Goal Recognition as a Deep Learning Task: the GRNet Approach

Mattia Chiari[1], Alfonso Emilio Gerevini[1], Francesco Percassi[2], Luca Putelli[1], Ivan Serina[1], Matteo Olivato[1]

[1]Dipartimento di Ingegneria dell'Informazione, Università degli Studi di Brescia, Via Branze 38, Brescia, Italy
[2]School of Computing and Engineering, University of Huddersfield, Queensgate, Huddersfield HD1 3DH, United Kingdom
{mattia.chiari, alfonso.gerevini, luca.putelli1, ivan.serina, m.olivato}@unibs.it, f.percassi@hud.ac.uk

Abstract

Recognising the goal of an agent from a trace of observations is an important task with many applications. The state-of-the-art approach to goal recognition (GR) relies on the application of automated planning techniques. We study an alternative approach, called GRNet, where GR is formulated as a classification task addressed by machine learning. GRNet is primarily aimed at solving GR instances more accurately and more quickly by learning how to solve them in a given domain, which is specified by a set of propositions and a set of action names. The goal classification instances in the domain are solved by a Recurrent Neural Network (RNN). The only information required as input of the trained RNN is a trace of action labels, each one indicating just the name of an observed action. A run of the RNN processes a trace of observed actions to compute how likely it is that each domain proposition is part of the agent's goal, for the problem instance under consideration. These predictions are then aggregated to choose one of the candidate goals. An experimental analysis confirms that GRNet achieves good performance in terms of both goal classification accuracy and runtime, obtaining better results w.r.t. a state-of-the-art GR system over the considered benchmarks. Moreover, such a state-of-the-art system and GRNet can be combined achieving higher performance than with each of the two integrated systems alone.

Introduction

Goal Recognition is the task of recognising the goal that an agent is trying to achieve from observations about the agent's behaviour in the environment (Van-Horenbeke and Peer 2021; Geffner 2018). Typically, such observations consist of a trace (sequence) of executed actions in an agent's plan to achieve the goal, or a trace of world states generated by the agent's actions, while an agent goal is specified by a set of propositions. Goal recognition has been studied in AI for many years, and it is an important task in several fields, including human-computer interactions (Batrinca et al. 2016), computer games (Min et al. 2016), network security (Mirsky et al. 2019), financial applications (Borrajo, Gopalakrishnan, and Potluru 2020), and others.

In the literature, several systems to solve goal recognition problems have been proposed (Meneguzzi and Pereira 2021). The state-of-the-art approach is based on transforming a plan recognition problem into one or more plan generation problems solved by classical planning algorithms (Ramírez and Geffner 2009; Pereira, Oren, and Meneguzzi 2020; Sohrabi, Riabov, and Udrea 2016). In order to perform planning, this approach requires domain knowledge consisting of a model of each agent's action specified as a set of preconditions and effects, and a description of the initial state of the world, in which the agent performs the actions. The computational efficiency (runtime) largely depends on the planning algorithm performance, which could be inadequate in a context demanding fast goal recognition (e.g., in real-time/online applications).

In this paper, we investigate an alternative approach in which the goal recognition problem is formulated as a classification task, addressed through machine learning, where each candidate goal (a set of propositions) of the problem can be seen as a value class. Our primary aim is making goal recognition more accurate as well as faster by learning how to solve it in a given domain. Given a planning domain specified by a set of propositions and a set of actions names, each one denoting an agent's action whose execution can be observed, we tackle the goal classification instances in the domain through a Recurrent Neural Network (RNN). A run of our RNN processes a trace of observed actions to compute how likely it is that each domain proposition is part of the agent's goal, for the problem instance under considerations. These predictions are then aggregated through a goal selection mechanism to choose one of the candidate goals.

The proposed approach, that we call GRNet, is generally faster than the model-based approach to goal recognition based on planning, since running a trained neural network can be much faster than plan generation. Moreover, GRNet operates with minimal information, since the only information required as input for the trained RNN is a trace of action labels (each one indicating just the name of an observed action), and the initial state can be completely unknown.

The RNN of GRNet is trained only once for a given domain, i.e., the same trained network can be used to solve a large set of goal recognition instances in the domain. On the other hand, as usual in supervised learning, a (possibly large) dataset of solved goal recognition instances for the domain under consideration is needed for the training. When such data are unavailable or scarce, they can be synthetized via planning. In such a case, the resulting overall system can be seen as a combined approach (model-based for generating training data, and model-free for the goal classification task) that outperforms the pure model-based approach in terms of both classification accuracy and classification runtime. Indeed, an experimental analysis confirms that GRNet achieves good performance obtaining better results with respect to the state-of-the-art goal recognition system LGR (Pereira, Oren, and Meneguzzi 2020) for the considered benchmark domains.

Moreover, we propose an effective method to integrate GRNet and LGR in an ensemble fashion, and we experimentally show that this system combination performs consistently better than both GRNet and LGR considered alone.

In the reminder of the paper, after giving the necessary background and preliminaries, we describe the GRNet approach and how it can be combined with LGR, we present the experimental results, and we discuss the related work.[1]

Preliminaries

We describe the problem of goal recognition, starting with its relation to activity and plan recognition.

Activity, Goal and Plan Recognition

Activity, plan, and goal recognition are related tasks (Geib and Pynadath 2007). Since in the literature sometime they are not clearly distinguished, we begin with an informal definition of them following (Van-Horenbeke and Peer 2021).

Activity recognition concerns analyzing sequences of (typically low-level) data generated by humans, or other autonomous agents acting in an environment, to identify the corresponding activity that they are performing. E.g., data can be collected from wearable sensors, accelerometers, or images to recognize human activities such as running, cooking, driving, etc. (Vrigkas, Nikou, and Kakadiaris 2015).

Goal recognition (GR) is the problem of identifying the intention (goal) of an agent from observations about the agent behaviour in an environment. These observations can be represented as an ordered sequence of discrete actions (each one possibly identified by activity recognition), while the agent's goal can be expressed either as a set of propositions or a probability distribution over alternative sets of propositions (each one forming a distinct candidate goal).

Plan recognition is more general than GR and concerns both recognising the goal of an agent and identifying the full ordered set of actions (plan) that have been, or will be, performed by the agent in order to reach that goal; as GR, typically plan recognition takes as input a set of observed actions performed by the agent (Carberry 2001).

Model-based and Model-free Goal Recognition

In the approach to GR known as "goal recognition over a domain theory" (Ramírez and Geffner 2010; Van-Horenbeke and Peer 2021; Santos et al. 2021; Sohrabi, Riabov, and Udrea 2016), the available knowledge consists of an underlying model of the behaviour of the agent and its environment. Such a model represents the agent/environment states and the set of actions A that the agent can perform; typically it is specified by a planning language such as PDDL. The states of the agent and environment are formalised as subsets of a set of propositions F, called *fluents* or *facts*, and each domain action in A is modelled by a set of preconditions and a set of effects, both over F. An *instance of the GR problem* in a given domain is then specified by:

- an initial state I of the agent and environment ($I \subseteq F$);
- a sequence $O = \langle obs_1, .., obs_n \rangle$ of observations ($n \geq 1$), where each obs_i is an action in A performed by the agent;
- and a set $\mathcal{G} = \{G_1, .., G_m\}$ ($m \geq 1$) of possible goals of the agent, where each G_i is a set of fluents over F that represents a partial state.

