Autonomous Robots (2023) 47:1013–1033
https://doi.org/10.1007/s10514-023-10129-1

Learning modular language-conditioned robot policies through attention

Yifan Zhou[1] · Shubham Sonawani[1] · Mariano Phielipp[2] · Heni Ben Amor[1] · Simon Stepputtis[3]

Received: 2 May 2023 / Accepted: 26 July 2023 / Published online: 30 August 2023

Abstract
Training language-conditioned policies is typically time-consuming and resource-intensive. Additionally, the resulting controllers are tailored to the specific robot they were trained on, making it difficult to transfer them to other robots with different dynamics. To address these challenges, we propose a new approach called Hierarchical Modularity, which enables more efficient training and subsequent transfer of such policies across different types of robots. The approach incorporates Supervised Attention which bridges the gap between modular and end-to-end learning by enabling the re-use of functional building blocks. In this contribution, we build upon our previous work, showcasing the extended utilities and improved performance by expanding the hierarchy to include new tasks and introducing an automated pipeline for synthesizing a large quantity of novel objects. We demonstrate the effectiveness of this approach through extensive simulated and real-world robot manipulation experiments.

Keywords Language-conditioned learning · Attention · Imitation · Modularity

1 Introduction

The word *robot* was introduced and popularized in the Czech play, "Rossum's Universal Robots", also known as R.U.R. In this seminal piece of theatre, robots understand and carry out a variety of verbal human instructions. Roboticists and AI researchers have long been striving to create machines with such an ability to turn natural language instructions into physical actions in the real world (Jang et al., 2022; Stepputtis et al., 2020; Lynch and Sermanet, 2021; Ahn et al., 2022; Shridhar et al., 2021). However, this task requires robots to interpret instructions in the current situational and behavioral context in order to accurately reflect the intentions of the human partner. Achieving such inference and decision-making capabilities demands a deep integration of multiple data modalities—specifically, the intersection of vision, language, and motion. Language-conditioned imitation learning (Lynch and Sermanet, 2021; Stepputtis et al., 2020) is a technique that can help address these challenges by jointly learning perception, language understanding, and control in an end-to-end fashion.

However, a significant drawback of this approach is that, once trained, these language-conditioned policies are only applicable to the specific robot they were trained on. This is because end-to-end policies are monolithic in nature, which means that robot-specific aspects of the task, such as kinematic structure or visual appearance, cannot be individually targeted and adjusted. While it is possible to retrain the policy on a new robot, this comes with the risk of catastrophic forgetting and substantial computational overhead. Similarly, adding a new aspect, behaviors, or elements to the task may also require a complete retraining.

This paper tackles the problem of creating modular language-conditioned robot policies that can be re-structured,

✉ Yifan Zhou
yzhou298@asu.edu

Shubham Sonawani
sdsonawa@asu.edu

Mariano Phielipp
mariano.j.phielipp@intel.com

Heni Ben Amor
hbenamor@asu.edu

Simon Stepputtis
stepputtis@cmu.edu

[1] School of Computing and Augmented Intelligence, Arizona State University, Tempe, AZ, USA

[2] Intel AI, Phoenix, AZ, USA

[3] The Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA

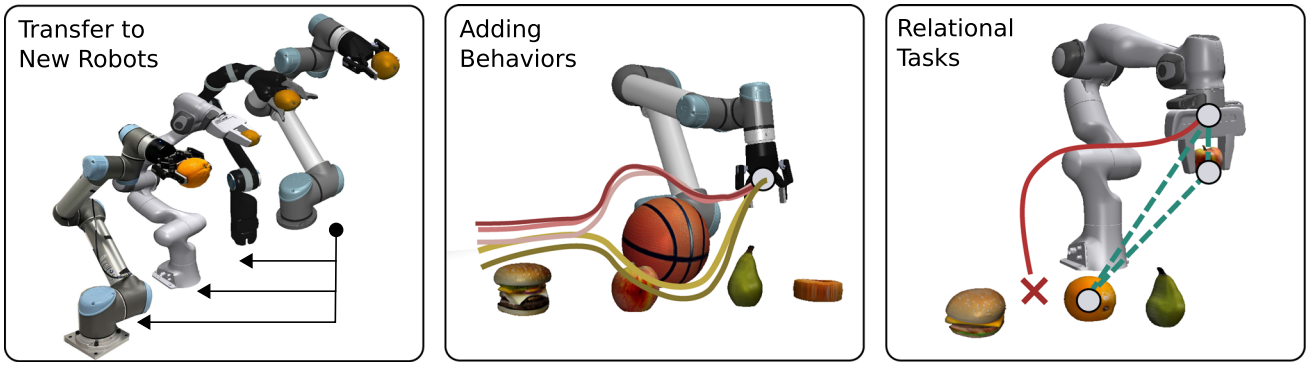


Fig. 1 Our proposed method demonstrates high performance on a variety of tasks. It is able to transfer to new robots in a data-efficient manner, while still keeping a high execution performance. It also accepts adding new behaviors to an existing trained policy. Besides them, we also demonstrate the ability to learn relational tasks, where there are two objects involved in the same sentence

extended and selectively retrained. Figure 1, depicts a set of scenarios that we want to address in this paper. For example, we envision an approach which allows for the efficient repurposing and transfer of a policy to a new robot. We also envision situations in which a new behavior may be added to an existing policy, e.g., incorporating obstacle avoidance into an existing motion primitive. Similarly, we envision situations in which the type of behavior is changed by incorporating additional modules into a policy, e.g., following human instructions that define a relationship between multiple objects, such as, *"Put the apple left of the orange!"*.

However, the considered modularity is at odds with the monolithic nature of end-to-end deep learning. To overcome this challenge, the paper proposes an attention-based methodology for learning reusable building blocks, or modules, that realize specialized sub-tasks. In particular, we discuss supervised attention which allows the user to guide the training process by focusing the attention of a sub-network (or module) on certain input–output variables. By imposing a specific locus of attention, individual sub-modules can be guided to realize an intended target functionality. Another contribution, called hierarchical modularity, is a training regime inspired by curriculum learning that aims to decompose the overall learning process into individual subtasks. This approach enables neural networks to be trained in a structured fashion, maintaining a degree of modularity and compositionality.

Our contributions can be summarize and extend our prior work in Zhou et al. (2022) as follows: (1) we propose a sample-efficient approach for training language-conditioned manipulation policies that allows for rapid transfer across different types of robots; (2) we introduce a novel method, which is based on two components called hierarchical modularity and supervised attention, that bridges the divide between modular and end-to-end learning and enables the reuse of functional building blocks; (3) we demonstrate that our method outperforms the current state-of-the-art methods [BC-Z (Jang et al., 2022) and LP (Stepputtis et al., 2020)]; (4) we extend the methodology by creating more complex tasks that incorporate obstacle avoidance and relational instruction following. Finally, we also perform an extensive number of experiments that shed light on generalization properties of the our methodology from different angles, e.g., dealing with occlusions, synonyms, variable objects, etc (Fig. 1).

2 Preamble: how generative AI helped write the paper

This paper largely centers around the training of generative models at the intersection of vision, language and robot control. Besides being the topic of the paper, generative models have also been instrumental in writing this paper. In particular, we incorporated such techniques into both (a) the text editing process when writing the manuscript, as well as (b) the process of generating 3D models and textures of manipulated objects.

For text editing, we utilized GPT-4 (OpenAI, 2023) to iteratively revise and refine our initial drafts, ensuring improved readability and clarity of the concepts discussed. We achieved this by conducting prompt engineering and formulating a specific prompt as follows:

"Now you are a professor at a top university, studying computer science, robotics and artificial intelligence. Could you please help me rewrite the following text so that it is of high quality, clear, easy to read, well written and can be published in a top level journal? Some of the paragraphs might lack critical information. If you notice that, could you please let me know? Let's do back and forth discussions on the writing and refine the writing."

We initiate each conversation with this impersonation prompt, followed by our draft text. GPT-4 then returns a revised version of the text, ensuring the semantics remained

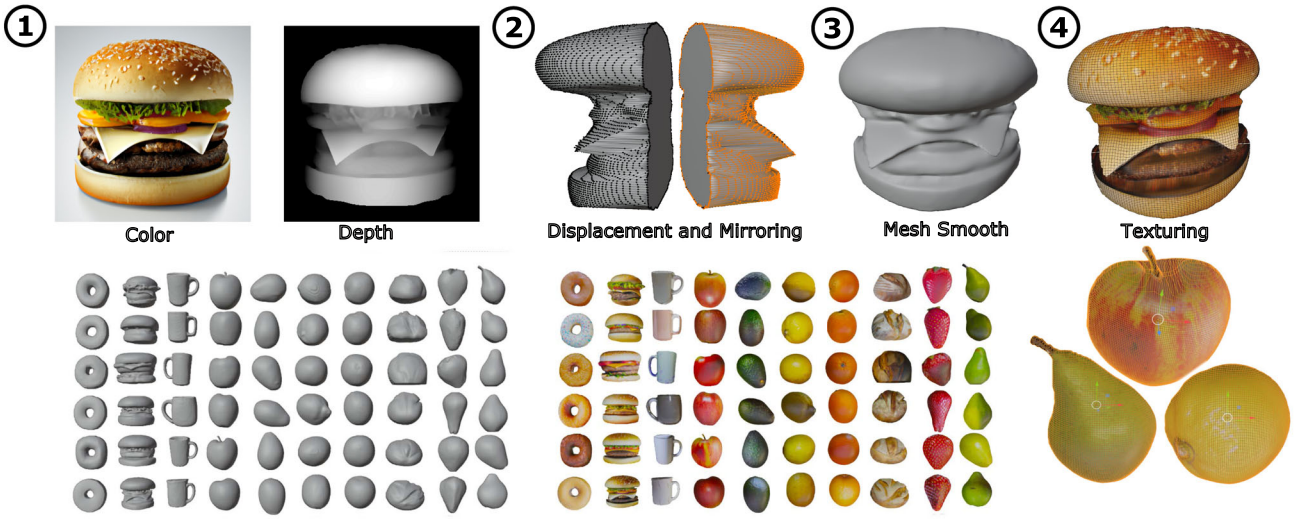


Fig. 2 Using generative models to automatically synthesize an unlimited set of 3D models

unaltered while updating the literary style to incorporate professional terminology and wording, as well as a clear logical flow. This prompt also encourages GPT-4 to solicit feedback on the revised text, thus facilitating back-and-forth conversations. We manually determine when a piece of writing has been fine-tuned to a satisfactory degree and bring the conversation to a close.

With regard to the generation of 3D models and assets, we created a new pipeline for automated synthesis of complete polygonal meshes. Figure 2(top row) depicts the individual steps of this process. First, we synthesize an image of the intended asset using latent diffusion models (Rombach et al., 2022) to produce an image of the required asset. We provide as input to the model a textual description of the asset, e.g., *"A front image of an apple and a white background."*. In turn, the resulting image is fed into a monocular depth-estimation algorithm (Ranftl et al., 2022) to generate the corresponding depth map. At this stage, each pixel in the image has both (1) a corresponding depth value and (2) an associated RGB texture value. To generate a 3D object, we take a flat mesh grid of the same resolution as the synthesized RGB image. We then perform displacement mapping (Zirr and Ritschel, 2019) based on the values present in the depth image. Within this process, each point of the originally flat grid gets elevated or depressed according to its depth value. The result is a 3D model representing the front half of the target object. For the sake of this paper, we assume a plane symmetry—a feature that is common among a large number of household objects. Accordingly, we can mirror the displacement map in order to yield the occluded part of the object. Finally, we also apply a Laplacian smoothing operation (Sorkine et al., 2004) on the final object. Texturing information is retrieved from the source image. This automated 3D synthesis process allows us to rapidly generate a potentially infinite number of variants of an object. This is particularly useful when studying the generalization capabilities of a model. It also completely removes any 3D modeling or texturing burden. At the moment, the pipeline is limited to symmetric objects.

