the goal recognition process. The model takes in incomplete observation traces as input and predicts the most likely next step. Observation traces are sequences of states that are given to the trained model, thus the model returns a prediction of the next state.

We assume that a model of the domain is known, or at least approximated through learning techniques. Thus, we are given a full goal recognition problem with the tuple $\mathcal{P}_{GR} = \langle \mathcal{D}, \mathcal{I}, \mathcal{G}, O \rangle$. Using the domain model, we verify in the set of observations O if a sequence of states is impossible by evaluating whether a transition between two states is valid, starting from the initial state I and the first observation. If the sequence of states is impossible, we conclude that an observation is missing at the point of the invalid transition of the observation trace. Hence, the problem of predicting missing observations is given by predicting n observations $\hat{O}$ that fit the set of observations O. Figure 2 illustrates this process, where $P - S_n$ is a possible observation that can fit the set of observations O.

We show the developed architecture for enhancing observations in goal recognition problems in Figure 3. The *Grounding* element in the diagram generates the set of all possible actions for a given domain instance. The *Observation Predictor* ele-

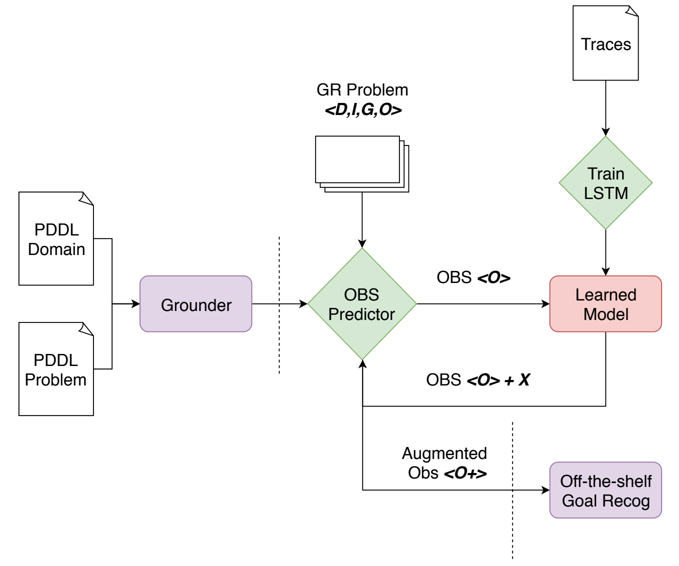


Fig. 3: Enhancing observations architecture.

ment takes as input the traces of a goal recognition problem and uses the ground actions to verify whether transitions between observations are valid. If a transition is not valid, it feeds the traces to the *Learned Model* element, a trained self-attention LSTM, which outputs a prediction for the missing observation. For each prediction, we select the most likely state following the given input observations, and test if it is a possible consecutive state. Then, the predicted observation is fed back to the *Observation Predictor*, and, if it constitutes a valid transition, it is appended to the sequence of observations. If the predicted state is impossible to apply in the given observations, we iterate the top-3 outputs of the network layer and evaluate whether one of them is a valid transition. If none of the top-3 selected predictions are valid transitions, we do not add any observations. If the predicted state is included in the set of candidate goals $\mathcal{G}$, we stop predicting observations further from that state. Finally, when the model can no longer predict valid transitions to the current sequence of observations, the enhanced observation traces are sent to the *Off-the-shelf Goal Recognizer* element.

Our model architecture consists of 5 layers. The first layer is an embedding in which we tokenize the input. Second is an LSTM with 1024 hidden neurons, which is connected to a self-attention layer with a softmax activation. The output is sent to a flatten layer that feeds a fully connected layer activated with softmax. The final activated output is a vector with the size of unique tokens (vocabulary), where each token represents an observation. In Figure 4 we illustrate the architecture of our learning model, where *max-len* is the maximum length of an input sequence of observations in the dataset and *vocab* is the number of unique tokens of the problem (all possible states that the model can predict). We employ an Adam optimizer and a Categorical Cross-Entropy loss function to train the network.