The observations form a trace of the full sequence π of actions performed by the agent to achieve a goal G^*. Such a plan trace is a selection of (possibly non-consecutive) actions in π, ordered as in π. Solving a GR instance consists of identifying the G^* in $\mathcal{G}$ that is the hidden goal of the agent.

The approach based on a model of the agent's actions and of the agent/environment states, that we call *model-based goal recognition* (MBGR), defines GR as a reasoning task addressable by automated planning techniques (Ghallab, Nau, and Traverso 2016; Meneguzzi and Pereira 2021).

An alternative approach to MBGR is *model-free goal recognition* (MFGR) (Geffner 2018; Borrajo, Gopalakrishnan, and Potluru 2020). In this approach, GR is formulated as a classification task addressed through machine learning. The domain specification consists of a fluent set F, and a set of possible actions A, where each action $a \in A$ is specified by just a label (a unique identifier for each action).

A MFGR instance for a domain is specified by an observation sequence O formed by action labels and, as in MBGR, a goal set $\mathcal{G}$ formed by subsets of F. MFGR requires minimal information about the domain actions, and can operate without the specification of an initial state, that can be completely unknown. Moreover, since running a learned classification model is usually fast, a MFGR system is expected to run faster than a MBGR system based on planning algorithms. On the other hand, MFGR needs a data set of solved GR instances from which learning a classification model.

Example 1 *As a running example, we will use a very simple GR instance in the well-known* BLOCKSWORLD *domain, in which the agent has the goal of building one or more stacks of blocks, and only one block may be moved at a time. There are four types of actions:* `Pick-Up` *a block from the table,* `Put-Down` *a block on the table,* `Stack` *a block on top of another one, and* `Unstack` *a block that is on another one. We assume that a GR instance in the domain involves at most 22 blocks. In* BLOCKSWORLD *there are three types of facts (predicates):* `On`*, that has two blocks as arguments, plus* `On-Table` *and* `Clear` *that have one argument. Therefore, the fluent set F consists of $22 \times 21 + 22 + 22 = 506$ propositions. The goal set $\mathcal{G}$ of the instance example consists of the two goals $G_1 = \{$*`(On`

[1]Additional material including source code and hyperparamentes of GRNet, the test data sets, and further experimental results are available from https://github.com/mattia93/GRNet.

`Block_F Block_C), (On Block_C Block_B)`$\}$ *and* $G_2 = \{$`(On Block_G Block_H), (On Block_H Block_F)`$\}$*; the observation sequence* O *is* $\langle$`(Pick-Up Block_C), (Stack Block_C Block_B), (Pick-Up Block_F)`$\rangle$.

Goal Recognition through GRNet

Our approach to goal recognition, called GRNet, is depicted in Figure 1. It consists of two main components. The first component takes as input the observations of the GR instance to solve, and gives as output a score (between 0 and 1) for each proposition in the domain proposition set F. This component, called *Environment Component*, is general in the sense that it can be used for every GR instance over F (training is performed once for each domain). The second component, called *Instance Component*, takes as input the proposition ranks generated by the environment component for a GR instance, and uses them to select a goal from the candidate goal set $\mathcal{G}$.

GRNet can be used alone or it can be combined with the MBGR system LGR (Pereira, Oren, and Meneguzzi 2020), as shown in the last part of this section.

The Environment Component of GRNet

Given a sequence of observations, represented on the left side of Figure 1, each action a_i corresponding to an observation is encoded as a vector e_i of real numbers by an embedding layer (Bengio et al. 2003). In Figure 1, the observed actions are displayed from top to bottom in the order in which they are executed by the agent. The embedding layer is initialised with random weights, and trained at the same time with the rest of the environment component.

The index of each observed action is simply the result of an arbitrary order of the actions that is computed in the preprocessing phase, only once for the domain under consideration. Note that two consecutively observed actions a_i and a_j may not be consecutive in the full plan of the agent, which may contain any number of actions in between a_i and a_j.

The Environment Component is based on a Long Short-Term Memory network (LSTM), which is a kind of RNN especially suitable for processing sequential data like signals or text documents (Hochreiter and Schmidhuber 1997) (in our case the sequence of observed actions).[2] A LSTM layer is composed by cells, which process each element of the input sequence (each observed action) considering also the previous inputs (actions in the sequence). The output of each cell is processed by an Attention Mechanism (Bahdanau, Cho, and Bengio 2015), in particular, the variant proposed by (Yang et al. 2016), which computes the weights representing the contribution of each element of the sequence, and generates a unique representation (also called the *context vector*) of the entire plan trace. The context vector is then passed to a feed-forward layer, which has N output neurons with *sigmoid* activation function. N is the number of the domain fluents (propositions) that can appear in any goal of $\mathcal{G}$ for any GR instance in the domain; for our experiments N was set to the size of the domain fluent set F, i.e., $N = |F|$. The output of the i-th neuron $\overline{o}_i$ corresponds to the i-th fluent f_i (fluents are lexically ordered), and the activation value of $\overline{o}_i$ gives a rank for f_i being true in the agent's goal (with rank equal to one meaning that f_i is true in the goal). In other words, our network is trained as a multi-label classification problem, where each domain fluent can be considered as a different binary class. As loss function, we used standard binary crossentropy.

As shown in Figure 1, the dimension of the input and output of our neural networks depend only on the selected domain and some basic information, such as the maximum number of possible output facts that we want to consider. The dimension of the embedding vectors, the dimension of the LSTM layer and other hyperaparameters of the networks are selected using the Bayesian-optimisation approach provided by the Optuna framework, (Akiba et al. 2019), with a validation set formed by 20% of the training set, while the remaining 80% is used for training the network.

The Instance Specific Component of GRNet

After the training and optimisation phases of the environment component, the resulting network can be used to solve any goal recognition instance in the domain through the instance-specific component of our system (right part of Figure 1). Such component performs an evaluation of the candidate goals in $\mathcal{G}$ of the GR instance, using the output of the environment component fed by the observations of the GR instance. To choose the most probable goal in $\mathcal{G}$ (solving the multi-class classification task corresponding to the GR instance), we designed a simple score function that indicates how likely it is that G is the correct goal, according to the neural network of the environment component. This score is defined as $S_{\mathsf{GRNet}}(G) = \sum_{f \in G} \overline{o}_f$ where $\overline{o}_f$ is the network output for fact f of the current GR instance. For each candidate goal $G \in \mathcal{G}$, we consider only the output neurons that have associated facts in G. By summing only these predicted values, we derive an overall score for G being the correct goal. The element with the highest score is the most probable goal in $\mathcal{G}$.

