Following Instructions by Imagining and Reaching Visual Goals

John D. Kanu, Eadom Dessalene, Xiaomin Lin, Yiannis Aloimonos
Department of Computer Science, University of Maryland, College Park
jdkanu@cs.umd.edu, edessale@cs.umd.edu, xlin01@umd.edu, yiannis@cs.umd.edu

Abstract

While traditional methods for instruction following typically assume prior linguistic and perceptual knowledge, recent work in reinforcement learning has advanced the end-to-end learning of policies for instruction following, but typically involves learning a single flat policy that maps an instruction and observation directly to an action without explicitly separating reasoning from action. We present a novel hierarchical architecture for instruction following, combining ideas from unsupervised representation learning, visual goal reaching, and text-conditional image transformation. Our architecture learns to sequentially imagine transformations of scene structure and choose appropriate actions to fulfill structural transformations. Our architecture operates on raw pixel images and assumes no prior linguistic or perceptual knowledge. The only extrinsic reward signal comes from a measure of task completion. We demonstrate the effectiveness of our method for controlling a robot arm to execute object manipulation instructions in a simulated interactive 3D environment. Our method outperforms flat policies with raw-image and ground-truth states, and a hierarchy of policies with ground-truth states.[1]

Introduction

Building agents that can follow instructions in a physical environment is a longstanding challenge in the development of Artificial Intelligence, originally introduced as SHRDLU in the early 1970s (Winograd 1971). Traditional methods assume prior linguistic, perceptual, and procedural knowledge in order to ground and execute instructions (Tellex et al. 2011). Recent works have shown that deep reinforcement learning (RL) may be a promising framework for instruction-following in simulated 2D and 3D environments, for tasks in navigation (Hermann et al. 2017; Chao, Cakmak, and Thomaz 2011) and manipulation (Misra, Langford, and Artzi 2017; Bahdanau et al. 2018; Jiang et al. 2019). Naturally, instructions often represent temporally extended tasks, whose successful execution requires language grounding, structured reasoning, and complex skills. However, learning to effectively ground language instructions and the skills to achieve them in the deep RL framework is still an open challenge. In this work, we consider the question: how can we incorporate an explicit spatial reasoning objective to accomplish temporally extended tasks, with no assumption of prior linguistic or perceptual knowledge?

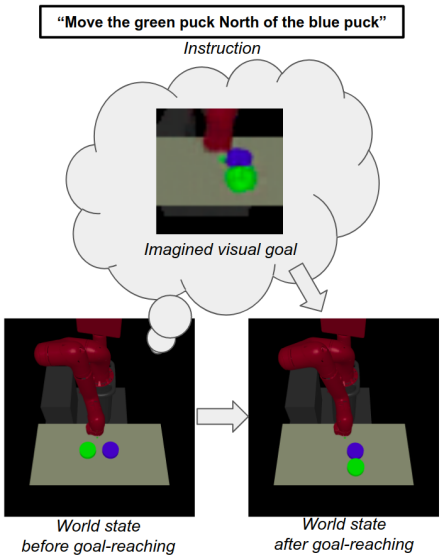

Figure 1: Sample instruction in an interactive 3D environment with a robotic arm. The agent learns to generate and reach latent visual goals to execute instructions from scratch. A reconstruction of a latent goal is shown.

In several previous works, models are trained to ground language using a multimodal state representation (Janner, Narasimhan, and Barzilay 2018; Chaplot et al. 2018), and acquire skills through a flat policy that maps this state to an action. However, such techniques are based on an inductive bias that limits modeling flexibility. (Jiang et al. 2019) employ hierarchical reinforcement learning (HRL) to train a hierarchy of policies to learn temporally extended tasks through goal-directed behavior, with a high-level policy issuing goals to a low-level policy in the form of language. Like the flat policies, their low-level policy must select an

[1]Code is available at `https://github.com/jdkanu/IFIG`.

appropriate action without any explicit language-grounding objective.

We consider abstractions other than language. One possibility is to use hard-coded abstractions (Sutton, Precup, and Singh 1999; Konidaris and Barto 2007; Heess et al. 2016; Peng et al. 2017), though this technique limits the flexibility to perform arbitrary tasks specified by language. Another possibility is to use raw sensory signals; however, photometric loss functions such as Euclidean distance often do not correspond to meaningful differences between states (Ponomarenko et al. 2015; Zhang et al. 2018) and are thus ineffective for training agents. Another possibility is to learn a structured latent representation of the image. As demonstrated in (Nair et al. 2018; Pong et al. 2019), motor policies can be trained on image embeddings with reward based on the distance between states.

