CLIPORT: What and Where Pathways for Robotic Manipulation

Anonymous Author(s)
Affiliation
Address
email

Abstract: How can we imbue robots with the ability to manipulate objects precisely but also to reason about them in terms of abstract concepts? Recent works in manipulation have shown that end-to-end networks can learn dexterous skills that require precise spatial reasoning, but these methods often fail to generalize to new goals or quickly learn transferable concepts. In parallel, there has been great progress in learning generalizable semantic representations for vision and language by training on large-scale internet data, however these representations lack the spatial understanding necessary for fine-grained manipulation. To this end, we propose a framework that combines the best of both worlds: a two-stream architecture with semantic (*what*) and spatial (*where*) pathways for vision-based manipulation. Specifically, we present CLIPORT, a language-conditioned imitation-learning agent that combines the broad semantic understanding of CLIP [1] with the spatial precision of Transporter [2]. Our end-to-end framework is capable of solving a variety of language-specified tabletop tasks from packing unseen objects to folding cloths, all without any explicit representations of object poses, instance segmentations, history, symbolic states, or syntactic structures. Experiments in simulation and hardware show that our approach is data efficient and generalizes effectively to seen and unseen semantic concepts. We even learn one multi-task policy for 10 simulated and 9 real-world tasks that shows better or comparable performance to single-task models.

Keywords: Manipulation, Vision Language Grounding, CLIP

1 Introduction

Ask a person to "get a scoop of coffee beans" or "fold the cloth in half" and they can naturally take abstract concepts like *scoop* or *fold* and ground them in concrete physical actions within an accuracy of a few centimeters. We humans do this intuitively, without explicit geometric or kinematic models of coffee beans or cloths. Moreover, we can generalize to a broad range of tasks and concepts from a minimal set of examples on what needs to be achieved. How can we imbue robots with this ability to efficiently ground abstract semantic concepts in precise spatial reasoning?

Recently, a number of end-to-end frameworks have been proposed for vision-based manipulation [2, 3, 4, 5]. While these methods do not use any explicit representations of object poses, instance segmentations, or symbolic states, they can only replicate demonstrations with a narrow range of variability and have no notion of the semantics underlying the tasks. Switching from packing red pens to blue pens involves collecting a new training set [2], or if using goal-conditioned policies, involves the user providing a goal-image from the scene [5, 6]. In realistic human-robot interaction settings, collecting additional demonstrations or providing goal-images is often infeasible and unscalable. A natural solution to both these problems is to condition policies with natural language. Language provides an intuitive interface for specifying goals and also for implicitly transferring concepts across tasks. While language-grounding for manipulation has been explored in the past [7, 8, 9, 10], these pipelines are limited by object-centric representations that cannot handle granular or deformable objects and often do not reason about perception and action in an integrated manner. In parallel, there has been great progress in learning models for semantic understanding [11, 12] and aligning representations of vision and language [13, 14, 15] by training on large-scale internet data. However, these representations lack the spatial understanding necessary for fine-grained manipulation.

Submitted to the 5th Conference on Robot Learning (CoRL 2021). Do not distribute.

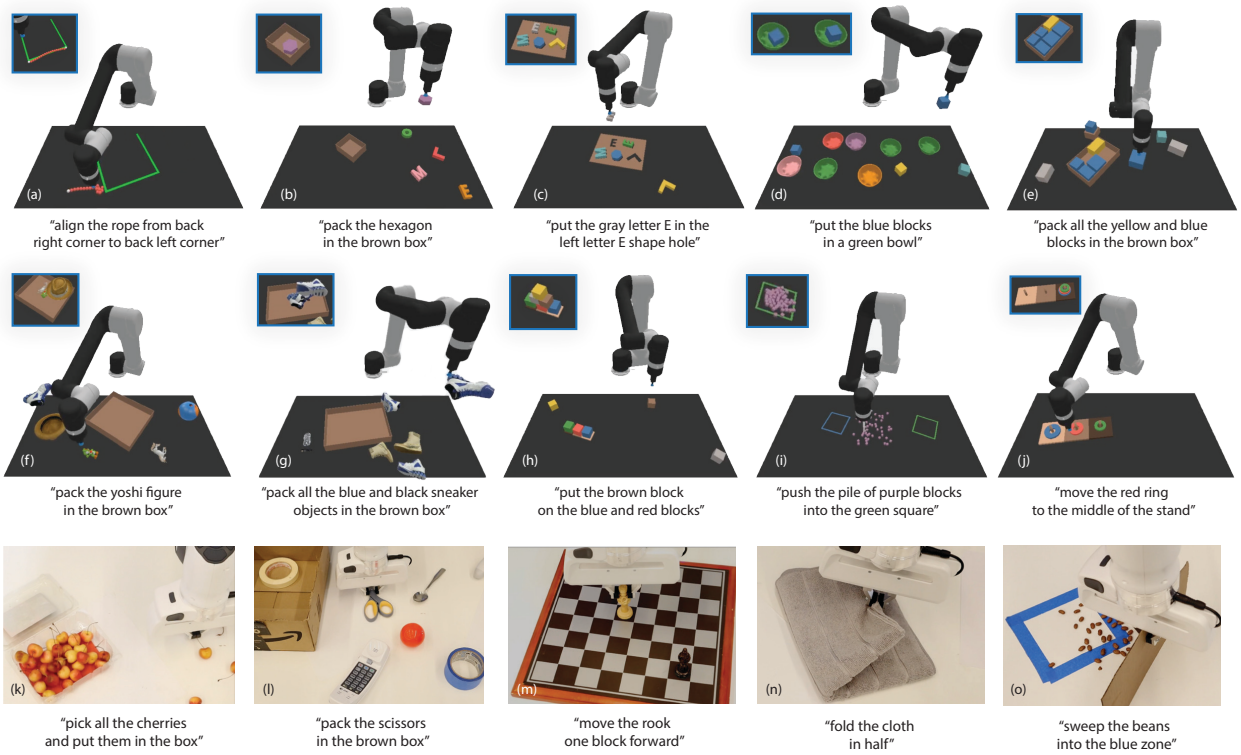

Figure 1. **Language-Conditioned Manipulation Tasks:** CLIPORT is a broad framework applicable to a wide range of language-conditioned manipulation tasks in tabletop settings. We conduct large-scale experiments in Ravens [2] on 10 simulated tasks (a-j) with 1000s of unique instances per task. See Appendix A for challenges pertaining to each task. CLIPORT can even learn one multi-task model for all 10 tasks that achieves better or comparable performance to single-task models. Similarly, we demonstrate our approach on a Franka Panda manipulator with one multi-task model for 9 real-world tasks (k-o; only 5 shown) trained with just 179 image-action pairs.