Example 2 *In our running example, since we are assuming that the GR instances involve at most 22 blocks, we have that the action set* A *is formed by* 22 `Pick-Up` *actions, 22* `Put-Down` *actions,* $22 * 21 = 462$ `Stack` *actions and 462* `Unstack` *actions, for a total of* $968 = |A|$ *different actions in the domain. Suppose that the three observed actions* `(Pick-Up Block_C)`, `(Stack Block_C Block_B)` *and* `(Pick-Up Block_F)` *forming the observation sequence* O *have ids corresponding to indices* 5, 17 *and* 21*, respectively. In the Environment Component of GRNet, after being processed by the embedding layer, the input* O *is represented by the sequence of vectors* e_{05}*,* e_{17} *and* e_{21}*. Then this sequence is fed to the LSTM layer and subsequently to the attention mechanism, producing a context vector* c *representing the entire plan trace formed by the observed actions. Finally, the vector* c *is processed by a final feed-forward layer made of* $|F| = 506$ *output neurons. In this representation, each neuron corresponds to a distinct proposition in* F*. Considering two possible goals,*

[2]We considered also using standard RNN or Gated Recurrent Units, but both performed worse than LSTM for our GR task.

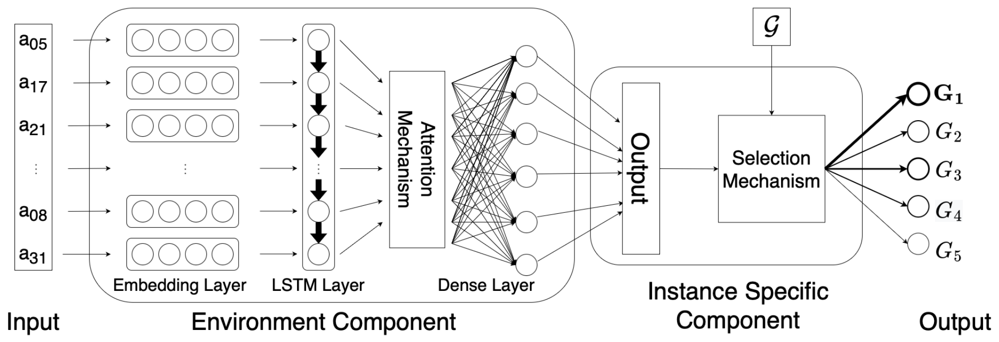

Figure 1: Architecture of GRNet. The input observations are encoded by embedding vectors and then fed to a LSTM neural network. After that the attention mechanism computes the context vector, which is used by a feed-forward layer to define the corresponding output values. This layer is composed by $|F|$ neurons, each one representing a possible fluent in the domain. The output of the neural network is then used by the instance component for selecting the goal with the highest score (G_1 in the example of the figure). The observed actions a_{05}, a_{17}, ..., a_{31} are ordered from top to bottom according to their execution order.

G_1, made by `(On Block_F Block_C)` *and* `(On Block_C Block_B)`, *and G_2 made by* `(On Block_G Block_H)` *and* `(On Block_H Block_F)`, *if the network has to predict G_1, the neurons associated to the different propositions should have value 1, while the neurons of the propositions in G_2 should have value zero.*

Therefore, in the Instance Component of GRNet, the prediction values of G_1 and G_2 is the sum of the predictions for the neurons representing their facts. Suppose that $\overline{o}_{(\texttt{On Block\_F Block\_C})} = 0.017$, $\overline{o}_{(\texttt{On Block\_C Block\_B})} = 1.000$, $\overline{o}_{(\texttt{On Block\_G Block\_H})} = 0.000$, $\overline{o}_{(\texttt{On Block\_H Block\_F})} = 0.003$, we have that the final score of G_1 is 1.017, while the final score of G_2 is 0.003. The goal with the highest score (G_1) is selected as the most probable goal solving the GR instance.

Integrating GRNet and LGR

GRNet can be integrated with an approach based on plan generation such as the state-of-the-art system LGR (Pereira, Oren, and Meneguzzi 2020). GRNet and LGR focus on different aspects of the problem. While GRNet has the capability of learning from experience the relations among actions and fluents belonging to the goal, LGR exploits domain knowledge and automated reasoning for selecting the most probable goal. Combining these two ways of addressing goal recognition can lead to better accuracy results, overcoming the limits of the automated reasoning approach, especially in the presence of incomplete plan traces, and improving GRNet when the learned experience is inadequate to solve the task. Therefore, we have created an integrated system, called LGRN, that combines LGR with GRNet.

The integration is simple and effective. Both GRNet and LGR provide a numerical score for each goal in $\mathcal{G}$, and each system considers the goal with the highest score as the most probable one. Therefore, we can combine them by using an aggregated score defined as the normalized sum of the scores of the two individual systems. More formally, given a GR instance, for each candidate goal $G \in \mathcal{G}$, the score $S_{\mathsf{LGRN}}(G)$ computed by LGRN for G is:

$$\sigma([S_{\mathsf{LGR}}(G_i)|G_i \in \mathcal{G}])_G + \sigma([S_{\mathsf{GRNet}}(G_i)|G_i \in \mathcal{G}])_G$$

Domain	$\lvert A\rvert$	$\lvert F\rvert$	$\lvert G_i\rvert$	$\lvert\mathcal{G}\rvert$
BLOCKSWORLD	968	506	[4,16]	[19,21]
DEPOTS	13050	150	[2,8]	[7,10]
DRIVERLOG	4860	156	[4,11]	[6,10]
LOGISTICS	15154	154	[2,4]	[10,12]
SATELLITE	33225	629	[4,9]	[6,8]
ZENOTRAVEL	23724	66	[5,9]	[6,11]

Table 1: Size of A, F, $G_i \in \mathcal{G}$ and $\mathcal{G}$ in the considered GR instances for each considered domain. Interval $[x, y]$ indicates a range of integer values from x to y.

where $S_{\mathsf{LGR}}(G)$ is the score of LGR for G, and $\sigma([\cdot])_G$ is the output for G of the softmax function applied to the input score vector $[\cdot]$. The softmax normalisation is chosen in order to have the same value distribution space for the two scores, i.e., values in the range $[0, 1]$ with sum 1. Furthermore, the softmax function tends to flatten the score values, which helps to avoid cases where the sum is extremely biased towards one approach or the other.

Experimental Analysis

Benchmark Suite and Data Sets

We consider six well-known benchmark domains: BLOCKSWORLD, DEPOTS, DRIVERLOG, LOGISTICS, SATELLITE and ZENOTRAVEL (McDermott 2000; Long and Fox 2003). Of course GRNet can be trained and tested also using other domains.

