ACT-Thor: A Controlled Benchmark for Embodied Action Understanding in Simulated Environments

Michael Hanna[*†]
University of Amsterdam
m.w.hanna@uva.nl

Federico Pedeni[*]
University of Trento
federico.pedeni@studenti.unitn.it

Alessandro Suglia
Heriot-Watt University
a.suglia@hw.ac.uk

Alberto Testoni
University of Trento
alberto.testoni@unitn.it

Raffaella Bernardi
University of Trento
raffaella.bernardi@unitn.it

Abstract

Artificial agents are nowadays challenged to perform embodied AI tasks. To succeed, agents must understand the meaning of verbs and how their corresponding actions transform the surrounding world. In this work, we propose ACT-Thor, a novel controlled benchmark for embodied action understanding. We use the AI2-THOR simulated environment to produce a controlled setup in which an agent, given a before-image and an associated action command, has to determine what the correct after-image is among a set of possible candidates. First, we assess the feasibility of the task via a human evaluation that resulted in 81.4% accuracy, and very high inter-annotator agreement (84.9%). Second, we design both unimodal and multimodal baselines, using state-of-the-art visual feature extractors. Our evaluation and error analysis suggest that only models that have a very structured representation of the actions together with powerful visual features can perform well on the task. However, they still fall behind human performance in a zero-shot scenario where the model is exposed to unseen (action, object) pairs. This paves the way for a systematic way of evaluating embodied AI agents that understand grounded actions.

1 Introduction

Recently, embodied agents have been increasingly proposed and evaluated on their capacity to navigate virtual environments, and execute actions within them according to instructions (Suhr et al., 2019; Hahn et al., 2020; Shridhar et al., 2020; Padmakumar et al., 2022). Action execution as a means of evaluating language understanding is a long-standing idea (e.g. (Winograd, 1972)) which has now become feasible thanks to the release of various virtual environment platforms (Kolve et al., 2017; Savva et al., 2019; Yan et al., 2018). Thus, the study of embodied agents is an interesting venue to push forward due its both theoretical and practical appeal. However, fully embodied tasks involve a variety of issues, making it hard to understand where we stand with respect to the challenges posed. Here, we focus on one core aspect of such embodied tasks: action grounding.

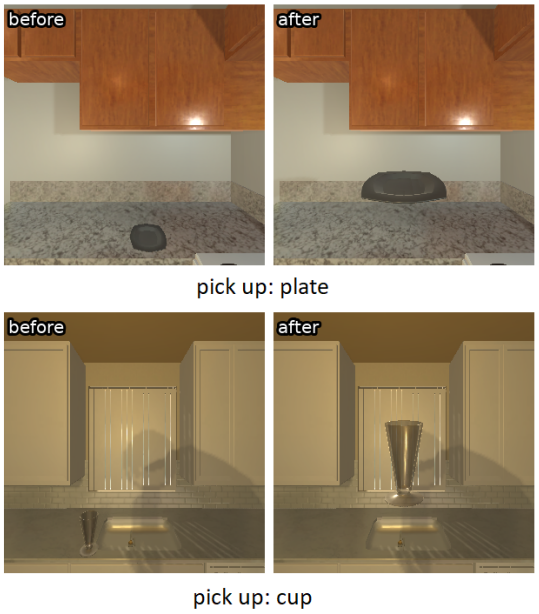

Figure 1: The image on the left (before-image) is the view the robot has while holding a plate or a cup; the image on the right is the view the robot has after having executed the command of picking up that plate/cup.

Tamari et al. (2020) advocate for embodied language understanding as a necessary step to overcome the limitations of the mainstream deep-learning paradigm. Among the priorities of their agenda is evaluating models' ability to simulate the effects of actions. Evidence from neuroscience favors an embodied meaning of words and specifically of verbs; for instance, the natural theory of

[*]Equal contribution
[†]Work performed while at the University of Trento

Proceedings of the 29th International Conference on Computational Linguistics, pages 5597–5612
October 12–17, 2022.

language (Feldman and Narayanan, 2004) is based on the assumption that people understand language by imagining (or simulating) the situation being described.

In this paper, we focus on grounded verb understanding and predicting the visual transformation their corresponding action causes. More specifically, we consider the idea of verbs as functions (Frege, 1892), and see them as transformations of an object within a scene before and after the verb/action is executed. To achieve this, we create a dataset combining ideas and methods from two main papers about learning physical commonsense knowledge (Gao et al., 2018; Zellers et al., 2021). As in Gao et al. (2018), we focus on the action-effect relation, namely the physical change that an action causes when executed on an object. And, as in Zellers et al. (2021), we leverage a virtual environment, AI2-THOR (Kolve et al., 2017), to generate transitions between object states.

In sum, we propose `ACT-Thor`, a dataset for learning the effects of actions, generated using a virtual environment.[1] For each example in our dataset, we collect an image of an object, a label describing an action performed on the object, and an after-image resulting from executing the action (Figure 1). Thus, unlike in Gao et al. (2018) and Zellers et al. (2021), both the before and after world states are images; the linking action acts as a function transforming the former into the latter. Then, via the use of images whose generation is carefully controlled, we design a task that intends to capture a challenge embodied AI agents will face in a real setting.

Out of this collection of image pairs and their corresponding action-label, we design a prediction task based on carefully selected contrast sets containing candidate images representing the effect of other actions performed on the very same object in the same scene (Figure 2). Agents that truly ground verbs, given the before-image and the action label, should be able to select the after-image out of the contrast set. We also evaluate human performance on a sample of the dataset. Annotators achieve high accuracy and inter-annotator agreement, evidencing the quality of the design method we propose, and setting an upper bound for multimodal models.

We evaluate baseline models based on state-of-the-art visual features (Singh et al., 2021; Radford et al., 2021). We put together ideas and results from different disciplines to shed light on different ways to study action in neural models and pave the way for future research on action grounding. From formal semantics and computational semantics, we take the view of verbs as functions (Frege, 1892), and therefore as matrices (Baroni and Zamparelli, 2010) and learn, for each action, a matrix able to transform the before-image, into the corresponding after-image. We compare this model with multimodal neural network models of increasing complexity. Our experiments show that

- virtual environment platforms can be used to generate interesting diagnostic datasets of static images having certain properties and satisfying certain constraints;
- the task we propose is easy for humans despite the use of virtual environments they are not familiar with;
- though the baseline models we have evaluated succeed on the task when evaluated on unseen scenes, they have difficulties when generalizing the learned action transformations to unseen objects.

2 Related Work

Actions/verbs have been studied from a variety of perspectives. They have been the subject of thorough investigation in formal semantics (Fillmore, 1982; Beth, 1993), and have been the focus of widely-used annotated resources (Fillmore et al., 2002; Schuler, 2005).

Multimodal approaches to action/verb learning have been especially fruitful. Misra et al. (2015) learn to ground visual representations of actions to formal semantic representations thereof. Prior studies have connected visual action representations to concepts such as intention (Pezzelle et al., 2020; Ignat et al., 2021) and causality (Gao et al., 2016; She and Chai, 2017; Yoo et al., 2021). Moreover, text-based models of similarity between actions have been shown to substantially improve when text is combined with visual information (Regneri et al., 2013). The breadth of these works suggests the importance of verbs and actions for the development of embodied agents and natural language understanding more broadly. We contribute to this line of research by presenting a new technique for creating datasets that evaluate computational models

[1] The dataset and all relevant code are available at `https://github.com/hannamw/ACT-Thor`.

for action grounding. We build on ideas introduced in (Gao et al., 2018; Wang et al., 2016) by leveraging a virtual environment platform: rather than filtering and annotating images from the web via crowdsourcing, we generate a controlled dataset using an agent in a virtual environment.