To this end, we propose the first framework that combines the best of both worlds: end-to-end learning for fine-grained manipulation with the broad generalization capability of vision-language grounding systems. We introduce a two-stream architecture for manipulation with **semantic** (*what*) and **spatial** (*where*) pathways broadly inspired by (or vaguely analogous to) the two-stream hypothesis is cognitive psychology [16, 17, 18]. Specifically, we present CLIPORT, a language-conditioned imitation-learning agent that integrates the semantic understanding of CLIP [1] with the spatial precision of Transporter [2]. Transporter has been applied to a wide range of rearrangement tasks from industrial packing [2] to manipulating deformable objects [6]. The key insight of the approach is formulating tabletop manipulation as a series of pick-and-place affordance predictions, where the objective is to directly *detect actions* rather than to *detect objects* and then learn a policy. This action-centric approach [19] is data efficient and effective at overcoming the limitations of explicit object representations. However, Transporter is a tabula rasa system that learns all visual representations from scratch and so every new task or goal requires collecting a new set of demonstrations. To address this problem, we bake in a strong semantic prior while learning policies. We condition our **semantic** stream with visual and language-goal features from a pre-trained CLIP model [1]. Since CLIP was trained to align image and language features from large-amounts of image-caption pairs from the internet, it provides a powerful prior for grounding semantic concepts that are common across tasks like categories, parts, shapes, colors, texts, and other visual attributes, all without a top-down pipeline that requires bounding boxes or segmentations [13, 14, 15, 20]. This architecture effectively allows us to formulate tabletop rearrangement as a series of language-conditioned affordance predictions, a predominantly vision-based inference problem, and thus benefit from the strengths of data-driven paradigms like scale and robustness.

To study these benefits, we conduct large-scale experiments in the Ravens [2] framework with a simulated suction-gripper robot. We propose 10 language-conditioned tasks with 1000s of unique instances per task that require both semantic and spatial reasoning (see Figure 1 a-j). CLIPORT is not only effective at solving these tasks, but surprisingly, it can even learn a multi-task model for all 10 tasks that achieves better or comparable performance to single-task models. Further, our evaluations indicate that our multi-task model can effectively transfer attributes like "pink block" across tasks, having never seen pink blocks or the word 'pink' in the context of the evaluation task. We also demonstrate our approach on a Franka Panda manipulator with one multi-task model for 9 real-world tasks trained with just 179 image-action pairs (see Figure 1 k-o).

In summary, our contributions are as follows:

- An extended **benchmark of language-grounding tasks** for manipulation in Ravens [2].
- **Two-stream architecture** for using internet pre-trained vision-language models for conditioning precise manipulation policies with language goals.
- **Empirical results** on a broad range of manipulation tasks, including multi-task models, validated with real-robot experiments.

We will release our code, models, and language-grounding benchmark upon publication.

2 Related Work

Vision-based Manipulation. Traditionally, perception for manipulation has centered around object detectors, segmentors, and pose estimators [21, 22, 23, 24, 25, 26]. These methods cannot handle deformable objects, granular media, or generalize to unseen objects without object-specific training data. Alternatively, dense descriptors [27, 28, 29] and keypoint representations [30, 31, 32] forgo segmentation and pose representations, but do not reason about sequential actions and struggle to represent scenes with variable numbers of objects. On the other hand, end-to-end perception-to-action models can learn precise sequential policies [2, 4, 6, 33, 34, 35], however, these methods have limited understanding of semantic concepts and rely on goal-images to condition the policies. Recently, Yen-Chen et. al [36] showed that pre-training on semantic tasks like classification and segmentation helps in improving efficiency and generalization of grasping predictions.

Semantic Models. With the advent of large-scale models [37, 38, 39], a number of methods for learning joint vision and language representations have been proposed [13, 14, 15, 20, 40]. However, these methods are restricted to bounding box or segmentation detections, which make them inapplicable for detecting things like piles of coffee beans or squares on a chessboard. Alternatively, recent works in contrastive learning forgo top-down object-detection and learn continuous representations by pre-training on unlabeled data [11, 12]. Recently, CLIP [1] applied a similar approach to align vision and language representations by training on image-caption pairs from the internet.

Language Grounding for Robotics. Several works have proposed systems for instructing robots with natural language [7, 8, 9, 10, 41, 42, 43, 44, 45, 46, 47]. However, these methods use disentangled pipelines for perception and action with the language primarily being used to guide the perception. As such, these pipelines lack the spatial precision necessary for tasks like folding cloths. Recently, Lynch et. al [48] proposed an end-to-end system for grounding language in continuous control, but it requires several hours of human teleoperation data for a single simulated desk setting.

Two-Stream Architectures are prevalent in action-recognition networks [49, 50, 51] and audio-recognition systems [52, 53]. In robotics, Zeng et. al [54] and Jang et. al [55] have proposed two-stream pipelines for affordance predictions of novel objects. The former requires goal-images and the latter is restricted to one-step grasps with single-category goals. In contrast, our framework provides a rich and intuitive interface with composable language commands for sequential tasks.

3 CLIPORT

CLIPORT is an imitation-learning agent based on four key principles: (1) Manipulation through a two-step primitive where each action involves a start and final end-effector pose. (2) Spatially equivariant visual representations [56, 57] that are robust to translations and rotations. (3) Two separate pathways for semantic and spatial information. (4) Language-conditioned policies for specifying goals and also transferring concepts across tasks. Combining (1) and (2) from Transporter with (3) and (4) allows us to achieve generalizable policies that go beyond just imitating demonstrations.

Section 3.1 describes the problem formulation, gives an overview of Transporter [2], and presents our language-conditioned model. Section 3.2 provides details on the training approach.

3.1 Language-Conditioned Manipulation

We consider the problem of learning a goal-conditioned policy π that generates actions $\mathbf{a}_t$ given input $\gamma_t = (\mathbf{o}_t, \mathbf{l}_t)$ consisting of a visual observation $\mathbf{o}_t$ and an English language instruction $\mathbf{l}_t$:

$$\pi(\boldsymbol{\gamma}_t) = \pi(\mathbf{o}_t, \mathbf{l}_t) \rightarrow \mathbf{a}_t = (\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}}) \in \mathcal{A} \tag{1}$$