3 Related work

Imitation learning offers a straightforward and efficient method for learning agent actions based on expert demonstrations (Dillmann and Friedrich, 1996; Schaal, 1999; Argall et al., 2009). This approach has proven effective in diverse tasks including helicopter flight (Coates et al., 2009), robot control (Maeda et al., 2014), and collaborative assembly. Recent advancements in deep learning have enabled the acquisition of high-dimensional inputs, such as vision and language data (Duan et al., 2017a; Zhang et al., 2018a; Xie et al., 2020)—partially stemming from improvements in image and video understanding domains (Lu et al., 2019; Kamath et al., 2021; Chen et al., 2020; Tan and Bansal, 2019; Radford et al., 2021; Dosovitskiy et al., 2021), but also in language comprehension (Wang et al., 2022; Ouyang et al., 2022). Specifically, the work presented in Radford et al. (2021) paved the way for multimodal language and vision alignment. The generalizability of such large multimodal models (Singh et al., 2022; Alayrac et al., 2022; Ouyang et al., 2022; Zhu et al., 2023) enables a variety of downstream tasks, including image captioning (Laina et al., 2019; Vinyals et al., 2015; Xu et al., 2015), visual question answering systems (VQA) (Antol et al., 2015; Johnson et al., 2017), and multimodal dialog systems (Kottur et al., 2018; Das et al., 2017). However, most importantly, these

models have shown their utility when learning language-conditioned robot policies (Shridhar et al., 2021; Nair et al., 2022) that conduct a variety of manipulation tasks (Lynch and Sermanet, 2021; Stepputtis et al., 2020; Jang et al., 2022). Utilizing multimodal inputs for task specification and robot control (Anderson et al., 2019; Kuo et al., 2020; Rahmatizadeh et al., 2018; Duan et al., 2017b; Zhang et al., 2018b; Abolghasemi et al., 2019; Mees et al., 2022) plays a crucial role, as the environment and verbal instruction needs to be grounded across modalities. Most notably, BC-Z (Jang et al., 2022) proposes a large multimodal dataset which is trained via imitation learning in order to complete a variety of diverse household tasks. Similar in spirit, LanguagePolicies (LP) (Stepputtis et al., 2020) learns a language-conditioned policy to comprehend commands that describes what, where and how to do a task, but describes the outputs of the policy in terms of a dynamic motor primitive (DMP) (Schaal, 2006). Going beyond single instruction following, SayCan (Ahn et al., 2022) focuses on planning of longer horizon tasks and incorporates prompt engineering. Most recently, even large language models have achieved impressive performance on embodied agents (Vemprala et al., 2023), with a push to generally capable agents that can play Atari, caption images, chat, and stack blocks with a real robot arm (Reed et al., 2022).

While these model achieve impressive performance, they usually require large quantities of data and are mostly "black box" approaches that do not lend themselves well to human interpretation in case the policy behavior is not performing as desired. A potential solution to this problem that retains the end-to-end training benefits of deep learning is the utilization of a modularized approach, allowing the creation of entire policies from a set of modules that can afford additional insights into the inference process of the neural network. Such modularization can be achieved by introducing auxiliary tasks that have shown to improve policy performance (Huang et al., 2022). Recent works on modularity investigate the question of whether "modules implementing specific functionality emerge" in neural networks automatically (Csordás et al., 2021; Filan et al., 2020). However, in contrast to these emergent modularity approaches, our prior work (Zhou et al., 2022) introduced supervised attention, together with a hierarchical learning regime akin to curriculum learning. Originating in machine translation (Liu et al., 2016), supervised attention and hierarchical modularity allow for such functional modules to be implemented in a top-down manner. In this work, we delve deeper into the benefits of this approach by investigating how it can be extended to more complex tasks including obstacle avoidance and instructions utilizing referential expressions across tasks that utilize a large qunatity of automatically generated scene objects.

4 Methodology

In this section, we present our approach for modularity in language-conditioned robot policies. The main objective of the approach is to build neural networks out of composable building blocks which can be reused, retrained and repurposed whenever changes to the underlying task occur. A distinguishing feature of our approach is its modular training, while maintaining end-to-end learning benefits. In particular, the shift from training individual components to training the complete network occurs progressively, yet modules can be trained quickly without requiring gradient propagation throughout the entire network. Owing to its modularity, $\pi_{\boldsymbol{\theta}}$ can be transferred to a new robot in a sample-efficient manner. The modular nature of the resulting neural networks also enables easy introspection into the intermediate computation steps at runtime.

The introduced methodology builds upon two essential components, namely supervised attention and hierarchical modularity—two ingredients that are used in conjunction to crystallize individual modules within an end-to-end deep learning system. Subsequently, we first introduce the problem statement underlying language-conditioned imitation learning. Thereafter, a detailed description of the training process is provided. Initially, we focus on efficient training of language-conditioned policies that can be transferred across a variety of robots. Thereafter, we shift our focus to the question of how new modules can be incorporated or how multiple modules can be interrelated.

4.1 Problem statement

In Language-Conditioned Imitation Learning (Lynch and Sermanet, 2021; Stepputtis et al., 2020), the goal is to learn a policy $\pi_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}, \boldsymbol{I})$ that can execute a human instruction while taking into account situational and environmental conditions. The result of the learning process is a deep neural network parameterized by weight vector $\boldsymbol{\theta}$. Input $\boldsymbol{s}$ is a verbal task instruction provided by a human whereas $\boldsymbol{I}$ is an image captured by an RGB camera mounted on the robot. Throughout this paper, policy $\pi_{\boldsymbol{\theta}}$ is trained to generate an action $\boldsymbol{a} \in \mathbb{R}^7$ containing the Cartesian position (x, y, z) and orientation (r, p, y) for the robot end-effector, as well as a binary label $g = \{open, closed\}$ indicating the gripper state. Policies are trained following the imitation learning paradigm from a dataset $\mathcal{D} = \{\boldsymbol{d}_0, \ldots, \boldsymbol{d}_N\}$ of N expert demonstrations and corresponding verbal commands. In this dataset, each demonstration $\boldsymbol{d}_n$ represents a sequence with T steps $((\boldsymbol{a}_0, \boldsymbol{s}_0, \boldsymbol{I}_0), \ldots, (\boldsymbol{a}_T, \boldsymbol{s}_T, \boldsymbol{I}_T))$. Each step in demonstration $\boldsymbol{d}_n$ is defined as a tuple $(\boldsymbol{a}_t, \boldsymbol{s}_t, \boldsymbol{I}_t)$ containing the action, language command, and image at time step t. Upon completion of training, the policy is expected to execute novel configurations of the task.

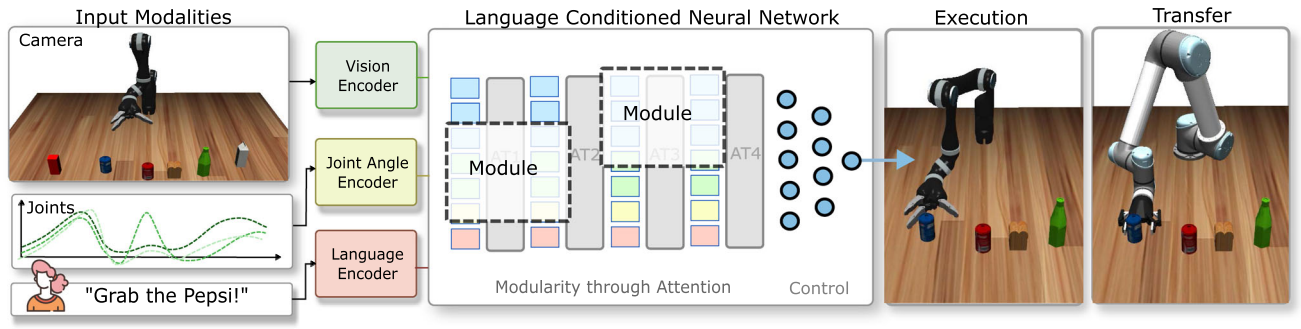


Fig. 3 Overview: different input modalities, i.e., vision, joint angles and language are fed into a language-conditioned neural network to produce robot control values. The network is setup and trained in a modular fashion—individual modules address sub-aspects of the task. The neural network can efficiently be trained and transferred onto other robots and environments (e.g. Sim2Real)

4.2 Training modular language-conditioned policies

Our overall method is illustrated in Fig. 3. First, camera image $\boldsymbol{I}$ is processed together with a natural language instruction $\boldsymbol{s}$ and the robot's proprioceptive data (i.e., joint angles) through modality-specific encoders to generate their respective embeddings. The resulting embeddings are subsequently supplied as input tokens to a transformer-style (Vaswani et al., 2017) neural network consisting of multiple attention layers. This neural network is responsible for implementing the overall policy $\pi_{\boldsymbol{\theta}}$ and produces the final robot control signals.

The encoding process ensures that distinct input modalities, e.g., language, vision and motion, can effectively be integrated within a single model. To that end, *Vision Encodings* $\boldsymbol{e_I} = f_{\mathcal{V}}(\boldsymbol{I})$ are generated using an input image $\boldsymbol{I} \in \mathbb{R}^{H \times W \times 3}$. Taking inspiration from (Carion et al., 2020; Locatello et al., 2020), we maintain the original spatial structure while encoding the image into a sequence of lower-resolution image tokens. The resolution is reduced via a convolutional neural network while increasing the number of channels, yielding $\boldsymbol{eI} \in \mathbb{R}^{(H/s)\times(W/s)\times d}$, with s representing a scaling factor and d denoting the embedding size. Consequently, the low-resolution pixel tokens are transformed into a sequence of tokens $\boldsymbol{eI} \in \mathbb{R}^{Z\times d}$, where $Z = (H \times W)/s^2$ through a flattening operation.

By contrast, *Language Encodings* $\boldsymbol{e_s} = f_{\mathcal{L}}(\boldsymbol{s}) \in \mathbb{R}^{1\times d}$ are produced via a pre-trained and fine-tuned CLIP (Radford et al., 2021) model. Particularly, each instruction $\boldsymbol{s}$ is represented as a sequence of words $[w_0, w_1, \ldots, w_n]$ in which each word $w_i \in \mathcal{W}$ is a member of vocabulary $\mathcal{W}$. During training, we employ automatically generated, well-formed sentences; however, after training, we allow any free-form verbal instruction that is presented to the model, including sentences affected by typos or bad grammar. Finally, *Joint Encodings* $\boldsymbol{e_j} = f_{\mathcal{J}}(\boldsymbol{a}) \in \mathbb{R}^{1\times d}$ are created by transforming the current robot state $\boldsymbol{a}$ into a latent representation using a simple multi-layer perceptron. The main purpose of this step is to transform the joint representation into a compatible shape that aligns with the other input embeddings.

4.2.1 Supervised attention

After encoding, the inputs are processed within a single neural network in order to produce robot control actions. However, a unique element of our approach is the formation of semantically meaningful sub-modules during the learning process. These modules may solve a specific sub-task, e.g., detecting the robot end-effector or calculating the distance between the robot and the target object. To achieve this effect, we build upon modern attention mechanisms (Vaswani et al., 2017) in order to manage the flow of information within attention layers, thereby explicitly guiding the network to concentrate on essential inputs.

More specifically, we adopt a *supervised attention* mechanism in order to enable user-defined information routing and the formation of modules within an end-to-end neural network. The main idea underlying this mechanism is that information about optimal token pairings may be available to the user. In other words, if we know which key tokens are important for the queries to look at, we can treat their similarity score as a maximization goal. In Fig. 4, we see the information routing for three modules. The first module LANG is supposed to identify the target object within the sentence. Hence, the corresponding attention layer is trained to only focus on the language input. The attention for the robot joint values and vision input is trained to be zero. In order to provide the output of this module to the attention layer in the next level, we use so-called *register slots*. Register slots are used to store the output of a module so that it can be accessed in subsequent modules in the hierarchy. Accordingly, each module within our method has corresponding register slot tokens. The role of the register slots is to provide access to the output of previously executed modules within the hierarchy. Coming back to Fig. 4, the second module EE2D locates

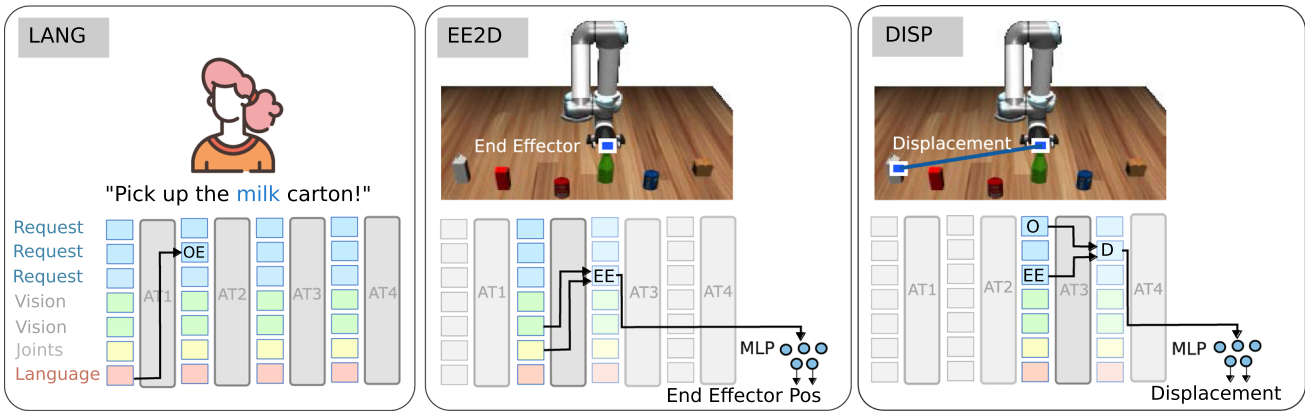


Fig. 4 Different sub-aspects of the tasks are implemented as modules (via supervised attention). LANG identifies the target object. EE2D locates the robot end-effector

the robot end-effector in the image. Accordingly, the attention for this module is trained such that the focus is on the vision and language inputs only. In turn, the result is written into the corresponding register slot. The final module in Fig. 4, DISP, calculates the distance between the end effector (EE) and the target object (O). Since this module is higher up in the hierarchy, it accesses the register slots of lower-level modules as inputs in order to calculate the distance.