Building upon the HRL framework and methods for state representation learning and goal-reaching in (Nair et al. 2018; Pong et al. 2019), we introduce a framework, called Instruction-Following with Imagined Goals (IFIG), in which an agent learns to sequentialy imagine and reach states corresponding to the execution of an instruction from the current state. The architecture consists of three main parts: (1) a state representation module, which learns a structured latent representation of image states, (2) a goal-generating module, which learns instruction-conditional structured state transformations to synthesize goal states to be reached in order to complete the instruction, and (3) a goal-reaching module, which learns the skills to move between arbitrary states. Learning structured latent representations according to an image according to a reconstruction objective allows the agent to represent the information in the state without manual engineering, while learning to generate and reach goals using separate objectives allows us to decouple the visual grounding of the instruction and task-directed spatial reasoning, from the lower-level acquisition of skills.

We conducted experiments inside a simulated interactive 3D environment, converting physical object-arrangement instructions to continuous control signals on a robot arm. We benchmark our model against a flat policy with raw-image states, a flat policy with ground-truth states, and a hierarchy of policies with ground-truth states. Experiments show that our model outperforms all three, with accuracy measured by distance between each puck and its goal position. Through visualization of goal reconstructions, we show that our model has learned instruction-conditional state transformations that suggest an ability to perform spatial reasoning.

Related Work

Language grounding in robotics. Traditional methods exploit the compositional structure of language, using computational approaches to generate action plans. Several methods infer action plans using pre-defined templates for semantic parsing, object localization, and spatial relations (Tellex et al. 2011; Guadarrama et al. 2013; Misra et al. 2016). These methods make limited relational modelling assumptions for the environment and spatial reasoning. These methods require explicit representations of object locations and attributes in the world model.

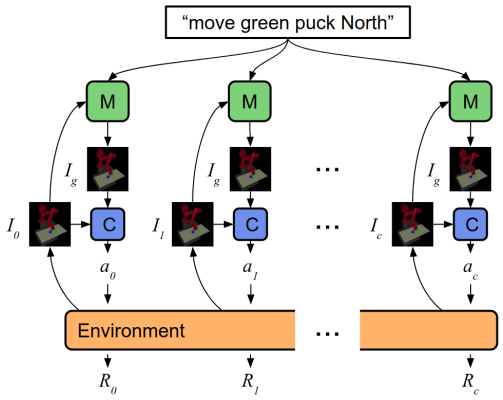

Figure 2: A schematic depiction of instruction following with imagined goals (IFIG). Given an instruction and raw pixel image observations I_t, the meta-controller (M) generates a latent goal z_g (represented by an image I_g for simplicity). At each time step t, the controller (C) maps the current image latent z_t and goal z_g to an action a_t. The environment processes the action a_t, performs a single forward-step in the simulation, and a reward R_t is generated according to the instruction. This process repeats until either the goal is reached, or the maximum number of steps is exceeded.

To our knowledge, our work is the first to visually ground language for robotic manipulation using reinforcement learning. Each of these methods assumes access to world state and attributes of the environment objects.

Mapping instructions to actions. Early methods rely on object-level representations and relational modelling, exploiting the compositional structure of language to parse the instruction into a formal language (Artzi and Zettlemoyer 2013; Misra et al. 2016; Chen and Mooney 2011; Kuhlmann et al. 2004). In recent years, deep reinforcement learning (RL) has been applied to instruction following tasks for navigation and manipulation in simulated 2D and 3D environments. Agents typically consist of a mixing module which learns a multimodal representation of the instruction and state. Simple concatenation is used in (Mei, Bansal, and Walter 2016; Misra, Langford, and Artzi 2017; Hermann et al. 2017), however, more sophisticated methods for visual grounding of language have been explored. (Chaplot et al. 2018) use an attention mechanism based on multiplicative interactions between instruction and image state, which is fed into a policy that generates an action. (Janner, Narasimhan, and Barzilay 2018) learn to generate a convolutional kernel from the instruction, which is used to convolve the image before passing it to the policy. (Jiang et al. 2019; Hu et al. 2019) use language as the interface between a high-level policy and a low-level policy, without any explicit grounding objective. (Prabhudesai et al. 2019) train generative networks to map instructions and an image to a 3-dimensional visual feature representation of the desired state, but they assume the existence of a pre-trained object detector. In contrast to these works, we learn to generate a sequence of visual goal states. We train our model end-