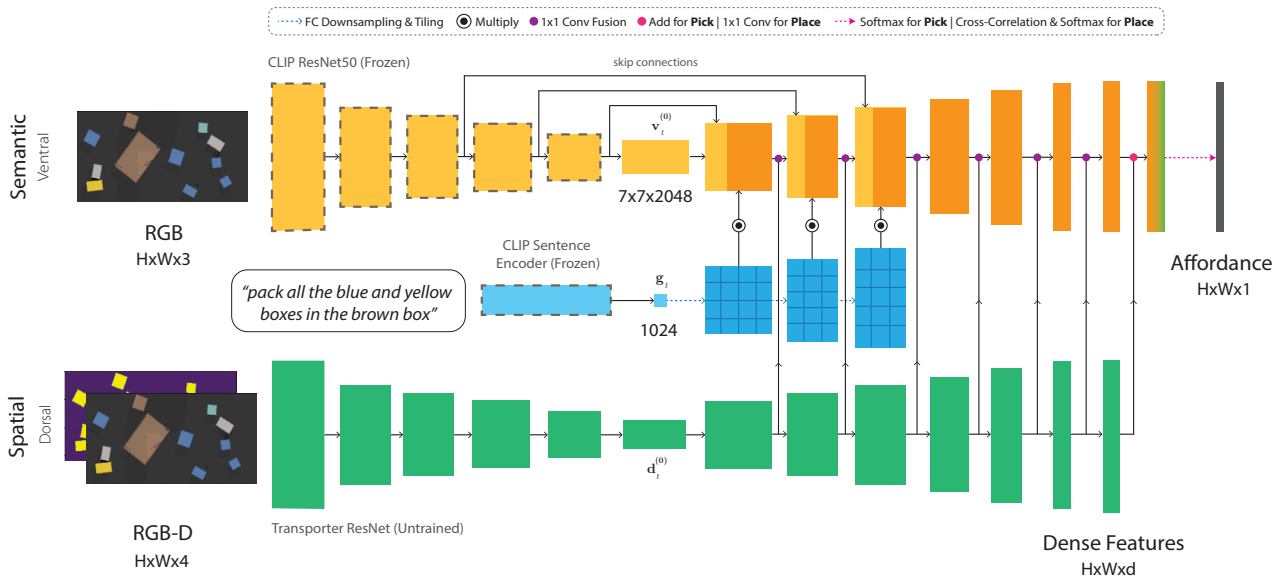

Figure 2. **CLIPORT Two-Stream Architecture.** An overview of the **semantic** and **spatial** streams in the 3 Fully-Convolutional-Networks f_{pick}, Φ_{query}, and Φ_{key} where f_{pick} is used to predict pick actions, and Φ_{query} and Φ_{key} are used to predict place actions. The **semantic** stream uses a frozen CLIP ResNet50 [1] to encode RGB input, and its decoder layers are conditioned with tiled language features from the CLIP sentence encoder. The **spatial** stream encodes RGB-D input, and its decoder layers are laterally fused with the **semantic** stream. The final output is dense pixelwise features which are used for pick or place affordance predictions. See Appendix B for the exact architecture.

The actions $\mathbf{a} = (\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}})$ specify the end-effector pose for picking and placing, respectively. We consider tabletop tasks where $\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}} \in \mathbf{SE}(2)$. The visual observation $\mathbf{o}_t$ is a top-down orthographic RGB-D reconstruction of the scene where each pixel corresponds to a point in 3D space. The language instruction $\mathbf{l}_t$ either specifies step-by-step instructions e.g. *"pack the scissors"* $\rightarrow$ *"pack the purple tape"* $\rightarrow$ etc., or a single goal description for the whole task e.g *"pack all the blue and yellow boxes in the brown box"*. See Figure 4 for specific examples.

We assume access to a dataset $\mathcal{D} = \{\zeta_1, \zeta_2, \ldots, \zeta_n\}$ of n expert demonstrations with associated discrete-time input-action pairs $\zeta_i = \{(\mathbf{o}_1, \mathbf{l}_1, \mathbf{a}_1), (\mathbf{o}_2, \mathbf{l}_2, \mathbf{a}_2), \ldots\}$ where $\mathbf{a}_t = (\mathcal{T}_{\text{pick}}, \mathcal{T}_{\text{place}})$ corresponds to expert pick-and-place coordinates at timestep t. These expert demonstrations are used to supervise the policy π.

Transporter for Pick-and-Place. The policy π is trained with Transporter [2] to perform spatial manipulation. The model first (i) attends to a local region to decide where to pick, then (ii) computes a placement location by finding the best match through cross-correlation of deep visual features.

Following Transporter [2, 6], the policy π is composed of two action-value modules (Q-functions): The pick module $\mathcal{Q}_{\text{pick}}$ decides where to pick, and conditioned on this pick action the place module $\mathcal{Q}_{\text{place}}$ decides where to place. These modules are implemented as Fully-Convolutional-Networks (FCNs) that are translationally equivariant by design. As we will describe in more detail below, we extend these networks to two-stream architectures that can handle language input. The pick FCN f_{pick} takes input $\boldsymbol{\gamma}_t = (\mathbf{o}_t, \mathbf{l}_t)$ and outputs a dense pixelwise prediction $\mathcal{Q}_{\text{pick}} \in \mathbb{R}^{H \times W}$ of action-values, where are used to predict a pick action $\mathcal{T}_{\text{pick}}$:

$$\mathcal{T}_{\text{pick}} = \underset{(u,v)}{\operatorname{argmax}}\, \mathcal{Q}_{\text{pick}}((u,v)|\boldsymbol{\gamma}_t) \tag{2}$$

Since $\mathbf{o}_t$ is an orthographic heightmap, each pixel location (u, v) can be mapped to a 3D picking location using the known camera calibration. f_{pick} is trained in a supervised manner to predict the pick action $\mathcal{T}_{\text{pick}}$ that imitates the expert demonstration with the specified language instruction at timestep t.

The second FCN Φ_{query} takes in $\boldsymbol{\gamma}_t[\mathcal{T}_{\text{pick}}]$, which is a $c \times c$ crop of $\mathbf{o}_t$ centered at $\mathcal{T}_{\text{pick}}$ along with the language instruction $\mathbf{l}_t$, and outputs a query feature embedding of shape $\mathbb{R}^{c \times c \times d}$. The third FCN Φ_{key} consumes the full input $\boldsymbol{\gamma}_t$ and outputs a key feature embedding of shape $\mathbb{R}^{H \times W \times d}$. The place action-values $\mathcal{Q}_{\text{place}}$ are then computed by cross-correlating the query and key features:

$$\mathcal{Q}_{\text{place}}(\Delta\tau|\gamma_t, \mathcal{T}_{\text{pick}}) = \big(\Phi_{\text{query}}(\boldsymbol{\gamma}_t[\mathcal{T}_{\text{pick}}]) * \Phi_{\text{key}}(\boldsymbol{\gamma}_t)\big)[\Delta\tau] \tag{3}$$

where $\Delta\tau \in \boldsymbol{SE}(2)$ represents a potential placement pose. Since $\mathbf{o}_t$ is an orthographic heightmap, rotations in the placement pose $\Delta\tau$ can be captured by rotating the crop of $\mathbf{o}_t$ for k discrete angles before passing it through the query network Φ_{query}. Then $\mathcal{T}_{\text{place}} = \text{argmax}_{\Delta\tau}\, \mathcal{Q}_{\text{place}}(\Delta\tau|\gamma_t, \mathcal{T}_{\text{pick}})$, where the place module is trained to imitate the placements in the expert demonstrations. For all models, we use $c = 64$, $k = 36$ and $d = 3$. As in Transporter [2, 6], our framework can be extended to handle any motion primitive like pushing, sliding, etc. that can be parameterized by two end-effector poses at each timestep. For more details, we refer the reader to the original paper [2].

Two-Stream Architecture. In CLIPORT, we extend the network architecture of all three FCNs $f_{\text{pick}}, \Phi_{\text{query}}$ and Φ_{key} from Transporter [2] to allow for language input and reasoning about high-level semantic concepts. We extend the FCNs to two-pathways: **semantic** (ventral) and **spatial** (dorsal). The **semantic** stream is conditioned with language features at the bottleneck and fused with intermediate features from the **spatial** stream. See Figure 2 for an overview of the architecture.

The **spatial** stream is identical to the ResNet architecture in Transporter – a tabula rasa network that takes in RGB-D input $\mathbf{o}_t$ and outputs dense features through an hourglass encoder-decoder model. The **semantic** stream uses a frozen pre-trained CLIP ResNet50 [1] to encode the RGB input[1] $\tilde{\mathbf{o}}_t$ up until the penultimate layer $\tilde{\mathbf{o}}_t \rightarrow \mathbf{v}_t^{(0)} : \mathbb{R}^{7\times7\times2048}$, and then introduces decoding layers that upsample the feature tensors to mimic the **spatial** stream $\mathbf{v}_t^{(l-1)} \rightarrow \mathbf{v}_t^{(l)} : \mathbb{R}^{h\times w\times C}$ at each layer l.