Registers serve multiple purposes and can either be used as inputs to a module, in which case they serve as a learnable latent embedding, or be used to store the output of a particular module. An output register of a module is calculated by utilizing the standard transformer architecture. In particular, we define a transformer based attention module over queries ($\boldsymbol{Q}$), keys ($\boldsymbol{K}$), and values ($\boldsymbol{V}$), which are subsequently processed as follows:

$$\boldsymbol{r}_{\text{out}} = \text{Attn.}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right)\boldsymbol{V} \tag{1}$$

where d_k is the dimensionality of the keys. In our use case, the queries are initialized with either learnable and previously unused register slots, or with registers that have been set by modules operating in prior layers, thus encoding their respective results. Our keys are equivalent to the values and are initialized with all formal inputs (language, vision, and joint embeddings) as well as all previously set registers from prior layers. In contrast to common practice, we control the information flow when learning each module via our proposed *supervised attention*, which is a specific optimization target for attention layers.

As an illustrative example, consider a query identifying the location of the end-effector, as demonstrated in Fig. 5 (first key and query combination in the top left) or finding the target object (key and query combination near the center). For simplicity, we omit the other formal inputs and only focus on the visual input. However, the `Tar Reg.` would also

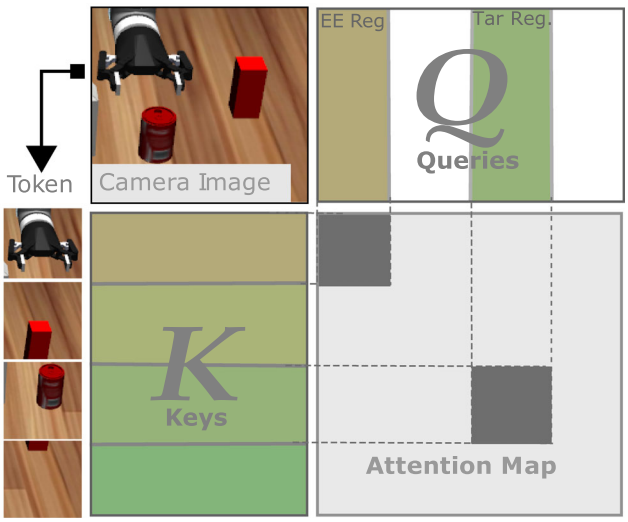


Fig. 5 Supervised attention example for the second layer of processing information throughout our overall policy

depend on the language register from the prior LANG module. Following common practice, the keys and values derive from the input image, with each image embedding vector corresponding to an image patch (Fig. 5 left). In this particular example, the EE uses a trainable, previously unused register as query, while the Tar register utilizes the output register of the language module to find the correct object (Fig. 5 top). The EE register is supervised to focus on the robot's gripper image patch, thereby creating a sub-module for detecting the robot end-effector. Similarly, the target register attends to the target object's image patch, forming a sub-module responsible for identifying the target object. When these queries accurately attend to their respective patches, these patches will primarily contribute to the output register's embedding vector, which can then be used as subsequent module inputs.

More formally, we maximize the similarity between query $\boldsymbol{q}_i$ and key $\boldsymbol{k}_j$ if a connection should exist, thus optimizing

Table 1 Explanation of the various modules utilized in our hierarchical attention module

Module	Layer	Formulation, supervised attention mask, and explanation
LANG	1	$\boldsymbol{r}_{LANG} = f_{\text{LANG}}(\boldsymbol{r}_0, \{\mathbf{e}_s, \mathbf{e}_j, \mathbf{e}_I\})$
		Supervised to connect $\boldsymbol{r}_1 \leftrightarrow \mathbf{e}_s$
		Identify the target object in the user's language
EE2D	2	$\boldsymbol{r}_{EE2D} = f_{\text{EE2D}}(\boldsymbol{r}_1, \{\mathbf{e}_s, \mathbf{e}_j, \mathbf{e}_I, \boldsymbol{r}_{LANG}\})$
		Supervised to connect $\boldsymbol{r}_2 \leftrightarrow \mathbf{e}_I$
		Identify the robot's end-effector in the image patches
TAR2D	2	$\boldsymbol{r}_{TAR2D} = f_{\text{TAR2D}}(\boldsymbol{r}_2, \{\mathbf{e}_s, \mathbf{e}_j, \mathbf{e}_I, \boldsymbol{r}_{LANG}\})$
		Supervised to connect $\boldsymbol{r}_{LANG} \leftrightarrow \mathbf{e}_I$
		Identify visual representation of the target object in the image
EE3D	3	$\boldsymbol{r}_{EE3D} = f_{\text{TAR2D}}(\boldsymbol{r}_3, \{\mathbf{e}_s, \mathbf{e}_j, \mathbf{e}_I, \boldsymbol{r}_{LANG}, \boldsymbol{r}_{EE2D}, \boldsymbol{r}_{TAR2D}\})$
		Supervised to connect $\boldsymbol{r}_3 \leftrightarrow \mathbf{e}_j, \mathbf{r}_{EE2D}$
		Similar to forward kinematics, calculates the end-effector's 3D position
TAR3D	3	$\boldsymbol{r}_{TAR3D} = f_{\text{TAR3D}}(\boldsymbol{r}_4, \{\mathbf{e}_s, \mathbf{e}_j, \mathbf{e}_I, \boldsymbol{r}_{LANG}, \boldsymbol{r}_{EE2D}, \boldsymbol{r}_{TAR2D}\})$
		Supervised to connect $\boldsymbol{r}_4 \leftrightarrow \boldsymbol{r}_{TAR2D}$
		Identify the robot's end-effector in 3D space
DISP	4	$\boldsymbol{r}_{DISP} = f_{\text{DISP}}(\boldsymbol{r}_5, \{\mathbf{e}_s, \mathbf{e}_j, \mathbf{e}_I, \boldsymbol{r}_{LANG}, \boldsymbol{r}_{EE2D}, \boldsymbol{r}_{EE3D}, \boldsymbol{r}_{TAR2D}, \boldsymbol{r}_{TAR3D}\})$
		Supervised to connect $\boldsymbol{r}_5 \leftrightarrow \boldsymbol{r}_{EE3D}, \boldsymbol{r}_{TAR2D}$
		Calculate the displacement between the end-effector and the target object
CTRL	5	$\boldsymbol{r}_{CTRL} = f_{\text{CTRL}}(\boldsymbol{r}_6, \{\mathbf{e}_s, \mathbf{e}_j, \mathbf{e}_I, \boldsymbol{r}_{LANG}, \boldsymbol{r}_{EE2D}, \boldsymbol{r}_{EE3D}, \boldsymbol{r}_{TAR2D}, \boldsymbol{r}_{TAR3D}, \boldsymbol{r}_{DISP}\})$
		Supervised to connect $\boldsymbol{r}_5 \leftrightarrow \boldsymbol{r}_{EE3D}, \boldsymbol{r}_{TAR3D}, \boldsymbol{r}_{DISP}, \boldsymbol{e}_s, \boldsymbol{e}_j$
		Calculates the controll signal for the robot

The bold "**e**" refer to the embeddings as introduced in Sec. 4.2. the bold "**r**" refers to the registers as introduced in Equation 1

$\arg\max_{\boldsymbol{\theta}} \boldsymbol{q}_i \boldsymbol{k}_j^T$. This process is equivalent to maximizing the corresponding attention map element $\boldsymbol{M}_{ij}$, where $\boldsymbol{M}_i = \text{softmax}(\frac{\boldsymbol{Q}_i \boldsymbol{K}^T}{\sqrt{d_k}})$. Since each element $\boldsymbol{M}_{ij} < 1$, we minimize the distance between $\boldsymbol{M}_{ij}$ and 1 according to Eq. 2. We assume that N supervision pairs are provided in a set $\mathcal{S}$, indicating the query and key tokens that should pay attention to each other. Each pair $(i, j) \in \mathcal{S}$ contains the indices defining which queries $\boldsymbol{q}_i$ should attend to which corresponding keys $\boldsymbol{k}_j$. Individual supervision pairs in this set can be addressed by $\mathcal{S}(p) = (i_p, j_p)$. We then define the cost function for supervised attention as follows:

$$\mathcal{L}(\mathcal{S}) = \sum_{n=0}^{N} \left(\text{softmax}\left(\frac{\boldsymbol{q}_r \boldsymbol{k}_s^T}{\sqrt{d_k}} \right) - 1 \right)^2 \tag{2}$$

where (r, s) correspond to the indices held by the n-th supervision pair $(r, s) = \mathcal{S}(n)$. While Eq. 2 defines the loss as a minimization problem with a mean squared error loss, other cost functions such as the cross-entropy (de Boer et al., 2004) can also be applied, but have empirically resulted in lower performance.

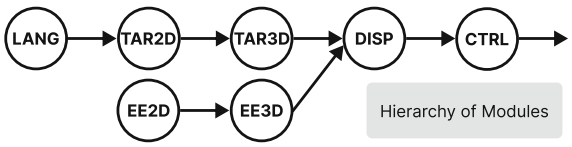


Fig. 6 Hierarchy of the modules used in our method

4.2.2 Hierarchical modularity

In this section, we describe hierarchical modularity—an algorithm for training hierarchies of modules which is inspired by curriculum learning (Bengio et al., 2009). The previously introduced supervised attention mechanism enables the training of modules or building blocks relevant to the task. However, such modules also have to be stacked and cascaded together in other to realize the overall goal of the policy. In that sense, one module's output becomes the subsequent module's input. This can be represented as a directed graph, as shown in Fig. 6 (top), in which a cascade of specialized modules implements the overall control policy. Here, each module is represented by a node, while edges represent the information flow between nodes.

Table 1 formally defines each of the modules (nodes) by introducing their functionality, queries, keys, and supervised attention mask. Broadly speaking, each module follows

Algorithm 1 Hierachical Modularity: training algorithm returns network weights θ.

Input: $(\mathcal{D}, \{\mathcal{S}_k\}_{k=1}^K, \{\mathcal{L}_k\}_{k=1}^K, \{\Psi_k\}_{k=1}^K)$
Output: Weights θ
for subtask $k \leftarrow 1$ to K **do**
  while not converged **do**
    $E_k \leftarrow \sum_{t=0}^{k} \mathcal{L}_t(\mathcal{S}_t) + \Psi_t$
    $\theta \leftarrow \text{Train}(\mathcal{D}, \{\mathcal{S}_1, \ldots, \mathcal{S}_k\}, E_k)$
  end while
end for
return θ

equation Eq. 1 with keys being the set of original sensor modalities, as well as registers. In the first layer of Fig. 6, the LANG module identifies the target object, as referred to in the verbal command, and stores the result in the $\boldsymbol{r}_{\text{LANG}}$ register. Subsequently, in the second layer, the f_{TAR2D} module utilizes the $\boldsymbol{r}_{\text{LANG}}$ register as a query while the f_{EE2D} module utilizes a new, previously unused register as a query. This chain continues until the final control output of the robot is generated in the CTRL module.