Within the NLP community, predicting the consequences of actions is often framed as part of learning commonsense knowledge. We share with Gao et al. (2018) a focus on what they call *naive* physical action-effect relations, namely the causal relation between an action (verb) and the change in physical world state caused by that action. Similarly, we propose an "effect prediction" task in which the model predicts an effect (an image) given the action and the object on which the action is performed. In their task, the action and object pair are expressed as verbs and nouns, whereas in our case the noun is presented as a (before) image; hence the model is challenged to learn the visual transformation objects undergo through the action execution.

Zellers et al. (2021) use AI2-THOR to build the PIGPeN dataset, from which models can learn world dynamics. They do so by letting a virtual agent navigate and interact with its environment to solve certain tasks. However, when building these trajectories, the robot's visual perspective (an image) is ignored in favor of a symbolic representation of the world state (i.e., as a set of visual attributes). Hence, models trained on PIGPeN receive symbolic and natural language inputs representing world information, and produce outputs in the same modality. To achieve our goal, we propose studying the physical action-effect relation highlighted by Gao et al. (2018), utilizing the image generation approach used in Zellers et al. (2021).

Finally, as in (Gao et al., 2018), we implement this concept as a classification task. However, inspired by Gardner et al. (2020), we adopt a contrastive setup. Our use of a virtual environment and its metadata allows us to build carefully controlled sets of candidates in which the very same object in the very same scene depicted in the before-image has undergone different visual transformations/different actions.

Statistic	ACT-Thor
action-object pairs	403
before-i, action, after-i	11154
unique before-i	3746
unique after-i	11154
scenes	120
objects	62
actions	12

Table 1: Statistics of the ACT-Thor collection.

3 Datasets

To build our dataset, we generated images using AI2-THOR[2], the virtual environment platform introduced in (Kolve et al., 2017). In AI2-THOR, the user controls a robot as it navigates a room of a virtual house environment, filled with household objects. While navigating the rooms, the robot can interact with objects it encounters by performing actions on them. Position and state metadata are available for the robot and all objects in the scene.

We use the robot to generate an image representing a scene before and after an action is taken. In the following section, we describe the image generation process, which led to the collection of triples consisting of before and after-images and their linking action. Then, we describe ACT-Thor, a dataset built out of this collection to evaluate models' abilities to simulate the consequences of actions on objects situated in virtual scenes.

3.1 Image Generation

AI2-THOR provides a rich environment for image generation. It allows an agent to traverse 120 distinct virtual rooms (scenes), equally distributed among the following four categories: kitchen, bathroom, living room, or bedroom. These scenes contain 125 distinct objects, from large, static objects such as desks or sinks, to smaller objects like pots or cups. Crucially for the creation of an action-oriented dataset, the robot can execute 24 actions, 21 of which act directly on objects.

In this virtual world, as in reality, each object has a certain number of affordances (Gibson, 1977), in other words, not all actions can be performed on all objects. Moreover, the state of an object affects these affordances as well; for example, objects can only be put down or thrown if they are currently held. We are interested in studying actions as visual

[2]We used version v4.2.0 of AI2-THOR, https://ai2thor.allenai.org/.

transformations, but not all actions result in clearly visible changes. A preliminary analysis showed that *pickupable* objects, those that can be picked up, could be the target of a greater variety of actions compared to static objects, or objects that could be moved but not picked up. Thus, we focused on these, as our dataset requires objects to be the targets of multiple, diverse actions.

Selecting pickupable objects and filtering out the rest, we consider 62 object types across all 120 scenes; each object type occurs in on average 28.5 scenes (SD: 14.65). Of the 21 actions affecting objects, we focused on 15 actions. We prioritized actions that cause visible state changes in their target objects, while also including some that physically move objects (some of the physical actions are similar in character, and not very visually distinct).

Based on these observations, our image generation procedure was as follows. For each of AI2-THOR's scenes, we generated the list of *pickupable* objects by selecting those on which the action "pick up" could be executed and identified a surface on which objects could be placed. Then for each scene, we performed two sets of trials with each pickupable object at the predetermined location. In the first set of trials, the object started in the robot's grasp (held object) while in the second set, the object started on the surface (placed object). For each object and set of trials, one before-image was recorded, and for each action performed, one after-image was recorded.

In the first trial (held object), all objects could have at least 3 distinct actions performed on them: they could all be dropped, thrown, and put down. Some objects permitted additional actions: dirtying, toggling, filling, breaking, cooking, and opening. In the second trial (placed object), all objects could be pushed, pulled, and picked up. Again, some objects allowed additional actions, the same as for held objects, plus slicing. We executed all possible actions for each object to generate the after-images, making sure that every before-action-after triple in a given scene takes place in the same location. This approach yielded 13958 before-action-after triples (6979 for each trial). There were 412 object-action pairs, across 15 actions and 62 objects, across all 120 scenes.

We then cleaned the data, removing from the dataset any triples whose associated actions failed; for example, placing an object can fail if there is no space in which to place it. We also removed any triples where the object was no longer visible in the after-image, as well as those whose after-image was not visibly different from the before-image, as determined by pixel-wise distance between images. Finally, we removed actions that could be executed on only one object. This filtering removed 3 actions: "use up", "slice", and "cook".

As summarized in Table 1, after filtering, our collection contained 11154 triples, built out of 12 actions, 62 object types, and 403 action-object pairs, across 120 scenes. It contained in total 3746 unique before-images and 11154 unique after-images.[3] Note also that while this collection is not large, it can also be expanded, as it was collected from a virtual environment. Specifically, by randomizing object color, scene lighting, agent perspective, and other aspects of the virtual environment, it would be possible to increase visual diversity while not compromising the controlled nature of the image generation.

In the next section, we focus on building a dataset for a contrast set task; however, we note that this collection could be assembled into datasets for other tasks as well. For example, an *action inference* task could be created by presenting a matching before- and after-image, and training a model to predict the action connecting the two images.

3.2 Dataset Creation

We converted our collection of triples described above into contrast sets consisting of 4 images. For each (before-image, action, after-image) triple, the corresponding contrast set consists of the true after-image, and 2 after-images generated starting from the same before-image by executing different actions on the same object. In addition, we include 1 image that is the result of executing the same action on the same object type situated in a scene distinct from the one depicted in the before-image. The inclusion of this image ensures that a model cannot simply pick images that are consistent with the given action; they must also correspond to the scene of the before-image. Below we provide the details of the dataset creation to highlight its fine-grained design.

Some before-images had fewer than 3 after-images (same object, same scene, different action); thus, triples containing this before-image could not

[3] The list of actions, together with the number of object types on which they have been executed, is provided in the Supplementary Material (Table 5)

Action	# Objs.	Action	# Objs.
Push	20	Pull	20
Pick Up	20	Put Down	20
Throw	20	Drop	20
Break	11	Fill	7
Dirty	7	Toggle	3
Open	2	Close	2

Table 2: Task: Number of object types per action (before-image-action pair) in the filtered contrast set dataset.