The language instruction $\mathbf{l}_t$ is encoded with CLIP's Transformer-based sentence encoder to produce a goal encoding $\mathbf{l}_t \rightarrow \mathbf{g}_t : \mathbb{R}^{1024}$. This goal encoding $\mathbf{g}_t$ is downsampled with fully-connected layers to match the channel dimension C and tiled to match the spatial dimensions of the decoder features such that $\mathbf{g}_t \rightarrow \mathbf{g}_t^{(l)} : \mathbb{R}^{h\times w\times C}$. The decoder features are then conditioned with the tiled goal features through an element-wise product $\mathbf{v}_t^{(l)} \odot \mathbf{g}_t^{(l)}$ (Hadamard product). Since CLIP was trained with contrastive loss on the dot-product alignment between pooled image features and language encodings, the element-wise product allows us to use this alignment while the tiling preserves the spatial dimensions of the visual features. This language conditioning is repeated for three subsequent layers after the bottleneck inspired by LingUNet [58]. We also add skip connections to these layers from the CLIP ResNet50 encoder to utilize different levels of semantic information from shapes to parts to object-level concepts [59]. Finally, following existing two-stream architectures in video-action recognition [51], we add lateral connections from the **spatial** stream to the **semantic** stream. These connections involve concatenating two feature tensors and applying 1×1 `conv` to reduce the channel dimension $[\mathbf{v}_t^{(l)} \odot \mathbf{g}_t^{(l)}; \mathbf{d}_t^{(l)}] : \mathbb{R}^{h\times w\times C_{\mathbf{v}}+C_{\mathbf{d}}} \rightarrow \mathbb{R}^{h\times w\times C_{\mathbf{v}}}$, where $\mathbf{v}_t^{(l)}$ and $\mathbf{d}_t^{(l)}$ are the **semantic** and **spatial** tensors at layer l, respectively. For the final fusion of dense features, `addition` for f_{pick} and 1×1 `conv` fusion for Φ_{query} and Φ_{key} worked the best empirically. See Appendix B for details on the exact architecture.

3.2 Implementation Details

Training from demonstrations. Similar to Transporter [2] we train CLIPORT using imitation learning on a set of expert demonstrations $\mathcal{D} = \{\zeta_1, \zeta_2, \ldots, \zeta_n\}$ consisting of discrete-time input-action pairs $\zeta_i = \{(\mathbf{o}_1, \mathbf{l}_1, \mathbf{a}_1), (\mathbf{o}_2, \mathbf{l}_2, \mathbf{a}_2), \ldots\}$. During training, we randomly sample an input-action pair from the dataset and supervise the model end-to-end with one-hot pixel encodings of demonstration actions $Y_{\text{pick}} : \mathbb{R}^{H\times W\times k}$ and $Y_{\text{place}} : \mathbb{R}^{H\times W\times k}$ with k discrete rotations. In simulated experiments with the suction-gripper, we use $k = 1$ for pick actions and $k = 36$ for place actions. Following Transporter [2], the model is trained with cross-entropy loss: $\mathcal{L} = -\mathbb{E}_{Y_{\text{pick}}}[\log \mathcal{V}_{\text{pick}}] - \mathbb{E}_{Y_{\text{place}}}[\log \mathcal{V}_{\text{place}}]$ where $\mathcal{V}_{\text{pick}} = \text{softmax}(\mathcal{Q}_{\text{pick}}((u, v)|\gamma_t))$ and $\mathcal{V}_{\text{place}} = \text{softmax}(\mathcal{Q}_{\text{place}}((u', v', \omega')|\gamma_t, \mathcal{T}_{\text{pick}}))$. Compared to the original Transporter models that were trained for 40K iterations, we train our models for 200K iterations (with data augmentation; see Appendix D) to account for additional semantic variation in tasks – randomized colors, shapes, objects. All models are trained on a single commodity GPU for 2 days with a batch size of 1.

Training multi-task models. Multi-task training is nearly identical to single-task training except for the sampling of training data. First, we randomly sample a task, and then select a random input-action pair from that task in the dataset. Using this strategy, all tasks are equally likely to be sampled but longer horizon tasks are less likely to reach full coverage of input-action pairs available in the dataset. To compensate for this, we train all multi-task models $3\times$ longer for 600K iterations or 6 GPU days.

[1] We cannot use depth information with CLIP since it was trained with RGB-only image-caption pairs from the internet.

4 Results

We perform experiments both in simulation and hardware aimed at answering the following questions: 1) How effective is the language-conditioned two-stream architecture for fine-grained manipulation compared to one-stream alternatives and other simpler baselines? 2) Is it possible to train a multi-task model for all tasks, and how well does it perform and generalize? 3) How well do these models generalize to seen and unseen semantic attributes like colors, shapes, and object categories?

4.1 Simulation Setup

Environment. All simulated experiments are based on a Universal Robot UR5e with a suction gripper. The setup provides a systematic and reproducible environment for evaluation, especially for benchmarking the ability to ground semantic concepts like colors and object categories. The input observation is a top-down RGB-D reconstruction from 3 cameras positioned around a rectangular table: one in the front, one on the left shoulder, and one on the right shoulder, all pointing towards the center. Each camera has a resolution of 640×480 and is noiseless.

Language-Conditioned Manipulation Tasks. We extend the Ravens benchmark [2] set in PyBullet [60] with 10 language-conditioned manipulation tasks. See Figure 1 for examples and Table 3 for challenges associated with each task. Each task instance is constructed by sampling a set of objects and attributes: poses, colors, sizes, and object categories. 8 of the 10 tasks have two variants, denoted by seen and unseen, depending on whether the task has unseen attributes (e.g. color) at test time. For colors: $\mathbb{T}_{\text{seen colors}} = \{\texttt{yellow, brown, gray, cyan}\}$ and $\mathbb{T}_{\text{unseen colors}} = \{\texttt{orange, purple, pink, white}\}$ with 3 overlapping colors $\mathbb{T}_{\text{all}} = \{\texttt{red, green, blue}\}$ used in both the seen and unseen spilts. For packing objects, we use 56 tabletop objects from the Google Scanned Objects dataset [61] and split them into 37 seen and 19 unseen objects. The language instructions are constructed from templates for simulated experiments, and human-annotated for real-world experiments. For more details about individual tasks, see Appendix A.

Evaluation Metric. We adopt the 0 (fail) to 100 (success) scores proposed in the Ravens benchmark [2]. The score assigns partial credit based on the task, e.g. $3/5 \Rightarrow 60.0$ for packing 3 out of 5 objects specified in the instructions, or $30/56 \Rightarrow 53.6$ for pushing 30 out of 56 particles into the correct zone. See Appendix A for the specific evaluation metric used in each task. During an evaluation episode, an agent keeps interacting with the scene until an oracle indicates task-completion. We report scores on 100 evaluation runs for agents trained with $n = 1, 10, 100, 1000$ demonstrations.

4.2 Simulation Results

Table 1 presents results from our large-scale experiments in Ravens [2] and Figure 3 summarizes these results with average scores across seen and unseen splits.

