GoalNet: Interleaving Neural Goal Predicate Inference with Classical Planning for Generalization in Robot Instruction Following

Jigyasa Gupta[1, 2*], **Shreya Sharma**[1 *], **Shreshth Tuli**[1, 3], **Rohan Paul**[1], **Mausam**[1]

[1]Indian Institute of Technology Delhi, India
[2]Samsung R&D Institute India, Delhi
[3]Happening Technology, UK

Abstract

Our goal is to enable a robot to learn how to sequence its actions to perform high-level tasks specified as natural language instructions, given successful demonstrations from a human partner. Our novel neuro-symbolic solution GoalNet builds an iterative two-step approach that interleaves (i) inferring next subgoal predicate implied by the language instruction, for a given world state, and (ii) synthesizing a feasible subgoal-reaching plan from that state. The agent executes the plan, and the two steps are repeated. GoalNet combines (i) *learning*, where dense representations are acquired for language instruction and the world state via a neural network prediction model, enabling generalization to novel settings and (ii) *planning*, where the *cause-effect* modeling by a classical planner eschews irrelevant predicates, facilitating multi-stage decision making in large domains. GoalNet obtains 78% improvement in the goal reaching rate in comparison to several state-of-the-art approaches on benchmark data with multi-stage instructions. Further, GoalNet can generalize to novel instructions for scenes with unseen objects. Source code available at `https://github.com/reail-iitd/goalnet`.

1 Introduction

Robots may be placed in scenarios where they learn from humans on how to perform tasks. Hence, they must possess the ability to understand high-level task specifications, communicated in natural language (NL) by humans, and successfully generate goal-reaching plans, possibly in novel environments. A popular paradigm for training a robot is *imitation learning*, wherein models are trained directly from such demonstrations (Tuli et al. 2022; Mei, Bansal, and Walter 2016; Suhr and Artzi 2018). However, prior work highlights that such methods tend to lack the ability to scale with environment complexity (Misra et al. 2018).

On the other hand, symbolic planners allow us to scale well, while circumventing the challenges posed by side-effect or irrelevant predicates (Misra et al. 2018). However, planners need symbolic goal predicates and cannot directly take NL instructions as input. We can decompose the problem into first predicting the final goal predicates and using them to generate actions via a planner. This will entail training a model to output intended goal predicates for a given NL instruction. Preliminary experiments show that this approach faces limitations. First, the hypothesis space of conjunctive goals is exponential (powerset of all predicates). Second, irrelevant predicates or side-effects in demonstrated plans can lead to inaccurate predictions. Finally, there may be stochasticity or erroneous plan executions that may not exactly lead to the intended goal state.

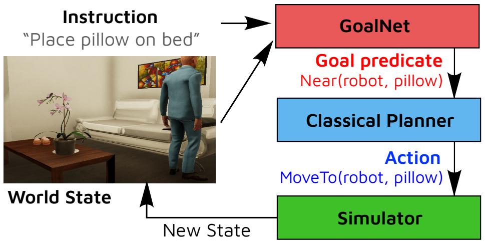


Figure 1: GoalNet infers goal predicates (red), which are forwarded to a classical planner (blue) and actions are executed in a simulator (green) to reach a goal state.

In response, we propose an interleaved strategy (see Fig. 1). Here, a neuro-symbolic model, namely GoalNet, predicts a *subgoal predicate* (one each for positive and negative) from the current world state, which is forwarded to a classical planner that outputs a sequence of actions. The plan is executed to obtain the next world state. This is repeated iteratively till the final goal state is reached. GoalNet's architecture has several benefits: prediction of *single* subgoals reduces hypothesis space, prediction of *intermediate* goals makes it more robust to side-effects, and interleaving allows the model to recover from execution errors and earlier mistakes in subgoal prediction. Furthermore, to handle scenes with unseen objects, we extend GoalNet to be agnostic to the object set in the world state – we call this GoalNet*.

We experiment on a benchmark dataset by Misra et al. (2015) collected from a mobile manipulator in a kitchen or living-room environment possessing a rich set multi-stage interactions. Our evaluation shows that the interleaved architecture of GoalNet outperforms state-of-the-art methods that use (i) a non-neural subgoal predictor with a similar interleaving, (ii) a conjunctive goal predictor without interleaving, and (iii) a pure imitation learner. Our results demon-

[*] These authors contributed equally.

strate a significant improvement in the *task completion rate* of at least 77.8% compared to our baselines. Further, GOALNET and GOALNET* outperform baselines by giving up to 1.36 times higher task completion rate in settings with *unseen* objects and *novel* paraphrased instructions.

2 Related Work

Robot instruction following (also called language grounding) considers the task of synthesizing a context-dependent robot plan from varied natural language instructions from a human partner. These approaches learn an association between language constructs and an action specification for the robot to execute. For example, Tellex et al. (2011) maps verbs in instructions to robot trajectories where as Paul et al. (2018) infer motion constraints from manipulation instructions. These approaches require explicit supervision for linguistic phrases in terms of robot actions; often requiring expert annotators. Note that, unlike semantic parsing, this work does not consider the presence of instruction annotations with precise logical forms, which is the case in standard semantic parsers such as SEMPRE (Berant et al. 2013).

Alternative approaches leverage human demonstrations to learn a model for translating language to plans. Boularias et al. (2015) use Inverse RL to infer a latent reward function for an instruction from human labelled trajectories of navigation. Liao et al. (2019); Tuli et al. (2021) propose an object-centric model of the world to inferring plans involving rich agent-object interactions. Branavan et al. (2012); Misra et al. (2016) learn symbolic plan strategies from textual plan descriptions, incorporating a classical planner to assess goal reach-ability in order to score candidate plans. (Silver et al. 2022) learns action operators grounded in the continuous action space allowing reasoning over the metric space. Such approaches have been successful in grounding instructions amenable to direct translation to short-horizon robot plans. In contrast, this work assumes access to low-level skills and focuses on inferring multi-step plans with rich inter-object interactions; particularly in novel scenes.

Within classical planning, works such as (Sebastia, Onaindia, and Marzal 2006) discover sub-goals for a complex planning problem by analyzing negative/positive interactions. Further, Lipovetzky and Geffner (2012) introduce a notion of *planning width* to characterize the complexity of planning problems. In this work, we adopt a *learning-to-plan* paradigm that leverages demonstrations to learn which sub-goals are important for goal reach-ability. Further, the iterative planning and execution approach allows goal-attainment even in problems with higher width.