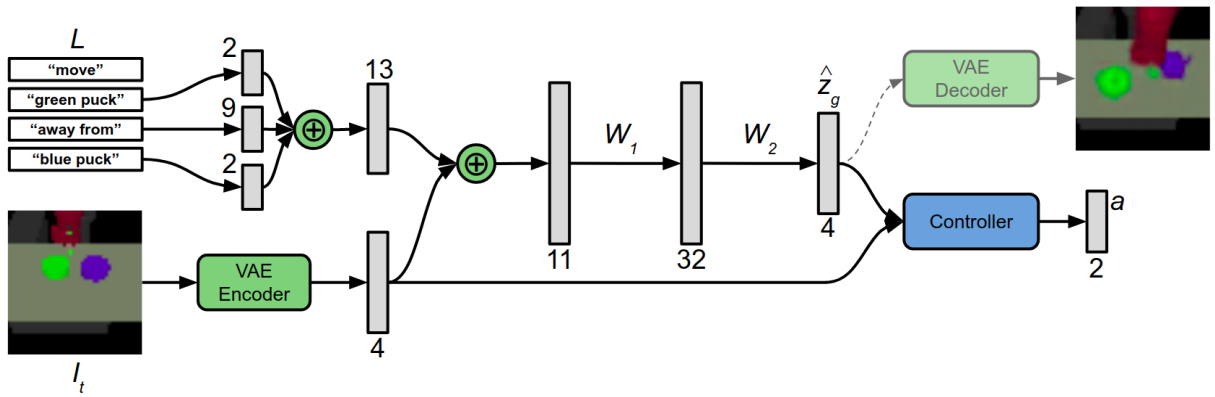

Figure 3: A schematic depiction of the dataflow for mapping instructions and images to actions. The meta-controller generates a goal in latent space, given a fixed-length one-hot vector g representing the instruction, and the current latent state z_t.

to-end with no assumption of prior linguistic or perceptual knowledge.

Preliminaries

Our method combines methods for unsupervised representation learning, goal-conditioned reinforcement learning (Nair et al. 2018; Pong et al. 2019), and text-conditional image transformations. We briefly review the methods we employ.

Goal-conditioned reinforcement learning. Our meta-controller and controller are each trained using goal-conditioned reinforcement learning, where goals are instructions for the high-level policy and goal states for the low-level policy. The goal-conditioned RL problem considers an Augmented Markov Decision Process (MDP) defined by the tuple (S, G, A, T, R, γ), where S is the state space, G is the goal space, A is the action space, $T : S \times A \times S \to [0, \infty)$ represents the probability density of transitioning from $s_t \in S$ to $s_{t+1} \in S$ under the action $a \in A$, $R : S \times A \times G \to [r_{\text{min}}, r_{\text{max}}]$ represents the scalar reward for each transition under a given goal, and $\gamma \in [0, 1)$ is the discount factor. The objective is to learn a policy $\pi(a_t|s_t, g)$ that maximizes the expected discounted return $R_t = \mathbb{E}[\sum_{i=t}^{T} \gamma^{(i-t)} r_i]$, where $r_i = r(s_i, a_i, g)$.

Reinforcement learning with imagined goals. Reinforcement learning with imagined goals (RIG) is a framework for jointly learning a latent representation of raw sensory states and a policy for reaching arbitrary latent goal states (Nair et al. 2018). We summarize the algorithm as follows. First, an exploration policy is used to collect a dataset D of visual observations. In our case, we use Skew-Fit (Pong et al. 2019) for exploration, a self-supervised state-covering method which provides a formal exploration objective that maximizes the entropy of the goal distribution. Second, a beta variational autoencoder (β-VAE) is trained on D to learn an embedding of raw pixel images, which has been shown to learn structured representations of high-dimensional data (Kingma and Welling 2013). The VAE consists of an encoder $q_\phi(z|s)$ that models the probability of latent distribution z corresponding to the visual state s given parameters ϕ, and a decoder $p_\psi(s|z)$ that models the probability of visual state s corresponding to the latent distribution z given parameters ψ. The encoder q_ϕ and decoder p_ψ are approximated by neural networks mapping between image states and latent states. After training the VAE, a policy $\pi(z, z_g) = a$ is trained to generate actions a that move an agent from latent state z to z_g, where z_g is a latent goal sampled from the latent variable model. Reward is computed as $r(s, g) = -||e(s) - e(g)|| = -||z - z_g||$, the Euclidean distance between the current latent and the latent goal.