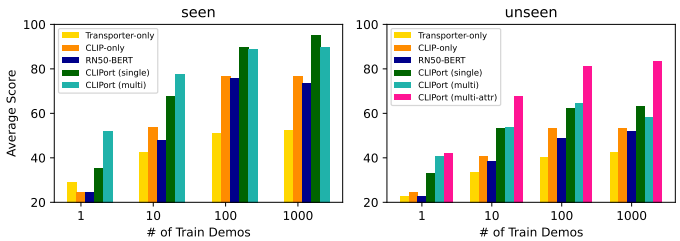

Figure 3. Average scores across seen and unseen splits for all tasks in Table 1.

Baseline Methods. To study the effectiveness of our two-stream architecture, we broadly compare against two baselines: Transporter-only and CLIP-only. Transporter-only is the original Transporter [2], or equivalently, the **spatial** stream of CLIPORT with RGB-D input. Although Transporter-only does not receive any language goals, it shows what can be achieved through chance by exploiting the most likely actions seen during training. On the other hand, CLIP-only is just the **semantic** stream of CLIPORT with RGB and language input. CLIP-only shows what can be achieved by fine-tuning a pre-trained semantic model for manipulation without spatial information, particularly depth.

Two-Stream Performance. Figure 3 (seen) captures the essence of our main claims. The performance of Transporter-only saturates at 50% since it doesn't use the language instruction to ground the desired goal. CLIP-only does have a goal, but lacks the spatial precision to go the last mile and thus saturates at 75%. Only CLIPORT (single) achieves more than 90%, which indicates that both the semantic and spatial streams are crucial for fine-grained manipulation. Further, CLIPORT (single) achieves 90% on most tasks with just 100 demonstrations, showcasing its efficiency.

In addition to these baselines, we present various ablations and alternative one-stream and two-stream models in Appendix E. To briefly summarize these results, CLIP is essential for few-shot

Method	packing-box-pairs seen-colors				packing-box-pairs unseen-colors				packing-seen-google objects-seq				packing-unseen-google objects-seq				packing-seen-google objects-group				packing-unseen-google objects-group			
	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
Transporter-only [2]	48.9	57.2	59.4	60.6	37.8	52.3	54.5	60.7	29.0	39.1	39.5	41.5	25.9	35.1	41.7	37.9	53.9	52.0	54.4	52.9	29.9	53.5	52.7	54.7
CLIP-only	37.1	72.3	87.4	90.9	36.1	61.8	67.2	62.9	29.3	76.0	86.4	**97.0**	36.5	46.2	58.8	58.0	51.8	66.6	89.5	94.7	46.3	61.0	75.3	77.4
RN50-BERT	40.0	64.4	94.7	90.5	42.1	58.7	62.4	72.2	29.7	49.8	90.4	94.6	39.9	41.8	57.5	57.2	48.5	56.9	83.1	93.6	44.8	55.3	71.7	77.9
CLIPORT (single)	51.9	84.7	95.9	**98.0**	47.1	66.9	70.0	71.9	15.3	60.3	**92.9**	96.5	25.9	52.0	62.5	61.6	51.4	71.5	88.8	**95.6**	53.6	68.4	77.8	72.4
CLIPORT (multi)	**68.6**	**90.0**	**96.0**	96.3	**55.9**	**70.3**	**76.6**	**72.9**	**43.6**	**76.2**	85.1	82.2	**48.9**	**60.1**	**67.2**	**67.4**	**68.4**	**84.8**	**92.0**	93.2	**67.4**	**72.7**	**82.7**	**81.6**
CLIPORT (multi-attr)	–	–	–	–	*63.2*	*85.0*	*91.5*	*87.0*	–	–	–	–	*38.0*	*62.0*	*78.0*	*82.0*	–	–	–	–	*70.2*	*80.8*	*87.8*	*91.5*

	stack-block-pyramid seq-seen-colors				stack-block-pyramid seq-unseen-colors				separating-piles seen-colors				separating-piles unseen-colors				towers-of-hanoi seq-seen-colors				towers-of-hanoi seq-unseen-colors			
	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
Transporter-only [2]	4.8	4.0	6.8	5.7	4.8	5.3	5.0	5.0	42.8	52.9	54.7	55.6	47.8	53.4	52.6	54.8	25.1	74.4	**100**	**100**	25.6	46.4	77.0	81.7
CLIP-only	5.5	30.0	58.7	59.0	2.0	16.3	5.7	19.3	39.7	69.6	90.4	92.9	46.4	61.6	76.9	74.4	10.9	48.1	88.6	52.9	15.9	44.7	67.1	58.1
RN50-BERT	5.7	35.5	94.0	98.0	5.2	10.5	19.7	33.3	33.3	55.9	53.0	48.7	35.7	52.2	53.1	57.0	26.4	68.1	92.7	95.9	16.3	75.0	82.0	84.3
CLIPORT (single)	29.0	68.8	95.0	**99.3**	15.8	29.0	32.7	**41.8**	45.1	58.6	**96.8**	**99.9**	50.7	56.5	**83.8**	**83.0**	55.3	**94.1**	99.9	**100**	**66.6**	**91.9**	**96.4**	**100**
CLIPORT (multi)	**38.3**	**71.0**	**97.0**	97.3	**27.8**	**31.8**	**39.3**	33.3	**53.2**	**73.0**	92.7	89.2	**55.5**	**71.2**	79.5	76.7	**67.6**	94.0	99.1	**100**	55.6	68.6	79.1	67.0
CLIPORT (multi-attr)	–	–	–	–	*28.7*	*56.3*	*84.5*	*93.8*	–	–	–	–	*51.2*	*66.1*	*82.2*	*89.8*	–	–	–	–	*53.4*	*94.0*	*99.6*	*100*

	align-rope				packing-unseen-shapes				assembling-kits-seq seen-colors				assembling-kits-seq unseen-colors				put-blocks-in-bowls seen-colors				put-blocks-in-bowls unseen-colors			
	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
Transporter-only [2]	6.3	24.7	39.8	48.2	28.0	34.0	27.0	32.0	6.8	15.2	30.8	32.6	9.4	15.6	30.4	30.0	18.8	45.2	63.2	69.0	12.2	16.8	20.5	21.7
CLIP-only	15.4	47.6	76.7	74.3	26.0	36.0	40.0	**43.0**	1.4	6.4	19.0	27.2	4.2	5.6	12.0	16.2	22.3	62.2	94.7	98.5	15.8	29.7	38.3	24.7
RN50-BERT	6.8	26.9	69.8	61.1	22.0	31.0	29.0	30.0	2.4	6.8	15.2	23.0	2.2	7.6	15.2	19.4	10.8	46.3	82.3	92.2	14.0	24.2	29.7	27.7
CLIPORT (single)	14.8	**66.2**	**93.2**	**98.2**	22.0	**42.0**	35.0	40.0	11.0	28.8	**51.6**	**72.0**	**17.2**	23.2	33.0	**38.0**	21.7	73.0	98.2	**100**	17.2	32.5	40.2	**48.3**
CLIPORT (multi)	**19.2**	52.4	80.2	72.2	**29.0**	**42.0**	**47.0**	41.0	**17.4**	**37.2**	48.2	57.6	12.2	**23.8**	**36.4**	29.0	**59.7**	**94.0**	**100**	**100**	**33.8**	**42.7**	**55.3**	43.3
CLIPORT (multi-attr)	–	–	–	–	–	–	–	–	–	–	–	–	*3.6*	*35.6*	*49.0*	*48.2*	–	–	–	–	*29.5*	*61.7*	*76.8*	*73.0*

Table 1. **Language-Conditioned Task Evaluations.** Task success scores (mean %) from 100 evaluation instances vs. # of training demonstrations (1, 10, 100, or 1000). See Appendix A for challenges pertaining to each task. CLIPORT (single) models are trained on seen splits, and evaluated on both seen and unseen splits. CLIPORT (multi) models are trained on seen splits of all 10 tasks with $1\mathbb{T}$, $10\mathbb{T}$, $100\mathbb{T}$, and $1000\mathbb{T}$ demonstrations where $\mathbb{T} = 10$. CLIPORT (multi-attr) indicate CLIPORT (multi) models trained on seen-and-unseen splits from all tasks *except* for that one particular heldout task, for which it is trained only the seen split. See Figure 3 for an overview with average scores.

learning (i.e. $n \geq 10$) in lieu of **semantic** stream alternatives like ImageNet-trained ResNet50 [62] with BERT [38]. Image-goal models outperform CLIPORT (single) in packing Google objects, but this is only because they do not have to solve the language-grounding problem.