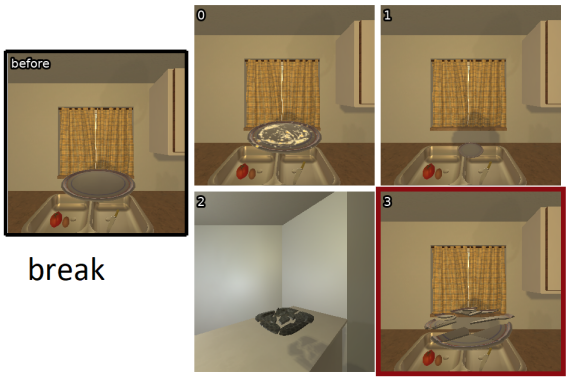

Figure 2: Simulation task: given a before-image and an action label, select from a contrast image set the one that represents the action simulation (after-image).

be converted into contrast sets. We also discarded examples where certain action pairs occur in the same contrast set, as their outcomes are too visually similar. For example, the outcome of putting any given object down is not consistently distinguishable from the outcome of throwing it; similarly, pushing and pulling objects can be visually similar. This particular filtering step results in significant cuts to the dataset size. This approach yields 4451 datapoints, 152 object-action pairs over 12 actions and 20 objects, across 109 scenes. The object-action distribution is given in Table 2.

Task Figure 2 illustrates the task: given the before-image (the robot is in a kitchen, holding a plate) and the action label, the model has to select the corresponding after-image (image 3) from the given contrast set, whose candidates are generated from the before-image by having the robot dirty (image 0), throw (image 1), and break (image 3) the plate. Finally, image 2 results from the robot breaking the plate in a different kitchen.

Human experiment To determine how challenging this task was for humans, we also solicited annotations from students. Each of the 7 annotators was randomly assigned 60 contrast sets to annotate using the `makesense.ai` platform. Annotators saw images like Figure 2 but without the red box indicating the correct response. They were told to select the index of the image corresponding to performing the listed action on the object in the before-image.

As each contrast set was annotated by two annotators, this yielded annotations for 210 contrast sets. Overall, annotators achieved high accuracy (81.4%) and annotator agreement (84.9%), compared to random chance (25% for both). This evidences the quality of the data and generation process, as well as the feasibility of the task.

4 Models

In this section, we describe baseline models designed to assess the role of the embeddings involved: those of the action-label, before- and after-images. We compare models using frozen visual features extracted from various state-of-the-art feature extractors.[4]

Training method Considering the contrastive nature of our task, inspired by CLIP's loss function (Radford et al., 2021), we use the InfoNCE loss proposed by van den Oord et al. (2018). This loss minimizes the distance with the ground-truth vector while jointly maximizing distances from the "negative" elements in the contrast set (i.e. the ones generated from different actions). Let $C = (\{\mathbf{c}_1, \mathbf{c}_2, ..., \mathbf{c}_n\}, k)$ be a contrast set where $\mathbf{c}_i$ are the vector representations for each image, and k is the index of the true after-image. Furthermore, let $\tilde{\mathbf{a}}$ be the predicted after-image representation. Then we compute the probability that image i is the true after-image as follows:

$$p(i|C, \tilde{\mathbf{a}}) = \frac{e^{\mathbf{c}_i \cdot \tilde{\mathbf{a}}^\top}}{\sum_{j=1}^{n} e^{\mathbf{c}_j \cdot \tilde{\mathbf{a}}^\top}}$$

The training loss is $\mathcal{L}(C, \tilde{\mathbf{a}}) = -\log(p(k|C, \tilde{\mathbf{a}}))$, cross-entropy loss. At test time, our model predicts the true after -mage index as $\text{argmax}_i\, p(i|C, \tilde{\mathbf{a}})$. The models presented in the following paragraphs all use the approach described above, with the exception of the `visual-only` model, for which there is no training time.

[4]Details regarding model architectures and parameter counts can be found in the Supplementary Material (Figure 6, Table 8)

Unimodal Baselines Inspired by Thomason et al. (2019), we verify the impact of visual and language input by considering unimodal baselines. First, we evaluate a model that learns to select as after-image an image of the contrast set whose visual vector is closer to the before-image, based on cosine similarity (Hence, `visual-only`.) In this work, we explore two different visual extractors: 1) *MOCA*: we rely on the visual encoder pretrained using ALFRED data by Singh et al. (2021); 2) *CLIP*: we use the visual features extracted by the CLIP image encoder (Radford et al., 2021). We use the former because it has been extensively used for other Embodied AI tasks (e.g., (Pashevich et al., 2021)) while we use the latter because Khandelwal et al. (2022) showed that CLIP provides very effective representations for Embodied AI tasks.

Second, we evaluate a model that learns to select the after-image simply based on the action name (`action-name`). The model architecture is simple: first, the action label is converted into a (learned) embedding. Next, each of the after-images is turned into a feature vector using one of the aforementioned frozen pre-trained feature extractors. Then, both the action embedding and each of the after-image feature vectors are projected into a shared space via linear projections. The projected action embedding is taken to be $\tilde{\mathbf{a}}$, and the appropriate projected image representations to be $\mathbf{c}_1, \ldots, \mathbf{c}_n$. Then training and prediction are performed as described above.

Multimodal Baselines We design three multimodal baseline models. The first of them departs from the mainstream end-to-end approach used in Embodied AI, implementing the formal semantics view of "verbs as functions". Following Baroni and Zamparelli (2010), we represent each action as a matrix that is learned from its before- and after-images. Unlike Baroni and Zamparelli (2010), we rely on a feed-forward network with a single linear layer, with no bias vector or activation function (hence, `Action-Matrix`). Given the before-image representation (derived from either MOCA or CLIP), we use the action identifier to extract the associated *action matrix*, which is composed with the before-image vector through matrix multiplication to return the predicted after-image vector.[5]

In addition, we design two models trained end-to-end to select the after-image given the before-image representation and the action name represented as a vector that belongs to a learned embedding matrix. The two variants of this architecture are: `Concat-Linear`: a single matrix, acting as a linear layer, which is multiplied by the concatenation of *before* vector and action vector; and `Concat-Multi`: a 2-layer, feed-forward network, including batch-normalization, dropout and activation layers, taking as input the same concatenated vector as above.

5 Experiments

It is important that embodied AI agents be capable of making sound predictions in unseen scenarios is a very important requirement for real-world applications. For this reason, we carefully design two hold-out procedures when splitting the dataset into train, validation and test data, in order to estimate the generalization capabilities of the baselines

Unseen Scenes. Following the standard practice in AI2-THOR[6], we split the set of scenes $\mathbb{S}$ into *seen* scenes $\mathbb{S}_s$ and *unseen* scenes $\mathbb{S}_u$ and use the former to create the training and validation sets (2856 and 684 datapoints, resp.) and the latter for the test set (820 datapoints); in other words, at test time the models will receive images illustrating one of the four possible types of virtual rooms (e.g. a bathroom) in a configuration that they have not seen during training/validation.[7] This method allows us to test for generalization at the scene level, in order to see if the models' visual features are fine-grained enough to generalize across different configurations of objects appearing in different rooms.

Unseen Objects. Given the set of object classes $\mathbb{O}$, we split it into a *seen* subset $\mathbb{O}_s$, and an *unseen* subset $\mathbb{O}_u$. We define the training and validation splits so that they contain objects that belong to $\mathbb{O}_s$ only. On the other hand, we define the test split so that it contains only objects belonging to $\mathbb{O}_u$. We manually selected the unseen objects (*laptop*, *cup*, and *book*) so as to guarantee the complete coverage of actions at test time. The test set contains 908 datapoints, the training and validation sets amount to 2762 and 690 datapoints, respectively.[8] This

[5]The model we implemented is quite close to the Siamese Network proposed in (Wang et al., 2016) which we discovered while writing the paper. The code is not available; hence, we could not evaluate it.