Problem Formulation

We address the problem of instruction following in interactive environments, where an autonomous agent is issued a task in the form of an instruction, and must produce actions at each time step to successfully complete the task. Assume the agent exists inside an episodic environment. At the beginning of each episode, the environment is randomly initialized, and an instruction L is generated. At each time step t, the agent is given the instruction L and a raw pixel-level image I_t of the environment, and generates an action a_t. For simplicity, we consider the case that the instruction can be embedded into a fixed-length vector. We assume the existence of a function $f_L : S \to (r_{min}, r_{max})$ which outputs the agreement between the instruction L and state s_t. The episode terminates when either the agreement is maximized or the maximum number of steps is reached. The objective is to learn a policy $\pi(s, L) = a$ that maximizes the expected return, which corresponds to a successful completion of the task specified by the instruction within the limit of steps.

Instruction-Following with Imagined Goals

In this section, we present our framework for training an interactive agent to perform low-level continuous control conditioned on high-level language instructions from visual observations, by integrating a structured state representation, a high-level policy that generates visual goals, and a task-agnostic low-level policy that performs control to reach visual goals. We begin by formalizing the problem of synthesizing and reaching visual goals conditioned

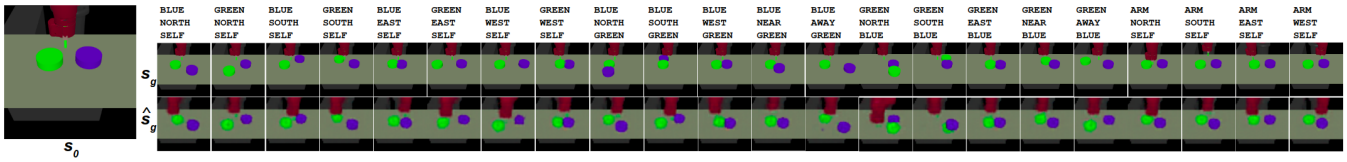

Figure 4: Illustrations of all 22 instructions executed on a sample initial state s_0. The top row shows an image of a valid goal state (s_g), and the bottom row shows latent goal reconstructions ($\hat{s_g}$) learned by our model.

on instructions. We then summarize a method for learning state representations $\boldsymbol{z}$ from visual observations $\boldsymbol{s}$, which is fully unsupervised (Nair et al. 2018). We summarize a method for training a low-level, task-agnostic goal-reaching policy $\pi_l(\boldsymbol{a}|\boldsymbol{z_t}, \boldsymbol{z_g})$, also fully unsupervised (Pong et al. 2019). We discuss our method for training a high-level policy $\pi_h(\boldsymbol{z_g}|\boldsymbol{z_t})$ to generate goal-directed structured transformations of visual inputs, using such a low-level policy. We refer to this architecture as *Instruction Following with Imagined Goals* (IFIG, Figure 2). A schematic depiction of instruction following with imagined goals (IFIG) is given in Figure 2. The meta-controller architecture is illustrated in Figure 2.

We factorize the agent into a *meta-controller*, which generates a goal given the instruction and current state, and a *controller*, which maps the current state and goal to an action. The meta-controller is trained to map pairs of images and instructions to goal images in latent space. During test-time, the meta-controller will generate a goal given the current state and instruction, and the controller will map the current state and goal state to a sequence of actions.

State representation learning

In order to map instructions and states to goals, and states to actions, we must find a suitable state representation. The state representation reflect the factors of variation in the scene relevant to instruction following tasks, such as the number, position, and attributes of objects, such as color and size. Raw pixel-level images solve this problem, but learning is harder. Our method uses unsupervised state representation learning using a variational autoencoder (VAE) to acquire a latent distribution, which can be used in training the controller and meta-controller. For the controller, this allows us to generate structured representations of raw sensory inputs and compute reward for goal reaching using distances in latent space. For the meta-controller, the latent space naturally lends itself to learning a transformation of environment structure conditioned on an instruction.

Following the approach of Nair et al. (2018), we collect a dataset of state observations $\{s^{(i)}\}$ from the execution of a random policy (Pong et al. 2019), train a β-VAE on the dataset. We subsequently fine-tune the VAE. We embed the state s_t and goals s_g into a latent space $\mathcal{Z}$ using an encoder e to obtain a latent state $z_t = e(s_t)$ and latent goal $z_g = e(g)$. We then use the mean of the encoder as the state encoding, i.e. $z = e(s) \triangleq \mu_\phi(s)$. See Preliminaries for details.

Goal-reaching policy

To reach goals, we train a policy $\pi_l(a_t|z_t, z_g)$ that generates actions to move between a given current state z_t and goal state z_g, according to the RIG framework (Nair et al. 2018). This process is completely unsupervised. The reward function is given by the function $r(s, g) = -||e(s) - e(g)|| = -||z - z_g||$, the Euclidean distance between the current latent and the latent goal. See preliminaries for more details. Training of the controller and meta-controller are disjoint. With the trained low-level policy π_l, the agent can attempt to reach arbitrary given goals in the latent distribution Z.