Multi-Task Performance. In realistic scenarios, we want the robot to be capable of any task, not just one task. We investigate this through CLIPORT (multi) in Table 1 with one multi-task model trained on all 10 tasks. CLIPORT (multi) models are trained only on seen-splits of tasks, so an unseen attribute like 'pink' is consistent throughout single and multi-task settings. Surprisingly, CLIPORT (multi) outperforms single-task CLIPORT (single) models in $44/72 = 61\%$ of the evaluations in Table 1. This trend is also evident in Figure 3, especially in instances with 100 demonstrations or less. Although CLIPORT (multi) is trained on more diverse data from other tasks, both CLIPORT (multi) and CLIPORT (single) have access to the same amount of data *per* task. This supports our premise that language is a strong conditioning mechanism for reusing concepts from other tasks without learning them from scratch. It also validates a trait of data-driven approaches where training on lots of diverse data leads to more robust and generalizable representations [1, 63]. However, CLIPORT (multi) performs worse on longer-horizon tasks like align-rope. We hypothesize that this is because longer-horizon tasks get less coverage of input-action pairs in the dataset. Future works could use better sampling methods that balance tasks according to their average time horizon.

Generalizing to Unseen Attributes. Tasks that require generalizing to novel colors, shapes, and objects are more difficult and all our agents achieve relatively lower performance on these tasks, as shown in Figure 3 (unseen). However, CLIPORT (single) models do substantially better than chance, i.e., Transporter-only. The lower performances are due to the difficulty of grounding unseen attributes such as 'pink' and 'orange' in the language instruction "put the pink block on the orange bowl", when the agent has never encountered words 'orange', 'pink' or their corresponding visual characteristics in the context of the physical environment. Although pre-trained CLIP has been exposed to the attribute 'pink', it could correspond to different concepts in the physical setting depending on factors like lighting condition, and thus requires at least few examples to condition the trainable **semantic** decoder layers. Additionally, we notice that CLIPORT (single) is also less prone to overfitting compared to Transporter-only. As evidenced in towers-of-hanoi-seq-unseen-colors task in Table 1, Transporter-only suffers from a performance drop because of rings with unseen colors despite the fact that Tower of Hanoi can be solved without attending to the colors and simply focusing on the ring size. We hypothesize that since CLIP was trained on diverse internet data, it enables our agent to focus on task-relevant concepts while ignoring irrelevant aspects of the task.

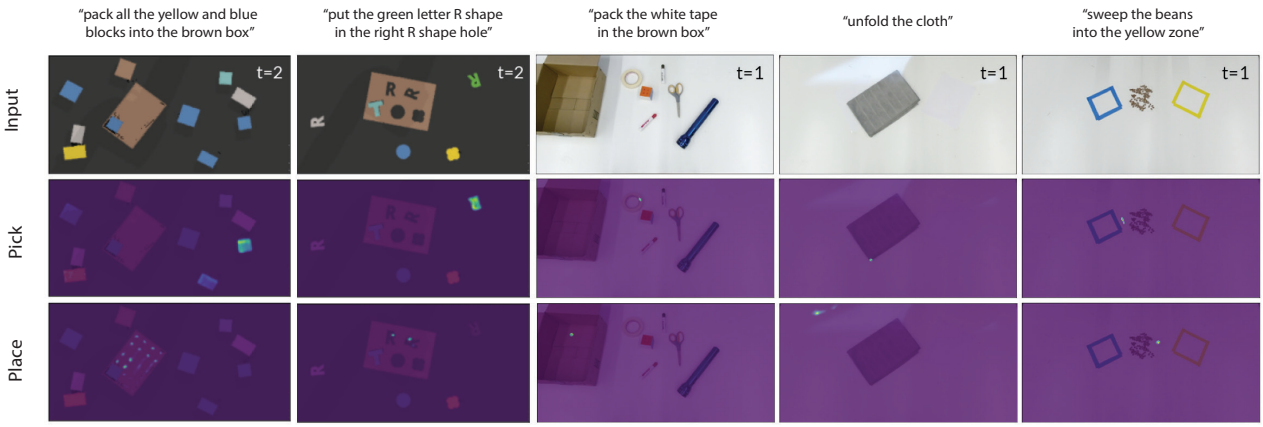

Figure 4. Affordance predictions from CLIPORT (multi) models in sim (left two) and real settings (right three). More examples in Appendix G.

Transferring Attributes across Tasks. One solution for dealing with unseen attributes is to explicitly learn these attributes from other tasks. We study this with CLIPORT (multi-attr) in Table 1 and Figure 3 (unseen). For these models, CLIPORT (multi) is trained on both seen-and-unseen splits from all tasks *except* for the task being evaluated on, for which it was only trained on the seen split. As such, this evaluation measures whether having seen pink blocks in put-blocks-in-bowl-unseen-colors helps solve "pack all the pink and cyan boxes" in packing-box-pairs-unseen-colors. Results indicate that such explicit transfers result in significant improvements. For instance, on the put-blocks-in-bowls-unseen-colors task for $n = 1000$, CLIPORT (multi)'s performance increases from 43.3 to 73.0.

4.3 Real-Robot Experiments

We validated our results in hardware with a Franka Panda manipulator. See Appendix C for setup details. Table 2 reports success rates for a multi-task model trained and evaluated on 9 real-world tasks. Due to COVID restrictions, we could not conduct large-scale user-studies, so we report on small train (5-10 demos) and test sets (5-10 runs) per task. Overall, CLIPORT (multi) is effective at few-shot learning with just 179 samples, and the performances roughly correspond to those in simulated experiments, with simple block manipulation tasks achieving $\sim 70\%$. We estimate that for more robust real-world performance at least 50 to 100 training demonstrations are necessary, as evident in Figure 3. Interestingly, we observed that the model sometimes exploits biases in the training data instead of learning to ground instructions. For instance, in Put Blocks in Bowl, the training set consisted of only one datapoint on "yellow blocks" being placed inside a "blue bowl". This made it difficult to condition the model to place "yellow blocks" in non-blue bowls. But instances with just one or two examples where a colored block went to different colored bowls was sufficient to make the model pay attention to the language. In summary, unbiased datasets containing both a good coverage of expected skills and invariances, and a decent number of training demonstrations, are crucial for good real-world performance.