Training sets In order to create the (solved) GR instances for the training and test sets in the considered domains, we used automated planning techniques. Concerning the training set, for each domain, we randomly generated a large collection of (solvable) plan generation problems of different size. We considered the same ranges of the numbers of involved objects as in the experiments of Pereira, Oren, and Meneguzzi (2020). For each of these problems, we computed up to four (sub-optimal) plans solving them. As planner we used LPG (Gerevini, Saetti, and Serina 2003;

Gerevini and Serina 2003), which allows to specify the number of requested different solutions for the planning problem it solves. From the generated plans, we derived the observation sequences for the training samples by randomly selecting actions from the plans (preserving their relative order). The selected actions are between 30% and 70% of the plan actions.[3]

The generated training set consists of pairs (O, G^*) where O is a sequence of observed actions obtained by sampling a plan π, and G^* is the hidden goal corresponding to the goal of the planning problem solved by π. For each considered domain, we created a training set with 55000 pairs.

Test sets For evaluating GRNet, we generated two test sets formed by GR instances *not seen at training time*: TS$_{PerGen}$ and TS$_{Rec}$. Such test instances were generated as for the train instances, except that the observation sequences were derived from plans computed by LAMA (Richter and Westphal 2010), while for the training instances we used plans computed by LPG; this change is to make the testing more robust.

TS$_{PerGen}$ is a generalisation and extension of the test set used in (Pereira, Oren, and Meneguzzi 2020) for the same domains that we consider. We indicate Pereira's test set with TS$_{Per}$. TS$_{PerGen}$ includes all TS$_{Per}$ instances; moreover, the goal sets ($\mathcal{G}$) of TS$_{Per}$ and TS$_{PerGen}$ are the same. The additional instances in TS$_{PerGen}$ are motivated by the limited number and particular structure of those in TS$_{Per}$. The structure is limited because the observations in the instances of TS$_{Per}$ are created from plans for the goals in $\mathcal{G}$ that are all generated from the *same* initial state. In TS$_{PerGen}$, the GR instances are created combining different initial states with the candidate goal sets, obtaining a richer diversification of the observation traces and a larger number of test instances. In particular, for each of DEPOTS, DRIVERLOG, SATELLITE and ZENOTRAVEL, TS$_{Per}$ contains only 84 instances, while TS$_{PerGen}$ contains 1000 instances for each domain.

For each plan generated for being sampled, we randomly derived five different action traces formed by 10%, 30%, 50%, 70% and 100% of the plan actions, respectively. This gives five groups of test instances, for each considered domain, allowing to evaluate the performance of GRNet also in terms of different amounts of available observations.

Table 1 gives information about the size of the GR instances in our test and training sets for each domain, in terms of number of possible actions ($|A|$), facts ($|F|$), min/max size of a goal ($|G_i|$) in a goal set $\mathcal{G}$, and min/max size of a goal set ($|\mathcal{G}|$).

Test set TS$_{Rec}$ was created to evaluate how well the compared systems behave on GR instances of different difficulty. We focus this analysis on a specific domain (ZENOTRAVEL). In TS$_{Rec}$, the generated GR instances are grouped into several classes according to their difficulty. As difficulty measure, we used the notion of *recognizability of the hidden goal*, which is inspired by the notion of the "uniqueness of landmarks" introduced by Pereira, Oren, and Meneguzzi (2020). Specifically, the recognizability $R(G)$ of a goal $G \in \mathcal{G}$ is defined as

$R(G) = \sum_{f \in G} \frac{1}{|\{G' \mid G' \in \mathcal{G} \wedge f \in G'\}|}$.

The lower $R(G)$ is, the more difficult recognising G is; vice versa, the higher $R(G)$ is, the more discernible G is. We normalize $R(G)$ as a value between 0 and 1, denoted $R_Z(G)$. E.g., if $\mathcal{G} = \{G_1, G_2, G_3\}$, with $G^* = G_1 = \{a, b, c\}$, $G_2 = \{a, e, f\}$ and $G_3 = \{g, h, i\}$, then $R(G^*) = \frac{1}{2} + 1 + 1 = \frac{5}{2}$ and $R_Z(G^*) = 0.75$ (high recognizability). If $\mathcal{G} = \{G_1, G_2, G_3\}$ with $G^* = G_1 = \{a, b, c\}$, $G_2 = \{a, b, x\}$ and $G_3 = \{a, b, y\}$, then $R(G^*) = \frac{1}{3} + \frac{1}{3} + 1 = \frac{5}{3}$, and so $R_Z(G^*) = 0.33$ (low recognizability).

Using different values for $R_Z(G^*)$, we generated nine classes of GR instances, denoted $C_1, ..., C_9$. For each GR instance in class C_i, we have $0.1 \cdot i \leq R_Z(G^*) < 0.1 \cdot (i+1)$, for $i = 1...9$. Each class consists of 100 GR instances.

Evaluation measures We use the GR *accuracy* for a set of test instances as the main evaluation criteria, which is defined as the percentage of instances whose goals are correctly identified (predicted) over the total number of instances in the test set. If for an instance the evaluated system provides k different goals with the same highest score, then, in the overall count of the solved instances, this instance has value $1/k$ if the true goal is one of these k goals, 0 otherwise.

Following the methodology in (Pereira, Oren, and Meneguzzi 2020), we also analyze the GR θ-*accuracy*. This measure assumes that the scores assigned to the goals by the GR system are values between 0 and 1, and uses a θ threshold to select the set of candidate goals whose score is greater than or equal to the highest assigned score minus θ. If the true goal belongs to the set of selected goals, the instance is considered correctly identified, and this instance obtains θ-accuracy 1 (otherwise θ-accuracy is 0). For evaluating the θ-accuracy for GRNet and LGRN, their prediction scores (positive numbers) are scaled using the min-max normalization which sets each score to a value in the $[0, 1]$ range. For $\theta = 0$ the system selects the $k \geq 1$ goals with the highest score, and if the true goal belongs to this set, the instance is considered solved obtaining value 1, instead of $1/k$ as in the standard accuracy metric.

The θ-accuracy is analysed together with the *Spread* measure, i.e., the average number of predicted goals for an instance according to the used θ threshold.

Experimental Results

We experimentally evaluate GRNet and LGRN, and we compare them with the state-of-the-art system LGR. For the goal selection in LGR we used heuristic h_{uniq} because the authors stated that it performs better than heuristic h_{gc}.

Accuracy Results for TS$_{PerGen}$ Table 2 summarizes the performance results of LGR, GRNet and LGRN in terms of GR accuracy using TS$_{PerGen}$. Bold results for GRNet and LGRN indicate better performance w.r.t. to LGR. GRNet and LGRN perform generally well, and they improve their performances with the increase of the percentage of the observed actions. With 30% of the actions, the accuracy of GRNet improves w.r.t. 10% of the action in every domain by more than 20 points. For instance, in DRIVERLOG, GRNet improves from 39.8 to 65.4. Similar improvements can be

[3]Note that the *test* sets (described next) also include GR instances with lower and higher percentages of observed actions.