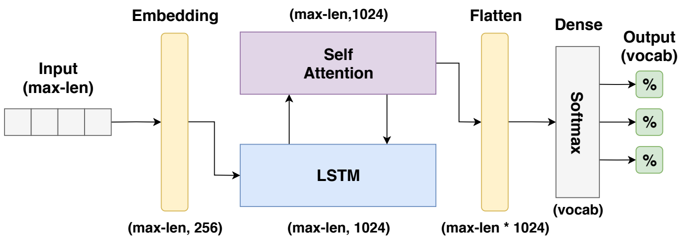


Fig. 4: Model architecture.

IV. Datasets and training

In this section we detail how we built the datasets for training and testing the models. First, we detail the datasets used to train models for the the classical AIPS domains. Then, we detail the datasets created to train the model for domains in latent space. Finally, we detail how we train our models and test their performance.

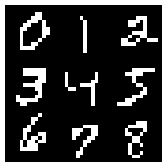
A. Classical domains dataset

To train our models for classical problems, we use two classical PDDL domains from AIPS, i.e., blocks-world and logistics. These are well known planning domains used in classical goal recognition approaches in the literature. We build one dataset to each domain and model, where we use 100 planning problems for each domain. We then solve these problems using a standard PDDL planner called PyperPlan[1], which computes an optimal plan to solve each problem instance. This plan is a sequence of states, the solution from the initial state to the goal state. After computing a plan for each problem, we separate the data into a training set, containing 80 plan instances, and a test set with 20 plan instances. We augment the data used in training by generating subsets of the training instances as new data instances. For example, if we have the train instance $x_1 = [s1, s2, s3, s4, s5]$, where s_n is a state, and $y_1 = [s6]$ is the label to this training instance, we create new training instances, such as $x_{1a} = [s1, s2, s3, s4]$ using $y_{1a} = [s5]$ as label. The subsets maintain the contiguity of states, and are generated until the minimum lenght of three states. After augmentation, plan instances have on average a sequence of 7 states on these domains.

B. Latent space datasets

To test our approach in real-world data, we generated a number of image-based datasets based on existing goal recognition problems [14], [30], [31]. These datasets were first introduced in [12], using images to compose puzzle domains and employing a variational autoencoder to extract latent space features from each domain. We select two domains from [12], the MNIST 8-puzzle and the Lights-Out domain. The MNIST 8-puzzle, illustrated in Figure 5a, uses handwritten digits from the MNIST dataset as tiles of the puzzle, with the number 0 representing the blank space. Every image of the dataset uses the same handwritten digit for every repeating number. The

[1]https://bitbucket.org/malte/pyperplan


(a) MNIST

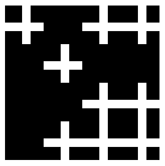

(b) LO Digital

Fig. 5: Sample state for each domain.

TABLE I: Model metrics for each domain.

Domain	Vocab. Size	Max. Length	Top-1 Acc	Top-3 Acc
Blocks-world	473	16	0.78	0.84
Logistics	1021	22	0.85	0.86
MNIST	566	10	0.83	0.91
LO Digital	709	8	0.85	0.88

Lights-Out puzzle game [32] consists of a 4 by 4 grid of lights that can be turned on and off, thus named Lights-Out Digital (LO Digital). LO Digital starts with a random number of lights initially on—toggling any of the lights also toggles every adjacent light—and the objective is to turn every light off. The LO Digital domain uses crosses to represent when a light is on, as illustrated in Figure 5b.

To generate the training dataset, we create 100 planning problems for each latent space domain. After creating the problems, we use LatPlan [31] to solve them and create a plan trace (sequence of states). Latent space states are represented as binary vectors extracted from the autoencoder latent features. We then split these plan instances, using 80 of them for training, and 20 for testing. We augment the training dataset for the latent space domains following the same procedure used for the classical domains, in order to increase the dataset size. After augmentation, plan instances have on average a sequence of 5 states on these domains.