Goal-generating policy

To execute an instruction, a goal-generating policy π_h generates a goal state for the goal-reaching policy. The meta-controller is implemented as a fully-connected multi-layer perceptron (MLP). This fully-connected architecture provides the ability to learn a mapping that transforms the latent vector to a new vector that satisfies the instruction. See Figure 3 for a schematic depiction of the architecture. The meta-controller generates a goal $\hat{z_g}$ in latent space. At each time step, the raw-pixel level image I_t is embedded into a vector z_t in latent space by the encoder network. We assume, for simplicity, the existence of a function $f_{enc} : L \to \mathbb{R}^N$, which embeds the instruction L to a fixed-length vector of dimensionality N, for some N. Thus, the joint representation of I_t and L is simply the concatention of the embedding z_t and $f_{enc}(L)$:

$$s_t = [z_t; f_{enc}(L)]$$

The meta-controller maps s_t to a goal $\hat{z_g}$ in latent space. We train the meta-controller using the soft actor-critic (SAC) algorithm. This is a popular algorithm for learning policies with continuous action space, and has demonstrated good performance on similar tasks (CITE). A continuous scalar reward signal $r \in (-\infty, 0]$ is issued by the environment based on the Euclidean distances between the actual and desired object positions:

$$r(s, L) = \sum_{o \in O} -||x_o - g_o||_2 \quad (1)$$

where x_o is the current position of o, g_o is the goal position of o under instruction L, and the negation ensures that the reward signal is proportional to the probability of instruction fulfillment.

It is important to note that the ability of the meta-controller to produce a goal state representing the successful completion of the instruction is dependent on the ability of the controller to reach the desired goal state, since reward

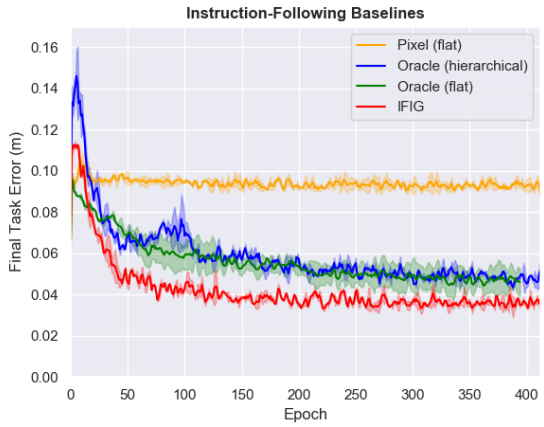

Figure 5: Cumulative distance to goals for all instructions over 3 random seeds. We note that 1 step in the metacontroller environment corresponds to 10 steps in the controller environment, so we normalize the training duration of the flat policies by a factor of 10.

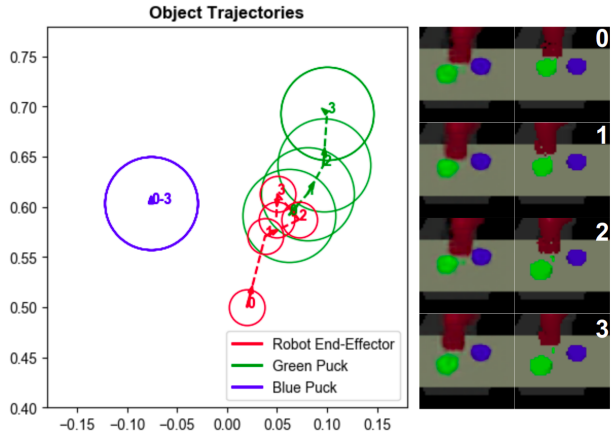

Figure 6: Sample trajectories of a blue puck, green puck, and arm for the instruction *move green puck away from blue puck*. The sequence of latent goal reconstructions, raw pixel images of reached states, and ground truth object positions during execution are labeled 0-3. Goals describe the sequence of moving to the green puck and increasing its distance to the blue puck.

is maximized only if the desired goal state is reached. One common issue with hierarchical policies comes from the updating of the controller during training, which often necessitates off-policy corrections (CITE). Since we keep the controller fixed, our method does not require off-policy corrections.

Experiments

We are interested in the end-to-end training of agents to perform goal-directed spatial reasoning in the execution of tasks with minimal supervision and no prior knowledge, by training a policy to transform structured visual observations according to symbolic instructions using reinforcement learning (RL). First, we test the ability of the method to transform visual states and reach goals. Then we compare the performance of our method and two benchmarks, a flat policy with no intermediate goals, and a flat policy with access to ground truth. We aim to answer the following two questions:

1. How effective is the visual modality for representing goals for instruction-following over other modalities?
2. How effective is the framework in which we imagine and reach goals, compared with methods that do not make any explicit goal objective?