Others factorize the problem of interpreting plans from language instructions as that of inferring a goal specification (Meneguzzi and Pereira 2021) and delegate the task of determining action sequences to a planner. Lesh and Etzioni (1995) introduced a goal recognizer that observes human actions to prune inconsistent actions or goals from an input graph state. Dragan et al. (2015) present a robot motion planner that generate legible plans using human-robot collaboration by uncovering latent goals from a task demonstration. She and Chai (2016) use linguistic and environment features to induce a hypothesis set of goal predicates that can be handed to a planner. hese approaches have demonstrated the ability to infer goals in small-sized domains (grid-world like domains) and their scalability to large environments with realistic interactions is limited. The reliance on hand-crafted features limits generalization to novel scenes and phrases.

3 Problem Formulation

Robot and Environment Models. We are interested in robots that have the ability to navigate domains and manipulate multiple objects in natural confined environments such as a kitchen or living room. We consider objects as symbolic entities that consist of (i) *identifying tokens* such as "apple", "stove" and "pillow", (ii) *object states* such as $\mathrm{Open/Closed}$, $\mathrm{On/Off}$, and (iii) *properties* such as $\mathrm{IsSurface}$, $\mathrm{IsContainer}$, $\mathrm{IsGraspable}$, etc. Relations between object pairs are also present, such as OnTop, Near, Inside and $\mathrm{ConnectedTo}$. Let s denote the world state – it has symbolic objects $\mathcal{O}(s)$ including their identifiers, states and properties. We denote the set of spatial-relation predicates and object state predicates by $\mathcal{S}$. Let $\mathcal{R}(s)$ denote the set of object relations in s. A *state variable* $r \in \mathcal{R}(s)$ is denoted by $R(o^1, o^2)$ that represents a relation predicate of type $R \in \mathcal{S}$ between objects $o^1 \in \mathcal{O}(s)$ and $o^2 \in \mathcal{O}(s)$, or as $R(o^1)$ in case of r being an object state, for example $\mathrm{OnTop}(\mathrm{pillow}_0, \mathrm{shelf}_0)$, $\mathrm{ConnectedTo}(\mathrm{fork}_0, \mathrm{robot})$ and $\mathrm{stateIsOpen}(\mathrm{tap}_0)$. Let s_0 denote the initial state.

Action and Transition Models. We denote the set of all possible symbolic actions by A. An action $a \in A$ is represented in its symbolic form as $I(o^1)$ or $I(o^1, o^2)$ where the *interaction* (action predicate) $I \in \mathcal{I}$ affects the state or the relation between o^1 and o^2. Examples of interactions in $\mathcal{I}$ include Grasp, MoveTo, $\mathrm{stateOn}$, and $\mathrm{PlaceOn}$. Interaction effects are considered to be deterministic. For instance, a $\mathrm{stateOn}$ action applies to an $\mathrm{IsTurnable}$ object, such as a tap, to swap its state between On and Off. Similarly, a $\mathrm{PlaceOn}(\mathrm{pillow}_0, \mathrm{couch}_0)$ action establishes the $\mathrm{OnTop}(\mathrm{pillow}_0, \mathrm{couch}_0)$ relation. Interactions are also associated with pre-conditions in the form of relations or properties. For instance, $\mathrm{PlaceOn}$ is allowed only when the object has a *grasped* relation with the agent. For more details see Appendix

Instructions and Goals. Given an initial state of the environment s_0 and transition model $\mathcal{T}$, the robot needs to perform a task expressed as a natural language *instruction* l – it is encoded in the form of a sequence $\langle l_0, \ldots, l_z \rangle$ where each element is a word token. Each instruction l is assumed to represent a goal, expressed as conjunctions over state variables (relations/object states). Based on which variables should be present or absent in the final goal state, we split them into two sets, Δ_l^+ (positive) and Δ_l^- (negative). For a successful execution, the final goal state must have all state variables from Δ_l^+, and none from Δ_l^-. For example, for the input state s with a pillow on the shelf and another inside a cupboard, the declarative goal, $l =$ "put the shelf pillow on the couch" can be expressed as sets of relations $\Delta_l^+ = \{\mathrm{OnTop}(\mathrm{pillow}_0, \mathrm{couch}_0)\}$ and $\Delta_l^- = \{\mathrm{OnTop}(\mathrm{pillow}_0, \mathrm{shelf}_0)\}$. To successfully execute an input instruction l from an initial state s_0, the agent must

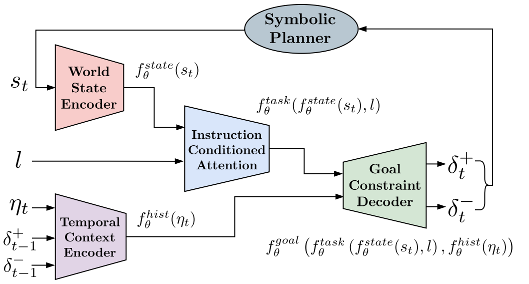

Symbolic Planner
s_t
World State Encoder
$f_\theta^{state}(s_t)$
$f_\theta^{task}(f_\theta^{state}(s_t), l)$
Instruction Conditioned Attention
l
Goal Constraint Decoder
δ_t^+
δ_t^-
η_t
δ_{t-1}^+
δ_{t-1}^-
Temporal Context Encoder
$f_\theta^{hist}(\eta_t)$
$f_\theta^{goal}\left(f_\theta^{task}\left(f_\theta^{state}(s_t), l\right), f_\theta^{hist}(\eta_t)\right)$

Figure 2: Using a symbolic planner and GoalNet in tandem.

synthesize a *plan*, as a sequence of actions $\langle a_0, \ldots, a_T \rangle$, such that the final state $s_T = \mathcal{T}(\ldots \mathcal{T}(s_0, a_0) \ldots, a_T)$ is a goal state. Let $\mathcal{G}(s, l)$ denote the *goal check* function that determines if the intended goal l is achieved by a state s.

Training Data and Task Definition. We wish to learn models through human demonstrations denoted as D_{Train} of N goal-reaching *trajectories* (tr), where the i^{th} datum consists of the initial state s_0^i, the instruction l^i and a trajectory $tr^i = \langle s_0^i, a_0^i, \ldots, s_{T_i-1}^i, a_{T_i-1}^i, s_{T_i}^i \rangle$ that achieves the goal intended by l^i. The task is to train a model, which given an instruction l and the current state, outputs the next action to execute towards achieving the goal intended in the instruction. To facilitate the use of symbolic planners, we also provide the PDDL description Λ of the domain .

4 Technical Approach

We design a neuro-symbolic approach where language grounding is done via a neural module, and planning is done symbolically. A natural approach is to create a pipeline, where a neural function $f_\theta(\cdot)$ outputs the final goal as a conjunction of positive and negative state variables, i.e., $f_\theta(s_0, l) = \langle \Delta_l^+, \Delta_l^- \rangle$. These state variables are then provided as a goal to a classical planner $\mathcal{P}(\cdot)$, along with the domain Λ, which outputs a plan $\vec{a}$. This plan is executed to reach final state s_T and success of the model is determined based on whether $\mathcal{G}(s_T, l) = 1$.