C. Training and Testing

After building the training and test datasets, we train 4 distinct models, one for each domain, using the architecture described in Figure 3. In each sequence of traces, the final trace is extracted and used as a label to the remainder of the sequence. Since LSTM models expect inputs with equal size, we identify the longest sequence of traces and apply a zero left-padding to all instances shorter than it, so that all inputs have the same length. During training, our model receives a trace sequence as input, and outputs a prediction, which is the probability of each state being the correct next state. Training is interrupted when there is no improvement in validation loss after 10 epochs straight (early stop).

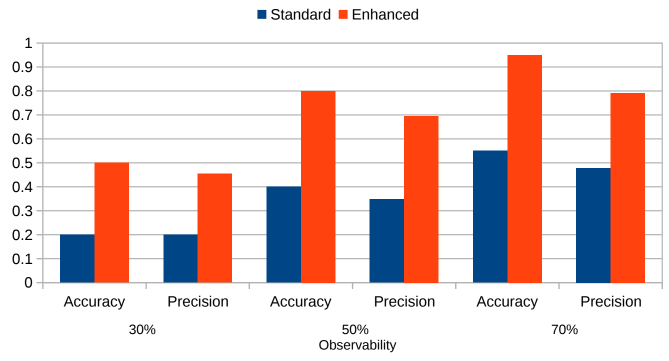
In Table I, we show the results of our training model in each dataset, where *Vocab. Size* represents the number of distinct states the model can predict, *Max. Length* the maximum sequence length that can be fed to the network, *Top-1 Acc* the standard accuracy of the model when predicting the correct missing observation, and *Top-3 Acc* the accuracy when considering the 3 higher probability states.

V. Experiments

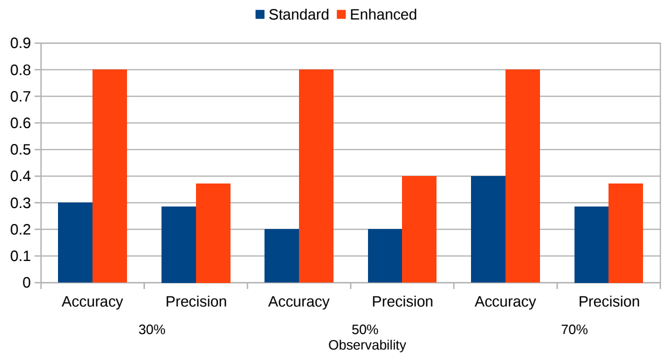
To run experiments in goal recognition problems and perform the prediction of missing observations using the trained model, we use the 20 planning problems of each domain's test dataset to generate observation traces. Goal recognition techniques use different rations of observability in order to test their efficiency. We remove observations from the test plans following ratios of 30, 50, and 70 percent of observability. This means that we randomly cut a percentage of observations out of a plan, resulting in traces such as illustrated in Figure 2. We refer to the resulting dataset with no predictions from our model as the *Standard* dataset. We generate a new dataset by feeding the observations to our model and try to fill gaps of missing observations, resulting the dataset we refer to as *Enhanced* dataset. Finally, we generate goal recognition problems using the traces of each domain in the datasets for each observability ratio, resulting in 60 goal recognition problems for each domain.

Both the *Standard* dataset (no predictions from the model) and the *Enhanced* dataset are used as input to a state-of-the-art goal recognizer proposed in [9]. The recognizer uses a landmark-based approach [9] for recognizing actual goals from a set of candidate goals, given a trace of observations. The recognizer uses the Uniqueness Landmark-based Heuristic, which uses the concept of a landmark's uniqueness value, representing the information value of the landmark for a particular candidate goal when compared to landmarks for all candidate goals. The estimates 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. The more landmarks an observation trace contains with regard to a candidate goal, the higher the score for that goal being the correct goal pursued by the agent. Thus, enhancing observations traces that are correct with regard to the plan an agent is pursuing will most likely increase the chances of recognizing the agent's actual goal.