Recall that sub-modules address intermediate tasks in the overarching control problem, making the output register $\boldsymbol{r}$ suitable for human interpretation and allowing for supervised training of the resulting embedding. To achieve this, we employ small multi-layer perceptron (MLP) decoders to convert the module outputs into their respective numeric outputs. For example, we train a small MLP on top of the $\boldsymbol{r}_{\text{EE2D}}$ register that predicts the end-effector location (ee_x, ee_y) via a single linear transformation. This approach enables our policy to predict intermediate module outputs, enhancing training accuracy and allowing monitoring and debugging during inference, which is particularly valuable when transferring the policy to different robots or scenarios.

Training Cascaded Modules

Intuitively, the cascaded modules can be trained in a manner inspired by curriculum learning, wherein each component is trained before further layers of the hierarchy are added to the training objective. This ensures that each module is trained until convergence before being employed for more sophisticated decision-making processes, ultimately leading to the prediction of robot control parameters. Algorithm 1 outlines the training procedure for our hierarchical approach in further detail. The algorithm trains each module of the hierarchy one after another, until the currently trained module is converged according to its respective loss function. After that, we progressively incorporate additional modules in a manner reminiscent of curriculum learning. Each module k is trained with an attention loss $\mathcal{L}_k$ given the supervision signal S of our proposed supervised attention approach, as well as a task-specific loss functions Ψ_k which trains the MLP decoder for every module. Thus, each module is optimized with regard to two targets. Note that the policy loss for the robot controller CTRL is also implemented as an MLP decoder, which also represents the overall prediction target of our training process. Notably, in our scenario, this decoder predicts the next ten goal positions at each timestep instead of predicting only the next action. This choice is inspired by Jang et al. (2022), which also allows for a fair comparison in the subsequent evaluation sections.

While the modular approach requires manually defining loss terms for each module, it is essential to note that all modules form a single overarching neural network implementing the robot policy, inherently learning necessary features in an end-to-end manner. Modularization arises solely from training the network with various supervised attention targets and a cost function that successively integrates more sub-tasks.

4.3 Use-cases and extensions of hierarchy

We present our model as a cascade of sub-modules, trained hierarchically, enabling seamless integration of additional modules. In this section, we discuss the incorporation of obstacle avoidance, tracking a predefined obstacle, and describe the generalization of this approach to arbitrary "referential objects" that let users specify commands that reference any other object. These enhancements are implemented by introducing new modules, as depicted in Fig. 8.

4.3.1 Runtime introspection

All sub-modules retain their functionality, even after training. Consequently, they can be used at runtime to query individual outputs (e.g., LANG, TAR2D, EE3D). This feature allows users to monitor the intermediate computations of the end-to-end network to identify potential deviations and misclassifications. Figure 7 visually depicts the outputs of each model during the execution on a real robot. A textual description (left upper corner) shows the currently identified object name, as well as the displacement (in cm) between the end-effector and the target object. The current attention map is visualized in yellow, whereas the end-effector position and the target position are highlighted by red and blue points. Computing these intermediate outputs of the network generates negligible to no computation overhead. In our specific system, we implemented a real-time visualization tool that can be used at all times to monitor the above features. Such tools for introspection can help in debugging and troubleshooting of the language-conditioned policy. For example, they can be used to detect when individual modules need to be retrained, or where in the hierarchy a problem is manifesting. In addition, such outputs can be used with formal runtime monitoring and verification systems, e.g., Yamaguchi and Fainekos (2021) and Pettersson (2005), to improve the safety of the neural network policy.

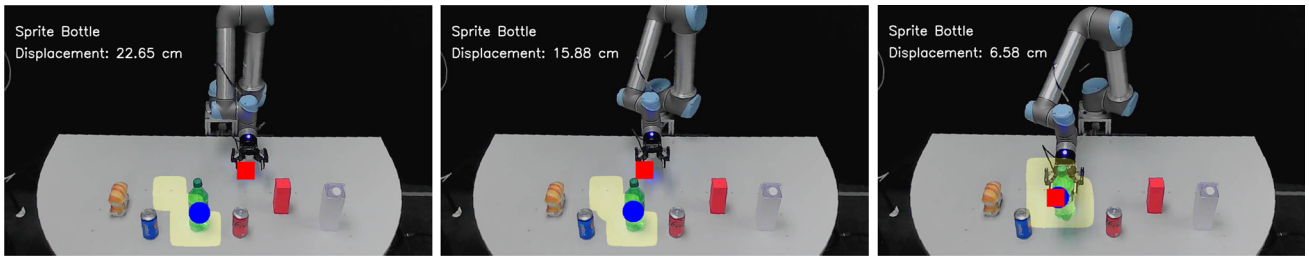


Fig. 7 Sequence of real-time outputs of the network modules: the object name (white) and visual attention (yellow region), the length of the displacement (white text), the object pos (blue), and end-effector pos (red). All values are generated from a single network that also produces robot controls (Color figure online)

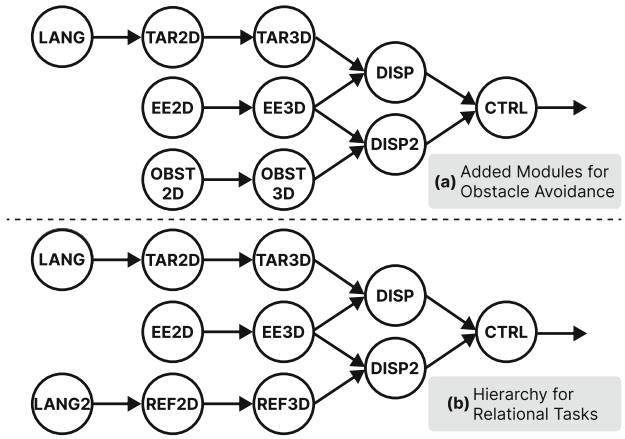


Fig. 8 Extensions of hierarchy. **a** The hierarchy used for obstacle avoidance. 3 new modules, OBST2D, OBST3D and DISP2 are plugged in post-training for detecting the obstacle and avoiding collision with it. **b** The hierarchy for the relational tasks. These tasks involve 2 objects in a sentence, e.g., "Put the apple right to the orange", where "orange" is the referral object. We add LANG2D, REF2D, REF3D and DISP2 for detecting the referral object and generating the according trajectory (Color figure online)

4.3.2 Adding new behaviors

An important benefit of the modular architecture of our approach is the ability to add new modules into a neural network, even after successful training. To demonstrate this functionality, we add an obstacle avoidance behavior into the system, i.e., the robot is expected to detect an obstacle and generate controls to avoid any collisions.

In our specific scenario, we introduce an obstacle in the form of an orange basketball that must be avoided when approaching the target object. To incorporate this ability into the existing system, we add new modules into the previous hierarchy. This can be seen in Fig. 8 (top).

In particular, we add OBST2D and OBST3D, which identify the obstacle's position, and DISP2, which computes the displacement between the end-effector and obstacle. Similar to target object detection, the obstacle is identified from object embeddings and ultimately results in a displacement value. The controller module incorporates the additional displacement as an additional input. In general, new modules can be added or existing modules removed according to the needs of the task.

4.3.3 Creating new types of behaviors by interconnecting modules

The modular approach also enables new types of behaviors to be incorporated; in particular, behaviors that interconnect multiple existing modules. For example, we may want to learn a robot policy that allows for relational queries, e.g., *"Put the coke can in front of the pepsi can."*. Such a feature would require the dynamic identification of a secondary object and its desired relationship to the target object. In the previous example, the model must infer the target object (*"coke can"*), the reference object (*"pepsi can"*), and their relation (*"in front of"*).

Figure 8 (bottom) shows the new hierarchy for this use case. Similar to the object avoidance case, we can incorporate additional modules REF2D, REF3D, and DISP2 for this purpose. However, in contrast to the obstacle avoidance case, an additional module LANG2 is added to extract the object reference from the user's instruction and subsequently informs the REF2D for further processing. This process of adding and removing modules allows for extensible language-conditioned policies whose complexity can be increased or reduced according to the necessities of the task. In the evaluation section, we will see that such an incremental approach has advantages over a complete retraining of the entire policy.

5 Evaluation

In this section, we present a set of experiments designed to evaluate various aspects of our approach. We firstly elaborate on the data collection process in Sect. 5.1. In Sect. 5.2, we investigating basic performance metrics of our approach

and compare them to other state-of-the-art methods. To this end, we carry out ablation studies in order to probe the impact of our hierarchical modularity and supervised attention modules, as well as structure of the hierarchy itself. Thereafter, we study the robustness of our approach when exposed to occlusions (Sect. 5.2.2) and linguistic variability (Sect. 5.2.3). In Sect. 5.3, we focus on the ability of our approach to transfer existing policies between different robots in simulation, but also demonstrate the transfer to real-world robots in a sample efficient manner. Section 5.4 examines the policy's ability to generalize to novel objects. Lastly, we explore the possibility of incorporating new modules into an existing hierarchy for the purposes of obstacle avoidance and relational instructions (Sect. 5.5).

We evaluate our method on a tabletop manipulation task of six Robosuite (Zhu et al., 2020) and up to 100 automatically generated objects across five different tasks. Our tasks include three basic objectives namely picking objects, pushing them across the table, and rotating them. Further, we have two obstacle avoidance task, focusing on a single object, or all non-target objects simultaniously. In addition, we also investigate a more complex placing task in which objects need to be placed in relation to other objects in the environment, thus requiring the understanding and correct interpretation of relational instructions. Tasks are performed on three different robots in simulation and one robot in the real world. Our simulated robots include a Franka Emika, Kinova Jaco, and Universal Robot UR5 compliant robot arm. In the real world scenario, we utilize a UR5 robot. The following sections will first provide the details of our experimental setup and data collection strategy and then discuss evaluation results.

Training Resources To train our method from scratch, a single Quadro P5000 GPU takes approximately 48 h until convergence. In conjunction with this paper, we will release our final code base (and dataset) which is capable of leveraging multi-GPU setups, thereby resulting in further speed-ups with regards to the absolute training time.

5.1 Data generation

We perform a series of simulated experiments in MuJoCo (Todorov et al., 2012), employing three distinct robotic platforms (Kinova, UR5, and Franka) that closely resemble our real-world experimental setup with a UR5 robot.

Figure 9 illustrates all four configurations, along with the six Robosuite objects utilized in our investigations, including a red cube, a Coke can, a Pepsi can, a milk carton, a green bottle, and a loaf of bread. Further, our comprehensive set of 100 procedurally generated objects is depicted in Fig. 3. Demonstrations are collected using a heuristic motion planner that orchestrates fundamental motion planning techniques to control each target robot. By contrast, real-world demonstrations are collected via kinesthetic teaching utilizing a gravity-compensated robot arm. Beyond the robots' motion, we store each action's respective command (e.g., *"Pick up the green bottle!"*) and the corresponding RGB video stream captured by an overhead camera from the same angle as shown in Fig. 9 with a resolution of 224×224 pixels.

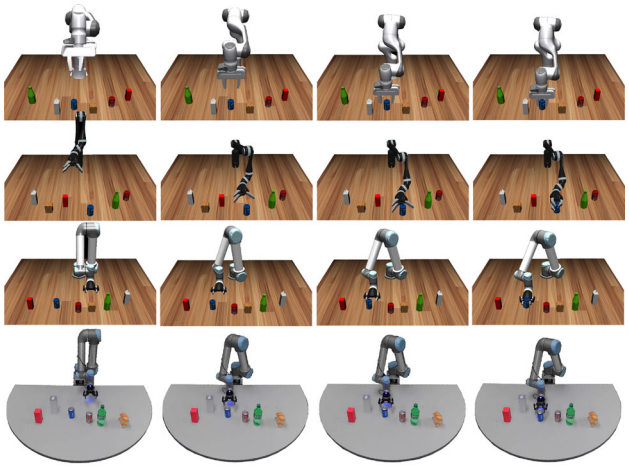


Fig. 9 A human instruction is turned into robot actions via a learned language-conditioned policy. The neural network is then successfully transferred to different robots in simulation and real-world

As a simple data augmentation technique, we utilize a templating system that generates syntactically correct sentences during the collection of training, validation, and testing data. These templates are derived from two human annotators who, after watching pre-recorded robot behavior videos, were assigned the task of providing instructions on what the robot was executing in the video. This small dataset served as the foundation for extracting command templates, as well as a collection of the used nouns, verbs, and adjectives. This collection is then extended with commonly available synonyms to allow the creation of an automated system for command generation during data collection. The template initially selects a random verb phrase in accordance with Table 10. Subsequently, a noun phrase is determined through random selections from Adj and Noun, as outlined in Table 9.

