Regularised Loss Function for Goal Recognition as a Deep Learning Task

Matteo Olivato[1], Mattia Chiari[1], Lorenzo Serina[1], Valerio Borelli[1], Massimiliano Tummolo[1], Ivan Serina[1], Nicholas Rossetti[1], and Alfonso Emilio Gerevini[1]

University of Brescia, Via Branze 38, 25121 Brescia, Italy
{matteo.olivato,mattia.chiari,lorenzo.serina,
valerio.borelli,massimiliano.tummolo,ivan.serina,
nicholas.rossetti,alfonso.gerevini}
@unibs.it

Abstract. Goal Recognition (GR) consists of recognising the goal of an agent from partial observations. The state of the art on particular planning domains is represented by GRNet, a model based on Recurrent Neural Networks that solves GR as a classification task. Compared to automated planning, the need for large training sets is the main disadvantage of these approaches. Therefore, we formalise a loss regularisation technique to reduce the number of training samples needed, to reduce the convergence time, and to increase the performance in GR instances with a small percentage of observations. We empirically evaluate its effectiveness through extensive experiments.

Keywords: Goal Recognition · Loss Regularisation · Recurrent Neural Networks · Deep Learning · Training Sample Reduction · Convergence Time · Partial Observations · Automated Planning

1 Introduction

Classical Planning is a fundamental part of Automated Planning, an important field of Artificial Intelligence. The model underlying classical planning can be described as a model $\mathcal{S} = \langle S, s_0, S_G, A, f, c \rangle$, where S is a finite set of possible states that are a representation of all the possible configurations of the environment; $s_0 \in S$ is the initial state; $S_G \subseteq S$ is the non-empty set of goal states; $A(s) \subseteq A$ is the set of actions in A that are applicable in each state $s \in S$; $f(a, s)$ is a deterministic transition function where $s' = f(a, s)$ is the state that follows s after applying action $a \in A(s)$; $c(a, s)$ is a positive cost for executing action a in the state s. A solution or *plan* π is a sequence of applicable actions $\pi = a_0, a_1, \ldots, a_n$ that generates a state sequence $s_0, s_1, \ldots, s_{n+1}$ where $a_i \in A(s_i)$, $s_i \in S$, $s_{i+1} = f(a_i, s_i)$ and s_{n+1} is a goal state (i.e., $s_{n+1} \in S_G$). Classical Planning domains and problems are often represented using the Planning Domain Definition Language (PDDL) [7,11], where a problem is represented

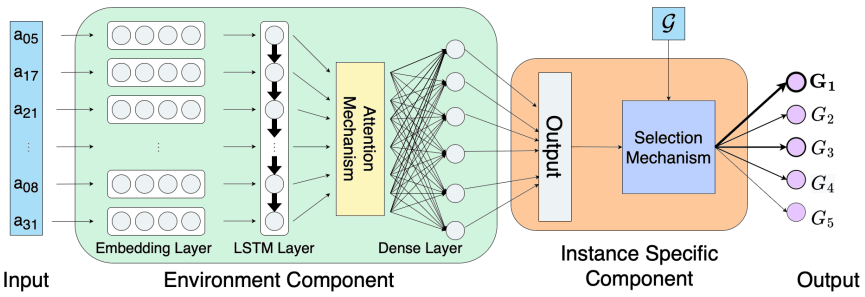


Fig. 1: The GRNet architecture where the EC, based on neural components, predicts the true facts from the sequence of observed actions and the ISC uses the predicted facts to identify the most probable goal G_i in the goal set $\mathcal{G}$.

by a tuple $P = \langle F, I, A, G \rangle$ where F represents the set of atoms (fluents) or propositions of interest; $I \subseteq F$ represents the initial state; A represents the set of actions; $G \subseteq F$ represents the goal.

GR involves identifying the objective to which an agent aims by monitoring the agent's actions within the environment [4,18]. The study of GR plays a crucial role in various domains such as human-computer interactions [1], computer games [13], network security [14], smart homes [8], financial applications [2], among others. One technique for solving the GR task is *model-free goal recognition* (MFGR) [4,2], where GR is formulated as a classification task addressed by machine learning methods. An instance of MFGR for a domain is specified by an observation sequence $O = \langle obs_1, ..., obs_n \rangle$, formed by action labels, and a goal set $\mathcal{G}$ formed by subsets of F. An advantage of the MFGR approach is that it requires minimal information about the domain actions and can operate without the specification of an initial state that can be completely unknown.