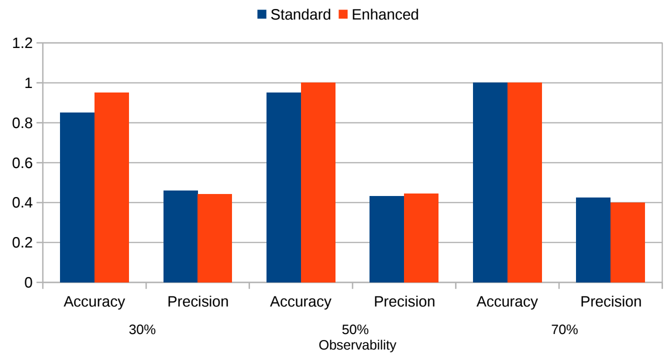
In Figure 6, we show the result metrics of the goal recognizer after applying our technique in the 4 proposed domains. To each observability ratio, we show the accuracy and precision metrics of the goal recognizer, in blue for the *Standard* dataset, and in red for the *Enhanced* dataset. Results show that our approach is capable of improving the accuracy of the state-of-the-art goal recognition algorithm, both in classical and in latent space domains. In the classical domains (Blocks-world Fig. 6a, and Logistics 6b), the accuracy gain is over 60% across all degrees of observability. In the Logistics domain, with 50% of observability, the accuracy triples, highlighting the ability of our method to improve observation traces. In latent space domains, the observation traces in the LO Digital domain improve significantly (Fig. 6d), while improvements in the MNIST 8-puzzle are slight in accuracy (Fig. 6c),



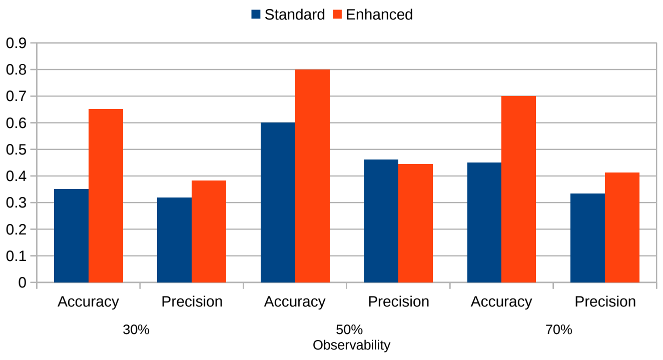
(a) Blocks-world



(b) Logistics



(c) MNIST 8-Puzzle



(d) LightsOut Digital

Fig. 6: Goal recognition accuracy in each domain.

with a slight decrease in precision for 30 and 50 percent of observability. This occurs because the accuracy in the MNIST 8-puzzle domain is already high, leaving small room for improvement and a higher chance of noise being added.

In Table II we report in detail the result metrics of the *Standard* and *Enhanced* datasets using the state-of-the-art goal recognizer. As data shows, our approach consistently improves the results across all metrics. In *Avg. # Obs*, we show the average number of observation traces given to the recognizer, and in *Recog. Goals*, the average number of identified goals returned by the goal recognizer (spread). Our approach is able to consistently add many observations across all domains, and in some cases triples the amount of observations. The improvements are significantly higher in the classical goal recognition problems, but our technique is still able to improve the results using latent space domains.

VI. Related Work

In [33], Min et. al. 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 we only predict missing observations, without the compromise of finding one of the candidate goals. This is a significant difference, because our approach can still provide results when dealing with goals that are not contained in the training dataset (referenced in Table I as *Vocab*). Thus, our approach focuses in improving state-of-the-art goal recognition approaches, and it is applicable to future goal recognition approaches.