Domain	10% of the plan			30% of the plan			50% of the plan			70% of the plan			100% of the plan		
	LGR	GRNet	LGRN	LGR	GRNet	LGRN	LGR	GRNet	LGRN	LGR	GRNet	LGRN	LGR	GRNet	LGRN
BLOCKSWORLD	20.39	**22.85**	**30.80**	38.92	**52.35**	**63.70**	53.02	**71.10**	**79.30**	72.25	**84.90**	**90.60**	88.62	**92.02**	**96.81**
DEPOTS	25.42	**32.95**	**40.30**	47.52	**59.80**	**71.90**	65.88	**74.70**	**87.60**	78.78	**84.90**	**93.60**	93.17	91.95	**97.25**
DRIVERLOG	25.43	**39.80**	**44.50**	41.27	**65.40**	**72.70**	59.07	**78.40**	**82.90**	76.32	**86.20**	**91.10**	89.94	**90.78**	**94.28**
LOGISTICS	27.20	**39.15**	**43.60**	55.50	**68.80**	**77.60**	73.93	**80.40**	**90.90**	85.06	**89.20**	**97.10**	90.14	**93.94**	**99.23**
SATELLITE	41.26	**45.50**	**58.70**	73.95	**75.10**	**85.50**	84.35	**88.30**	**94.50**	92.34	**96.10**	**97.80**	96.44	**98.73**	**99.36**
ZENOTRAVEL	28.37	**48.00**	**58.00**	49.70	**76.90**	**85.70**	69.90	**89.20**	**94.80**	88.18	**96.80**	**98.80**	98.71	**98.73**	**99.58**

Table 2: Goal recognition accuracy (% of GR instances correctly predicted) by LGR, GRNet and LGRN with test set $\text{TS}_{\textit{PerGen}}$. Results for GRNet and LGRN are in bold when they are better than the corresponding results for LGR.

observed considering 50%, 70% and 100% of the actions. E.g., with 70% of the observed actions, the accuracy of GRNet is higher than 96 in SATELLITE and ZENOTRAVEL. GRNet always obtains higher accuracy w.r.t. LGR except in DEPOTS with 100% of the observations, and in many cases the performance improvement is of several points (e.g., more than 12 points for BLOCKSWORLD, DRIVERLOG, LOGISTICS and ZENOTRAVEL with 30% of the actions).

Regarding the accuracy of LGRN, it always performs better than both LGR and GRNet. Moreover, in several domains, we can see a remarkable improvement w.r.t. both LGR and GRNet, especially with 30% and 50% of the actions. For instance, in DEPOTS the accuracy of LGRN for 30% of actions is almost 72, 24 points better than LGR and 12 points better than GRNet. With 100% of the actions, the accuracy scores of GRNet and LGRN are higher than or equal to 90%, and still generally better than (especially for LGRN) the accuracy scores of LGR.

Moreover, GRNet's performance does not seem to be affected by the diversity of the domains indicated by the four parameters of Table 1. While the remarkable performance obtained for ZENOTRAVEL might be correlated with the fact that in this domain the test instances have only 66 facts (see column $|F|$ of Table 1), the results for SATELLITE are not so distant even if the instances in this domain have 629 facts. Analysing the experimental results, it seems that also the number of the actions has no significant impact on the performance. In fact, while BLOCKSWORLD has only 968 actions, the other domains have more than 15000 actions, and GRNet obtains better results for them. This is probably due to the embedding layer that is able to learn a compact and informative representation even with a large vocabulary of actions. Overall, GRNet exhibits good robustness with respect to the size of the space of actions and the number of facts in the domains (the output of the network).

θ-accuracy results for $\text{TS}_{\textit{PerGen}}$ Table 3 compares GRNet and LGRN with LGR in terms of θ-accuracy and the corresponding spread in $\mathcal{G}$. Considering θ equal to either 0 or 0.1 and partial plan traces (from 10% to 70% of observed actions), GRNet obtains a better θ-accuracy w.r.t. LGR in 44 out of 48 configurations, while LGRN has better performance in 47 out of 48 configurations. In several cases, the improvement is by several points, such as in DRIVERLOG with 10% of the actions. With $\theta = 0.2$, overall GRNet and LGRN perform better than LGR. There are two notable exceptions, DEPOTS and SATELLITE. In DEPOTS, LGR obtains a higher θ-accuracy w.r.t. GRNet but not w.r.t. LGRN. However, this result should be analysed also in terms of spread in $\mathcal{G}$. LGR has a considerably higher spread than GRNet, especially considering low percentages of actions. According to the definition of θ-accuracy, an instance is considered correctly solved if the true goal belongs to the selected *set* of goals, and so a higher spread can lead to a higher θ-accuracy. The same can be said for SATELLITE with $\theta = 0.2$, in which GRNet and LGRN have considerably lower spreads and worse θ-accuracy (but in most cases performing similarly).

We can observe that GRNet and LGRN have lower spreads in all but three considered configurations. In particular, with $\theta = 0$ the spread of GRNet is always 1. With $\theta > 0$, especially when considering low percentages of actions, we have a remarkable improvement in terms of spread w.r.t. LGR alongside an improvement in terms of θ-accuracy (see, e.g., DRIVERLOG and ZENOTRAVEL with 30% of the actions).

Concerning instances with complete plan traces (100% of the actions), in terms of θ-accuracy, all three systems obtain very good results, in many cases close to 100%. Although for these cases LGR often has better θ-accuracy, considering also the lower spreads of GRNet and LGRN, we think that overall the results for the full plan traces are comparable.

In term of CPU time to solve (classify) a GR instance, GRNet is generally much faster than LGR. The average execution time of LGR is 1.158 seconds with a standard deviation of 0.87 seconds, while GRNet runs on average in 0.06 seconds with a standard deviation of 0.04 seconds.

Results for $\text{TS}_{\textit{Per}}$ While we consider the evaluation using the extended set $\text{TS}_{\textit{PerGen}}$ more significant and informative than using the restricted set $\text{TS}_{\textit{Per}}$, we compared LGR, GRNet and LGRN also with $\text{TS}_{\textit{Per}}$. Overall, in terms of accuracy, θ-accuracy and spread, the results are still in favor of GRNet compared with LGR, and substantially better for LGRN compared with LGR. However, this is not the case for domain SATELLITE with test instances that have 70% of the actions. In this case LGR reaches accuracy 93.4, while GRNet and LGRN have accuracy 84.5 and 88.1, respectively. Most of the errors made by GRNet are due to the restricted and particular set of instances in $\text{TS}_{\textit{Per}}$, which has instances with very similar goals in $\mathcal{G}$. These are not clearly distinguished by GRNet, making LGRN less effective in such cases.