While this pipeline architecture is intuitive, our initial experiments showed it not to be effective. The prediction of variable-sized sets Δ_l made the learning problem hard, and a single pipeline added robustness issues in execution, overall leading to low task completion rates. In response, we design a novel architecture GoalNet– the key idea is to *interleave* next subgoal prediction, planning and execution. In particular, subgoal predictor f_θ outputs a single positive and negative subgoal variable (instead of two variable-sized conjunctive sets). I.e., $f_\theta(s_t, l) = \langle \delta_l^+, \delta_l^- \rangle$. This is provided to planner $\mathcal{P}$ as the next goal, and the resulting plan $\vec{a_t}$ is executed to reach next state s_{t+1}. This state is given to f_θ for the next prediction, and the whole process continues iteratively.

Before we describe the design of the subgoal predictor, we recognize that we ought to postprocess D_{Train} to make it suitable as supervision for its input-output characteristics. For this, given a state s_t^i in training trajectory i, we generate its target subgoals using a single-step difference over the sets of variables between the two consecutive states:

$$\langle \hat{\delta}_t^+, \hat{\delta}_t^- \rangle = \langle s_{t+1}^i \setminus s_t^i, \quad s_t^i \setminus s_{t+1}^i \rangle \quad \forall t < T_i,$$
$$\langle \hat{\delta}_{T_i}^+, \hat{\delta}_{T_i}^- \rangle = \langle \emptyset, \emptyset \rangle.$$

The data $(s_t^i, l^i, \langle \hat{\delta}_t^+, \hat{\delta}_t^- \rangle)$ is used as supervision to train f_θ. If $|\hat{\delta}_t^+| + |\hat{\delta}_t^-| > 1$, we only use one predicate for training as planner compensates for side effects predicates.

Subgoal Predictor

While, in principle, s_t is the world state, and has all the information for planning and subgoal prediction, previous research has found that explicitly maintaining some history (in their case, action history) aids model learning (Tuli et al. 2021). For our subgoal predictor at time t, as additional input, we provide an explicit *subgoal history*: $\eta_t = \{(\delta_0^+, \delta_0^-), \ldots, (\delta_{t-1}^+, \delta_{t-1}^-)\}$. I.e., $f_\theta(s_t, l, \eta_t) = \langle \delta_t^+, \delta_t^- \rangle$.

At a high level (see Fig. 2), the subgoal predictor takes in the world state s_t in the form of an object-centric graph. The object encodings in the state are generated by $f_\theta^{state}(\cdot)$. The subgoal history is encoded as $f_\theta^{hist}(\cdot)$. Next, it attends over the object encodings conditioned on the input task instruction via $f_\theta^{task}(\cdot)$. Finally, the positive and negative subgoals are decoded autoregressively by $f_\theta^{goal}(\cdot)$. Overall:

$$\langle \delta_t^+, \delta_t^- \rangle = f_\theta(s_t, l, \eta_t) = f_\theta^{goal}\left(f_\theta^{task}\left(f_\theta^{state}(s_t), l\right), f_\theta^{hist}(\eta_t)\right).$$

We now describe each components of the subgoal predictor.

World State Encoder. The current world state s_t is encoded as an object-centric graph $G_t = (\mathcal{O}(s_t), \mathcal{R}(s_t))$ where each node represents an object $o \in \mathcal{O}(s_t)$. Each relation $r \in \mathcal{R}(s_t)$ of the form $R(o^1, o^2)$ is encoded as a directed edge from o^1 to o^2 with predicate as R. To represent the object states of an object o, GoalNet generates a binary feature vector $q_o = \{0, 1\}^u$ that represents the discrete object states for each of u state predicates that include $\text{Open}/\text{Closed}$, On/Off, etc. Similarly, it generates a binary feature vector $p_o = \{0, 1\}^v$ that represents the presence of various object properties (1 if present and 0 otherwise) for each of v properties such as isSurface and isContainer.

It also incorporates a function $\mathcal{C}(\cdot)$ that generates a dense vector representation for the input token of an object. For an object o, we represent this by $e_o = \mathcal{C}(o) \in \mathbb{R}^w$ as a w-dimensional embedding. We assume that $\mathcal{C}(o)$ of semantically similar objects (such as "apple" and "orange") appear close, whereas semantically different objects appear far apart (such as "fork" and "table") (Mikolov et al. 2018). Unless stated otherwise, we utilize ConceptNet embeddings (Speer, Chin, and Havasi 2019) to facilitate generalization to unseen objects (Tuli et al. 2021).

GoalNet concatenates the embeddings q_o, p_o and e_o for each object o to form the feature vector that initializes each node of the G_t. The relations of each object o in the edges $\mathcal{R}(s_t)$ are represented as an adjacency vector r_o. This relational information is first encoded using a d-layer Fully Connected Network (FCN) with Parameterized ReLU (PReLU) (He et al. 2015) activation to generate a relational embedding for each object o as r_o^d. Next, it fuses the semantic and

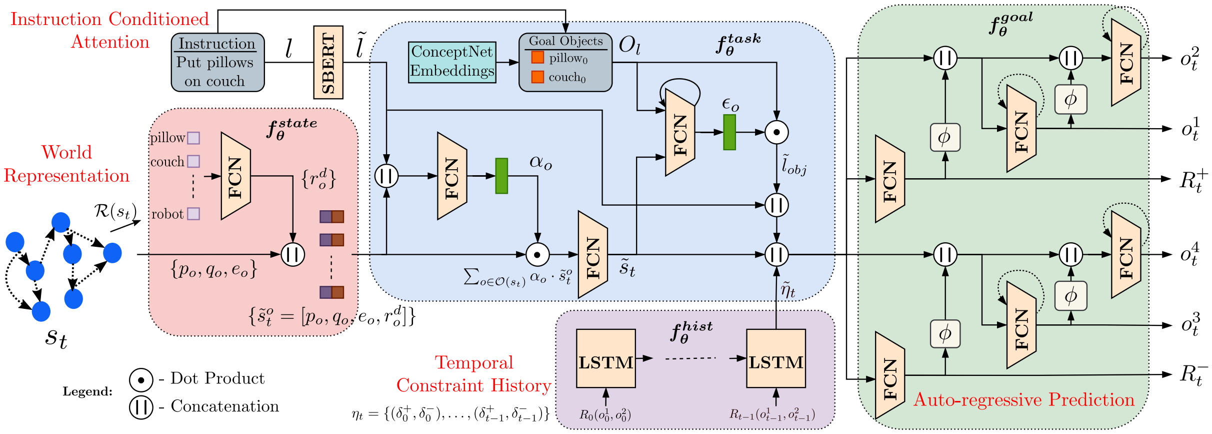


Figure 3: GoalNet subgoal predictor encodes the world state and NL instruction to attend over objects, and decodes the next subgoals to send to a symbolic planner. Dashed arrows represent factored score prediction in GoalNet* for generalizing to unseen objects.