Task	# Train (Samples)	# Test	Succ. %
Stack Blocks	5 (13)	10	70.0
Put Blocks in Bowl	5 (10)	10	65.0
Pack Objects	10 (31)	10	60.0
Move Rook	4 (29)	10	70.0
Fold Cloth	9 (9)	10	57.0
Read Text	2 (26)	10	55.0
Loop Rope	4 (12)	10	60.0
Sweep Beans	5 (23)	5	60.6
Pick Cherries	4 (26)	5	75.0

Table 2. Success rates (%) of a multi-task model trained an evaluated 9 real-world tasks (see Figure 1). Samples indicate total image-action pairs, e.g 1 in Figure 8.

5 Conclusion

We introduced CLIPORT, an end-to-end framework for language-conditioned fine-grained manipulation. Our experiments, specifically with multi-task models, indicate that data-driven approaches to generalization have yet to be fully-exploited in robotics. Coupled with the right action abstraction and spatio-semantic priors, end-to-end methods can quickly learn new skills without requiring top-down pipelines that need task-specific engineering.

While CLIPORT can solve a range of tabletop tasks, extending it to dexterous 6-DOF manipulation that goes beyond the two-step primitive remains a challenge. As such, it cannot handle complex partially-observable scenes, or output continuous control for multi-fingered hands, or predict task-completion (see Appendix H for an extended discussion). But overall, we are excited by the confluence of data and structural priors for building scalable and generalizable robotic systems.

References

[1] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning Transferable Visual Models From Natural Language Supervision. *arXiv:2103.00020 [cs]*, Feb. 2021.

[2] A. Zeng, P. Florence, J. Tompson, S. Welker, J. Chien, M. Attarian, T. Armstrong, I. Krasin, D. Duong, V. Sindhwani, and J. Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *Conference on Robot Learning (CoRL)*, 2020.

[3] I. Akkaya, M. Andrychowicz, M. Chociej, M. Litwin, B. McGrew, A. Petron, A. Paino, M. Plappert, G. Powell, R. Ribas, et al. Solving rubik's cube with a robot hand. *arXiv preprint arXiv:1910.07113*, 2019.

[4] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *Conference on Robot Learning (CoRL)*, 2018.

[5] D. Kalashnikov, J. Varley, Y. Chebotar, B. Swanson, R. Jonschkowski, C. Finn, S. Levine, and K. Hausman. Mt-opt: Continuous multi-task robotic reinforcement learning at scale. *arXiv preprint arXiv:2104.08212*, 2021.

[6] D. Seita, P. Florence, J. Tompson, E. Coumans, V. Sindhwani, K. Goldberg, and A. Zeng. Learning to rearrange deformable cables, fabrics, and bags with goal-conditioned transporter networks. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021.

[7] M. Shridhar and D. Hsu. Interactive visual grounding of referring expressions for human-robot interaction. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[8] C. Matuszek, L. Bo, L. Zettlemoyer, and D. Fox. Learning from unscripted deictic gesture and language for human-robot interactions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 28, 2014.

[9] M. Bollini, S. Tellex, T. Thompson, N. Roy, and D. Rus. Interpreting and executing recipes with a cooking robot. In *Experimental Robotics*, pages 481–495. Springer, 2013.

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

[11] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[12] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. Momentum contrast for unsupervised visual representation learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9729–9738, 2020.

[13] J. Lu, D. Batra, D. Parikh, and S. Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. In *Advances in Neural Information Processing Systems (NeuRIPS)*, 2019.

[14] Y. C. Chen, L. Li, L. Yu, A. El Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu. Uniter: Universal image-text representation learning. In *European Conference on Computer Vision*, pages 104–120. Springer, 2020.

[15] H. Tan and M. Bansal. Lxmert: Learning cross-modality encoder representations from transformers. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.

[16] D. H. Hubel and T. N. Wiesel. Receptive fields and functional architecture in two nonstriate visual areas (18 and 19) of the cat. *Journal of neurophysiology*, 28(2):229–289, 1965.

[17] M. Livingstone and D. Hubel. Segregation of form, color, movement, and depth: anatomy, physiology, and perception. *Science*, 240(4853):740–749, 1988.

[18] A. Derrington and P. Lennie. Spatial and temporal contrast sensitivities of neurones in lateral geniculate nucleus of macaque. *The Journal of physiology*, 357(1):219–240, 1984.

[19] J. J. Gibson. *The ecological approach to visual perception: classic edition*. Psychology Press, 2014.

[20] A. Kamath, M. Singh, Y. LeCun, I. Misra, G. Synnaeve, and N. Carion. Mdetr–modulated detection for end-to-end multi-modal understanding. *arXiv preprint arXiv:2104.12763*, 2021.

[21] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[22] Y. Xiang, T. Schmidt, V. Narayanan, and D. Fox. Posecnn: A convolutional neural network for 6d object pose estimation in cluttered scenes. In *Proceedings of Robotics: Science and Systems (RSS)*, 2018.

[23] M. Zhu, K. G. Derpanis, Y. Yang, S. Brahmbhatt, M. Zhang, C. Phillips, M. Lecce, and K. Daniilidis. Single image 3d object detection and pose estimation for grasping. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3936–3943. IEEE, 2014.

[24] A. Zeng, K.-T. Yu, S. Song, D. Suo, E. Walker, A. Rodriguez, and J. Xiao. Multi-view self-supervised deep learning for 6d pose estimation in the amazon picking challenge. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 1386–1383. IEEE, 2017.

[25] X. Deng, Y. Xiang, A. Mousavian, C. Eppner, T. Bretl, and D. Fox. Self-supervised 6d object pose estimation for robot manipulation. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3665–3671. IEEE, 2020.

[26] C. Xie, Y. Xiang, A. Mousavian, and D. Fox. The best of both modes: Separately leveraging rgb and depth for unseen object instance segmentation. In *Conference on Robot Learning (CoRL)*, pages 1369–1378. PMLR, 2020.

[27] P. R. Florence, L. Manuelli, and R. Tedrake. Dense object nets: Learning dense visual object descriptors by and for robotic manipulation. In *Conference on Robot Learning (CoRL)*, 2018.

[28] P. Florence, L. Manuelli, and R. Tedrake. Self-supervised correspondence in visuomotor policy learning. *IEEE Robotics and Automation Letters*, 5(2):492–499, 2019.

[29] P. Sundaresan, J. Grannen, B. Thananjeyan, A. Balakrishna, M. Laskey, K. Stone, J. E. Gonzalez, and K. Goldberg. Learning rope manipulation policies using dense object descriptors trained on synthetic depth data. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9411–9418. IEEE, 2020.

[30] L. Manuelli, W. Gao, P. Florence, and R. Tedrake. kpam: Keypoint affordances for category-level robotic manipulation. In *International Symposium on Robotics Research (ISRR)*, 2019.

[31] T. D. Kulkarni, A. Gupta, C. Ionescu, S. Borgeaud, M. Reynolds, A. Zisserman, and V. Mnih. Unsupervised learning of object keypoints for perception and control. *Advances in neural information processing systems (NeuRIPS)*, 32:10724–10734, 2019.

[32] X. Liu, R. Jonschkowski, A. Angelova, and K. Konolige. Keypose: Multi-view 3d labeling and keypoint estimation for transparent objects. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11602–11610, 2020.