Table 2 presents the datasets utilized in our experimental setup. Each sample is collected with 125 Hz, resulting in trajectories containing 100–500 steps, depending on the distance between the robot's initial position and the target object, as well as the task being executed. The smaller datasets in rows three to five are for transferring a previously trained policy from one robot or task to another. The transfer learning datasets are purposefully over-provisioned, as we assess the minimal size required to achieve performance comparable to a policy trained from scratch in Sect. 5.3. Finally, the datasets in rows 4, 6, 7, 8, and 9 undergo evaluation in an interactive, live setting in which a user engages with a deployed policy, either within a simulation or the real world; thus, these datasets do not have a formal test split.

Table 2 Utilized dataset during our experiments

Dataset	Samples Train/val/test	Objects Robosuite	Gen.	Tasks Base	Place	Env.	Robot	Purpose
$\mathcal{D}^{\text{UR5}}$	1600/400/400	✓		✓		Simulation	UR5	Training from scratch
$\mathcal{D}^{\text{Kinova}}$	1600/400/400	✓		✓		Simulation	Kinova	Training from scratch
$\mathcal{D}^{\text{UR5}}_{\text{TF}}$	320/80/80	✓		✓		Simulation	UR5	Transfer Kinova → UR5
$\mathcal{D}^{\text{UR5}}_{\text{RW}}$	260/80/live	✓		✓		Real-World	UR5	Transfer UR5 → real-world
$\mathcal{D}^{\text{Franka}}_{\text{TF}}$	320/80/80	✓		✓		Simulation	Franka	Transfer Kinova → Franka
$\mathcal{D}^{\text{UR5}}_{\text{OBST}}$	200/400/live	✓		✓		Simulation	UR5	Extend skill obstacle-avoidance
$\mathcal{D}^{\text{UR5}}_{\text{M-OBST}}$	200/400/live	✓		✓		Simulation	UR5	Multiple-obstacle-avoidance
$\mathcal{D}^{\text{UR5}}_{\text{NO}}$	3000/600/live		✓	✓		Simulation	UR5	Training 100 novel objects
$\mathcal{D}^{\text{UR5}}_{\text{ET}}$	3000/600/live		✓		✓	Simulation	UR5	Training relational task

A "live" designation indicates that testing has been conducted interactively and no formal test dataset exists

Table 3 Comparison with the state-of-the-art baseline as well as ablations in Mujoco

Model	Success rate (%)				Prediction error (cm)		
	Pick	Push	Putdown	Overall	TAR3D	EE3D	DISP
LP	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	–	–	–
Vanilla attention	17.7 ± 1.8	8.3 ± 0.0	12.5 ± 4.2	13.3 ± 0.7	–	–	–
BC-Z	81.6 ± 6.2	85.6 ± 7.9	49.1 ± 5.8	73.1 ± 4.5	–	–	–
Ours (UR5 sim)	$\mathbf{91.3 \pm 5.3}$	$\mathbf{97.2 \pm 2.0}$	55.6 ± 8.6	$\mathbf{82.4 \pm 4.9}$	$\mathbf{2.24 \pm 0.48}$	$\mathbf{0.51 \pm 0.09}$	$\mathbf{2.42 \pm 0.52}$
Ours: no S. Attn	88.9 ± 4.2	92.6 ± 4.7	55.1 ± 11.6	79.9 ± 5.8	3.18 ± 1.80	$\mathbf{0.42 \pm 0.10}$	3.10 ± 1.64
Ours: no H. Mod	44.4 ± 3.8	39.4 ± 7.1	22.7 ± 7.9	36.4 ± 3.3	22.96 ± 0.99	0.59 ± 0.16	23.16 ± 1.05
Ours: TAR	87.5 ± 4.8	93.8 ± 2.3	$\mathbf{59.3 \pm 11.5}$	80.9 ± 6.0	2.77 ± 0.80	0.71 ± 0.08	3.24 ± 1.45
Ours: no TAR	20.8 ± 1.8	13.9 ± 2.4	8.3 ± 4.2	15.0 ± 1.3	32.23 ± 0.05	0.81 ± 0.01	32.29 ± 0.29
Ours: no DISP	65.6 ± 3.1	83.3 ± 4.2	33.3 ± 8.3	61.3 ± 3.3	3.37 ± 0.17	0.69 ± 0.01	27.49 ± 0.20
Ours: Extra	$\mathbf{94.8 \pm 1.8}$	$\mathbf{98.6 \pm 2.4}$	$\mathbf{62.5 \pm 7.2}$	$\mathbf{86.3 \pm 2.5}$	$\mathbf{1.53 \pm 0.14}$	0.68 ± 0.13	$\mathbf{2.25 \pm 0.22}$
Ours: Joint	10.9 ± 2.2	25.0 ± 5.9	2.1 ± 2.9	12.5 ± 3.5	6.19 ± 0.35	1.71 ± 0.01	10.83 ± 0.53

The bold numbers are the ones with top 2 performance within all methods

5.2 Model performance and baseline comparison

In this section, we evaluate our model on the three basic actions across the six Robosuite objects, utilizing the $\mathcal{D}^{\text{UR5}}$ dataset. We also compare our method to two state-of-the-art baselines, specifically BC-Z (Jang et al., 2022) and LP (Stepputtis et al., 2020). As our third baseline, we investigate vanilla, unsupervised attention. In this scenario, the same network as before is trained, but without supervision of the attention process as introduced in this paper.

Table 3 summarizes these results in which each training and testing procedure was executed three times to provide a better understanding of the stability of the compared methods. We evaluate not only the overall success rates but also the performance of each individual module within our language-conditioned policy. Specifically, we employ the following metrics: (1) **Success Rate** describes the percentage of successfully executed trials among the test set, (2) Target Object Position Error (**TAR3D**) measures the Euclidean 3D distance between the predicted target object position and the ground truth, (3) End Effector Position Error (**EE3D**) quantifies the Euclidean 3D distance between the predicted end effector position and the ground truth, (4) Displacement Error (**DISP**) calculates the 3D distance between the predicted 3D displacement vector and corresponding ground truth vector.

Our method (line 4) outperformed BC-Z (line 3) on all basic tasks with an average success rate of 82.4%, as compared to 73.1% for BC-Z. Furthermore, we separately assessed the prediction error of the proposed network's components, namely EE3D, TAR3D, and DISP. We note that the end-effector pose prediction accuracy (approximately 0.5 cm) surpasses the target object's accuracy, which could be attributed to the presence of the robot's joint state information. The target object's position estimation deviates by around 2–3, possibly due to the absence of depth information in our input dataset (solely consisting of RGB).

By contrast, the LP model (line 1) is not able to successfully complete any of the tasks. We hypothesize that this low performance is due to the training dataset's significantly smaller size compared to the LP's usual training data size,

as indicated by Stepputtis et al. (2020). Finally, the vanilla, unsupervised attention approach (line 2) achieves a success rate of 13.3%. Qualitatively, we observe in this scenario that the vanilla attention model is not able to recognize the correct object. Similarly to the LP approach, we hypothesize that the issue could potentially be resolved with a larger dataset. However, for the sake of a fair comparison within this paper, we utilize the same dataset $\mathcal{D}^{\text{UR5}}$ across all methods.

5.2.1 Ablations

In order to evaluate the impact of our two main contributions—supervised attention and hierarchical modularity—we conduct an ablation study to investigate the impact of each contribution on training performance. In addition, we also ablate the structure of the hierarchy itself in order to investigate its resiliency to structural changes.

Results of the ablation experiments can be found in Table 3. Our model (line 3) has an overall success rate of 82.4% across three seeds. When ablating the usage of hierarchical modularity, performance drops to 36.4% (line 5). Utilizing our runtime introspection approach to investigate potential issues in the modules (Sect. 4.3.1), we find that the target and displacement errors increased to over 20 cm, which is likely the cause for the reduced performance. When removing the supervision signal (line 4) for the attention inside our modules (and instead relying on end-to-end training), we see a drop of $\approx$ 2.5% in performance to about 80%.

When ablating the hierarchy itself, we merged the TAR2D and TAR3D module (line 7) into a single module instead of maintaining two. The underlying rationale is that the separation of the target detection between 2D and 3D detection is not strictly necessary and thus a single target module may be sufficient. The resulting success rate in this case is 80.9% which is only slightly below the original rate of 82.4%. Next we removed the displacement module DISP (line 9) altogether, which results in a performance of about 61.3% (a loss of around 20%). Finally, we added spurious modules that are not necessary for the policy's success in these tasks (line 10). In particular, we added a specific module that only detects the "Coke" can. In this case, we achieved a success rate of 86.3% which is slightly higher than the original result.

As a general observation, the approach seems to be favorable to superfluous modules, combined modules, or variations of a hierarchy. However, the absence of certain *critical* modules, e.g., the DISP or TAR modules (lines 9 and 8 respectively), may have a more drastic effect on performance. In the above case of removing the DISP module (line 9), the performance reduces to about 61.3% which is below the corresponding value for BC-Z (73.1%).

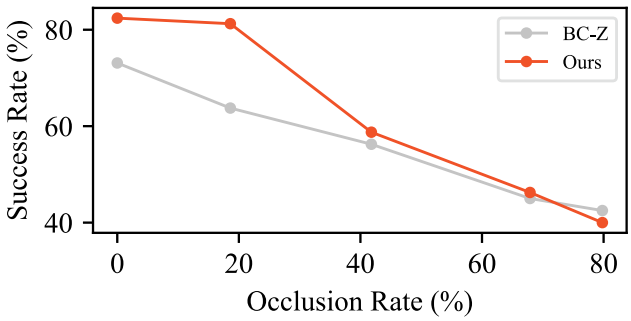


Fig. 10 Success rate when part of the target object is occluded

5.2.2 Occlusion

Next, we evaluate the robustness of our approach to partial occlusions of the target objects during task execution. To this end, occlusions are introduced by removing image patches in the camera feed of the simulated experiments. This step is performed by covering approximately 20%, 42%, 68% and 80% of the target object's total area; calculated via a pixel-based segmentation approach of the input image provided by the simulator. All experiments are conducted on all six Robosuite objects across all three basic tasks. The results are shown in Fig. 10. We observe that our method is robust to occlusions of up to 20% of the target object, while our baseline model, BC-Z, already experiences a significant drop in accuracy. While our model only loses about 1.1% in performance, BC-Z drops by 9.35%. However, for occlusions greater than 40%, our method performs on-par with BC-Z. We argue that our robustness to 20% occlusions is significant since small, partial, occlusions are more likely to occur during tabletop manipulation tasks.

5.2.3 Synonyms

Our final robustness experiment is concerned with the variability of free-form spoken language. While our system is trained with sentences from a template-based generator, we evaluate its performance when exposed to a set of additional synonyms, as well as free-form spoken language from a small set of human subjects. When replacing synonyms, as shown in Table 8, in the single-word and short-phrase case, we observe that our model achieves a 82.5% success rate on the pushing task. When using BC-Z, on the same task with the same synonyms, performance drops to 28.57%, indicating the robustness of our methods to variations in the language inputs. Finally, we also evaluate the performance on 30 examples of free-form natural language instructions that were collected from human annotators and report a success rate of 73.3%. The sentences used by the annotators can be found in Table 11 and show that our model can work with unconstrained language commands.

5.3 Transfer to different robots and real-world

In this section, we evaluate the ability of our approach to efficiently transfer policies between different robots that may have different morphologies. Rather than retraining our model from scratch to accommodate the altered dynamics between different robots, we posit that our modular approach enables the transfer of substantial portions of the prior policy. This necessitates only minimal fine-tuning, consequently resulting in a reduced demand for data collection on the different robots. In particular, we evaluate fine-tuning of the entire policy, and fine-tuning of only the modules affected by a change in visual appearance of robot morphology.

5.3.1 Transfer in simulation

Our initial policy is trained from scratch on the $\mathcal{D}^{\text{Kinova}}$ dataset while the transfer of the trained policy to the Franka and UR5 robot is realized with the $\mathcal{D}_{\text{TF}}^{\text{Franka}}$ and $\mathcal{D}_{\text{TF}}^{\text{UR5}}$ datasets respectively.