We devise a set of experiments to address these questions. For the first question, we compare IFIG to an oracle hierarchical architecture that uses ground truth object states for observations and reward, instead of latent visual states. For the second question, we implement two flat policies, one of which operates on raw images and another that operates on ground truth states.

Environment

A natural example of this problem occurs when a robotic arm is tasked with physically manipulating objects in an episodic 3D environment, where goal configurations are issued in the form of an instruction. Consider the problem of visual object-manipulation, where an agent controls a robotic arm inside an episodic 3D environment. At the beginning of each episode, the object and arm configurations are randomly initialized in the environment, and an instruction $\boldsymbol{g}$ is generated, describing a set of spatial relations among objects to be achieved by the agent. We do not filter out instructions that are already satisfied, to train the agent to take no action in this case. At each time step, the robotic arm produces an action specifying the velocity of its end-effector. The environment processes the action by updating its state according to a, updating the joint configuration of the robot arm via inverse kinematics, performing collision detection and response, and updating the position of all physical entities in the scene. For evaluation, we use the MuJoCo simulator with a Sawyer arm with continuous control.

In all experiments, the action space is 2-dimensional, describing the change in the end-effector position at each time step. The state is represented as a top-down view RGB image of the task from a static viewpoint. Each puck has height 0.02m and diameter of 0.06m. At the beginning of each episode both puck positions are sampled from uniformly random distributions of positions within a workspace. The environment is reset after the instruction is completed or after 200 steps.

Instructions

We define a set of 22 instructions representing goal arrangements of object and arm. We assume a vocabulary of 14 tokens referring to the blue puck, the green puck, the arm, cardinal directions (North, South, East, and West), binary relations (representing the prepositions “on top of”, “next to”,

Benchmarks	Parameters	Unary	Binary
IFIG	314,752	**25.3%**	**42.3%**
Oracle (Flat)	68,088	20.3%	41.8%
Oracle (Hier.)	116,788	16.2%	37.2%
Pixel (Flat)	210,116	2.4%	4.1%

Table 1: The success rates for the execution of unary and binary instructions. Higher is better. We label an executed instruction as a success if the pucks are displaced within at least 0.04 cm of the intended position for the cardinal instructions, or if the pucks are displaced at least 5 cm apart for the AWAY instruction.

and "away from"), and a self-identifier token. Each instruction, such as *"move blue puck North"* or *"move the green puck North of the blue puck"*, is uniquely tokenized by a fixed-length sequence of tokens $L = (l_1, l_2, l_3)$. For unary instructions, l_1 identifies the object that should be moved (blue puck, green puck, or arm), l_2 identifies the cardinal direction (North, South, East, or West), and l_3 takes the value of a self-identifier token. For binary instructions, l_1 refers to a puck, l_2 describes the desired spatial relation, and l_3 refers to the other puck.

We encode the tokenized instruction $L = (l_1, l_2, l_3)$ as a concatenation of one-hot vectors. For each i, we categorize the tokens that appear at l_i to produce a one-hot vector x_i for each instruction. Each instruction L is thus encoded as $f_{enc}(L) = [x_1; x_2; x_3]$. This uniquely represents each instruction as a vector to condition the high-level policy.

Benchmarks

Oracle hierarchical architecture. In order to interpret the effectiveness of the visual modality at encoding goals for an agent in the execution of a task, we compare IFIG to a hierarchy of policies with ground truth object states instead of latent visual states. The observation a 6-dimensional vector containing the 2-d positions of both pucks and the arm. The controller is trained unsupervised to map current state and goal states in ground truth to actions that reach the goal. Since ground-truth states describe the positions of objects and arms in 3-dimensional Cartesian space, reward is computed based on the Euclidean distance between current state and goal state: $r(s, g) = -||s - g||$. This follows the same fashion as the IFIG controller (Nair et al. 2018). The metacontroller is trained similar to our meta-controller, with the same reward function measuring task completion given by Equation 1.

The architectures for the controller and metacontroller are nearly identical with both being fully connected layer networks with 128 hidden layer nodes and RELU activation units. The number of layers in the controller and metacontroller policies are 4 and 3 respectively, making a total of 66,040 weights for the controller policy and 50,748 weights for the metacontroller policy. The controller has 6 inputs corresponding to the ground truth XY positions of the hand, green puck, and blue puck, and outputs a 2D velocity vector for the end effector. The input to the metacontroller is a concatenated vector of the 6D ground truth positions and the 12D instruction embedding.