relational embeddings to generate an embedding of each object o as $[q_o, p_o, e_o, r_o^d]$. Late fusion of the relational information enables the downstream predictors to exploit the semantic and relational information independently, improving inference performance as demonstrated in prior work (Tuli et al. 2021). Thus, the output of the state-encoder becomes

$$f_\theta^{state}(s_t) = \{\tilde{s}_t^o = [p_o, q_o, e_o, r_o^d] | \, \forall o \in \mathcal{O}(s_t)\}. \quad (1)$$

Temporal Context Encoder. As discussed above, the model is explicitly informed of the local context, which may suggest the objects to manipulate in the future, since often sequential actions are temporally correlated. For instance, in the task of placing pillows on the couch, the agent first moves towards a pillow, grasps it and then places it on the couch. This entails that the sequence of subgoals would initially have $\text{Near}(\text{robot}, \text{pillow}_0)$, followed by $\text{ConnectedTo}(\text{pillow}_0, \text{robot})$ and then $\text{OnTop}(\text{pillow}_0, \text{couch}_0)$. This example demonstrates the high correlation between the interactions and manipulated objects in two adjacent time steps.

Formally, GoalNet encodes the temporal history of subgoals η_t using an LSTM (Hochreiter and Schmidhuber 1997). For each subgoal $r = R_{t-1}(o_{t-1}^1, o_{t-1}^2) \in \mathcal{R}(s_{t-1})$ predicted in the previous time step $t-1$ in $\delta_{t-1}^+ \cup \delta_{t-1}^-$, it defines an encoding $\tilde{r} = [\vec{R}_{t-1}, \mathcal{C}(o_{t-1}^1), \mathcal{C}(o_{t-1}^2)]$, where $\vec{R}_{t-1}$ is a one-hot encoding for the predicate $R_{t-1} \in \mathcal{S}$ of the form $\{0,1\}^{|\mathcal{S}|}$. $\mathcal{C}(o_{t-1}^1)$ and $\mathcal{C}(o_{t-1}^2)$ are the dense embeddings of the tokens of objects or state of o_{t-1}^1. At each time step t, the subgoal history η_t is encoded as $\tilde{\eta}_t$ where

$$f_\theta^{hist} = \tilde{\eta}_t = \text{LSTM}([\vec{R}_{t-1}, \mathcal{C}(o_{t-1}^1), \mathcal{C}(o_{t-1}^2)], \tilde{\eta}_{t-1}).$$

Instruction Conditioned Attention. This component aligns the information of the input instruction with the scene to learn task-relevant context by allocating appropriate attention weights to objects. This relieves the downstream predictors from dealing with all objects, which can focus only on the ones related to the task, allowing the model to scale with the number of objects in the input world state.

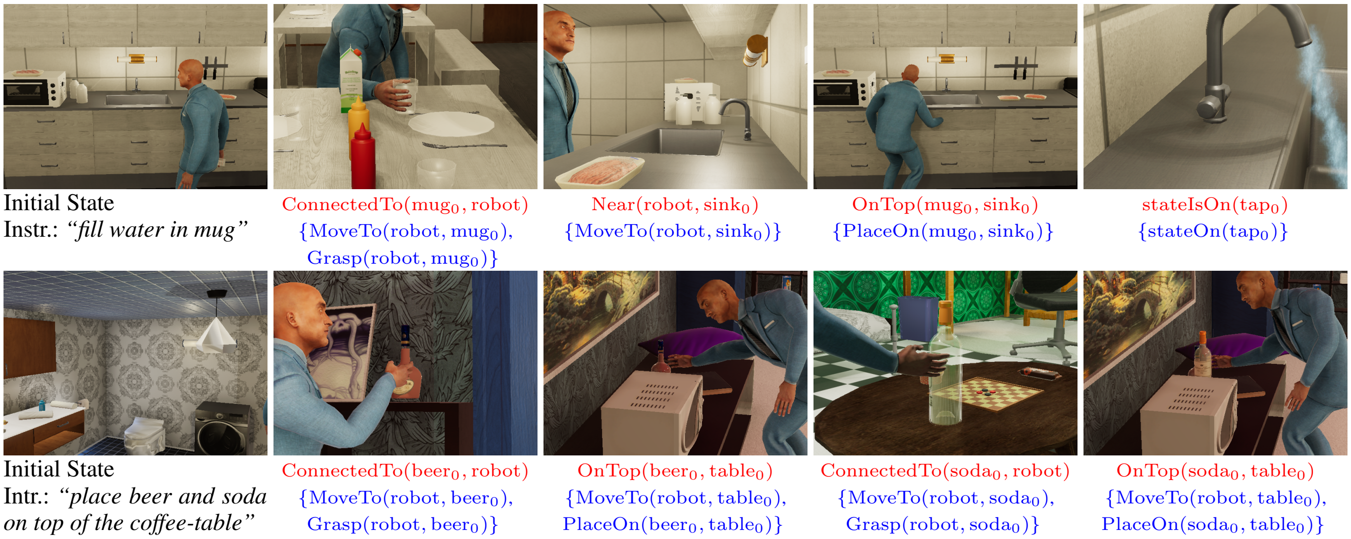
Specifically, instruction l is encoded using a sentence embedding model $\mathcal{B}(\cdot)$ to generate the encoding $\tilde{l} = \mathcal{B}(l)$. GoalNet uses SentenceBERT as its $\mathcal{B}(\cdot)$ function – a pre-trained model, it is a modification of the pre-trained BERT model (Devlin et al. 2019) to derive semantically meaningful sentence embeddings (Reimers et al. 2019).