[6]The standard practice in AI2-THOR is reported in the official documentation.

[7]Detailed statistics in Supplementary Material (Table 7)

[8]Detailed statistics in Supplementary Material (Table 6)

procedure will help showing how well models can understand the effect of known actions on novel objects.

6 Results

6.1 Accuracy

As illustrated in Figure 3, the unimodal baseline models are close to chance levels both in the unseen scenes and unseen objects setting. In both cases, the visual-only model performs better than the action-name one, independently of the visual features used. In general, for the visual-only baseline, CLIP features seem to be better than MOCA ones, possibly due to their higher expressiveness when used for visual similarity, as is the case here. The drop in performance for the model `action-name` on unseen scenes is justified by the fact that the model is learning to make predictions based on the action embeddings only. At training time, the model is exposed to seen scenes only; therefore, the action embeddings are likely to encode patterns associated with these scenes only. Such representations do not truly consider other context-specific information and therefore do not allow the model to generalize well on unseen scenes.

In both splits, all models perform better when based on the MOCA visual features than CLIP. In particular, MOCA-based models generalize better across scenes. This could be associated with the fact that our task requires very fine-grained information that a CLIP model may not be able to retain. Thanks to the embodied nature of the task solved by MOCA, its visual features learn more fine-grained representations of the AI2-THOR environment.

The `Action Matrix` and `Concat-Multi` models reach a similar accuracy in both settings; however, the latter suffers less from the coarser-grained representation provided by the CLIP features in the holding out scene setup. When comparing model and human performance, it is clear that the weak point of these models is the crucial ability to generalize their action grounding to unseen objects. Although the images are not fully naturalistic, humans reach 83% accuracy, whereas the best models are around 70% in the unseen objects split when given the MOCA features that are specifically fine-tuned on AI2-THOR, and lower than 60% with CLIP.

Another important consideration is that, following previous work on affordance prediction (Deng et al., 2021), our baselines benefit from the gold information of the action being performed. The `Action-Matrix` model is designed to learn a transformation matrix for each action in the dataset as in Wang et al. (2016). This demonstrates that having a way to learn "verb" related information is useful to match human performance and generalize better. We believe that this represents a very interesting challenge for embodied AI models that are fully end-to-end (e.g. (Suglia et al., 2021)). Introducing models that can reason over time by "simulating" the action being performed represents a potentially interesting avenue for future work.

6.2 Error Analysis

In order to understand models' performance, we completed a detailed analysis of the best performing models, `Action Matrix` and `Contact-Multi`, focusing on the unseen object split. First, we computed accuracy by action. Looking at Table 3, the most successful action is by far "drop" for all models: this could be due to the change of perspective that happens after dropping an object. Moreover, this action can be applied to all the objects in the dataset, thus we can assume that the model has more data to learn from and generalizes better.

While the MOCA-based `Concat-Multi` model holds the majority of the best values, it can be noted that the CLIP-based version scores better on actions for which there are fewer datapoints in the training set ("close", "open", "toggle"). We hypothesize this is due to the larger pretraining dataset that is used by CLIP, whose higher capacity might lead to better generalization with less data.

Second, we computed the accuracy by objects. This analysis shows that of the three objects unseen during training, "cup" is the one with highest scores across actions. We hypothesize that this is due to the presence of other visually similar objects in the training set, for instance "mug" and "bowl".[9]

Finally, we exploited the directly interpretable action representations computed by the `Action Matrix` MOCA-based model, and computed the nearest neighbors of each verb. As reported in Table 4, the model groups together actions that cause a change of position (e.g., "drop", "push", "pull"), and actions that cause a change in object appearance (e.g. "break", "dirty", "toggle".) This is a first promising step towards grounded verb

[9] Details in Table 9 of the Supplementary Material

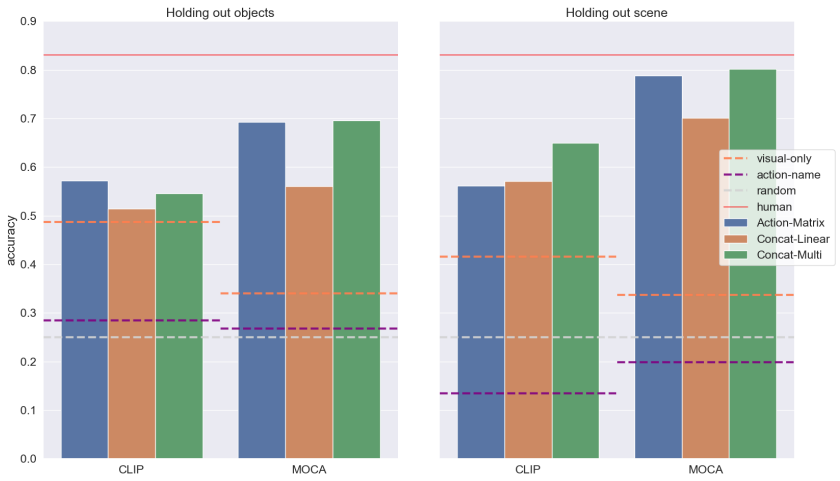

Figure 3: Results for contrast-task accuracy, training with contrastive-loss (average over 5 random initializations). Dashed lines represent baselines: gray for chance, orange for visual-similarity, purple for the action-name after-image-only model. The red line represents human accuracy averaged over all procedures.

Model	break	close	dirty	drop	fill	open	pickUp	pull	push	put	throw	toggle
AM-CLIP	0.406	0.516	0.553	0.907	0.495	0.348	0.540	0.439	0.572	0.628	0.619	0.447
CM-CLIP	0.576	**0.573**	0.502	0.627	0.546	**0.722**	0.593	0.521	0.546	0.607	0.622	**0.630**
AM-MOCA	0.580	0.541	0.678	0.991	0.814	0.409	0.772	0.449	**0.675**	0.782	**0.769**	0.567
CM-MOCA	**0.651**	0.482	**0.719**	**0.964**	**0.848**	0.426	**0.805**	**0.536**	0.668	**0.855**	0.670	0.573

Table 3: Accuracy per action of the best scoring models; results for the test set of the **object split**. Best values on columns in bold.

Action	Nearest neighbors (sorted)		
break	dirty	open	toggle
close	break	dirty	put
dirty	break	pull	open
drop	push	pull	pickUp
fill	put	pull	throw
open	dirty	break	fill
pickUp	dirty	fill	break
pull	put	push	throw
push	pull	throw	put
put	throw	pull	push
throw	put	push	pull
toggle	dirty	break	pull

Table 4: 3 nearest action neighbor actions for each action, using action representations from the MOCA Action-Matrix model (trained on the object split).

representations. We believe much could be learned by combining formal semantics findings on verbs and the embodied AI literature.

7 Conclusion

In this work, we define `ACT-Thor`, a controlled benchmark for embodied action understanding in simulated environments. Compared to similar benchmarks presented in the literature, we propose a more systematic benchmark for studying the capability of multimodal machine learning models to understand the effects of actions on the objects. Additionally, our contrast set formulation allows us to precisely pinpoint the distinctions that models should learn, providing a robust and reliable benchmark for studying action grounding.