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

Domain	$\lvert A\rvert$	$\lvert F\rvert$	$\lvert G_i\rvert$	$\lvert\mathcal{G}\rvert$	$\lvert Train\rvert$	$\lvert Val\rvert$
BLOCKSWORLD	968	506	[4,16]	[19,21]	55000	28148
DEPOTS	13050	150	[2,8]	[7,10]	55000	26654
DRIVERLOG	4860	156	[4,11]	[6,10]	55000	34702
LOGISTICS	15154	154	[2,4]	[10,12]	55000	31410
SATELLITE	33225	629	[4,9]	[6,8]	55000	23477
ZENOTRAVEL	23724	66	[5,9]	[6,11]	55000	37075

The research in [3] leveraged this formalisation to treat the GR task as a Deep Learning problem, presenting GRNet, an architecture built on Recurrent Neural Networks (LSTM cells) shown in Fig. 1. This model can process a sequence of observations depicted as the agent's actions and output the set of binary predicates (fluents) representing its goal.

The input of the model is the agent's observation trace, which may be incomplete. It addresses the GR task as a multi-label classification problem in the EnvironmentComponent (EC) part of the model, hence having an output size of $|F|$ equal to the number of possible fluents (facts) in the goal. It uses binary cross-entropy (BCE) as the loss function, which in a multi-label setup could be formalised as the following.

$$\mathcal{L}_{BCE}(y, \hat{y}) = -\sum_{j=1}^{C}(y_j \cdot log(\hat{y}_j) + (1 - y_j) \cdot log(1 - \hat{y}_j)) \tag{1}$$

where C is the number of fluents (classes) in the label, and y_j is the label of the fluent j with respect to the goal G_i, and the $\hat{y}_j$ is the prediction for the fluent j.

The goal selection process is a post-processing step by the so-called InstanceSpecificComponent (ISC). The network outputs (logits) are multiplied by the masks corresponding to each potential goal to retain the values at the positions of true facts. These values are then summed to calculate a score for each goal, and the highest score determines the correct goal predicted by the network.

GRNet is proven to be more efficient than model-based techniques, as the inferences in deep learning models are typically faster than logical planners. Nevertheless, during the training phase it shows some issues. First, the accuracy metric for the predicted facts ($\text{ACC}_{\text{FACTS}}$), the output of EC, remains near zero for many epochs during this initial period, approximately from 5 to 10 depending on the domain, and the loss function (BCE) slowly decreases. Second, the number of epochs in which $\text{ACC}_{\text{FACTS}}$ is almost zero increases with the reduction of the training size. A possible reason could be found in the difference between the high number of possible facts $|F|$ as the output space of EC and the few facts that identify a goal $|G_i|$, as indicated in Table 1. For example, in BLOCKSWORLD the possible predictable facts are more than 500 but the facts made true by an observation sequence are usually less than 16 (on average 10). This difference produces a label with high numbers of zeros (false facts) and a few numbers of ones (true facts), which is hard to correctly predict because the statistical power of the true facts is really low (e.g. $10/500 = 2\%$).

As a countermeasure, we try to improve the performance of the model by changing the loss used during the training of the EC. We study the performance of the model with this new loss, with respect to the $\text{ACC}_{\text{FACTS}}$ and the ACC_{GR} downstream of ISC, the epochs needed to converge, and the number of training instances provided to the model.

2 Background and Related Works

The binary cross-entropy (BCE) loss function ($\mathcal{L}_{BCE}$) is widely used in binary classification problems, but it can be susceptible to overfitting, especially when dealing with imbalanced datasets or noisy labels. Consequently, various regularisation techniques have been proposed to mitigate these issues. A common approach to regularisation involves adding penalty terms to the loss function that usually is based on the model weights (e.g., $||w||_2^2$ for L_2 regularisation) but formally is an arbitrary parameter.

In some cases, regularisation can be applied multiplicatively and the regularisation term directly scales the loss dynamically.