GoalNet uses the language instruction encoding $\tilde{l}$ as query to attend over the world objects using a Bahadanau attention mechanism (Bahdanau, Cho, and Bengio 2014) and generates a final state embedding $\tilde{s}_t$ as an attended summary over object embeddings $\tilde{s}_t^o$. Using a tokenizer (Bird, Klein, and Loper 2009), GoalNet also extracts the set of objects O_l in instruction l. It then generates the encoding $\tilde{l}_{obj}$ of these objects by using self-attention with $\tilde{s}_t$ as the query. Overall, $f_\theta^{task}(\tilde{s}_t^o, l) = [\tilde{s}_t, \tilde{l}_{obj}, \tilde{l}]$. The equations for computing $\tilde{l}_{obj}$ are as follows (those for $\tilde{s}_t$ are similar).

$$\alpha_o = \text{softmax}(W_l[\mathcal{C}(o), \tilde{s}_t] + b_l), \quad \tilde{l}_{obj} = \sum_{o \in O_l} \alpha_o \cdot \mathcal{C}(o).$$

Subgoal decoder. GoalNet takes the instruction attended world state $\tilde{s}_t$, encoding of the subgoal history $\tilde{\eta}_t$, instruction objects encoding $\tilde{l}_{obj}$ and the sentence encoding $\tilde{l}$ to predict a pair of positive and negative subgoals $R_t^+(o_t^1, o_t^2)$ and $R_t^-(o_t^3, o_t^4)$. To predict each of the three components, i.e., relation and the two objects, it computes the likelihood score for each relation predicate from $\mathcal{S}$ and objects in $\mathcal{O}(s_t)$. It then selects the relation or object with the highest likelihood scores, decoded in an auto-regressive fashion. For instance, to predict $R_t^+(o_t^1, o_t^2)$, the likelihood scores of R_t^+ are forwarded to predict o_t^1, and likelihood scores of both R_t^+ and o_t^1 to predict o_t^2. Instead of using an argmax of the likelihood vector, it forwards the Gumbel-Softmax of the vector (Jang, Gu, and Poole 2017) (denoted by $\phi(\cdot)$). This is a variation of softmax function that allows us to generate a one-hot vector while also allowing gradients to backpropagate (as argmax is not differentiable). It generates the scores using a fully connected network with PreLU activation.

$$\begin{aligned} \tilde{R}_t^+ &= \text{softmax}(\text{FCN}([\tilde{s}_t, \tilde{\eta}_t, \tilde{l}_{obj}, \tilde{l}])), \\ \tilde{o}_t^1 &= \text{softmax}(\text{FCN}([\tilde{s}_t, \tilde{\eta}_t, \tilde{l}_{obj}, \tilde{l}, \phi(\tilde{R}_t^+)])), \\ \tilde{o}_t^2 &= \text{softmax}(\text{FCN}([\tilde{s}_t, \tilde{\eta}_t, \tilde{l}_{obj}, \tilde{l}, \phi(\tilde{R}_t^+), \phi(\tilde{o}_t^1)])). \end{aligned} \quad (2)$$



Initial State
Instr.: *"fill water in mug"*

$\mathrm{ConnectedTo}(\mathrm{mug}_0, \mathrm{robot})$
$\{\mathrm{MoveTo}(\mathrm{robot}, \mathrm{mug}_0), \mathrm{Grasp}(\mathrm{robot}, \mathrm{mug}_0)\}$

$\mathrm{Near}(\mathrm{robot}, \mathrm{sink}_0)$
$\{\mathrm{MoveTo}(\mathrm{robot}, \mathrm{sink}_0)\}$

$\mathrm{OnTop}(\mathrm{mug}_0, \mathrm{sink}_0)$
$\{\mathrm{PlaceOn}(\mathrm{mug}_0, \mathrm{sink}_0)\}$

$\mathrm{stateIsOn}(\mathrm{tap}_0)$
$\{\mathrm{stateOn}(\mathrm{tap}_0)\}$

Initial State
Intr.: *"place beer and soda on top of the coffee-table"*

$\mathrm{ConnectedTo}(\mathrm{beer}_0, \mathrm{robot})$
$\{\mathrm{MoveTo}(\mathrm{robot}, \mathrm{beer}_0), \mathrm{Grasp}(\mathrm{robot}, \mathrm{beer}_0)\}$

$\mathrm{OnTop}(\mathrm{beer}_0, \mathrm{table}_0)$
$\{\mathrm{MoveTo}(\mathrm{robot}, \mathrm{table}_0), \mathrm{PlaceOn}(\mathrm{beer}_0, \mathrm{table}_0)\}$

$\mathrm{ConnectedTo}(\mathrm{soda}_0, \mathrm{robot})$
$\{\mathrm{MoveTo}(\mathrm{robot}, \mathrm{soda}_0), \mathrm{Grasp}(\mathrm{robot}, \mathrm{soda}_0)\}$

$\mathrm{OnTop}(\mathrm{soda}_0, \mathrm{table}_0)$
$\{\mathrm{MoveTo}(\mathrm{robot}, \mathrm{table}_0), \mathrm{PlaceOn}(\mathrm{soda}_0, \mathrm{table}_0)\}$

Figure 4: Sample plan in kitchen (top) and living-room (bottom) domains. Visualizations developed using the VirtualHome Simulator (Puig et al. 2018) and a human-like agent with functionality akin to a single-arm manipulator. Predicted goal predicates shown in red. Executed plan at each time step shown in blue. *soda* is unseen at training time and GoalNet* reaches a goal state. Only positive predicates shown.

The predicted relation predicate R_t^+ is $\mathrm{argmax}_{R \in \mathcal{S}} \tilde{R}_t^+$, the first object o_t^1 is $\mathrm{argmax}_{o \in \mathcal{O}(s_t)} \Omega(\tilde{o}_t^1, R_t^+)$ and o_t^2 as $\mathrm{argmax}_{o \in \mathcal{O}(s_t)} \Omega(\tilde{o}_t^2, R_t^+, o_t^1)$. A similar mechanism is followed to predict the negative subgoal $R_t^-(o_t^3, o_t^4)$. Here, Ω denotes PDDL-based grammar mask, which forces the likelihood scores of infeasible objects to 0. For instance, the OnTop relation only accepts the objects as the second argument that have the isSurface property. We also mask out the likelihood scores of $\tilde{o}_t^2$ and $\tilde{o}_t^4$ if R_t^+ and R_t^- are predicates on the object states (which only need one argument). Thus, the predicted subgoals are represented as

$$\langle \delta_t^+, \delta_t^- \rangle = \langle R_t^+(o_t^1, o_t^2), R_t^-(o_t^3, o_t^4) \rangle = f_\theta^{goal}(\tilde{s}_t, \tilde{\eta}_t, \tilde{l}_{obj}, \tilde{l}). \quad (3)$$

Generalization to unseen objects. GoalNet assumes a fixed-size likelihood vector and only works on the fixed number (and type) of objects. We extend it to GoalNet*, which can predict relations grounded over any number of (unseen) scene objects. To generalize to objects unseen at training time, GoalNet* predicts likelihood scores of objects independently for each object o using their embeddings $\mathcal{C}(o)$. This is depicted using the dotted loop-arrows in Fig. 3. GoalNet has constant inference time with the size of the input world state ($O(1)$). Factored prediction in GoalNet* has an inference time that varies linearly ($O(|\mathcal{O}(s_t)| + |\mathcal{R}(s_t)|$).