Domain	plan %	LGR						GRNet						LGRN					
		θ-Acc (↑)			Spread in $\mathcal{G}$ (↓)			θ-Acc (↑)			Spread in $\mathcal{G}$ (↓)			θ-Acc (↑)			Spread in $\mathcal{G}$ (↓)		
		0	0.1	0.2	0	0.1	0.2	0	0.1	0.2	0	0.1	0.2	0	0.1	0.2	0	0.1	0.2
BLOCKS	10	21.7	43.8	64.2	1.07	3.65	7.94	**23.0**	**56.9**	**65.9**	**1.00**	**3.39**	**4.46**	**31.2**	**53.4**	**68.3**	**1.00**	**2.49**	**4.51**
	30	40.4	63.3	77.9	1.07	3.07	6.43	**52.5**	**79.5**	**86.5**	**1.01**	**2.26**	**3.00**	**65.8**	**78.6**	**87.5**	**1.00**	**1.56**	**2.28**
	50	56.1	77.3	88.7	1.10	2.69	5.60	**71.3**	**90.3**	**94.3**	**1.00**	**1.89**	**2.55**	**80.2**	**88.3**	**93.3**	**1.00**	**1.28**	**1.66**
	70	75.9	89.8	96.2	1.11	2.14	4.40	**85.2**	**96.2**	**98.3**	**1.01**	**1.65**	**2.24**	**91.4**	**95.7**	**98.3**	**1.01**	**1.15**	**1.33**
	100	98.5	100.0	100.0	1.25	1.66	3.19	92.2	99.9	99.9	**1.00**	**1.46**	**1.96**	**98.6**	99.5	100.0	**1.00**	**1.05**	**1.13**
DEPOTS	10	27.2	47.3	69.6	1.15	2.49	4.56	**33.0**	**50.5**	56.4	**1.00**	**1.78**	**2.19**	**40.9**	**57.4**	**73.3**	**1.00**	**1.87**	**3.07**
	30	49.3	67.0	82.3	1.10	2.10	3.79	**59.8**	**72.6**	79.0	**1.00**	**1.45**	**1.73**	**72.8**	**81.4**	**86.9**	**1.00**	**1.34**	**1.72**
	50	68.0	80.8	92.1	1.07	1.78	3.08	**74.7**	**87.4**	90.9	**1.00**	**1.36**	**1.54**	**88.9**	**92.3**	**94.7**	**1.00**	**1.12**	**1.28**
	70	80.8	90.9	97.2	1.05	1.52	2.43	**84.9**	**92.4**	95.0	**1.00**	**1.25**	**1.38**	**94.2**	**96.4**	**97.7**	**1.00**	**1.05**	**1.13**
	100	96.2	99.2	99.8	1.06	1.30	1.63	91.9	96.6	98.2	**1.00**	**1.16**	**1.24**	**97.7**	98.1	98.5	**1.00**	**1.01**	**1.04**
DRIVERLOG	10	26.9	46.1	67.6	1.06	2.12	3.61	**39.8**	**58.4**	**71.4**	**1.00**	**1.72**	**2.40**	**45.6**	**54.2**	65.6	**1.00**	**1.38**	**1.90**
	30	43.1	64.1	80.8	1.07	1.97	3.23	**65.4**	**82.5**	**89.8**	**1.00**	**1.61**	**2.23**	**73.8**	**79.8**	**85.4**	**1.00**	**1.18**	**1.45**
	50	61.9	76.8	91.3	1.09	1.77	2.77	**78.4**	**90.5**	**96.4**	**1.00**	**1.45**	**1.96**	**84.3**	**88.6**	**91.6**	**1.00**	**1.11**	**1.24**
	70	80.3	90.3	96.3	1.11	1.58	2.30	**86.2**	**95.2**	**98.4**	**1.00**	**1.37**	**1.75**	**92.3**	**94.9**	**97.1**	**1.00**	**1.06**	**1.17**
	100	97.2	99.0	99.6	1.18	1.32	1.78	90.8	98.0	99.3	**1.00**	**1.27**	**1.57**	95.7	97.4	98.4	**1.00**	**1.04**	**1.10**
LOGISTICS	10	29.0	47.3	65.6	1.10	2.44	4.58	**39.2**	**57.1**	63.8	**1.00**	**1.66**	**2.06**	**44.2**	**64.1**	**79.7**	**1.00**	**2.02**	**3.70**
	30	57.4	71.5	82.3	1.07	1.71	2.82	**68.8**	**81.6**	**85.3**	**1.00**	**1.54**	**1.75**	**77.7**	**87.9**	**93.9**	**1.00**	**1.31**	**1.78**
	50	75.8	84.5	92.2	1.06	1.41	1.94	**80.4**	**91.0**	**92.6**	**1.00**	1.43	**1.58**	**90.9**	**95.3**	**97.9**	**1.00**	**1.14**	**1.33**
	70	89.0	93.7	98.0	1.11	1.25	1.50	**89.2**	**96.5**	98.0	**1.00**	1.32	**1.45**	**97.3**	**98.8**	**99.9**	**1.00**	**1.05**	**1.11**
	100	99.6	100.0	100.0	1.22	1.24	1.31	93.9	99.8	100.0	**1.00**	**1.18**	**1.24**	99.2	100.0	100.0	**1.00**	**1.01**	**1.02**
SATELLITE	10	47.4	84.5	97.7	1.27	3.25	5.27	45.5	68.9	81.1	**1.00**	**1.92**	**2.52**	**60.1**	83.5	94.2	**1.11**	**2.14**	**3.18**
	30	78.4	92.2	98.4	1.20	1.96	3.27	75.1	91.4	95.8	**1.00**	**1.56**	**1.82**	**87.1**	**94.0**	97.6	**1.11**	**1.34**	**1.61**
	50	87.9	95.7	98.8	1.16	1.52	2.20	**88.3**	**97.0**	98.6	**1.00**	**1.26**	**1.40**	**95.5**	**97.5**	**98.9**	**1.11**	**1.18**	**1.26**
	70	95.3	98.4	99.7	1.14	1.29	1.59	**96.1**	**99.4**	99.7	**1.00**	**1.13**	**1.23**	**97.9**	**99.0**	99.4	**1.11**	**1.14**	**1.17**
	100	98.5	99.6	99.8	1.12	1.19	1.37	**98.7**	99.6	99.8	**1.00**	**1.08**	**1.19**	**99.7**	**99.8**	99.8	**1.12**	**1.12**	**1.13**
ZENO	10	29.6	46.1	63.1	1.05	1.89	3.04	**48.0**	**71.7**	**89.6**	**1.00**	1.97	**2.98**	**59.4**	**70.0**	**80.8**	**1.00**	**1.43**	**2.00**
	30	50.5	67.4	80.0	1.03	1.74	2.65	**76.9**	**88.0**	**94.7**	**1.00**	**1.32**	**1.65**	**87.5**	**91.0**	**94.3**	**1.00**	**1.12**	**1.28**
	50	70.5	83.9	92.1	1.02	1.53	2.24	**89.2**	**94.7**	**97.7**	**1.00**	**1.15**	**1.27**	**95.7**	**97.4**	**98.6**	**1.00**	**1.04**	**1.09**
	70	88.6	94.4	98.8	1.01	1.27	1.64	**96.8**	**99.0**	**99.6**	**1.00**	**1.05**	**1.10**	**99.4**	**99.5**	**99.8**	**1.00**	**1.00**	**1.02**
	100	99.8	99.9	100.0	1.02	1.04	1.15	98.7	99.4	99.8	**1.00**	**1.01**	**1.04**	99.8	99.8	99.9	**1.00**	**1.00**	**1.01**

Table 3: θ-accuracy and spread of LGR, GRNet and LGRN for test set TS_{PerGen}. Results for GRNet and LGRN are in bold when they are better than the corresponding results for LGR.