$$\mathcal{L}_{reg} = \mathcal{L}_{BCE} \cdot \lambda\mathcal{R}$$

where $\mathcal{R}$ is the regularisation term considered as a variable scaling factor and λ is a constant scaling factor. The focal loss proposed by Lin et al. [9] is an example of this regularisation approach. They focus on training on hard-to-classify examples by down-weighting the contribution of well-classified examples addressing the issue of class imbalance. The focal loss modifies the standard $\mathcal{L}_{BCE}$ by introducing a multiplicative modulating factor:

$$\mathcal{L}_{focal} = \mathcal{L}_{BCE} \cdot \lambda\mathcal{R} = -\log(p_t) \cdot \alpha_t(1-p_t)^\gamma$$

where p_t is the predicted probability for the true class, $\log(p_t)$ is the standard binary cross-entropy component, α_t is a weighting factor λ for class imbalance and $(1-p_t)^\gamma$ is the regularisation term R. The weighting factor α_t assigns a higher weight to the minority class, helping to balance the contribution of different classes to the overall loss. When an example is easily classified (i.e., p_t is close to 1), this factor approaches 0, reducing the loss contribution. The parameter γ is the focussing parameter that controls the rate at which easy examples are down-weighted. This regularisation effectively handles class imbalance and, for instance, improves the performance of one-stage object detectors.

A different regularisation approach uses the labels as regularisation terms. For instance, *label smoothing* is a regularisation technique that replaces hard labels (0 or 1) with soft labels, introducing uncertainty, and preventing the model from becoming overconfident [17] improving model calibration and reducing overfitting. For binary classification, this can be achieved by replacing the target labels y with:

$$y_{smoothed} = (1-\epsilon)y + \epsilon(1-y)$$

where ϵ is a small constant.

This paper explores a novel regularisation technique for the binary cross-entropy loss function which combines the multiplicative regularisation approach and information contained in the labels with the aim of further improving the generalisation performance and robustness of GRNet.

3 Proposal

As previously introduced, EC of GRNet showed convergence problems during the initial training phase. The model decreases the loss value and increases $\text{ACC}_{\text{FACTS}}$ very slowly over the validation set at the beginning. This convergence issue affects both the maximum performance of GRNet in terms of the final ACC_{GR}, limits the reduction in training size, and indirectly increases the environmental impact of model training in terms of CO_2 production.

To address these issues, we propose a novel loss function $\mathcal{L}_{TFS}$ as the combination of the BCE and a multiplicative regularisation parameter, TFS, which is the sum of true facts in the label for each sample. We multiply $\mathcal{L}_{BCE}$ by the reciprocal of the regularisation parameter to obtain a regularised loss function suitable for multi-label classification. The formulas for the proposed TFS regularisation and the relative loss function are the following.

$$TFS(y) = \sum_{j=1}^{C} y_j + \epsilon \tag{2}$$

$$\mathcal{L}_{TFS}(y, \hat{y}) = \mathcal{L}_{BCE}(y, \hat{y}) \cdot \mathcal{R} \tag{3}$$

$$= \mathcal{L}_{BCE}(y, \hat{y}) \cdot \frac{1}{TFS(y)} \tag{4}$$

where y is the facts label, $\hat{y}$ the facts label predicted by EC of GRNet, y_j is the true value of the fact at position j in the label y, the $\mathcal{L}_{BCE}$ is the standard binary cross-entropy loss and the reciprocal of $TFS(y)$ is the regularisation term $\mathcal{R}$ which depends on y. We added $\epsilon = 10^{-7}$ to avoid division by zero errors, adopting the default value of our deep learning framework[1].

The intuition behind this regularisation is that we reduce the error importance on those observation sequences that have many true facts in their label, and, conversely, we increase the importance of those samples with few true facts in their label. We normalise by the number of true facts (ones present in the label) to balance the importance of different samples, reducing the bias of learning samples with many facts despite the others. For example, in BLOCKSWORLD an observation sequence can have between 4 and 16 true facts, as shown in Table 1. For an untrained model, the magnitude of the standard $\mathcal{L}_{BCE}$ for samples with a label having 4 true facts (ones) is four times smaller than for samples with 16 true facts. Therefore, the model focusses on learning samples with many true facts until the magnitude of the error on those samples becomes lower than those with few true facts. For instance, the model should be able to correctly predict around 12 over 16 ones until it really starts to optimise over samples with 4 ones. This affects the overall convergence time. Moreover, the final goal sets to predict are usually disjoint, which implies that if the model (via the EC) can correctly predict just a few of the true facts, then it would be able to correctly select

[1] https://keras.io/api/utils/config_utils/#epsilon-function

the right goal in the post-processing phase (via the ISC). Therefore, learning to correctly predict facts (even few) for many different samples increases the final goal recognition performance with respect to learning to correctly predict many facts for few samples.

4 Experimental Results

In this section, we compare the results obtained in terms of ACC_{GR} by $\mathcal{L}_{BCE}$ and $\mathcal{L}_{TFS}$ in different planning domains, considering different training sizes and percentages of observations. Then, we analyse the convergence time on the validation set of the EC using the two loss functions in different domains and different training sizes, taking as the main metric the accuracy over true facts in the label y.