Sohrabi et al. [34] developed a probabilistic approach for recognizing goals and plans that deals explicitly with unreliable and spurious observations (i.e., missing and noisy observations). To do so, Sohrabi et al. modify the domain model by introducing costs in the action descriptions according to an interpretation about what is missing or noisy observation. We note that their approach does not aim to enhance the information in the observation sequence by inferring what is missing between the observations, whereas our approach does.

TABLE II: Results for Goal Recognition problems

Domain	Observability	Dataset	Accuracy	Precision	Recall	F1-score	Avg. # Obs.	Recog. Goals
Blocks World	30%	Standard	0.20	0.20	0.20	0.20	4.15	1.00
		Enhanced	0.50	0.45	0.50	0.48	13.60	1.10
	50%	Standard	0.40	0.35	0.40	0.37	6.85	1.15
		Enhanced	0.80	0.70	0.80	0.74	16.80	1.15
	70%	Standard	0.55	0.48	0.55	0.51	9.85	1.15
		Enhanced	0.95	0.79	0.95	0.86	16.00	1.20
Logistics	30%	Standard	0.30	0.29	0.30	0.29	5.50	1.05
		Enhanced	0.80	0.37	0.80	0.51	19.30	2.15
	50%	Standard	0.20	0.20	0.20	0.20	8.65	1.00
		Enhanced	0.80	0.40	0.80	0.53	19.15	2.00
	70%	Standard	0.40	0.29	0.40	0.33	12.40	1.40
		Enhanced	0.80	0.37	0.80	0.51	19.85	2.15
MNIST 8-Puzzle	30%	Standard	0.85	0.46	0.85	0.60	2.75	1.85
		Enhanced	0.95	0.44	0.95	0.60	8.10	2.15
	50%	Standard	0.95	0.43	0.95	0.59	3.95	2.20
		Enhanced	1.00	0.44	1.00	0.62	8.80	2.25
	70%	Standard	1.00	0.43	1.00	0.60	5.50	2.35
		Enhanced	1.00	0.40	1.00	0.57	9.85	2.50
LO Digital	30%	Standard	0.35	0.32	0.35	0.33	2.70	1.10
		Enhanced	0.65	0.38	0.65	0.48	4.35	1.70
	50%	Standard	0.60	0.46	0.60	0.52	3.70	1.30
		Enhanced	0.80	0.44	0.80	0.57	5.25	1.80
	70%	Standard	0.45	0.33	0.45	0.38	5.00	1.35
		Enhanced	0.70	0.41	0.70	0.52	6.65	1.70

Moreover, our approach can be used along with theirs during the recognition process to infer missing states.

Granada et al. [35] developed 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. Differently from our work, this work relies on plan libraries, a different domain formalization for goal recognition, which requires a handmade structure, defining a set of plans to achieve the goals.

In [12], Asai et. al. developed 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. In this work, we demonstrated that our approach is able to improve results for goal recognition in latent space domains.

VII. Discussion and Future Work

In this work, we developed an approach for predicting missing observation in goal recognition problems. We tested our approach in four distinct domains, including domains based on real world data. Results show that our approach is capable of enhancing traces with missing observations for all domains, significantly improving accuracy across varying degrees of observability. As shown in the results, in some cases our approach more than doubles the accuracy of the recognizer, with minimal increase of spread as drawback. Thus, our main contribution is a novel technique to aid state-of-the-art goal recognition techniques, in classical and real-world data domains, without performing goal recognition in itself. Hence, our approach can be used in future state-of-the-art goal recognition approaches.

For future work, we propose two main improvements:

1) extend the idea of this work to continuous domains, instead of using only discrete domains;
2) apply this approach in online goal recognition and plan recognition.