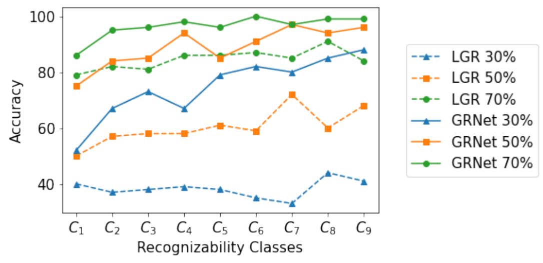

Figure 2: Accuracy results of LGR and GRNet on GR instances grouped into classes of decreasing difficulty with test set TS_{Rec}. C_1 is the most difficult class; C_9 is the easiest one.

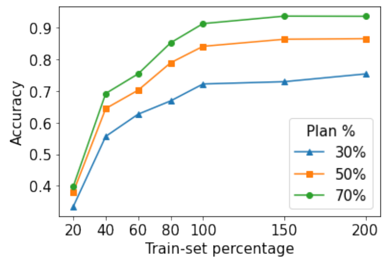

Figure 3: Accuracy of GRNet trained using data sets of different sizes (% of the original train set) using test set TS_{PerGen} in domain SATELLITE.

Results for TS_{Rec} and sensitiveness to the training set size Figure 2 shows the accuracy of the two compared systems considering different classes of test sets with decreasing difficulty measured using R_Z. We focus this analysis on instances with 30-50-70% of observed actions. As expected, the accuracy of GRNet depends on the difficulty of the problem, since there is an increasing trend in terms of accuracy for each observation percentage. This trend is more evident when we have 30% of the actions and becomes less marked as the number of observations grows. LGR appears to be more stable over the recognizability classes than GRNet. However, GRNet always performs significantly better than LGR regardless the value of R_Z.

Since the predictive performance of a machine learning system can be deeply influenced by the number of training instances, we experimentally investigated how much GRNet is sensible to this issue. We focus the analysis on the domain SATELLITE, training several neural networks with different fractions of our training set: 20%, 40%, 60% and 80%. Figure 3 shows how accuracy increases for TS_{PerGen} when the training set size increases. In particular, for TS_{PerGen} we can observe that using only 20% of the training instances gives accuracy lower than 40 in all three cases (30-50-70% of observed actions), but accuracy rapidly improves reaching more than 60 using 60% of the training instances.

We evaluated GRNet also for larger training sets, up to twice the number of instances in the original training set. As it can be seen in Figure 3, the enlarged training set for TS_{PerGen} produces only a small improvement in accuracy.

Related Work

Goal recognition has been extensively studied through model-based approaches exploiting planning techniques (Meneguzzi and Pereira 2021; Ramírez and Geffner 2010; Sohrabi, Riabov, and Udrea 2016; Santos et al. 2021; Pereira, Oren, and Meneguzzi 2020) or matching techniques relying on plan libraries (e.g., (Mirsky et al. 2016)). LGR is a state-of-the-art model-based approach exploiting plan generation and landmarks (Pereira, Oren, and Meneguzzi 2020). Differently from GRNet, LGR uses domain knowledge and performs no learning from previous experiences.

Several approaches to human activity recognition adopt architectures based on a RNN (Yin et al. 2022; Chen et al. 2016). These systems address a problem different from goal recognition; they deal with noisy input data from sensors, and perform a specific classification task (with fixed classification values). Consequently, these architectures provide solutions to very specific problems. The work in this paper addresses goal recognition, it deals with the lack of observability in the actions of the agent's plan, and proposes a more general approach allowing to solve, by the same trained network, different goal recognition instances in the domain.

Concerning GR systems using neural networks, some works use them for specific applications, such as game playing (Min et al. 2016). GRNet is more general, as it can be applied to any domain of which the sets of fluents and actions (F and A) are known. In order to extract useful information from image-based domains and perform goal recognition, Amado et al. (2018) used a pre-trained encoder and a LSTM network for representing and analysing a sequence of observed states, rather than actions as in our approach. Amado et al. (2020) trained a LSTM-based system to identify missing observations about states in order to derive a more complete sequence of states by which a MBGR system can obtain better performance.

Borrajo, Gopalakrishnan, and Potluru (2020) investigated the use of XGBoost and LSTM neural networks for goal recognition using only traces of plans, similarly to our approach. However, they train a specific machine learning model *for each goal recognition instance* (the goal set $\mathcal{G}$ is fixed), using instance-specific datasets for training and testing. In our approach, we train a general-purpose neural network that can be used to solve a large number of different goal recognition instances, without the need of designing or training a new model. Moreover, the experimental evaluation of the networks proposed in (Borrajo, Gopalakrishnan, and Potluru 2020) use peculiar goal recognition benchmarks with custom-made instances. Instead, in our work we evaluate GRNet and LGRN much more in depth by a substantially larger experimental analysis.

Maynard, Duhamel, and Kabanza (2019) compared model-based techniques and approaches based on deep learning for goal recognition. However, as in (Borrajo, Gopalakrishnan, and Potluru 2020), such a comparison is made using specific instances, and several kinds of neural networks are trained to directly predict the goal among a set of possible ones, instead of the facts that belong to the goal, as in our approach. This makes the trained networks in (Maynard, Duhamel, and Kabanza 2019) specific for the considered GR instances in a domain, while our approach is more general since it trains a single network for the domain. Another substantial difference is that, while in a typical goal recognition problem we can have missing observations across the entire plan of the agent(s), the work in (Maynard, Duhamel, and Kabanza 2019) considers only observations from the start of the plan to a given percentage of it, treating every possible successive observation as missing.

Amado, Mirsky, and Meneguzzi (2022) proposed a framework that combines off-the-shelf model-free reinforcement learning and state-of-the-art goal recognition techniques achieving promising results. However, similarly to (Borrajo, Gopalakrishnan, and Potluru 2020), their approach is designed to solve a specific goal recognition instance where the goal set $\mathcal{G}$ is fixed. On the contrary, the trained RNN of GRNet is used to solve all GR instances definable over the fluent and action sets.

Conclusions

We have proposed GRNet, an approach to address goal recognition as a deep learning task. Our system learns to solve (classify) goal recognition tasks from past experience in a given domain. Learning consists in training only one neural network for the considered domain, allowing to solve a large collection of GR instances in the domain by the same trained network. Moreover, GRNet can be effectively integrated with the state-of-the-art model-based system LGR. An experimental analysis shows that GRNet and LGRN, our system integrating GRNet and LGR, perform generally well for the considered benchmark domains, in terms of accuracy, θ-accuracy and spread.