4.1 Methodology

We followed the work of Chiari et al. [3] which focus on six benchmark domains: BLOCKSWORLD, DEPOTS, DRIVERLOG, LOGISTICS, SATELLITE and ZENOTRAVEL [12,10]. Concerning the training set, they randomly generated a large collection of (solvable) planning problems of different sizes for each domain, considering the same ranges for the number of objects as in [15]. For each of these problems, they computed up to four (sub-optimal) solution plans via the LPG planner [5,6]. From the generated plans, the authors randomly selected between 30% and 70% of the actions (preserving their relative order) to obtain the observation sequences. Therefore, more formally, the generated training set consists of pairs (O, G^*) where O is a observation sequence 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 specific domain considered, they created a training set $Train$ with 55000 pairs of GR instances (O, G^*), and a fixed validation set Val of different size depending of the domain, as indicated in Table 1.

To test GRNet, Chiari et al. generated a test set called TS_{PerGen} of unseen GR instances following the previous methodology but using a different planner, LAMA [16], to improve the robustness of the results. TS_{PerGen} is a generalisation and extension of the test set used in Pereira et al. [15], called TS_{Per}, where the goal sets ($\mathcal{G}$) are the same and the TS_{Per} instances are included. For each plan generated for being sampled, the authors randomly derived four different partial action traces formed by 10%, 30%, 50%, 70% of the plan actions, respectively. This creates four groups of test instances, allowing them to evaluate the performance of GRNet across different amounts of available observations for each domain. Table 1 gives information on the size of the GR instances in the train and test sets for each domain, in terms of the number of possible actions ($|A|$), facts ($|F|$), min/max size of a goal ($|G_i|$) in a goal set $\mathcal{G}$, that is the number of true facts needed to identify a goal i, and min/max size of a goal set ($|\mathcal{G}|$). As an evaluation measure, Chiari et al. used the GR *accuracy* (ACC_{GR}) as the main evaluation criterion, which is defined as the percentage of instances whose goals

are correctly identified (predicted) over the total number of instances in the test set.

In order to validate our proposal, we compare the $\mathcal{L}_{BCE}$ and the $\mathcal{L}_{TFS}$ extending the methodology by Chiari et al. by also considering different training sizes. In particular, we tried to reduce the training size by considering only a subset of the original observation sequences for each domain. For each training set, we create a subset of 70%, 80%, or 90% of the original 55000 samples in $Train$, thus having 38500, 44000, 49000, respectively. These subsets are randomly extracted by stratifying the observation traces by the sum of true facts in their label y, that is, the value $TFS(y)$. The stratification is needed with a reduction in the training samples to avoid overfitting on a particular kind of sequences, e.g. observation sequences with a small number of true facts or vice versa. To test our approach and obtain a significant comparison, we used the validation set Val and test set TS_{PerGen} same as those used by Chiari et al. without modifications.

4.2 Results: GR Accuracy

Table 2: Goal recognition accuracy, Acc_{GR}, by $\mathcal{L}_{BCE}$ and $\mathcal{L}_{TFS}$ on TS_{PerGen} considering different observation and training set percentages. Best in bold.

Domain	Train%	10%		30%		50%		70%	
		BCE	TFS	BCE	TFS	BCE	TFS	BCE	TFS
Blocksworld	70%	22.83	**27.97**	49.43	**61.50**	61.77	**78.12**	75.45	**89.80**
	80%	25.35	**27.57**	52.77	**59.81**	69.31	**76.63**	81.28	**89.16**
	90%	22.64	**26.05**	46.68	**47.93**	61.54	**62.12**	**73.95**	73.17
Depots	70%	**33.79**	32.98	59.39	**59.46**	73.71	**74.64**	85.52	**85.83**
	80%	32.95	**33.93**	59.64	**60.86**	74.84	**76.73**	86.31	**87.60**
	90%	32.64	**33.87**	58.50	**59.75**	74.81	**76.48**	84.42	**87.83**
Driverlog	70%	**33.69**	33.27	54.56	**55.45**	67.84	**68.30**	**77.36**	77.26
	80%	**35.00**	34.11	53.18	**55.58**	64.14	**68.88**	73.02	**77.96**
	90%	32.41	**35.70**	51.60	**56.62**	63.84	**68.30**	73.93	**76.73**
Logistics	70%	18.61	**36.10**	21.02	**59.52**	24.06	**74.75**	25.52	**85.73**
	80%	34.98	**36.35**	61.12	**61.82**	76.53	**77.17**	87.22	**87.32**
	90%	34.38	**35.92**	**60.43**	58.74	**75.72**	73.17	**86.43**	84.72
Satellite	70%	39.94	**41.87**	58.82	**62.07**	67.99	**73.33**	73.95	**79.01**
	80%	41.84	**44.50**	63.11	**65.35**	74.80	**76.44**	81.95	**82.18**
	90%	40.24	**45.53**	64.19	**70.67**	78.51	**81.37**	86.35	**86.64**
Zenotravel	70%	46.50	**47.55**	76.71	**77.13**	89.03	**90.10**	96.12	**96.15**
	80%	45.49	**46.55**	74.97	**76.80**	88.29	**89.57**	94.99	**96.22**
	90%	44.02	**45.87**	70.70	**75.24**	85.64	**89.11**	94.02	**95.86**