A different model would be needed in order to apply our technique in continuous domains. However, there is much research about learning approaches capable of approximating continuous domains, and even applied in goal recognition [36]–[38]. Our approach could easily fit in the problem of online goal recognition, predicting the next possible observation as we observe acting agents. By solving such problem, we could extend to plan recognition (a superset of goal recognition problems) predicting the entire plan as we observe an acting agent.

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] M. Ramírez and H. Geffner, "Plan Recognition as Planning," in *IJCAI*, 2009.

[5] M. Ramírez and H. Geffner, "Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners," in *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[6] Y. E. Martín, M. D. R. Moreno, and D. E. Smith, "A Fast Goal Recognition Technique Based on Interaction Estimates." in *Proceedings of the 24th International Joint Conference on Artificial Intelligence*, 2015.

[7] S. Sohrabi, A. V. Riabov, and O. Udrea, "Plan Recognition as Planning Revisited," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

[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 Thirty-Second AAAI Conference on Artificial Intelligence*, 2017.

[10] ——, "Landmark-based approaches for goal recognition as planning," *Artificial Intelligence*, vol. 279, p. 103217, 2020.

[11] D. McDermott, M. Ghallab, A. Howe, C. Knoblock, A. Ram, M. Veloso, D. Weld, and D. Wilkins, "PDDL – The Planning Domain Definition Language," 1998.

[12] M. Asai and A. Fukunaga, "Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary," in *AAAI*, 2018.

[13] S. A. Chien, S. Kambhampati, and C. A. Knoblock, Eds., *Proceedings of the Fifth International Conference on Artificial Intelligence Planning Systems, Breckenridge, CO, USA, April 14-17, 2000*. AAAI, 2000.

[14] L. Amado, R. F. Pereira, J. P. Aires, M. Magnaguagno, R. Granada, and F. Meneguzzi, "Goal recognition in latent space," in *IJCNN*, 2018.

[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] 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.

[17] 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.

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

[19] 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.

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

[21] R. Akita, A. Yoshihara, T. Matsubara, and K. Uehara, "Deep learning for stock prediction using numerical and textual information," in *Proceedings of the 2016 IEEE/ACIS 15th International Conference on Computer and Information Science*, ser. ICIS'16. IEEE, 2016, pp. 1–6. [Online]. Available: https://doi.org/10.1109/ICIS.2016.7550882

[22] M. Sundermeyer, H. Ney, and R. Schlüter, "From feedforward to recurrent lstm neural networks for language modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 517–529, March 2015. [Online]. Available: https://doi.org/10.1109/TASLP.2015.2400218

[23] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, Mar 1994. [Online]. Available: https://doi.org/10.1109/72.279181

[24] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[25] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," 2014.

[26] J. Cheng, L. Dong, and M. Lapata, "Long short-term memory-networks for machine reading," 2016.

[27] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li, "Opentag," *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, Jul 2018. [Online]. Available: http://dx.doi.org/10.1145/3219819.3219839

[28] P. Vincent, H. Larochelle, Y. Bengio, and P.-A. Manzagol, "Extracting and composing robust features with denoising autoencoders." in *ICML*, 2008.

[29] E. Jang, S. Gu, and B. Poole, "Categorical reparameterization with gumbel-softmax," *CoRR*, vol. abs/1611.01144, 2016.

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

[31] 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.

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

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

[34] S. Sohrabi, A. V. Riabov, and O. Udrea, "Plan Recognition as Planning Revisited," in *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*, 2016.

[35] 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.

[36] B. Say, G. Wu, Y. Q. Zhou, and S. Sanner, "Nonlinear hybrid planning with deep net learned transition models and mixed-integer linear programming," 2017.

[37] G. Wu, B. Say, and S. Sanner, "Scalable planning with tensorflow for hybrid nonlinear domains," in *NIPS*, 2017.

[38] R. F. Pereira, M. Vered, F. Meneguzzi, and M. Ramirez, "Online probabilistic goal recognition over nominal models," in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence (IJCAI)*, 2019, pp. 5547–5553.