As noted earlier, the $\mathcal{D}_{\text{TF}}$ datasets are intentionally over-provisioned to allow an evaluation regarding how much data is required in order to match the performance of the transferred policy to a policy that is trained from scratch on the same robot. In order to shed some light on this, we subsampled the transfer datasets to a total size of 80, 160, 240 and 320 demonstrations and conducted the training. Figure 11 shows the results of this analysis (reported as "Ours") given the varying dataset sizes when fine-tuning the entire policy initialized with the Kinova weights. With 160 demonstrations, our model achieves a success rate of 80%, which is only slightly below the policy's performance when trained on the full 1600 demonstrations from scratch. Further, given the full 320 demonstrations of the transfer dataset, the policy reaches a performance that is on-par with one trained from scratch. When fine-tuning BC-Z with the same dataset splits, we observe that our model consistently outperforms BC-Z. Interestingly, we also observe that our model performs similarly when transferring to the Franka and UR5 robots across the dataset splits, while BC-Z seems to initially perform worse when transferring the Franka robot. Note here that Franka is a 7 degree of freedom (DoF) robot while the source policy, which operates over the Kinova robot, only has six. This discrepancy likely affects robot dynamics thereby affecting the transfer process.

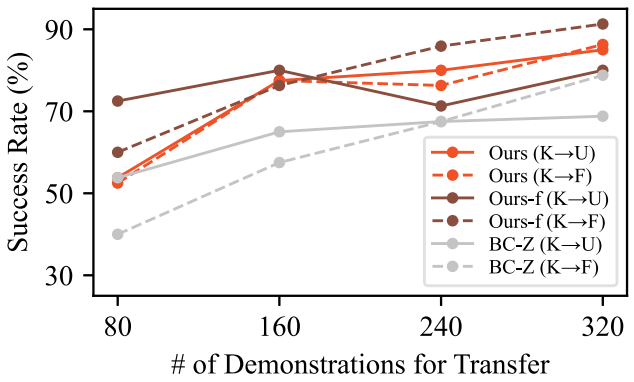


Fig. 11 Results of transferring policies from Kinova (K) robot to UR5 (U) and Franka (F) robots. "Ours-f" refers to freezing parts of our model during transferring. Experiments are performed in Mujoco simulator

Further, we conducted experiments in which we froze parts of our model during transfer of a pre-trained policy from the Kinova to the UR5 and Franka robot. In particular, the TAR3D, EE3D, and DISP prediction modules are unaffected by the change in visual appearance and morphology of the new robot and, thus, do not need to be retrained. Note, however, that we retrain TAR2D since partial occlusions by the new robot could lead to false positives for target objects. We have conducted further experiments with the same fine-tuning datasets and report their results in Fig. 11 (reported as "Ours-f"). In this setting, with a dataset of only 80 demonstrations, the partially frozen module produces a result of 60% and 72.5% when transferring to the Franka and UR5 respectively. This poses a substantial performance improvement of up to 18% in the case of transfer to the UR5 robot while utilizing less data than fine-tuning the entire model. This result further underlines the gains in data-efficiency that can be achieved through the hierarchical modularity.

5.3.2 Real-world transfer

Having demonstrated the ability of our approach to efficiently transfer policies between robots in simulation, we demonstrate that a policy can also be transferred to the real world (Sim2Real Transfer) in a sample-efficient way. To this end, we first trained a policy for the UR5 robot in simulation utilizing the $\mathcal{D}^{\text{UR5}}$ dataset and subsequently transferred it with a substantially smaller real-world dataset $\mathcal{D}_{\text{RW}}^{\text{UR5}}$. More specifically, 260 demonstrations on the real-robot are collected for transfer—this corresponds to about $\frac{1}{6}$-ththe size of the original training set. The overall robot setup can be seen in Fig. 12. The scene is observed via an external RGB camera and robot actions are calculated in a closed-loop fashion by providing the current camera image and language instruction to the policy.

To investigate the contributions of our proposed methods, we conduct experiments under 3 different baseline settings. These include directly applying the simulated policy on the real robot, fine-tuning the simulated policy using the real-world dataset $\mathcal{D}_{\text{RW}}^{\text{UR5}}$, and transferring the simulated policy to the real world using our proposed method. Image sequences of real-robot executions can be seen in Figs. 7 and 9. As expected, the policy trained in simulation is unable to complete any task when being directly applied to the real robot despite coordinate systems and basic dynamics being

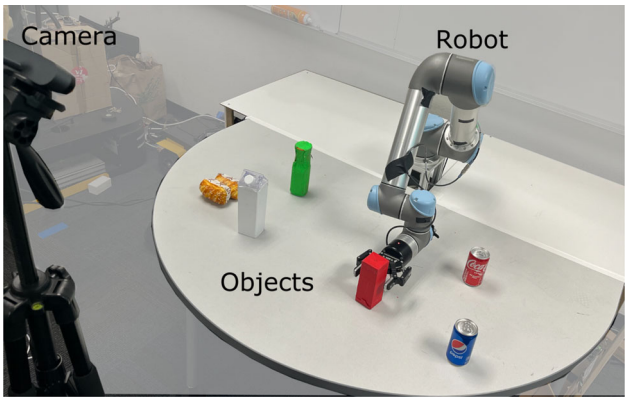


Fig. 12 Experimental setup of real-robot experiments. Objects are seen through an external camera and actions are generated in a closed-loop

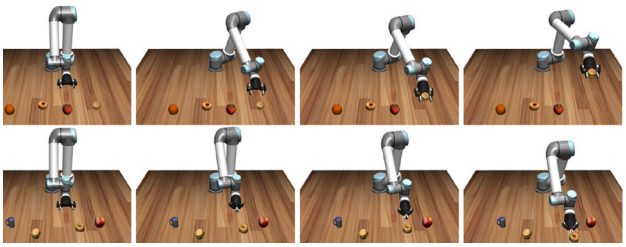


Fig. 13 Robot performing a task with objects which are generated automatically generated. The top row is the robot picking up an object, while the second row is the robot pushing an object

matched between the simulation and the real world. This failure is due to the substantial variation in visual appearance of the robot and objects. When using a naive fine-tuning approach that does not use our core contributions, the resulting success rate is 56.7% over 30 trials, thus demonstrating partial success. However, we observe that the noise in the attention maps is unusually high, which we attribute to the intricacies of real-world vision and dynamics. Finally, when training the system with our approach, including supervised attention and hierarchical modularity, the approach achieve a success rate of 80% in the real world when prompted with 30 commands issued by a human operator.

5.4 Generalization to novel objects

To investigate the importance of modularity for generalization, we extend the experiment setup to include a more challenging scenario. In particular, we incorporate a total of 100 objects, which are automatically generated following the approach outlined in Sect. 2. They are comprised of 10 unique classes, each with 10 objects. We utilize 3 objects from each class for training, while the remaining 7 objects, which were previously unobserved by the model, are reserved for testing. For this experiment, we utilize the $\mathcal{D}_{\text{NO}}^{\text{UR5}}$ dataset and perform an evaluation with 100 trials (Fig. 13).

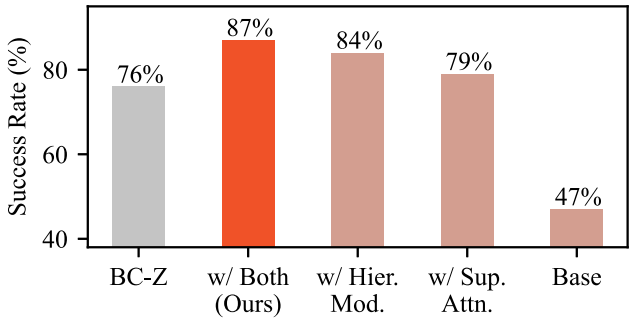


Fig. 14 Test results of simulated experiments with 70 unseen objects from 10 classes which are generated automatically using the data pipeline proposed in Sect. 2

As before, we compare our model's performance to BC-Z, as well as an ablated version of our model without supervised attention or hierarchical modularity. The results are shown in Fig. 14 on the 100 object generalization task. During this study, we removed either one or both of our components from the model during training to examine their individual and combined contributions. The most basic version of the model, identified as "Base" and not using supervised attention, nor hierarchical modularity, demonstrates poor performance with a score of 47%. Models without Supervised Attention ("w/ Sup. Attn") and those without Hierarchical Modularity ("w/ Hier. Mod.") each exhibit significantly better performance compared to the base model. Notably, the optimal model is our proposed full model ("Ours"), which combines both Supervised Attention and Hierarchical Modularity, resulting in an impressive success rate of 87%. For comparison, the baseline model BC-Z attains a 76% success rate, which is surpassed by our proposed model by 9%.

5.5 Hierarchy extension

In this section, we explore two extension to our hierarchy by introducing new models that allow the policy to conduct new tasks. As we have shown in prior sections, our modular approach allows for easy transfer between different robots; however, this approach can also be utilized to introduce novel tasks to the policy. The following sections introduce an obstacle avoidance task in which an object is placed in the path between the robot and the described target object which has to be detected and avoided. In a subsequent experiment, we further extend the hierarchy by not only focusing on a fixed "obstacle object", but allow the user to specify a secondary reference object, ultimately affording a novel placement task that allows objects to be placed in relation to others objects in the environment.

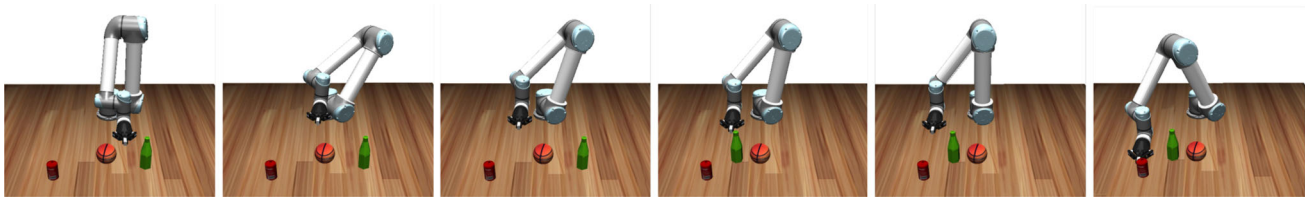


Fig. 15 Robot trained to avoid all obstacles in the scene. On the way to the Coke can, the robot first avoids a basketball and then the green bottle. We move the bottle in front of the robot to generate an instantaneous response

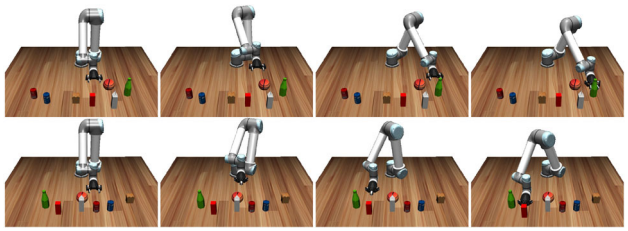


Fig. 16 Robot performing a task while avoiding a basketball. The top row shows a pick action and the bottom row shows a push action. In both cases, the robot changes its course to avoid collision with the obstacle

5.5.1 Obstacle avoidance

In this experiment, we demonstrate a seamless way to integrate new modules into an existing trained hierarchy by introducing an obstacle avoidance task. First, we discuss a setup wherein a single, *specific* object needs to be avoided, before extending the approach to avoid *any* obstacle in the scene (Figs. 15, 16).

In our first setting, a basketball is placed between the end-effector and the objects that are to be manipulated, serving as an obstacle. The robot must first identify the obstacle and subsequently formulate a trajectory to navigate around it effectively. In this task, new modules OBST2D and OBST3D are added to the hierarchy and trained to generate the location of the obstacle in image space and world space. More specifically, OBST2D identifies image patches that belong to the object. In turn, these patches are fed into OBST3D to generate a 3D world coordinate. We relate the obstacle's position to the robot by calculating a second displacement DISP2 which utilizes EE3D and OBST3D. Figure 8 shows the updated hierarchy. The output of DISP2 feeds into the calculation of the control value where it is combined with the output of DISP (the displacement of the end-effector to the target object).

The expert trajectories which avoid the obstacle are generated by using a potential field approach (Khatib, 1986). More specifically, the basketball is a repulsor that pushes the end-effector away from it. Using this approach, 200 training demonstrations are collected, forming the dataste $\mathcal{D}^{\text{UR5}}_{\text{OBST}}$. The policy for this task has been trained from a UR5 policy by utilizing the above datatet that introduces the novel task.

For evaluation, we define a successful trial as *the absence of any* collision between the robot and the obstacle. After training, our method achieves a success rate of 88% where failure cases mostly revolve around premature contact with the target object.