In Table 2 the results in terms of ACC_{GR} are shown for different training and observation percentages. For observation sequences that retain the 10% of the original plan, ACC_{GR} is favourable to $\mathcal{L}_{BCE}$ only few times, often with the lowest percentage of training 70%. This is due to the high difficulty of the prediction task and the intrinsic variability in predicting the most informative facts that identifies the correct goal. With so few actions, being able to predict that a fact is made true by the agent is really hard, and focussing on learning those sequences that have a small number of true facts in the label might be counter-productive. However, in general, the differences between ACC_{GR} obtained by $\mathcal{L}_{BCE}$ and $\mathcal{L}_{TFS}$ are small even in this situation. For example, DEPOTS and DRIVERLOG with the 70% train and 10% of observations have a reduction in the ACC_{GR} less than 1% (0.8% for DEPOTS and 0.42% for DRIVERLOG).

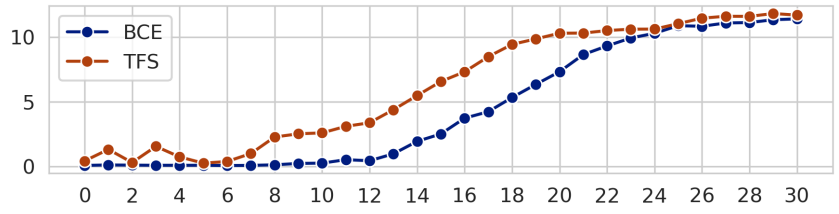
A similar behaviour is revealed when considering sequences with 70% of the original observations, where only 30% of the original actions are dropped. The $\mathcal{L}_{BCE}$ and $\mathcal{L}_{TFS}$ losses reached very similar performance in general, with the BCE that only a few times overcome the TFS regularisation performance. For instance, BLOCKSWORLD 90% and DRIVERLOG 70% with 70% of observations have a reduction in final performance using TFS of less than 1% (around 0.8% for BLOCKSWORLD and 0.10% for DRIVERLOG).

The proposed regularisation shows very interesting results, particularly in the 30% and 50% percentages of observed actions, where it not only shows a better ACC_{GR} than BCE but also shows to be robust in the case of reduction of train size with respect to all percentages considered (70%, 80%, 90%).