Training and Inference. The predicted subgoals are passed on to a planner $\mathcal{P}$ as goals for it to generate a plan, which is then executed by the agent. All pre-conditions and effects are encoded in Λ, which uses a classical Planning Domain Definition Language (PDDL) representation. This allows the planner to utilize the symbolic domain information.

Objects: (*kitchen*) sink, stove, mug, microwave, fridge, icecream, kettle, coke, plate, boiledegg, salt, stovefire, sinkknob. (*living room*) loveseat, armchair, coffeetable, tv, pillow, bagofchips, bowl, garbagebag, shelf, book, coke, beer.	
Object States: Open/Closed, On/Off, HasWater/HasChocolate/IsEmpty, DoorOpen/DoorClosed.	
Object Properties: IsSurface, IsTurnable, IsGraspable, IsPressable, IsOpenable, IsSqueezeable, IsContainer.	
Actions: Grasp, Release, MoveTo, PlaceOn, PlaceIn, Press, Pour, Squeeze, stateOn, stateOff, stateOpen, stateClose.	
Relations: OnTop, Near, ConnectedTo.	

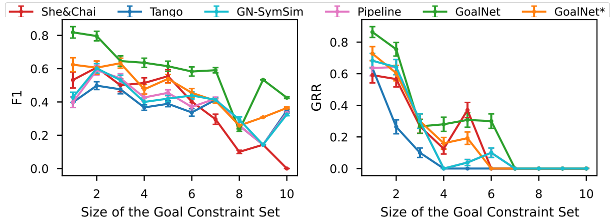
Table 1: Sample set of objects, states, properties, actions and relations (for complete description see Appendix

The whole subgoal predictor $f_\theta(\cdot)$ is trained using the sum of binary cross-entropy loss with ground-truth subgoals for the six predictors in Equation 3. We initially used teacher forcing, i.e., irrespective of predicted subgoal, always input the correct next state for training the model. However, at inference time, as we do not have ground-truth subgoals, we need to find the next state through simulation, causing the model to enter state regions it may not be exposed to, during training. To alleviate the effects of this exposure bias, we use teacher forcing with probability $1 - p$, and utilize a planner to generate next state with probability p (with the same target labels). The recurrent predicate prediction stops when $\delta_t^+ \cup \delta_t^- = \emptyset \vee t \geq T_i$.

This mitigates exposure bias, but running a planner can be slow. So, we use planners at two levels of fidelity: a *low-fidelity* symbolic simulator (that we call SymSim) and a *high-fidelity* planner by Rintanen (2012). In training, SymSim emulates the effects of executing actions corresponding to the subgoal $\langle \delta_t^+, \delta_t^- \rangle$, to generate the next state as $\hat{s}_{t+1}^i = s_t^i \cup \delta_t^+ \setminus \delta_t^-$. Here $\cup$ and $\setminus$ operations are performed on the relation set of the graph s_t^i to generate a new graph $\hat{s}_{t+1}^i$. SymSim reduces training time significantly. At test time, we use Rintanen for better performance.

5 Evaluation and Results

Dataset. For evaluation, we use a benchmark data set from Misra et al. (2015) consisting of NL instructions and as-



(a) F1 Score

(b) GRR Score

Figure 5: Performance of baseline and GOALNET model with the size of aggregate goal-predicate sets.

sociated trajectories collected through crowd-sourcing. The dataset has two domains: *kitchen* and *living room*, each containing 40 objects, with up to 4 instances of each object class. It consists of diverse instruction types ranging from short-horizon tasks such as *"go to the sink"* to tasks involving multiple extended interactions such as *"cook ramen in a pot of water"*. See Table 1 for a list of sample objects in the domain with the set of possible robot actions.

We extract intermediate states and target δ labels from this dataset, as described in previous section, yielding 1,117 datapoints, where we split 70%-15%-15% for our train, dev and test datasets as per She and Chai (2016). We also do data augmentation by perturbing the world states (and actions) via related object replacement based on ConceptNet embeddings. We perform this only for training and validation sets, wherein the replaced object is *unseen* in the original data. This allows us to increase the number of training datapoints by 25%, giving a total of 633 unique starting states in the dataset. At inference time, the prediction of goals (interleaved with the planner) continues till $\delta_t^+ \cup \delta_t^- = \emptyset \vee t \geq 30$.

Baselines. We evaluate using the following comparison baselines. (1) SHE&CHAI (She and Chai 2016) is a non-neural system that determines the hypothesis space for goals using *hand-crafted* rules and features. (2) TANGO (Tuli et al. 2021), an *imitation learning* model (which is adapted to take in NL instructions) that directly predict next actions. (3) GN-SYMSIM is a variation of GOALNET that uses the same interleaved architecture as GOALNET, but during test time, instead of RINTANEN, it uses SYMSIM planner. (4) PIPELINE is a variation of GOALNET where we infer the final goal at once (as a conjunctive set), and run RINTANEN once with this goal to generate the plan. (5) GAT is another variation of GOALNET, where we use a graph attention network (GAT) to encode the world state in lieu of an FCN. (6) GPT-3.5 TURBO follows ProgPrompt (Singh et al. 2023), wherein we prompt the GPT-3.5 Turbo model with natural language description of the world state knowledge of objects, relations and robot actions. Few shot context as 3 examples of language and goal predicates are also provided.

Evaluation Metrics. For i^{th} test datapoint, let the final state reached in gold trajectory be $s^i_{T_i}$, and similarly, in a model m's execution be $s^i_{T^m_i}$, where T^m_i is the number of execution steps for m. We compute aggregate state differences:

$$\hat{\Delta}^+_i = s^i_{T_i} \setminus s^i_0, \quad \hat{\Delta}^-_i = s^i_0 \setminus s^i_{T_i},$$
$$\Delta^+_i = \bar{s}^i_{T^m_i} \setminus s^i_0, \quad \Delta^-_i = s^i_0 \setminus \bar{s}^i_{T^m_i}.$$

Model	SJI	IED	F1	GRR
SHE&CHAI	0.448	0.450	0.512	0.370
TANGO	0.299	0.429	0.427	0.182
PIPELINE	0.364	0.291	0.483	0.340
GN-SYMSIM	0.366	0.294	0.487	0.344
GAT	0.176	0.327	0.223	0.149
GPT-3.5 TURBO	0.106	0.105	0.099	0.126
GOALNET	0.606	0.636	0.678	0.499
Model Ablations				
w/o Relational information	0.583	0.612	0.661	0.459
w/o Instance grounding	0.550	0.601	0.625	0.445
w/o δ^- prediction	0.425	0.448	0.534	0.450
w/o δ^+ prediction	0.191	0.223	0.252	0.134
w/o Temporal context encoding	0.282	0.349	0.344	0.188
w/o Grammar mask	0.578	0.612	0.655	0.459
GOALNET*	0.473	0.502	0.561	0.377
Training using RINTANEN	0.713	0.726	0.778	0.658

Table 2: A comparison of goal-prediction and goal-reaching performance for the baselines, GOALNET, and ablations. Two-sample t-tests comparing GoalNet/GoalNet* with baselines gives the p-values $\leq$0.0001, showing statistical significance.