To assess the quality of our dataset, we first completed a human evaluation demonstrating that humans can complete this task with 83% accuracy. Then, we evaluated several unimodal models, and multimodal models using several state-of-the-art visual feature extractors such as CLIP (Radford et al., 2021). By inspecting models that learn a matrix for each action, we show that the representations learned via our dataset favor the emergence of clusters of actions associated with a change of position, and actions that cause a change in object appearance. These are two salient visual transformations objects undergo, but more could be learned by extending the dataset with other actions or other objects, and by proposing controlled settings inspired by the theoretical view on verb semantics.

Our work demonstrates the potential of virtual environments to pursue this line of research.

We argue that `ACT-Thor` represents an experimental benchmark for studying the action understanding capabilities of machine learning models, an important skill for embodied AI agents. This is especially important moving forward for studying the capabilities of generalist agents that can solve multiple tasks (Reed et al., 2022).

Acknowledgments

Michael Hanna thanks the Erasmus Mundus European Masters Program in Language and Communication Technologies for its support.

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 conference on empirical methods in natural language processing*, pages 1183–1193.

Levin Beth. 1993. *English Verb Classes and Alternations. A Preliminary Investigation.* University of Chicago Press.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee.

Shengheng Deng, Xun Xu, Chaozheng Wu, Ke Chen, and Kui Jia. 2021. 3d affordancenet: A benchmark for visual object affordance understanding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1778–1787.

J Feldman and S. Narayanan. 2004. Embodied meaning in a neural theory of language. *Brain and Language*, 89(2):385–392.

Charles J. Fillmore. 1982. *Frame semantics*, pages 111–137. Hanshin Publishing Co., Seoul, South Korea.

Charles J. Fillmore, Collin F. Baker, and Hiroaki Sato. 2002. The FrameNet database and software tools. In *Proceedings of the Third International Conference on Languag Resources and Evaluation*, volume IV, Las Palmas. LREC, LREC.

G. Frege. 1892. Uber sinn und bedeutung. In *Zeitschrift fuer Philosophie un philosophische Kritik*.

Qiaozi Gao, Malcolm Doering, Shaohua Yang, and Joyce Chai. 2016. Physical causality of action verbs in grounded language understanding. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1814–1824, Berlin, Germany. Association for Computational Linguistics.

Qiaozi Gao, Shaohua Yang, Joyce Chai, and Lucy Vanderwende. 2018. What action causes this? towards naive physical action-effect prediction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 934–945, Melbourne, Australia. Association for Computational Linguistics.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. 2020. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1307–1323, Online. Association for Computational Linguistics.

James J Gibson. 1977. The theory of affordances. *Hilldale, USA*, 1(2):67–82.

Meera Hahn, Jacob Krantz, Dhruv Batra, Devi Parikh, James Rehg, Stefan Lee, and Peter Anderson. 2020. Where are you? localization from embodied dialog. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 806–822.

Oana Ignat, Santiago Castro, Hanwen Miao, Weiji Li, and Rada Mihalcea. 2021. WhyAct: Identifying action reasons in lifestyle vlogs. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4770–4785, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. 2022. Simple but effective: Clip embeddings for embodied ai. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 14829–14838.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Kumar Gupta, and Ali Farhadi. 2017. Ai2-thor: An interactive 3d environment for visual ai. *ArXiv*, abs/1712.05474.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing.

Dipendra Kumar Misra, Kejia Tao, Percy Liang, and Ashutosh Saxena. 2015. Environment-driven lexicon

induction for high-level instructions. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 992–1002, Beijing, China. Association for Computational Linguistics.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramithu, and Dilek Hakkani-Tur Gokhan Tur and. 2022. TEACh: Task-driven Embodied Agents that Chat. In *Conference on Artificial Intelligence (AAAI)*.

Alexander Pashevich, Cordelia Schmid, and Chen Sun. 2021. Episodic transformer for vision-and-language navigation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15942–15952.

Sandro Pezzelle, Claudio Greco, Greta Gandolfi, Eleonora Gualdoni, and Raffaella Bernardi. 2020. Be Different to Be Better! A Benchmark to Leverage the Complementarity of Language and Vision. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2751–2767, Online. Association for Computational Linguistics.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning transferable visual models from natural language supervision. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8748–8763. PMLR.

Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, et al. 2022. A generalist agent. *arXiv preprint arXiv:2205.06175*.

Michaela Regneri, Marcus Rohrbach, Dominikus Wetzel, Stefan Thater, Bernt Schiele, and Manfred Pinkal. 2013. Grounding action descriptions in videos. *Transactions of the Association for Computational Linguistics*, 1:25–36.

Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc.

Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. 2019. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9339–9347.

Karin Kipper Schuler. 2005. *VerbNet: A Broad-Coverage, Comprehensive Verb Lexicon.* Ph.D. thesis, PhD Dissertation.

Lanbo She and Joyce Chai. 2017. Interactive learning of grounded verb semantics towards human-robot communication. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1634–1644, Vancouver, Canada. Association for Computational Linguistics.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Kunal Pratap Singh, Suvaansh Bhambri, Byeonghwi Kim, Roozbeh Mottaghi, and Jonghyun Choi. 2021. Factorizing perception and policy for interactive instruction following. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1888–1897.

Alessandro Suglia, Qiaozi Gao, Jesse Thomason, Govind Thattai, and Gaurav Sukhatme. 2021. Embodied bert: A transformer model for embodied, language-guided visual task completion. *arXiv preprint arXiv:2108.04927*.

Alane Suhr, Claudia Yan, Jack Schluger, Stanley Yu, Hadi Khader, Marwa Mouallem, Iris Zhang, and Yoav Artzi. 2019. Executing instructions in situated collaborative interactions. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2119–2130, Hong Kong, China. Association for Computational Linguistics.

Ronen Tamari, Chen Shani, Tom Hope, Miriam R L Petruck, Omri Abend, and Dafna Shahaf. 2020. Language (re)modelling: Towards embodied language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6268–6281, Online. Association for Computational Linguistics.

Jesse Thomason, Daniel Gordon, and Yonatan Bisk. 2019. Shifting the baseline: Single modality performance on visual navigation & qa. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1977–1983.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Xiaolong Wang, Ali Farhadi, and Abhinav Gupta. 2016. Actions ~ transformations. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2658–2667.

Terry Winograd. 1972. Understanding natural language. *Cognitive Psychology*, 3:1–191.

Claudia Yan, Dipendra Misra, Andrew Bennnett, Aaron Walsman, Yonatan Bisk, and Yoav Artzi. 2018. Chalet: Cornell house agent learning environment. *arXiv preprint arXiv:1801.07357*.

Hongsang Yoo, Haopeng Li, Qiuhong Ke, Liangchen Liu, and Rui Zhang. 2021. Precondition and effect reasoning for action recognition. *arXiv preprint arXiv:2112.10057*.

Rowan Zellers, Ari Holtzman, Matthew Peters, Roozbeh Mottaghi, Aniruddha Kembhavi, Ali Farhadi, and Yejin Choi. 2021. PIGLeT: Language grounding through neuro-symbolic interaction in a 3D world. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2040–2050, Online. Association for Computational Linguistics.

A Probing visual features

For the purpose of our experiments, we require visual representations that are expressive and fine-grained so that it is possible to distinguish both object-dependent and scene-dependent features. Therefore, we rely on the object detection task—similar to the one performed in the ImageNet setup (Deng et al., 2009)—as a diagnostic task for the quality of the visual features. Particularly, we assume that high quality visual features will yield high performance in the selected diagnostic task. Furthermore, implementing this sanity check is important because most of our visual feature extractors are pretrained on real-world images, and we do not know how well they will perform on AI2Thor synthetic images.