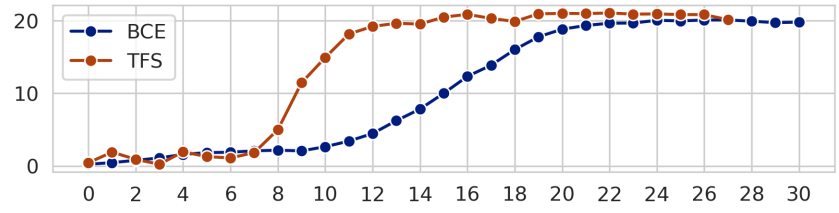
4.3 Results: Convergence Time

In Fig. 2 we show the validation accuracy in terms of facts prediction obtained by the EC for the training phases of BLOCKSWORLD 70%, DEPOTS 80% and LOGISTICS 90%. These validation plots compare the convergence time of the regularised loss function $\mathcal{L}_{TFS}$ with the baseline $\mathcal{L}_{BCE}$ with different training sizes. They show a reduction in the convergence time of the proposed loss with respect to the baseline as expected. In particular, for BLOCKSWORLD 70%, EC reached 10% $\text{ACC}_{\text{FACTS}}$ at epoch 18 using TFS regularisation and at epoch 24 without it, obtaining a savings of around 5 training epochs. For DEPOTS 80%, the model reached 20% $\text{ACC}_{\text{FACTS}}$ at nearly epoch 12 with $\mathcal{L}_{TFS}$ while around epoch 22 with $\mathcal{L}_{BCE}$, allowing a savings of approximately 10 epochs of training. Finally, in LOGISTICS 90%, the model reached over 30% of $\text{ACC}_{\text{FACTS}}$ at the epoch 16 and in the epoch 24 with the $\mathcal{L}_{TFS}$ and $\mathcal{L}_{BCE}$ respectively, anticipating by 8 epochs on average. In addition, it is important to note that in all the experiments shown the final fact accuracy is slightly higher for the training with the TFS regularisation with respect to the BCE. This small increase in $\text{ACC}_{\text{FACTS}}$ at the end of the training phase affects the precision of ISC that is able to better select the right goal perceived by the agent. In fact, for the previous examples, the TFS regularisation improved the ACC_{GR} of GRNet as shown in Table 2.

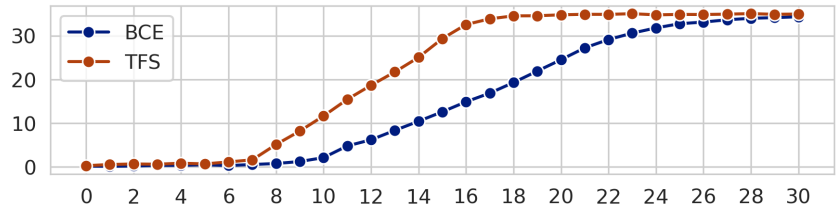
The behaviour previously described is consistent and generalised between different train and observation percentages, with some exceptions as shown in



(a) Blocksworld 70%



(b) Depots 80%



(c) Logistics 90%

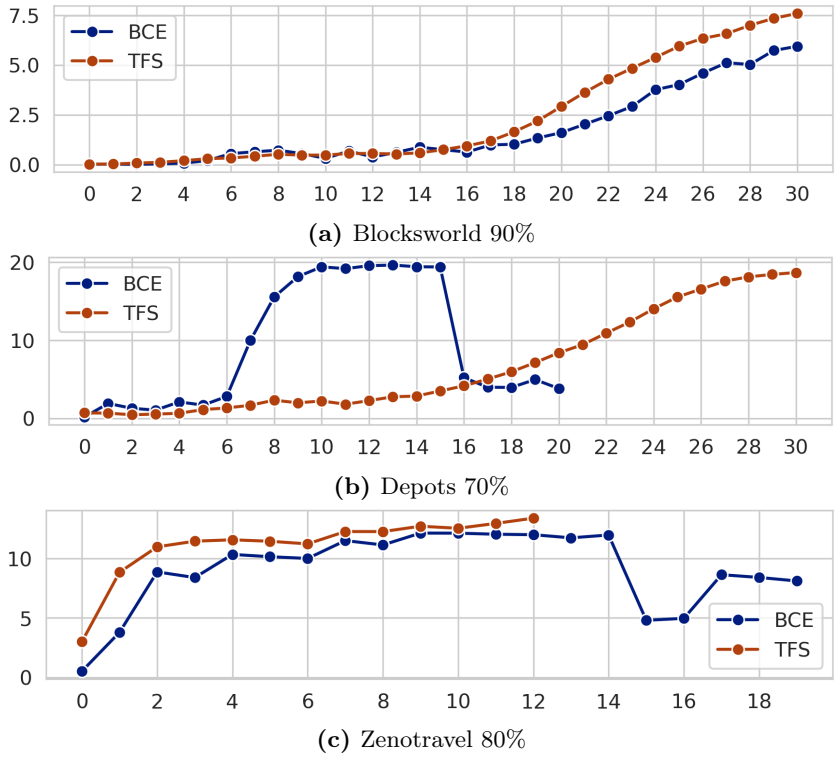
Fig. 2: The comparison of the validation accuracy (y-axis) on facts prediction over time between the $\mathcal{L}_{BCE}$ and the $\mathcal{L}_{TFS}$ over the epochs (x-axis).