Hyperparameter	Value
Metacont. Path Length	10
Controller Path Length	25
Latent State Dim.	6
Policy Learning Rate	0.005
Value Function Learning Rate	0.005
Target Update Rate	0.005
Batch Size	64
Discount Factor	0.99

Table 2: The hyperparameters to the training of the metacontroller.

Oracle flat policy. The oracle flat policy maps the instruction vector and 6-d ground-truth state directly to an action, without any intermediate goals. Reward is computed according to Equation 1. The architecture for the oracle flat policy is chosen to be a 4-layer MLP with a near-identical architecture to the controller used for the hierarchical architecture.

Raw-image flat policy. The raw-image flat policy maps the instruction vector and 48×48 pixel image directly to an action, without any intermediate goals. Reward is computed according to Equation 1. The architecture to the raw-image flat policy is chosen to be a 3-layer convolutional neural network, with kernal sizes: 5x3, 3x3, and 3x3, number of output filters: 16, 32, 64 and strides: 3, 2, 2. The input to the network is a 48x48 RGB channel image of the scene.

Results and Discussion

For all the models described in Section , the performance on both unary and binary instructions is shown in Table **??**.

Performance of IFIG. Figure 5 shows final task error for each epoch during training, with IFIG achieving the lowest error. From Table **??**, we observe that during test time, IFIG outperforms two flat policies trained on raw pixel images and ground-truth states. On both unary and binary instructions, IFIG achieves the highest task success rate, followed by the oracle flat policy, followed by the oracle hierarchy of policies, followed by the raw image flat policy. The margin of improvement is approx. 5.0% for unary instructions, but only 0.5% for binary instructions. Overall IFIG is trained with roughly 1.5x the parameter count as the raw image flat policy, while achieving over 10x the task success rate as the flat policy. Without access to a manually defined world state, IFIG outperforms both methods with access to ground truth state.

Policy execution. Figure 6 shows a policy execution of the IFIG model for the instruction *move green puck away from blue puck*. In this figure, we demonstrate the agent's ability to fulfill the instruction. Before the instruction is executed, the green puck is nearby the blue puck. During policy execution, the model generates a sequence of goal that describe a path of the green puck moving increasingly farther away

from the blue puck. We include a visualization of the visual goals generated by the meta-controller during this execution. The model has learned to localize an object by a reference token, and modify its position within the latent state according to the *away from* spatial relation.

Conclusion

In this paper we proposed a novel framework for training agents to execute language-like object-manipulation instructions from raw pixels. The framework, called Instruction-Following with Imagined Goals (IFIG), integrates a mechanism for sequentially synthesizing visual goal states and existing self-supervised methods for state representation learning and visual goal reaching. The architecture requires little supervision or manual engineering, and assumes no prior linguistic or perceptual knowledge. The only extrinsic reward comes from a measurement of task completion. We observe that IFIG, requiring no ground-truth state labelling, outperforms an oracle hierarchical architecture that performs goal-generating and goal-reaching with access to ground truth states. We also observe that our framework outperforms flat policies that lack an explicit language grounding objective. The visualization of the hierarchical policy execution indicates that the agent learns to decompose complex control tasks into a series of sub-tasks represented by visual goals.

References

[Artzi and Zettlemoyer 2013] Artzi, Y., and Zettlemoyer, L. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics* 1:49–62.

[Bahdanau et al. 2018] Bahdanau, D.; Hill, F.; Leike, J.; Hughes, E.; Hosseini, A.; Kohli, P.; and Grefenstette, E. 2018. Learning to understand goal specifications by modelling reward. *arXiv preprint arXiv:1806.01946*.

[Chao, Cakmak, and Thomaz 2011] Chao, C.; Cakmak, M.; and Thomaz, A. L. 2011. Towards grounding concepts for transfer in goal learning from demonstration. In *2011 IEEE International Conference on Development and Learning (ICDL)*, volume 2, 1–6. IEEE.

[Chaplot et al. 2018] Chaplot, D. S.; Sathyendra, K. M.; Pasumarthi, R. K.; Rajagopal, D.; and Salakhutdinov, R. 2018. Gated-attention architectures for task-oriented language grounding. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

[Chen and Mooney 2011] Chen, D. L., and Mooney, R. J. 2011. Learning to interpret natural language navigation instructions from observations. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

[Guadarrama et al. 2013] Guadarrama, S.; Riano, L.; Golland, D.; Go, D.; Jia, Y.; Klein, D.; Abbeel, P.; Darrell, T.; et al. 2013. Grounding spatial relations for human-robot interaction. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1640–1647. IEEE.