Borrowing from the literature in Computer Vision, we define our task as the combination of object classification and bounding-box regression: given an input image, the model should produce a tuple (x_0, y_0, x_1, y_1) of coordinates representing the top-left and bottom-right corners of the rectangular bounding-box, where $x_i \in [0, \text{max-image-width}]$ and $y_i \in [0, \text{max-image-height}]$, coupled with the hypothesized object label $l \in \{0, 1, ..., C-1\}$ where C is the number of unique objects in the dataset. Performance over these two tasks can be measured the Mean Average Precision (MAP) metric, which includes information of both intersection-over-union of the predicted boxes with the ground-truth and object class precision.

We use the TorchVision[10] implementation of an end-to-end object detection architecture, Faster RCNN (Ren et al., 2015): this is composed of a CNN backbone, coupled with a Region Proposal Network that gets trained in parallel and two distinct fully connected heads to predict bounding boxes (of size 4) and object class label (of size 62). This architecture is trained to predict both bounding box and object location of the only relevant object in the image, which is the one subject to the current action in the contrast set from which it comes; for the bounding box regression it is used an MSE loss, while for object detection it is used a Cross Entropy loss. In order to match the data distributions of the training set of these models, we normalize AI2Thor images channel-wise, by computing the mean and standard deviation for each channel over the whole dataset. We also crop inputs to size 224×224 for compatibility reasons.
Crucially, for evaluating our models as feature extractors we need to keep the convolutional backbone fixed to its initial weights: in this way, the better the visual features extracted, the easier it will be for the classification head to learn this diagnostic task. Nonetheless, for comparison we train also object detection models where the backbone is fine-tuned, and test if their performance is different from the frozen versions.

The TorchVision reference implementation is based on a ResNet50 backbone, which is the most similar to the models in our experiments, and it reaches a mAP value of 37.0 over COCO val2017 (Lin et al., 2014).

For frozen models, we obtain results that are lower than the baseline. We hypothesize this is due firstly to dataset size, which does not reach hundreds of thousands of samples as for other object detection benchmarks. However, it must be taken into account also the difference in our object detection task compared to more general ones: while in this work we assume just one interesting object with a single bounding box per sample, in works focused on object detection it is common to have a high number of boxes per image, spanning several different categories. This variation in the task framing could explain, at least in part, the differences in performance from broader works.

[10] https://pytorch.org/vision/stable/models.html#object-detection-instance-segmentation-and-person-keypoint-detection

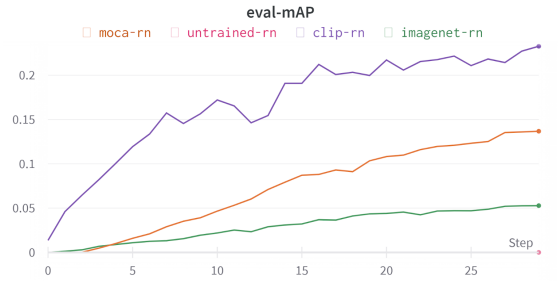

Figure 4: Object Detection: MAP on validation set per epoch.

Both MOCA and CLIP based models reach satisfying performances, therefore we consider their features to be good enough to be used in other experiments; we will include also the ImageNet model for comparison, despite its poor performance on this task. Additionally, this choice is also supported by previous work that shows that both MOCA and CLIP are appropriate vision models for AI2Thor synthetic scenes, as demonstrated by (Singh et al., 2021) and (Khandelwal et al., 2022), respectively.

B Additional Dataset, Model, and Results Information

Action	# Objs.	Action	# Objs.
Push	62	Pull	62
Pick Up	62	Put Down	62
Throw	62	Drop	62
Break	11	Fill	8
Dirty	8	Toggle	3
Open	2	Close	2

Table 5: List of the 12 actions used to generate after images and the number of object types per action.

Split	Object	# Actions	# Samples	Tot.
Test	Book	7	92	908
	Cup	9	316	
	Laptop	9	500	
Train	Bottle	8	52	2762
	Bowl	9	393	
	Box	7	188	
	Candle	7	115	
	CellPhone	8	128	
	Cloth	7	104	
	Egg	7	77	
	Kettle	8	98	
	Mug	9	350	
	Pan	7	151	
	Plate	8	278	
	Pot	8	221	
	Potato	6	65	
	Statue	7	290	
	Vase	7	259	
	WateringCan	6	23	
	WineBottle	7	35	
Valid	Bottle	8	14	690
	Bowl	9	92	
	Box	7	40	
	Candle	7	25	
	CellPhone	7	32	
	Cloth	6	32	
	Egg	4	11	
	Kettle	7	16	
	Mug	9	113	
	Pan	7	41	
	Plate	8	66	
	Pot	8	49	
	Potato	6	16	
	Statue	7	74	
	Vase	7	71	
	WateringCan	4	5	
	WineBottle	4	9	
				4360

Table 6: Action and sample counts for the Object split. 'Actions' column represents the number of unique actions performed on that object.

Split	Envt.	# Configs	# Samples	Total
Test	bathroom	3	36	820
	bedroom	5	158	
	kitchen	5	367	
	living room	5	277	
Train	bathroom	20	152	2856
	bedroom	20	472	
	kitchen	20	1513	
	living	20	773	
Valid	bathroom	5	44	693
	bedroom	5	117	
	kitchen	5	337	
	living	5	195	
				4360

Table 7: Environment information for the scene split.

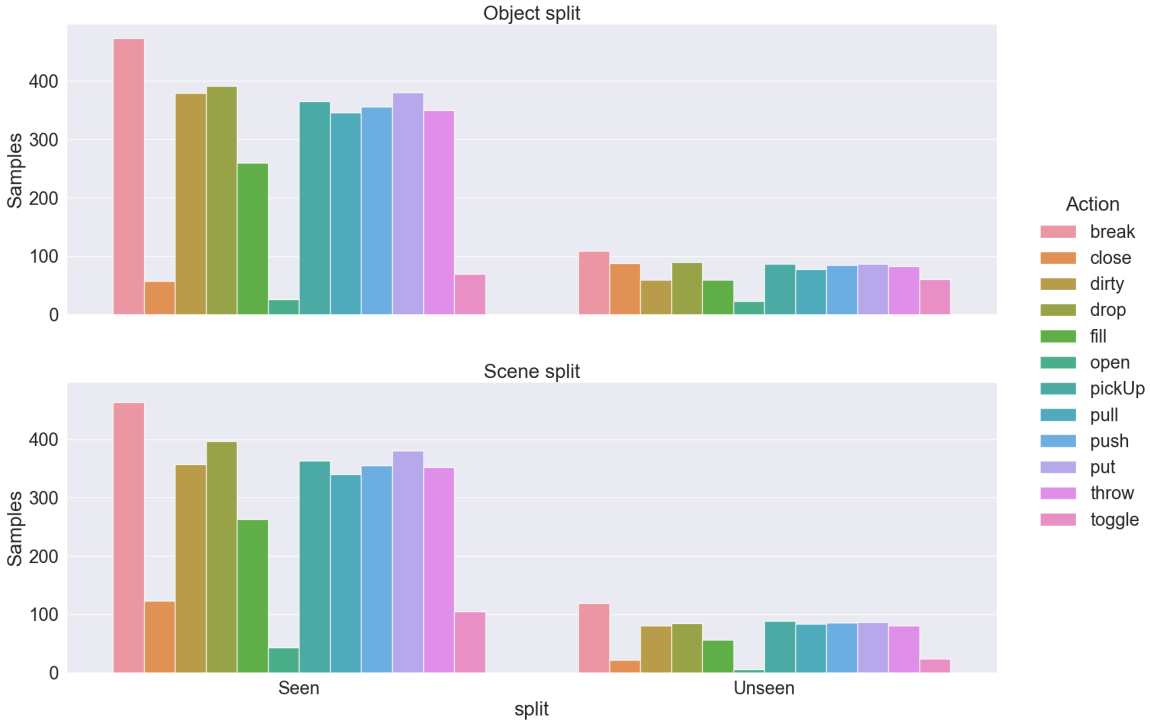

Figure 5: Distribution of actions between seen and unseen items per splitting procedure.