Following She and Chai (2016), for a dataset of size N, we use the following evaluation metrics:

- *SJI (State Jaccard Index)* is overlap between the predicted (Δ^+_i, Δ^-_i) and ground-truth ($\hat{\Delta}^+_i$, $\hat{\Delta}^-_i$) predicates
$$\text{SJI} = \frac{1}{N}\sum_{i=1}^{N} \frac{\|\hat{\Delta}^+_i \cap \Delta^+_i\| + \|\hat{\Delta}^-_i \cap \Delta^-_i\|}{\|\hat{\Delta}^+_i \cup \Delta^+_i\| + \|\hat{\Delta}^-_i \cup \Delta^-_i\|}.$$
- *IED (Instruction Edit Distance):* is the similarity between the predicted $\{\bar{a}^i_0, \ldots, \bar{a}^i_{T^m_i-1}\}$ and ground-truth plan $\{a^i_0, \ldots a^i_{T_i-1}\}$, using the edit distance d^i
$$\text{IED} = \frac{1}{N}\sum_{i=1}^{N} 1 - \frac{d^i}{\max(T_i, T^m_i)}.$$
- *GRR (Goal Reaching Rate)* is a proxy to evaluate if the intended goal was reached:
$$\text{GRR} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{1}(\Delta^+_i \subseteq \hat{\Delta}^+_i \wedge \Delta^-_i \subseteq \hat{\Delta}^-_i).$$
- *F1* evaluates the average of the F1 scores between the positive and negative aggregate state differences.

Training Details. We use Adam optimizer (Kingma and Ba 2014) with a learning rate of 5×10^{-4}. We use early stopping based on loss over validation set. We also decay the learning rate by $1/5$ every 50 epochs. We use a constant teacher-forcing probability of $p = 0.2$ of using the planner-generated next state during training.

Baseline Comparisons. Table 2 presents the scores for baselines and GOALNET. For fair comparison, all models include the instance groundings for all objects in the input instruction as part of the goal-object set O_l (see Section 4). GOALNET outperforms all baselines in all metrics by large (statistically significant) margins, obtaining 13 pt GRR improvement over the next best model. We additionally experiment with training GOALNET directly with RINTANEN planner, instead of SYMSIM. Even though it is 13x slower to train , we get further boost in all metrics including

Model	Verb Replacement				Paraphrasing				Unseen Objects			
	SJI	IED	F1	GRR	SJI	IED	F1	GRR	SJI	IED	F1	GRR
She&Chai	0.134	0.137	0.146	0.138	0.124	0.127	0.136	0.128	0.138	0.163	0.164	0.103
TANGO	0.193	0.249	0.287	0.084	0.176	0.267	0.270	0.171	0.082	0.110	0.113	0.098
Pipeline	0.249	0.226	0.339	0.242	0.210	0.115	0.306	0.250	0.113	0.145	0.123	0.127
GN-SymSim	0.263	0.220	0.355	0.250	0.201	0.098	0.294	0.237	0.091	0.082	0.121	0.123
GPT-3.5 TURBO	0.078	0.087	0.083	0.087	0.037	0.038	0.053	0.053	0.070	0.078	0.066	0.086
GoalNet	**0.398**	**0.422**	**0.476**	0.236	**0.376**	**0.489**	**0.464**	**0.303**	0.138	0.153	0.153	0.130
GoalNet*	0.316	0.339	0.393	**0.253**	0.206	0.322	0.272	0.184	**0.276**	**0.286**	**0.332**	**0.196**

Table 3: GoalNet demonstrates the ability to generalize in case of *verb replacement* and *paraphrasing* relative to the baseline. GoalNet* has the ability to generalize to scenes with a-priori unseen objects. Best results in bold.

16 pt GRR gain. This highlights the importance of training with all action side-effects and conflicts, handled by a symbolic planner. Fig. 4 shows sample trajectories generated by GoalNet in kitchen and living room domain, respectively.

Performance with size of aggregate goal set. Figure 5 characterizes the variation in performance, as the size of ground truth state difference ($(\|\Delta_i^+\| + \|\Delta_i^-\|)$ increases. We observe that all performances degrade, suggesting that more work is needed in training models that are agnostic to resulting plan length. GoalNet performs better for most cases, and this performance gap widens for larger constraint sets, suggesting that the neural approach in GoalNet is able to effectively encode the temporal context, enabling it to outperform the baseline in multi-stage long-horizon tasks. The imitation learning baseline, TANGO, is unable to perform in long-horizon cases as it needs to additionally learn cause-effect rules. She&Chai is unable to generalize due to the hypothesis space being restricted to the objects in training data. Further, GN-SymSim performs poorly due to errors and noise in plan execution. Finally, one shot pipelined approach without interleaving performs poorly due to the low quality predictions and the model ignoring execution errors. The interleaved solution is more robust and can potentially recover from erroneous predictions at a previous step.

Ablations. Table 2 also presents scores corresponding to GoalNet ablations. For a fair comparison, all ablation models have nearly same number of parameters. Each ablation performs worse, indicating that all model features add some value to the model. We find that without the relational information $\mathcal{R}(s_t)$ as input, the model is unable to capture change in the spatial relations among world objects and GRR drops by 4 pts. Similarly, the instance grounding of the objects also helps. For instance, without knowing that *coffee-table* in the instruction *"place beer on top of coffee-table"* is mapped to table_0, the agents needs to additionally infer the specific instance that needs to be manipulated (see Fig. 4). A drop of 5.5 pts is observed in this case.

Further, when we only predict positive subgoals (*w/o δ^- prediction*), GRR drops by 5 pts. Such a model is unable to predict the required subgoals when only relations are removed from the state (e.g., dropping objects). Unsurprisingly, without δ^+ prediction, GRR drops by a mammoth 37 pts. The inclusion of temporal context allows easier learning of correlated and commonly repeated action sequences. Ablating this component drops GRR by 31 pts. Without the grammar mask Ω, the GRR drops marginally (4 pts), indicating, to a certain extent, the model's ability to learn grammar-semantics from data. Finally, GoalNet* shows a drop in performance due to factored prediction, albeit with improved generalization, as discussed next.