Fig. 3. For BLOCKSWORLD 90%, there is no anticipation in the convergence epochs; instead, we notice that it starts to converge almost at the same time for both losses, but $\text{ACC}_{\text{FACTS}}$ of $\mathcal{L}_{TFS}$ ends with an increase of 2.5% with respect to $\mathcal{L}_{BCE}$. This improvement affects ACC_{GR} especially for small observation percentages as described in Table 2, with a considerable improvement with only 10% observations. In the case of DEPOTS 70%, the convergence of the fact accuracy begins much earlier for the $\mathcal{L}_{BCE}$ (around the 6 epoch) and is very fast. However, after the 14 epoch it drops until the training end. This happens because the Early Stopping mechanism of EC works on the validation loss as usual, stopping the training when it no longer decreases, without directly considering the value of $\text{ACC}_{\text{FACTS}}$ in validation over time. On the other hand, $\mathcal{L}_{TFS}$ starts to converge in many epochs after, but it ends slightly lower the maximum of BCE, around the 20% accuracy consistently. In Table 2, the ACC_{GR} are similar for both losses with a considerable margin only in the case of 50% of observation for $\mathcal{L}_{TFS}$ in this situation. A possible interpretation could be related to the fact that, with a reduced number of sequences (70%) in this particular domain that has a particular low number of facts $|F|$, the BCE incurs in overfitting over simple samples, meanwhile the TFS weighting more difficult one reduced the initial convergence time, but it obtains a smoother training. In contrast, in the same domain, when the number of examples is enough and both $\mathcal{L}_{BCE}$ and $\mathcal{L}_{TFS}$ are consistent over time, $\mathcal{L}_{TFS}$ is able to anticipate convergence as previously shown in Fig. 2.

Finally, for ZENOTRAVEL 80%, the $\text{ACC}_{\text{FACTS}}$ of $\mathcal{L}_{TFS}$ is always above that of $\mathcal{L}_{BCE}$ even if the convergence starts at the same time for both of them and terminating some epochs earlier due to early stopping. Similarly to the previous example, the BCE shows inconsistent behaviours at the end of the training phase with a drop in performance of more than 5 points after the 15 epoch, with some fluctuations afterwards. This difference in the dynamic of the training reflects the results of ACC_{GR} obtained in this condition shown in Table 2 where $\mathcal{L}_{TFS}$ overcome the baseline with a consistent margin, increasing considering small percentages of observation.

5 Conclusion and Future Works

In this paper, we showed the importance of proper regularisation of the loss function in the case of a learning model that aims to solve a GR task. In particular, we analysed the performance of GRNet using the BCE loss ($\mathcal{L}_{BCE}$) and the proposed regularised loss ($\mathcal{L}_{TFS}$) based on the sum of the true facts on the label y both in terms of ACC_{GR} and the convergence time of $\text{ACC}_{\text{FACTS}}$ for its EC. We considered different training sizes, different numbers of observable actions in the plans, and different planning domains. We showed that $\mathcal{L}_{TFS}$ the best ACC_{GR} the vast majority of the time with respect to both train and observation percentages in all domains. Moreover, evaluating the cost of training the EC, we showed that a proper regularisation is able to anticipate the convergence time and obtain a higher $\text{ACC}_{\text{FACTS}}$, saving computational and environmental resources. In



(a) Blocksworld 90%

(b) Depots 70%

(c) Zenotravel 80%

Fig. 3: The comparison of the validation accuracy (y-axis) on facts prediction over time between the $\mathcal{L}_{BCE}$ and the $\mathcal{L}_{TFS}$ over the epochs (x-axis) for combinations showing unexpected behaviours.

some particular conditions, when the anticipation effect was absent, we noticed a smoother training dynamic and a more consistent final performance.

In the future, our first objective is to thoroughly evaluate different regularisation approaches and other loss functions, such as the Binary-Focal Cross-Entropy loss ($\mathcal{L}_{focal}$), as $\mathcal{L}_{BCE}$. Second, we would reduce the training size even further and study the model behaviour in those extreme conditions, combining the reduction of the training size with a proper sample selection based on a smarter stratification mechanism. Third, we are interested in evaluating the GRNet model in a planning domain that is particularly difficult for planners, for example FLOORTILE, in the GR task. We want to study a possible replacement for ISC that is learnable as EC to enable end-to-end training and improve performance.

Acknowledgement

This work was supported by SERICS (PE00000014) and FAIR (B53C22003980006) projects under the NRRP MUR programme funded by the European Union - NextGenerationEU, and the MUR PRIN project RIPER (No. 20203FFYLK).