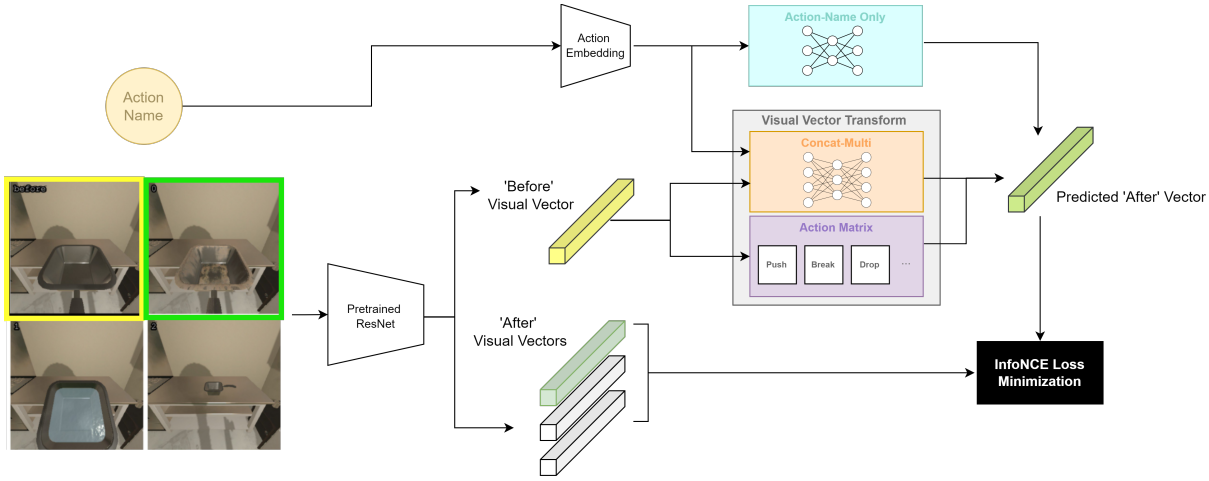

Figure 6: Schema of model architecture. Note: the arrow linking before-vector and vector transformation models represents, respectively: the dot product with the correct action matrix, for the Action-Matrix model; feedforwarding the vector as network input for the Concat-Multi model. We omit the Concat-Linear model since it uses the same procedure of the Concat-Multi.

VF	Embedding?	Model	Params
MOCA	No	Action-Matrix	259,308
		Concat-linear	23,520
		Concat-Multi	36,680
	Yes	Action-Matrix	259,308
		Concat-linear	375,296
		Concat-multi	539,520
CLIP	No	Action-Matrix	12,582,912
		Concat-linear	1,061,888
		Concat-Multi	1,602,421
	Yes	Action-Matrix	12,582,912
		Concat-linear	599,808
		Concat-multi	764,032

Table 8: Summary of model parameter counts. 'VF' is the visual feature extractor used, while 'Embedding' denotes whether the model uses an embedding space of the same dimension (256) for both actions and visual vectors instead of the 1-hot-vector action encoding (and raw visual vectors).

Object	break	close	dirty	drop	fill	open	pickUp	pull	push	put	throw	toggle
Book	//	//	//	**1.000**	//	**0.426**	0.508	0.538	0.646	0.780	0.420	//
Cup	**0.726**	//	**0.719**	0.993	**0.847**	//	**0.966**	**0.620**	**0.711**	**0.931**	**0.727**	//
Laptop	0.603	**0.482**	//	0.942	//	//	0.787	0.498	0.649	0.825	0.694	**0.573**

Table 9: Action-wise accuracies for the `Concat-Multi` MOCA model, computed for each object in the test set ('//' used when the action is not available for that object). Best values for columns in bold.

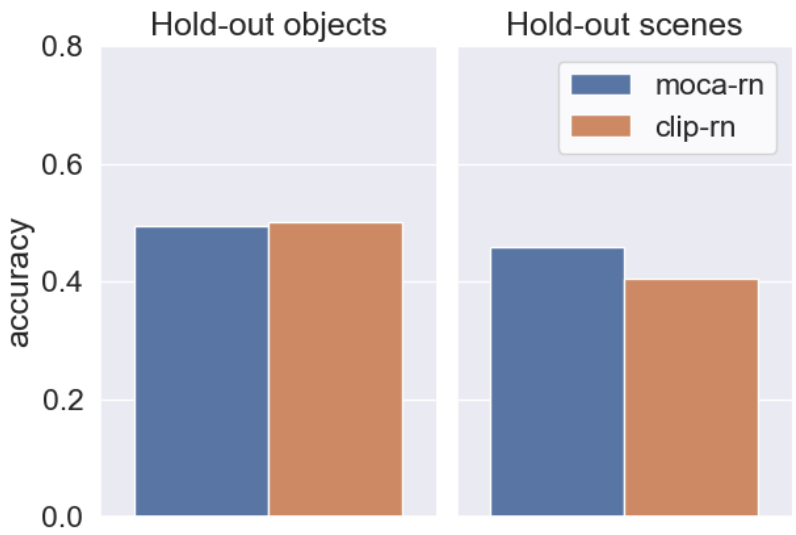

Figure 7: Results for accuracy of the least-squares regression model (average over 5 random splits).