Generalization. Next, we evaluate model generalization to *unseen* instruction by constructing three generalization data sets (see Table 3). We test the performance when replacing verb frames in the training data with those absent in the data set. For instance, we replace *boil* in *"boil milk"* with *heat*. Next, we paraphrase the language input to test instruction-level generalization. For example, we paraphrase *"gather all used plates and glasses, place into sink"* to *"collect all used dishes and glasses, keep in wash basin"*. Finally, all objects references in test data points are replaced by semantically similar objects not in training set. Table 3 presents the performance scores of baselines, GoalNet and GoalNet*. We observe that GoalNet generalizes much better in both paraphrasing and unseen verbs relative to the baselines. The performance over She&Chai baseline is mainly due to the presence of dense token (ConceptNet) and instruction (SBERT) representations as opposed to storing observed verb-frame hypotheses. Further, GoalNet* generalizes in the case of unseen objects due to its prediction pipeline being agnostic to the object set (Fig. 4).

Limitations. GoalNet struggles where there are multiple plausible subgoals, causing imperfect scores. Further, our models assume cause-effect modelling as PDDL input. Future work will investigate (i) use of learned symbolic actions models instead of formal PDDL input and (ii) modelling state and goal uncertainty inherent in realistic domains and (iii) resolving the rare cases (7 of 181 plans) where the subgoal predicts the same goal in loops.

6 Conclusions

We propose GoalNet, a novel neuro-symbolic architecture that learns to infer goal predicates for an input language instruction and world state, which when passed to an underpinning symbolic planner enables reaching goal states. As a departure from both pure imitation learning or a direct prediction of goal sets underlying the task, GoalNet interleaves subgoal prediction and symbolic planner in a loop combining the benefits of both. The use of neural world state and language encoding facilitates generalization to new instructions and scenes with new object types. Interleaving allows for potentially correcting subgoal predictor mistakes. We demonstrate significant gains in goal-reaching performance in both *a-priori* known and unknown instruction-environment settings compared to state-of-the-art baselines.

Acknowledgments

The work is supported by grants from Google, IBM, Verisk, Microsoft, and Huawei, and the Jai Gupta chair fellowship and Pankaj Gupta Young Faculty Fellowship by IIT Delhi. We thank the IIT-D HPC facility. We also thank Microsoft Accelerate Foundation Models Research (AFMR) program that helps provide access to OpenAI models.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Berant, J.; Chou, A.; Frostig, R.; and Liang, P. 2013. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 conference on empirical methods in natural language processing*, 1533–1544.

Bird, S.; Klein, E.; and Loper, E. 2009. *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.".

Boularias, A.; Duvallet, F.; Oh, J.; and Stentz, A. 2015. Grounding spatial relations for outdoor robot navigation. In *ICRA*, 1976–1982. IEEE.

Branavan, S.; Kushman, N.; Lei, T.; and Barzilay, R. 2012. Learning high-level planning from text. ACL.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *ACL*, volume 1, 4171–4186.

Dragan, A. D.; Bauman, S.; Forlizzi, J.; and Srinivasa, S. S. 2015. Effects of robot motion on human-robot collaboration. In *HRI)*, 51–58. IEEE.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *CVPR*, 1026–1034.

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

Jang, E.; Gu, S.; and Poole, B. 2017. Categorical reparameterization with gumbel-softmax. *ICLR*.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Lesh, N.; and Etzioni, O. 1995. A sound and fast goal recognizer. In *IJCAI*, volume 95, 1704–1710. Citeseer.

Liao, Y.-H.; Puig, X.; Boben, M.; Torralba, A.; and Fidler, S. 2019. Synthesizing environment-aware activities via activity sketches. In *CVPR*, 6291–6299.

Lipovetzky, N.; and Geffner, H. 2012. Width and serialization of classical planning problems. In *ECAI 2012*, 540–545. IOS Press.

Mei, H.; Bansal, M.; and Walter, M. R. 2016. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *ACL*, 720–730.

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

Mikolov, T.; Grave, É.; Bojanowski, P.; Puhrsch, C.; and Joulin, A. 2018. Advances in Pre-Training Distributed Word Representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*.

Misra, D.; Bennett, A.; Blukis, V.; Niklasson, E.; Shatkhin, M.; and Artzi, Y. 2018. Mapping Instructions to Actions in 3D Environments with Visual Goal Prediction. In *EMNLP*, 2667–2678.

Misra, D.; Tao, K.; Liang, P.; and Saxena, A. 2015. Environment-driven lexicon induction for high-level instructions. In *ACL*, 992–1002.

Misra, D. K.; Sung, J.; Lee, K.; and Saxena, A. 2016. Tell me dave: Context-sensitive grounding of natural language to manipulation instructions. *The International Journal of Robotics Research*, 35(1-3): 281–300.

Paul, R.; Arkin, J.; Aksaray, D.; Roy, N.; and Howard, T. M. 2018. Efficient grounding of abstract spatial concepts for natural language interaction with robot platforms. *The International Journal of Robotics Research*, 37(10): 1269–1299.

Puig, X.; Ra, K.; Boben, M.; Li, J.; Wang, T.; Fidler, S.; and Torralba, A. 2018. Virtualhome: Simulating household activities via programs. In *CVPR*, 8494–8502.

Reimers, N.; Gurevych, I.; Reimers, N.; et al. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *EMNLP*, 671–688.

Rintanen, J. 2012. Planning as satisfiability: Heuristics. *Artificial Intelligence*, 193: 45–86.

Sebastia, L.; Onaindia, E.; and Marzal, E. 2006. Decomposition of planning problems. *Ai Communications*, 19: 49–81.

She, L.; and Chai, J. 2016. Incremental acquisition of verb hypothesis space towards physical world interaction. In *ACL*, 108–117.

Silver, T.; Athalye, A.; Tenenbaum, J. B.; Lozano-Perez, T.; and Kaelbling, L. P. 2022. Learning Neuro-Symbolic Skills for Bilevel Planning. arXiv:2206.10680.

Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; and Garg, A. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 11523–11530. IEEE.

Speer, R.; Chin, J.; and Havasi, C. 2019. ConceptNet Numberbatch, the best pre-computed word embeddings you can use. *GitHub repository*.

Suhr, A.; and Artzi, Y. 2018. Situated Mapping of Sequential Instructions to Actions with Single-step Reward Observation. In *ACL*, 2072–2082.

Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M. R.; Banerjee, A. G.; Teller, S.; and Roy, N. 2011. Approaching the symbol grounding problem with probabilistic graphical models. *AI magazine*, 32(4): 64–76.

Tuli, S.; Bansal, R.; Paul, R.; and , M. 2021. TANGO: Commonsense Generalization in Predicting Tool Interactions for Mobile Manipulators. In *IJCAI*, 4197–4205.

Tuli, S.; Bansal, R.; Paul, R.; et al. 2022. TOOLTANGO: Common sense Generalization in Predicting Sequential Tool Interactions for Robot Plan Synthesis. *Journal of Artificial Intelligence Research*, 75: 1595–1631.