References

1. Batrinca, L., Mana, N., Lepri, B., Sebe, N., Pianesi, F.: Multimodal Personality Recognition in Collaborative Goal-Oriented Tasks. IEEE Transactions on Multimedia **18**(4), 659–673 (2016). https://doi.org/10.1109/TMM.2016.2522763
2. Borrajo, D., Gopalakrishnan, S., Potluru, V.K.: Goal recognition via model-based and model-free techniques. Proceedings of the 1st Workshop on Planning for Financial Services at the Thirtieth International Conference on Automated Planning and Scheduling, FinPlan 2020 (2020)
3. Chiari, M., Gerevini, A.E., Percassi, F., Putelli, L., Serina, I., Olivato, M.: Goal Recognition as a Deep Learning Task: The GRNet Approach. In: Koenig, S., Stern, R., Vallati, M. (eds.) Proceedings of the Thirty-Third International Conference on Automated Planning and Scheduling, July 8-13, 2023, Prague, Czech Republic. pp. 560–568. AAAI Press (2023). https://doi.org/10.1609/ICAPS.V33I1.27237, https://doi.org/10.1609/icaps.v33i1.27237
4. Geffner, H.: Model-free, Model-based, and General Intelligence. In: Lang, J. (ed.) Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI 2018, July 13-19, 2018, Stockholm, Sweden. pp. 10–17. ijcai.org (2018). https://doi.org/10.24963/IJCAI.2018/2, https://doi.org/10.24963/ijcai.2018/2
5. Gerevini, A., Saetti, A., Serina, I.: Planning Through Stochastic Local Search and Temporal Action Graphs in LPG. J. Artif. Intell. Res. **20**, 239–290 (2003)
6. Gerevini, A., Serina, I.: Planning as Propositional CSP: From Walksat to Local Search Techniques for Action Graphs. Constraints An Int. J. **8**(4), 389–413 (2003)
7. Gerevini, A.E.: An Introduction to the Planning Domain Definition Language (PDDL): book review. Artif. Intell. **280**, 103221 (2020). https://doi.org/10.1016/J.ARTINT.2019.103221, https://doi.org/10.1016/j.artint.2019.103221

8. Harman, H., Simoens, P.: Action Graphs for Performing Goal Recognition Design on Human-Inhabited Environments. Sensors **19**(12), 2741 (2019). https://doi.org/10.3390/s19122741, https://doi.org/10.3390/s19122741
9. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV). pp. 2980–2988 (Oct 2017)
10. Long, D., Fox, M.: The 3rd International Planning Competition: Results and Analysis. J. Artif. Intell. Res. **20** (2003)
11. McDermott, D., Ghallab, M., Howe, A.E., Knoblock, C.A., Ram, A., Veloso, M.M., Weld, D.S., Wilkins, D.E.: PDDL-the planning domain definition language. In: Proceedings of the International Conference on Artificial Intelligence in Planning Systems (AIPS) (1998), https://api.semanticscholar.org/CorpusID:59656859
12. McDermott, D.V.: The 1998 AI Planning Systems Competition. AI Mag. **21**(2), 35–55 (2000)
13. Min, W., Mott, B.W., Rowe, J.P., Liu, B., Lester, J.C.: Player Goal Recognition in Open-World Digital Games with Long Short-Term Memory Networks. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016. pp. 2590–2596. IJCAI/AAAI Press (2016)
14. Mirsky, R., Shalom, Y., Majadly, A., Gal, K., Puzis, R., Felner, A.: New Goal Recognition Algorithms Using Attack Graphs. In: Cyber Security Cryptography and Machine Learning - Third International Symposium, CSCML 2019, Proceedings. Lecture Notes in Computer Science, vol. 11527, pp. 260–278. Springer (2019). https://doi.org/10.1007/978-3-030-20951-3_23, https://doi.org/10.1007/978-3-030-20951-3_23
15. Pereira, R.F., Oren, N., Meneguzzi, F.: Landmark-based approaches for goal recognition as planning. Artif. Intell. **279** (2020). https://doi.org/10.1016/j.artint.2019.103217, https://doi.org/10.1016/j.artint.2019.103217
16. Richter, S., Westphal, M.: The LAMA Planner: Guiding Cost-Based Anytime Planning with Landmarks. J. Artif. Intell. Res. **39**, 127–177 (2010)
17. Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z.: Rethinking the inception architecture for computer vision. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 2818–2826 (June 2016)
18. Van-Horenbeke, F.A., Peer, A.: Activity, Plan, and Goal Recognition: A Review. Frontiers Robotics AI **8**, 643010 (2021). https://doi.org/10.3389/frobt.2021.643010, https://doi.org/10.3389/frobt.2021.643010