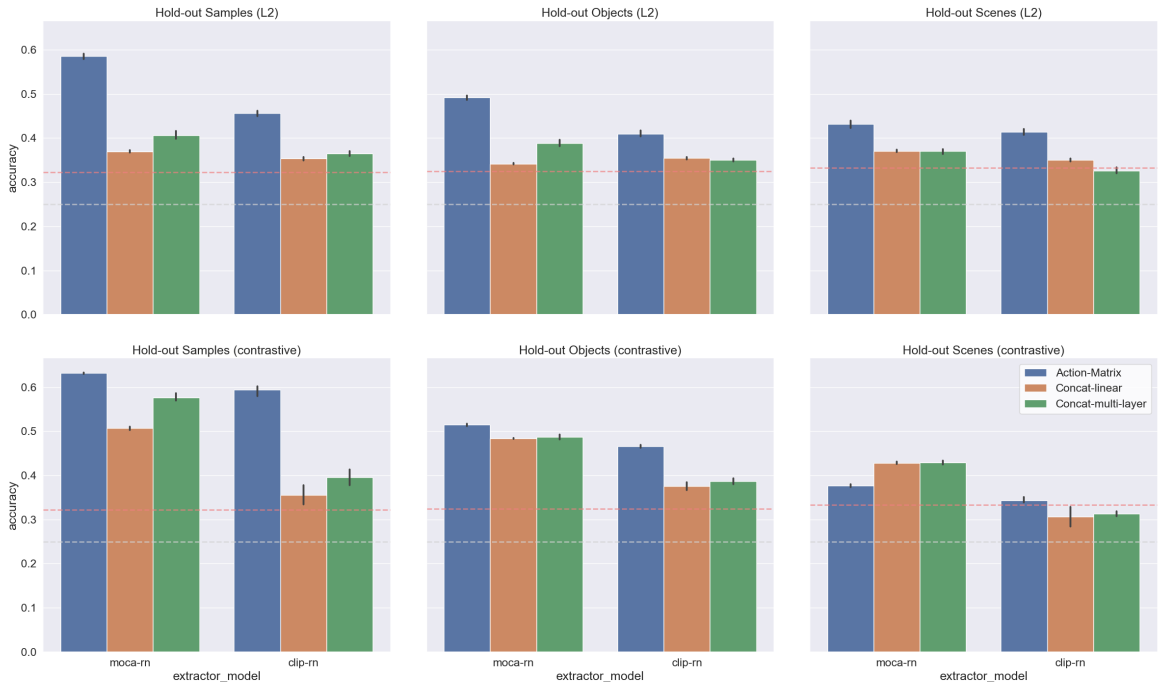

Figure 8: Comparison of L2 training procedure (top) and contrastive loss (bottom). Average over 5 random initializations. Dashed lines represent baselines (gray for chance, red for visual-similarity)

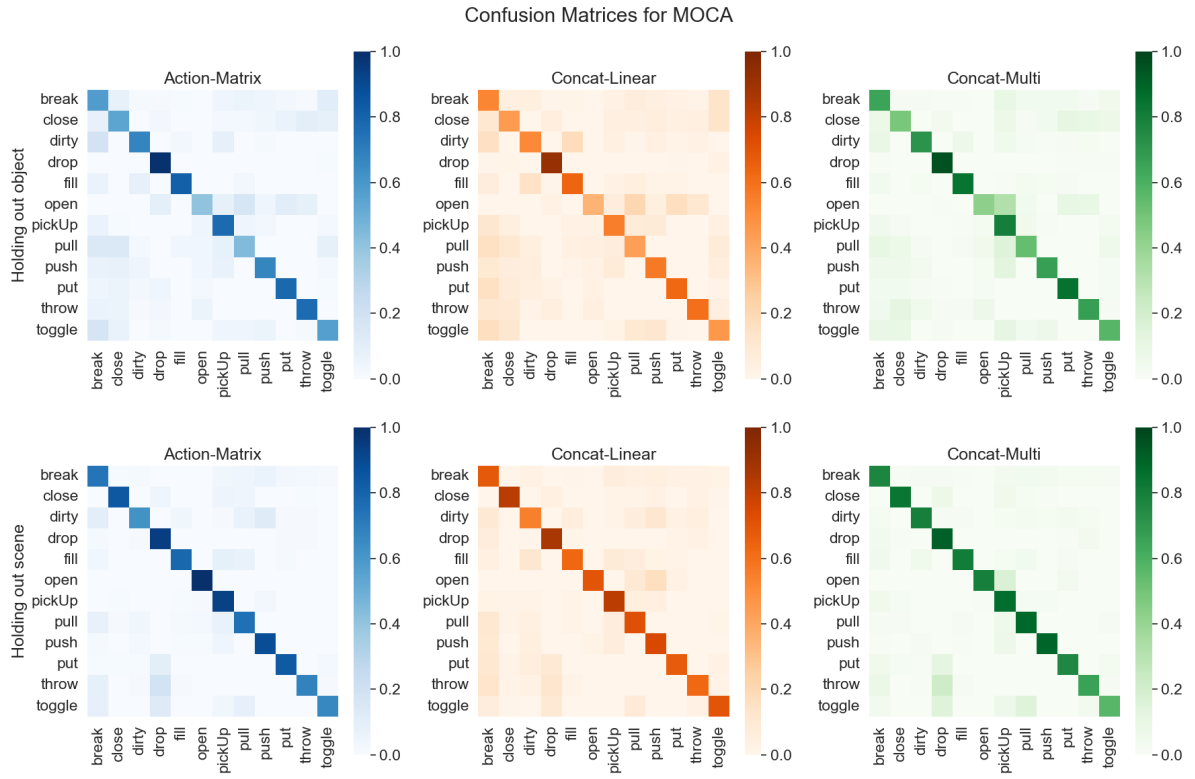

Figure 9: Action confusion matrix for the MOCA model with action embedding layer (row is ground truth, column represents prediction).

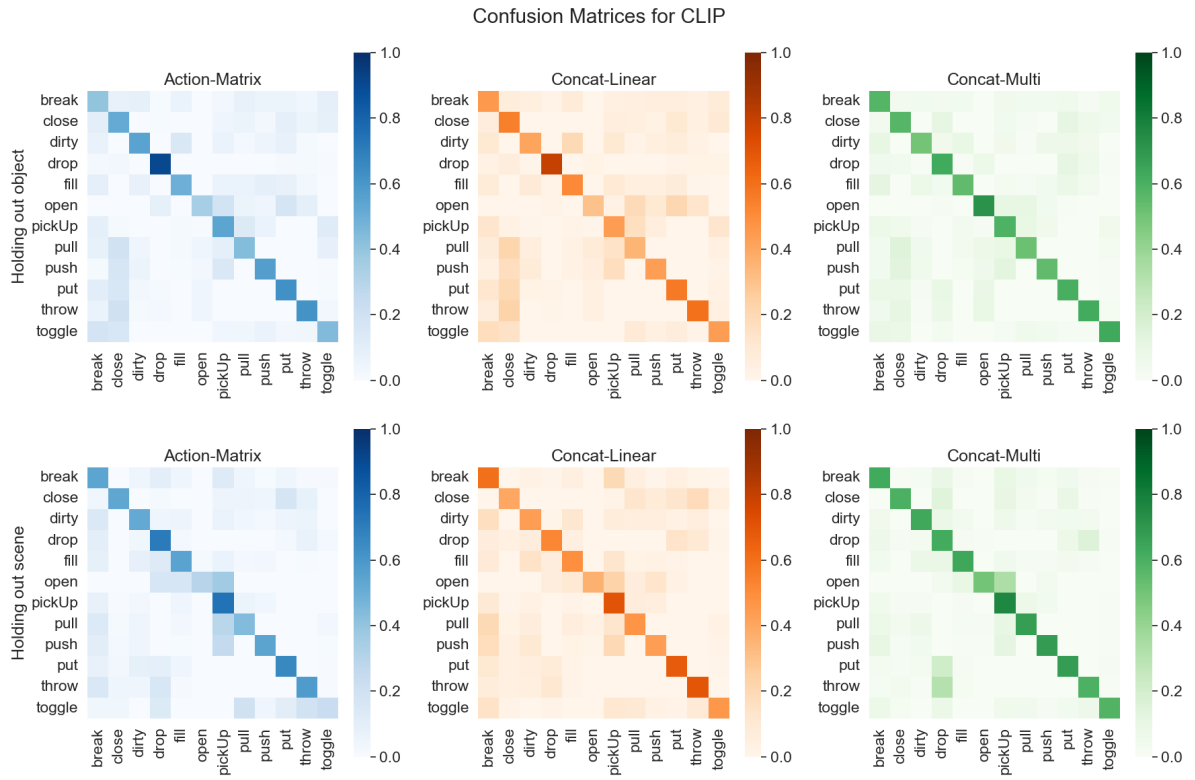

Figure 10: Action confusion matrix for the CLIP model with action embedding layer (row is ground truth, column represents prediction).