We further extend the capabilities of the proposed hierarchical approach by avoiding *any* object in the environment. To this end, we utilize a single module that is trained to focus on all objects with exception of the target object. This module can viewed as the inverse of the target detection module, i.e., all but the target object are highlighted. In this multiple-obstacle case, the trained network achieves a success rate of 83%. An image sequence of the resulting behavior can be seen in Fig. 15. Notice the robot response after a second obstacle (green bottle) is moved in front of it. The image sequence also highlights the closed-loop control underlying our approach—robot actions are constantly recalculated based on the current environmental conditions.

5.5.2 Relational reference

While the obstacle avoidance tasks showed the basic pipeline of adding a secondary object and defining a desired behavior for it, the approach can be extended to also allow the user to verbally specify this secondary object. For this purpose, we introduce a relational placing task in which a user specifies a reference object, requiring the system to identify the two task related objects and generating a control signal in accordance to it. Relational tasks involve instructions that not only specify a target object for manipulation (e.g., an "apple") but also mention an additional referential object (e.g., an "orange"), such as "Place the apple to the right of the orange." In these scenarios, the robot must identify the two objects and understand the intention behind the given language. We aim to demonstrate that our model can effectively handle such tasks even under generalization constraints. For this purpose we again utilize our 100 automatically generated objects and train a policy over 30 of them, which are composed of 3 objects per class, while evaluating it's generalization capabilities on the remaining 70 objects. For this experiment, we utilize the $\mathcal{D}^{\text{UR5}}_{\text{ET}}$ dataset (Fig. 17).

For this task, we have made modifications to the original hierarchies. Firstly, we introduce a LANG2 module

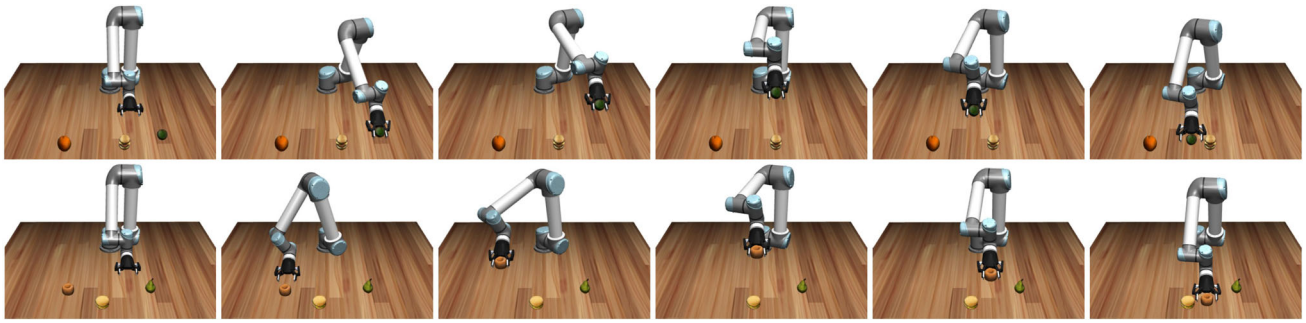


Fig. 17 Robot performing relational tasks with 2 objects involved in 1 command. The top row is the robot putting an avocado left to a hamburger, while the second row is the robot putting a donut right to a hamburger

to determine the referential object based on the language input. Besides that, we add TAR2D2 and TAR3D2 modules to identify the image patch corresponding to the second object and generate its 3D world coordinate, respectively. We also include a DISP2 module to calculate the displacement between the end-effector and the second object.

In this scenario, the robot is directed by a verbal sentence to identify the first object, pick it up, recognize the second object, and then place the first object either to the left or right of the second object according to the command given.

The entire process is carried out in the MuJoCo environment, evaluated on 100 test trials. For comparison, we also train and evaluate the BC-Z model. Our model achieves a success rate of 76%, which is a 7% improvement over the BC-Z performance. Considering the increased complexity of this task compared to previous ones—due to the need to identify two objects from both the sentence and image, and the more extended manipulation steps required—a 76% success rate and a 7% increment compared to the baseline are commendable results.

6 Discussion and limitations

The above experiments show a variety of benefits of the introduced modular approach. On one hand, it allows for new components and behaviors to be incorporated into an existing policy. This property is particularly appealing in robotics, since many popular robot control architectures are based on the concept of modular building-blocks, e.g., behavior-based robotics (Arkin, 1998) and subsumption architecture (Brooks, 1986). Modularity also enables the user to employ modern verification and runtime monitoring tools to better understand and debug the decision-making of the system. At the same time, the overall system is still end-to-end differentiable and was shown in the above experiments to yield practical improvements in sample-efficiency, robustness and extensibility.

However, a major assumption made in our approach is that a human expert correctly identifies the logical flow of components and subtasks into which a task can be divided. This process requires organizing these subtasks into a hierarchical cascade. Early results indicate that an inadequate decomposition can hamper, rather than improve, learning. Furthermore, the approach does not incorporate memory and therefore cannot perform sequential actions. In a few cases we observed a failure to stop after finishing a manipulation - the robot continues with random actions. Another open question is the scalability of the approach. In our investigations, we looked at behaviors with a small number of sub-tasks. Is it possible to scale the approach to hierarchies with hundreds or thousands of nodes? The prospect is appealing since this would bridge the divide between the expressiveness and plasticity of neural networks and the ability to create larger robot control systems which require the interplay of many subsystems.

For future work, we are particularly interested in using unsupervised and supervised attention side-by-side, i.e., several modules may be supervised by the human expert whereas other modules are adjusted in an unsupervised fashion. This would combine the best of both worlds, namely the ability to provide human structure and knowledge while at the same time maximally profiting from the network's plasticity. This is a particularly promising direction, since the ablation experiments indicate that having superfluous modules does not drastically alter the network performance. Further, we would like to investigate the potential of inferring a suitable hierarchy in a data-driven manner.

7 Conclusions

In this paper, we present a data-efficient approach for language-conditioned policies in robot manipulation tasks. We introduce a novel method called Hierarchical Modularity, and adopt supervised attention, to train a set of reusable sub-modules. This approach maintains the end-to-end learning advantages while promoting the reusability of the learned sub-modules. As a result, we are able to customize the hierarchy according to the specific task demand, or integrating new modules to an existing hierarchy for new tasks. Our

method demonstrates high performance in a comprehensive set of experiments including training manipulation policies with limited data, transferring between multiple robots, and extension of module hierarchies. We also develop an automated data generation pipeline for creating simulated objects to manipulate with, and show our model's generalization capability on unseen objects generated by such pipeline. Furthermore, we demonstrate that the learned hierarchy of sub-modules can be employed for introspection and visualization of the robot's decision-making processes.

Acknowledgements This work was partially supported by the National Science Foundation under Grant CNS-1932068.

Author Contributions YZ contributed to agent development, imitation learning, data collection, running and the analysis of results. SS contributed to engineering infrastructure, environment development, and automated 3D object generation. MP contributed to ideation and theoretical algorithm design. HBA contributed to advising, ideation, writing, project management, and development of generative AI model. SS contributed to Advising, ideation, writing and task design.

Declarations

Ethical approval This manuscript expands upon our prior conference paper, "Modularity through Attention: Efficient Training and Transfer of Language-Conditioned Policies for Robot Manipulation" by Zhou et al., previously published as part of the Conference on Robot Learning (CoRL) 2022 proceedings held in Auckland, New Zealand. Particularly the methodology described in Sects. 4.1 and 4.2 have previously been published. Similarly, the results discussed in Sects. 5.2 and 5.3 have been previously published in the above publication. We confirm that the manuscript, in its current form, has not been published elsewhere and is not under consideration by another journal. Further, we confirm that all authors have approved the manuscript and agree with its submission to the Autonomous Robots Special Issue on Large Language Models in Robotics.

Appendix A: Training details

A.1: Network architectures and hyperparameters

We use a convolutional neural network for image encoding, as shown in Table 4. We use fully connected layers for joint encoders, target position decoders, displacement decoders and controllers, which are shown in Tables 5, 6 and 7 respectively. We use 4 eight-head attention layers of 192 dimensions for modality fusing and interaction. The Adam optimizer with learning rate of 1e−4 is adopted for training (Tables 8, 9, 10, 11).

Table 4 Image encoder architecture

Layer	Kernel	Channel	Stride	Padding
CNN	7	64	1	3
CNN	3	128	2	1
CNN	3	256	2	1
CNN	3	256	2	1
ResBlock	3	256	1	1
ResBlock	3	256	1	1
ResBlock	3	256	1	1

Table 5 Joint encoder architecture

Layer	Dimension
FC	256
FC	128
FC	192

Table 6 Position and displacement decoder architecture

Layer	Dimension
FC	128
FC	9

Table 7 Controller architecture

Layer	Dimension
FC	2048
FC	1024
FC	256
FC	120

Table 8 Synonyms used in test

Milk carton	Bottle	Coke	Cube	Bread
Skimmed milk package	Soda	Coke zero	Brick	Cinnamon roll
Goat milk carton	Perrier	Round container	Block	Sourdough
Milk case	Tonic	Can	Cuboid	Brown bread
White packet	Flask	Coca cola	Bar	Loaf
Milk parcel	Pitcher	Red soda	Solid lump	Naan
Cream carton	Container	Cola	Rectangular object	Rye bread
Cream package	Decanter	Metal container	Solid piece	Toast
Heavy milk carton	Vial	Small soft drink	Slab	Gluten free food
Almond milk box	Vessel	Fizzy drink	Cuboidal slice	Light bread
Goat milk packs	Cruet	Diet coke	Square object	Food

Table 9 The noun phrase template

Object	Adj	Noun
Coke	Red	Can
	Coke	Bottle
	Cocacola	
Pepsi	Blue	Can
	Pepsi	Bottle
	Pepsi coke	
Bottle	green	Bottle
	Glass	
	‘’	
	Green glass	
Carton	Milk	Carton
	White	Box
Cube	Red	Object
	Maroon	Cube
		Square
Bread	‘’	Bread
		Yellow object
		Brown object

Table 10 The verb phrase template

Verb pick	Verb push	Verb put
Pick	Push	Put down
Pick up	Move	Place down
Raise		

Table 11 Sentences collected from annotators for evaluation purposes

Annotator labeled sentences	Success
Grab the loafs	F
Put down the lime soda	T
Lay down the red block	T
Tip over the azure can	T
Lift the white carton	F
Knock over the pastry	T
Lift the coke can	T
Put down the sprite	T
Grab the pepsi	T
Elevate the red cube	T
Pick up the red cube	T
Lift up the blue cylinder	T
Move away the brown object	T
Push away the white object	T
Lift the blue object	T
Put down the green sprite	T
Push the green sprite	T
Push the reddish can	T
Pick up the milk container	F
Hold up the milk carton	F
Please pick up the green thing	F
Lift the red colored coke can	T
Push the yellow bread	T
Grab the blue colored can	T
Nudge that green bottle	F
Put down the red colored cuboid	T
Lift the white box	T
Take the pepsi off the table	F
Push the green object forward	F
Put down the zero coke on the desk	T

Our model achieves 73.3% success rate on variations of languages

References

Abolghasemi, P., Mazaheri, A., Shah, M., et al. (2019). Pay attention!-robustifying a deep visuomotor policy through task-focused visual attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 4254–4262).

Ahn, M., Brohan, A., Brown, N., et al. (2022). Do as I can, not as I say: Grounding language in robotic affordances. arXiv:2204.01691

Alayrac, J. B., Donahue, J., Luc, P., et al. (2022). Flamingo: A visual language model for few-shot learning. *Advances in Neural Information Processing Systems, 35*, 23716–23736.

Anderson, P., Shrivastava, A., Parikh, D., et al. (2019). Chasing ghosts: Instruction following as Bayesian state tracking. In *Advances in neural information processing systems* (Vol. 32).

Antol, S., Agrawal, A., Lu, J., et al. (2015). Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision* (pp. 2425–2433).

Argall, B. D., Chernova, S., Veloso, M., et al. (2009). A survey of robot learning from demonstration. *Robotics and Autonomous Systems, 57*(5), 469–483.

Arkin, R. (1998). *Behavior-based robotics*. The MIT Press.

Bengio, Y., Louradour, J., Collobert, R., et al. (2009). Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning. Association for Computing Machinery, New York, NY, USA, ICML '09* (pp. 41–48). https://doi.org/10.1145/1553374.1553380

Brooks, R. (1986). A robust layered control system for a mobile robot. *IEEE Journal on Robotics and Automation, 2*(1), 14–23. https://doi.org/10.1109/JRA.1986.1087032

Carion, N., Massa, F., Synnaeve, G., et al. (2020). End-to-end object detection with transformers. In *European conference on computer vision* (pp. 213–229). Springer.