Differently from LGR, the GR instances addressable by GRNet are limited to those involving subsets of fluents and actions that were used in the training phase. If the GR instance to solve involves a new fluent, clearly such a fluent cannot be predicted in GRNet; if the instance involves a new action, such an action cannot be part of the input observed actions for GRNet. An interesting question for future work is how to extend GRNet to solve GR instances involving new actions and fluents; this can be performed considering the object names involved in the GR instance to solve, and defining a mapping with the object names of instances considered in the training phase. Preliminary results in this direction are encouraging. We also intend to investigate the use of other deep learning architectures (Serina et al. 2022), as well other ways of integrating model-free and model-based approaches.

Acknowledgements

This research has been carried out with the support of EU H2020 project AIPlan4EU (GA n. 101016442), EU ICT-48 2020 project TAILOR (No. 952215), and MUR PRIN project RIPER (No. 20203FFYLK). Francesco Percassi was supported by a UKRI Future Leaders Fellowship (grant number MR/T041196/9).

References

Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; and Koyama, M. 2019. Optuna: A next-generation hyperparameter optimization framework. In *Proceedings of the 25th ACM SIGKDD*, 2623–2631.

Amado, L.; Aires, J. P.; Pereira, R. F.; Magnaguagno, M. C.; Granada, R.; and Meneguzzi, F. 2018. LSTM-Based Goal Recognition in Latent Space. *CoRR*, abs/1808.05249.

Amado, L.; Licks, G. P.; Marcon, M.; Pereira, R. F.; and Meneguzzi, F. 2020. Using Self-Attention LSTMs to Enhance Observations in Goal Recognition. In *Proceedings of IJCNN 2020*. IEEE.

Amado, L.; Mirsky, R.; and Meneguzzi, F. 2022. Goal Recognition as Reinforcement Learning. In *Proceedings of AAAI 2022*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *ICLR 2015, Conference Track Proceedings*.

Batrinca, L.; Mana, N.; Lepri, B.; Sebe, N.; and Pianesi, F. 2016. Multimodal Personality Recognition in Collaborative Goal-Oriented Tasks. *IEEE Transactions on Multimedia*, 18(4).

Bengio, Y.; Ducharme, R.; Vincent, P.; and Janvin, C. 2003. A neural probabilistic language model. *The journal of machine learning research*, 3.

Borrajo, D.; Gopalakrishnan, S.; and Potluru, V. K. 2020. Goal recognition via model-based and model-free techniques. *Proceedings of FinPlan 2020*.

Carberry, S. 2001. Techniques for Plan Recognition. *User Model. User Adapt. Interact.*, 11(1-2): 31–48.

Chen, Y.; Zhong, K.; Zhang, J.; Sun, Q.; and Zhao, X. 2016. LSTM networks for mobile human activity recognition. In *Proceedings of ICAITA 2016*, 50–53. Atlantis Press.

Geffner, H. 2018. Model-free, Model-based, and General Intelligence. In *Proceedings of IJCAI 2018*.

Geib, C.; and Pynadath, D. 2007. Plan, Activity, and Intent Recognition. *AI Magazine*, 28(4): 124.

Gerevini, A.; Saetti, A.; and Serina, I. 2003. Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. *J. Artif. Intell. Res.*, 20: 239–290.

Gerevini, A.; and Serina, I. 2003. Planning as Propositional CSP: From Walksat to Local Search Techniques for Action Graphs. *Constraints An Int. J.*, 8(4): 389–413.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2016. *Automated Planning and Acting*. Cambridge University Press. ISBN 978-1-107-03727-4.

Hochreiter, S.; and Schmidhuber, J. 1997. Long Short-term Memory. *Neural computation*, 9.

Long, D.; and Fox, M. 2003. The 3rd International Planning Competition: Results and Analysis. *J. Artif. Intell. Res.*, 20.

Maynard, M.; Duhamel, T.; and Kabanza, F. 2019. Cost-Based Goal Recognition Meets Deep Learning. *Proceedings of PAIR 2019*.

McDermott, D. V. 2000. The 1998 AI Planning Systems Competition. *AI Mag.*, 21(2): 35–55.

Meneguzzi, F.; and Pereira, R. F. 2021. A Survey on Goal Recognition as Planning. In *Proceedings of IJCAI 2021*.

Min, W.; Mott, B. W.; Rowe, J. P.; Liu, B.; and Lester, J. C. 2016. Player Goal Recognition in Open-World Digital Games with Long Short-Term Memory Networks. In *Proceedings of IJCAI 2016*. IJCAI/AAAI Press.

Mirsky, R.; Shalom, Y.; Majadly, A.; Gal, K.; Puzis, R.; and Felner, A. 2019. New Goal Recognition Algorithms Using Attack Graphs. In *CSCML 2019, Proceedings*, volume 11527. Springer.

Mirsky, R.; Stern, R.; Gal, Y. K.; and Kalech, M. 2016. Sequential Plan Recognition. In *Proceedings of IJCAI 2016*. IJCAI/AAAI Press.

Pereira, R. F.; Oren, N.; and Meneguzzi, F. 2020. Landmark-based approaches for goal recognition as planning. *Artif. Intell.*, 279.

Ramírez, M.; and Geffner, H. 2009. Plan Recognition as Planning. In *Proceedings of IJCAI 2009*.

Ramírez, M.; and Geffner, H. 2010. Probabilistic Plan Recognition Using Off-the-Shelf Classical Planners. In *Proceedings of AAAI 2010*. AAAI Press.

Richter, S.; and Westphal, M. 2010. The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. *J. Artif. Intell. Res.*, 39: 127–177.

Santos, L. R. A.; Meneguzzi, F.; Pereira, R. F.; and Pereira, A. G. 2021. An LP-Based Approach for Goal Recognition as Planning. In *Proceedings of AAAI 2021*. AAAI Press.

Serina, L.; Chiari, M.; Gerevini, A. E.; Putelli, L.; and Serina, I. 2022. A Preliminary Study on BERT applied to Automated Planning. In *Proceedings of IPS 2022*, volume 3345 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Sohrabi, S.; Riabov, A. V.; and Udrea, O. 2016. Plan Recognition as Planning Revisited. In Kambhampati, S., ed., *Proceedings of IJCAI 2016*. IJCAI/AAAI Press.

Van-Horenbeke, F. A.; and Peer, A. 2021. Activity, Plan, and Goal Recognition: A Review. *Frontiers Robotics AI*, 8.

Vrigkas, M.; Nikou, C.; and Kakadiaris, I. A. 2015. A Review of Human Activity Recognition Methods. *Frontiers Robotics AI*, 2: 28.

Yang, Z.; Yang, D.; Dyer, C.; He, X.; Smola, A. J.; and Hovy, E. H. 2016. Hierarchical Attention Networks for Document Classification. In Knight, K.; Nenkova, A.; and Rambow, O., eds., *Proceedings of NAACL HLT 2016*.

Yin, X.; Liu, Z.; Liu, D.; and Ren, X. 2022. A Novel CNN-based Bi-LSTM parallel model with attention mechanism for human activity recognition with noisy data. *Scientific Reports*, 12(1): 1–11.