[Heess et al. 2016] Heess, N.; Wayne, G.; Tassa, Y.; Lillicrap, T.; Riedmiller, M.; and Silver, D. 2016. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*.

[Hermann et al. 2017] Hermann, K. M.; Hill, F.; Green, S.; Wang, F.; Faulkner, R.; Soyer, H.; Szepesvari, D.; Czarnecki, W. M.; Jaderberg, M.; Teplyashin, D.; et al. 2017. Grounded language learning in a simulated 3d world. *arXiv preprint arXiv:1706.06551*.

[Hu et al. 2019] Hu, H.; Yarats, D.; Gong, Q.; Tian, Y.; and Lewis, M. 2019. Hierarchical decision making by generating and following natural language instructions. *arXiv preprint arXiv:1906.00744*.

[Janner, Narasimhan, and Barzilay 2018] Janner, M.; Narasimhan, K.; and Barzilay, R. 2018. Representation learning for grounded spatial reasoning. *Transactions of the Association for Computational Linguistics* 6:49–61.

[Jiang et al. 2019] Jiang, Y.; Gu, S.; Murphy, K.; and Finn, C. 2019. Language as an abstraction for hierarchical deep reinforcement learning. *arXiv preprint arXiv:1906.07343*.

[Kingma and Welling 2013] Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[Konidaris and Barto 2007] Konidaris, G., and Barto, A. G. 2007. Building portable options: Skill transfer in reinforcement learning. In *IJCAI*, volume 7, 895–900.

[Kuhlmann et al. 2004] Kuhlmann, G.; Stone, P.; Mooney, R.; and Shavlik, J. 2004. Guiding a reinforcement learner with natural language advice: Initial results in robocup soccer. In *The AAAI-2004 workshop on supervisory control of learning and adaptive systems*. San Jose, CA.

[Mei, Bansal, and Walter 2016] Mei, H.; Bansal, M.; and Walter, M. R. 2016. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *Thirtieth AAAI Conference on Artificial Intelligence*.

[Misra et al. 2016] 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.

[Misra, Langford, and Artzi 2017] Misra, D.; Langford, J.; and Artzi, Y. 2017. Mapping instructions and visual observations to actions with reinforcement learning. *arXiv preprint arXiv:1704.08795*.

[Nair et al. 2018] Nair, A. V.; Pong, V.; Dalal, M.; Bahl, S.; Lin, S.; and Levine, S. 2018. Visual reinforcement learning with imagined goals. In *Advances in Neural Information Processing Systems*, 9191–9200.

[Peng et al. 2017] Peng, X. B.; Berseth, G.; Yin, K.; and Van De Panne, M. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Transactions on Graphics (TOG)* 36(4):41.

[Pong et al. 2019] Pong, V. H.; Dalal, M.; Lin, S.; Nair, A.; Bahl, S.; and Levine, S. 2019. Skew-fit: State-covering self-supervised reinforcement learning. *arXiv preprint arXiv:1903.03698*.

[Ponomarenko et al. 2015] Ponomarenko, N.; Jin, L.; Ieremeiev, O.; Lukin, V.; Egiazarian, K.; Astola, J.; Vozel,

B.; Chehdi, K.; Carli, M.; Battisti, F.; et al. 2015. Image database tid2013: Peculiarities, results and perspectives. *Signal Processing: Image Communication* 30:57–77.

[Prabhudesai et al. 2019] Prabhudesai, M.; Tung, H.-Y. F.; Javed, S. A.; Sieb, M.; Harley, A. W.; and Fragkiadaki, K. 2019. Embodied language grounding with implicit 3d visual feature representations. *arXiv preprint arXiv:1910.01210*.

[Sutton, Precup, and Singh 1999] Sutton, R. S.; Precup, D.; and Singh, S. 1999. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence* 112(1-2):181–211.

[Tellex et al. 2011] Tellex, S.; Kollar, T.; Dickerson, S.; Walter, M. R.; Banerjee, A. G.; Teller, S.; and Roy, N. 2011. Understanding natural language commands for robotic navigation and mobile manipulation. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

[Winograd 1971] Winograd, T. 1971. Procedures as a representation for data in a computer program for understanding natural language. Technical report, MASSACHUSETTS INST OF TECH CAMBRIDGE PROJECT MAC.

[Zhang et al. 2018] Zhang, R.; Isola, P.; Efros, A. A.; Shechtman, E.; and Wang, O. 2018. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 586–595.