Chen, Y. C., Li, L., Yu, L., et al. (2020). Uniter: Universal image-text representation learning. In: *Computer vision—ECCV 2020: 16th European conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXX* (pp. 104—120). Springer. https://doi.org/10.1007/978-3-030-58577-8_7

Coates, A., Abbeel, P., & Ng, A. Y. (2009). Apprenticeship learning for helicopter control. *Communications of the ACM, 52*(7), 97–105.

Csordás, R., van Steenkiste, S., & Schmidhuber, J. (2021). Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *International conference on learning representations*.

Das, A., Kottur, S., Gupta, K., et al. (2017). Visual dialog. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 326–335).

de Boer, P. T., Kroese, D. P., Mannor, S., et al. (2004). A tutorial on the cross-entropy method. *Annals of Operations Research, 134*, 19–67.

Dillmann, R., & Friedrich, H. (1996). Programming by demonstration: A machine learning approach to support skill acquision for robots. In *International conference on artificial intelligence and symbolic mathematical computing* (pp. 87–108). Springer.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., et al. (2021). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv:2010.11929.

Duan, Y., Andrychowicz, M., Stadie, B., et al. (2017a). One-shot imitation learning. In I. Guyon, U. V. Luxburg, S. Bengio, et al. (Eds.), *Advances in Neural Information Processing Systems.* (Vol. 30). Curran Associates Inc.

Duan, Y., Andrychowicz, M., Stadie, B., et al. (2017b). One-shot imitation learning. In *Advances in neural information processing systems* (Vol. 30).

Filan, D., Hod, S., Wild, C., et al. (2020). Neural networks are surprisingly modular. arXiv:2003.04881, cite Comment: 23 pages, 13 figures.

Huang, W., Xia, F., Xiao, T., et al. (2022). Inner monologue: Embodied reasoning through planning with language models. arXiv:2207.05608

Jang, E., Irpan, A., Khansari, M., et al. (2022). Bc-z: Zero-shot task generalization with robotic imitation learning. In *Conference on robot learning, PMLR* (pp. 991–1002).

Johnson, J., Hariharan, B., Van Der Maaten, L., et al. (2017). Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2901–2910).

Kamath, A., Singh, M., LeCun, Y., et al. (2021). Mdetr—modulated detection for end-to-end multi-modal understanding. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)* (pp. 1780–1790).

Khatib, O. (1986). The potential field approach and operational space formulation in robot control. In *Adaptive and learning systems* (pp. 367–377). Springer.

Kottur, S., Moura, J. M., Parikh, D., et al. (2018). Visual coreference resolution in visual dialog using neural module networks. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 153–169).

Kuo, Y. L., Katz, B., & Barbu, A. (2020). Deep compositional robotic planners that follow natural language commands. In *2020 IEEE international conference on robotics and automation (ICRA)* (pp. 4906–4912). IEEE.

Laina, I., Rupprecht, C., & Navab, N. (2019). Towards unsupervised image captioning with shared multimodal embeddings. In *Proceedings of the IEEE/CVF international conference on computer vision (ICCV)*.

Liu, L., Utiyama, M., Finch, A., et al. (2016). Neural machine translation with supervised attention. In *Proceedings of COLING 2016, the 26th international conference on computational linguistics: technical papers. The COLING 2016 Organizing Committee, Osaka, Japan* (pp. 3093–3102). https://aclanthology.org/C16-1291

Locatello, F., Weissenborn, D., Unterthiner, T., et al. (2020). Object-centric learning with slot attention. *Advances in Neural Information Processing Systems, 33*, 11525–11538.

Lu, J., Batra, D., Parikh, D., et al. (2019). Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems* (Vol. 32).

Lynch, C., & Sermanet, P. (2021). Language conditioned imitation learning over unstructured data. In *Proceedings of robotics: Science and systems*.

Maeda, G., Ewerton, M., Lioutikov, R., et al. (2014). Learning interaction for collaborative tasks with probabilistic movement primitives. In *2014 IEEE-RAS international conference on humanoid robots* (pp. 527–534). IEEE.

Mees, O., Hermann, L., Rosete-Beas, E., et al. (2022). Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks. *IEEE Robotics and Automation Letters, 7*(3), 7327–7334.

Nair, S., Rajeswaran, A., Kumar, V., et al. (2022). R3m: A universal visual representation for robot manipulation. arXiv:2203.12601

OpenAI. (2023). Gpt-4 technical report. arXiv:2303.08774

Ouyang, L., Wu, J., Jiang, X., et al. (2022). Training language models to follow instructions with human feedback. arXiv:2203.02155

Pettersson, O. (2005). Execution monitoring in robotics: A survey. *Robotics and Autonomous Systems, 53*(2), 73–88.

Radford, A., Kim, J. W., Hallacy, C., et al. (2021). Learning transferable visual models from natural language supervision. arXiv:2103.00020

Rahmatizadeh, R., Abolghasemi, P., Bölöni, L. et al. (2018). Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 3758–3765), IEEE.

Ranftl, R., Lasinger, K., Hafner, D., et al. (2022). Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 44*(3), 1623–1637.

Reed, S., Zolna, K., Parisotto, E., et al. (2022). A generalist agent. arXiv:2205.06175

Rombach, R., Blattmann, A., Lorenz, D., et al. (2022). High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 10684–10695).

Schaal, S. (1999). Is imitation learning the route to humanoid robots? *Trends in Cognitive Sciences, 3*(6), 233–242.

Schaal, S. (2006). Dynamic movement primitives—A framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines* (pp. 261–280). Springer.

Shridhar, M., Manuelli, L., & Fox, D. (2021). Cliport: What and where pathways for robotic manipulation. arXiv:2109.12098

Singh, A., Hu, R., Goswami, V., et al. (2022). Flava: A foundational language and vision alignment model. arXiv:2112.04482

Sorkine, O., Cohen-Or, D., Lipman, Y., et al. (2004). Laplacian surface editing. In *Proceedings of the 2004 Eurographics/ACM SIGGRAPH symposium on geometry processing. association for computing machinery, New York, NY, USA, SGP '04* (pp. 175–184). https://doi.org/10.1145/1057432.1057456

Stepputtis, S., Campbell, J., Phielipp, M., et al. (2020). Language-conditioned imitation learning for robot manipulation tasks. arXiv:2010.12083

Tan, H., & Bansal, M. (2019). Lxmert: Learning cross-modality encoder representations from transformers. In K. Inui, J. Jiang, V. Ng, et al. (Eds.) *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3–7, 2019. Association for Computational Linguistics* (pp. 5099–5110). https://doi.org/10.18653/v1/D19-1514

Todorov, E., Erez, T., & Tassa, Y. (2012). Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems* (pp. 5026–5033). IEEE.

Vaswani, A., Shazeer, N., Parmar, N., et al. (2017). Attention is all you need. In *Advances in neural information processing systems* (pp. 5998–6008).

Vemprala, S., Bonatti, R., Bucker, A., et al. (2023). Chatgpt for robotics: Design principles and model abilities. 2023.

Vinyals, O., Toshev, A., Bengio, S., et al. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)*.

Wang, Y., Mishra, S., Alipoormolabashi, P., et al. (2022). Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 conference on empirical methods in natural language processing* (pp. 5085–5109).

Xie, F., Chowdhury, A., De Paolis Kaluza, M. C., et al. (2020). Deep imitation learning for bimanual robotic manipulation. In: H. Larochelle, M. Ranzato, R. Hadsell, et al. (Eds.) *Advances in neural information processing systems* (Vol. 33, pp. 2327–2337). Curran Associates, Inc. https://proceedings.neurips.cc/paper/2020/file/18a010d2a9813e91907ce88cd9143fdf-Paper.pdf

Xu, K., Ba, J., Kiros, R., et al. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International conference on machine learning, PMLR* (pp. 2048–2057).

Yamaguchi, T., & Fainekos, G. (2021). Percemon: Online monitoring for perception systems. In *Runtime verification: 21st international conference, RV 2021, virtual event, October 11–14, 2021, Proceedings* (p. 297). Springer.

Zhang, T., McCarthy, Z., Jow, O., et al. (2018a). Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 5628–5635). https://doi.org/10.1109/ICRA.2018.8461249

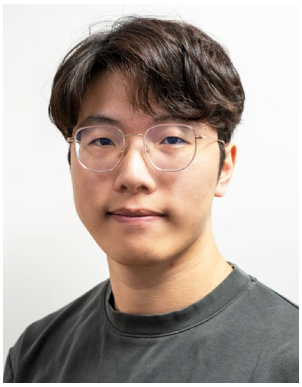
Zhang, T., McCarthy, Z., Jow, O., et al. (2018b). Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE international conference on robotics and automation (ICRA)* (pp. 5628–5635). IEEE.

Zhou, Y., Sonawani, S., Phielipp, M., et al. (2022). Modularity through attention: Efficient training and transfer of language-conditioned policies for robot manipulation. arXiv:2212.04573

Zhu, D., Chen, J., Shen, X., et al. (2023). Minigpt-4: Enhancing vision-language understanding with advanced large language models. arXiv:2304.10592

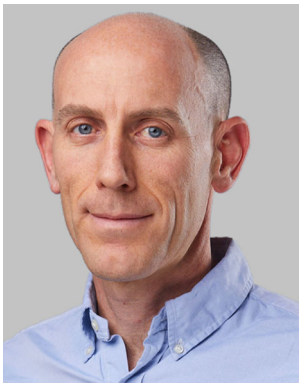
Zhu, Y., Wong, J., Mandlekar, A., et al. (2020). robosuite: A modular simulation framework and benchmark for robot learning. arXiv:2009.12293

Zirr, T., & Ritschel, T. (2019). Distortion-free displacement mapping. *Computer Graphics Forum*. https://doi.org/10.1111/cgf.13760



Yifan Zhou is a PhD student at Arizona State University, working in the Interactive Robotics Lab with Prof. Heni Ben Amor. His main focus is language-conditioned imitation learning for robot manipulation. Prior to that, he received his master's degree in Carnegie Mellon University and bachelor's degree in Southwest Jiaotong University.

Shubham Sonawani is a PhD student in Electrical Engineering at Arizona State University. He embarked on his academic journey at the Interactive Robotics Lab in Fall 2018. Prior to his endeavors at ASU, Shubham earned his Bachelor of Technology in Electrical Engineering from VJTI, India. His PhD research focuses on the confluence of Human-Robot Interaction, Grasping, Manipulation, and Mixed Reality. The primary objective of his research is to enhance collaboration between humans and robots by leveraging a range of information modalities, including but not limited to projection mapping, computer vision, and natural language processing.



Mariano Phielipp is currently employed at Intel AI Lab, which is situated within Intel Labs. His work encompasses research and development in deep learning, deep reinforcement learning, machine learning, and artificial intelligence. Since joining Intel, Dr. Phielipp has made significant contributions in various domains, including computer vision, face recognition, face detection, object categorization, recommendation systems, online learning, automatic rule learning, natural language processing, knowledge representation, energy based algorithms, and other machine learning and AI-related endeavors.

Heni Ben Amor Amor is an Associate Professor at Arizona State University where he leads the ASU Interactive Robotics Laboratory. He was previously a research scientist at Georgia Tech's Institute for Robotics and Intelligent Machines, where he led a project to improve robots for future applications in industrial settings, especially manufacturing. Prior to moving to Georgia Tech, Ben Amor worked with Jan Peters at the Technical University Darmstadt as a postdoctoral scholar. Ben Amor's research topics focus on artificial intelligence, machine learning, human-robot interaction, robot vision, and automatic motor skill acquisition. He received the highly competitive Daimler-and-Benz Fellowship as well as several best paper awards at major robotics and AI conferences. He also serves on the program committee of various AI and robotics conferences, including AAAI, IJCAI, IROS, and ICRA.

Simon Stepputtis is a Postdoctoral Fellow at Carnegie Mellon University's Robotics Institute. His research focuses on human-agent teaming and multi-agent systems, investigating how humans and robots can efficiently collaborate in complex scenarios by building trust and utilizing effective communication. Prior to moving to CMU, he received a Ph.D. from Arizona State University working on physical human-robot interaction. He has worked at Bosch and X, The Moonshot Factory on various aspects of industrial robot manipulation. Dr. Stepputtis's research topics include artificial intelligence, human-robot interaction, natural-language processing, interpretability, few-shot learning, multi-agent systems, and computer vision, for which he received spotlight presentations at various conferences, including NeurIPS and CoLLAs.