[33] K. Zakka, A. Zeng, J. Lee, and S. Song. Form2fit: Learning shape priors for generalizable assembly from disassembly. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9404–9410. IEEE, 2020.

[34] S. Song, A. Zeng, J. Lee, and T. Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *IEEE Robotics and Automation Letters*, 5(3): 4978–4985, 2020.

[35] Y. Wu, W. Yan, T. Kurutach, L. Pinto, and P. Abbeel. Learning to Manipulate Deformable Objects without Demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[36] L. Yen-Chen, A. Zeng, S. Song, P. Isola, and T.-Y. Lin. Learning to see before learning to act: Visual pre-training for manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[37] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeuRIPS)*, 2017.

[38] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[39] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[40] F. Yu, J. Tang, W. Yin, Y. Sun, H. Tian, H. Wu, and H. Wang. Ernie-vil: Knowledge enhanced vision-language representations through scene graph. *arXiv preprint arXiv:2006.16934*, 2020.

[41] Y. Bisk, D. Yuret, and D. Marcu. Natural language communication with robots. In *Proceedings of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 751–761, 2016.

[42] J. Thomason, S. Zhang, R. J. Mooney, and P. Stone. Learning to interpret natural language commands through human-robot dialog. In *Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI)*, 2015.

[43] J. Hatori, Y. Kikuchi, S. Kobayashi, K. Takahashi, Y. Tsuboi, Y. Unno, W. Ko, and J. Tan. Interactively picking real-world objects with unconstrained spoken language instructions. In *Proceedings of International Conference on Robotics and Automation (ICRA)*, 2018.

[44] Y. Chen, R. Xu, Y. Lin, and P. A. Vela. A Joint Network for Grasp Detection Conditioned on Natural Language Commands. *arXiv:2104.00492 [cs]*, Apr. 2021.

[45] V. Blukis, R. A. Knepper, and Y. Artzi. Few-shot object grounding for mapping natural language instructions to robot control. In *Conference on Robot Learning (CoRL)*, 2020.

[46] C. Paxton, Y. Bisk, J. Thomason, A. Byravan, and D. Fox. Prospection: Interpretable plans from language by predicting the future. In *International Conference on Robotics and Automation (ICRA)*, pages 6942–6948. IEEE, 2019.

[47] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, 2011.

[48] C. Lynch and P. Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020.

[49] K. Simonyan and A. Zisserman. Two-stream convolutional networks for action recognition in videos. *arXiv preprint arXiv:1406.2199*, 2014.

[50] C. Feichtenhofer, A. Pinz, and A. Zisserman. Convolutional two-stream network fusion for video action recognition. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1933–1941, 2016.

[51] C. Feichtenhofer, H. Fan, J. Malik, and K. He. Slowfast networks for video recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (CVPR)*, pages 6202–6211, 2019.

[52] E. Kazakos, A. Nagrani, A. Zisserman, and D. Damen. Slow-fast auditory streams for audio recognition. In *ICASSP 2021-2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 855–859. IEEE, 2021.

[53] F. Xiao, Y. J. Lee, K. Grauman, J. Malik, and C. Feichtenhofer. Audiovisual slowfast networks for video recognition. *arXiv preprint arXiv:2001.08740*, 2020.

[54] A. Zeng, S. Song, K.-T. Yu, E. Donlon, F. R. Hogan, M. Bauza, D. Ma, O. Taylor, M. Liu, E. Romo, et al. Robotic pick-and-place of novel objects in clutter with multi-affordance grasping and cross-domain image matching. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 3750–3757. IEEE, 2018.

[55] E. Jang, S. Vijayanarasimhan, P. Pastor, J. Ibarz, and S. Levine. End-to-end learning of semantic grasping. In *Conference on Robot Learning (CoRL)*, Proceedings of Machine Learning Research. PMLR, 2017.

[56] R. Kondor and S. Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. In *International Conference on Machine Learning (ICML)*, 2018.

[57] T. Cohen and M. Welling. Group equivariant convolutional networks. In *International conference on machine learning (ICML)*, 2016.

[58] D. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkhin, and Y. Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018.

[59] G. Goh, N. C. †, C. V. †, S. Carter, M. Petrov, L. Schubert, A. Radford, and C. Olah. Multimodal neurons in artificial neural networks. *Distill*, 2021. doi:10.23915/distill.00030. https://distill.pub/2021/multimodal-neurons.

[60] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. 2016.

[61] Google scanned objects dataset, 2020. URL `https://app.ignitionrobotics.org/GoogleResearch/fuel/collections/Google%20Scanned%20Objects`.

[62] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[63] J. Lu, V. Goswami, M. Rohrbach, D. Parikh, and S. Lee. 12-in-1: Multi-task vision and language representation learning. In *The IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[64] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*, 2019.

[65] V. Sanh, L. Debut, J. Chaumond, and T. Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[66] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[67] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

[68] D. Ding, F. Hill, A. Santoro, and M. Botvinick. Object-based attention for spatio-temporal reasoning: Outperforming neuro-symbolic models with flexible distributed architectures. *arXiv preprint arXiv:2012.08508*, 2020.

[69] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, pages 610–623, 2021.

A Task Details

Task	precise placing	multimodal placing	multi-step sequencing	unseen poses	unseen colors	unseen objects	language instruction
put-blocks-in-bowls-seen-colors*	✗	✓	✗	✓	✗	✗	goal
put-blocks-in-bowls-unseen-colors*	✗	✓	✗	✓	✓	✗	goal
assembling-kits-seq-seen-colors	✓	✓	✓	✓	✗	✓	step
assembling-kits-seq-unseen-colors	✓	✓	✓	✓	✓	✓	step
packing-unseen-shapes	✗	✓	✗	✓	✓	✓	goal
stack-block-pyramid-seq-seen-colors	✓	✓	✓	✓	✗	✗	step
stack-block-pyramid-seq-unseen-colors	✓	✓	✓	✓	✓	✗	step
towers-of-hanoi-seq-seen-colors	✓	✓	✓	✓	✗	✗	step
towers-of-hanoi-seq-unseen-colors	✓	✓	✓	✓	✓	✗	step
packing-box-pairs-seen-colors*§	✓	✓	✓	✓	✗	✓	goal
packing-box-pairs-unseen-colors*§	✓	✓	✓	✓	✓	✓	goal
packing-seen-google-objects-seq§	✗	✓	✓	✓	✗	✗	step
packing-unseen-google-objects-seq§	✗	✓	✓	✓	✓	✓	step
packing-seen-google-objects-group*§	✗	✓	✗	✓	✗	✗	goal
packing-unseen-google-objects-group*§	✗	✓	✗	✓	✓	✓	goal
align-rope*†	✓	✓	✓	✓	✗	✗	goal
separating-piles-seen-colors*†	✓	✓	✓	✓	✗	✗	goal
separating-piles-unseen-colors*†	✓	✓	✓	✓	✓	✗	goal

§ tasks that are commonly found in industry.
* tasks that have more than one correct sequence of actions.
† tasks that require manipulating deformable objects and granular media.

Table 3. Language-conditioned tasks in Ravens [2] with their associated challenges.

We extend the Ravens benchmark [2] to 10 language-conditioned. 8 out of 10 tasks have two evaluation variants, denoted by seen and unseen in their names. See Table A for an overview of the challenges associated with each task and split. Figure 5 presents the full list of attributes, shapes, and objects across seen and unseen splits. All tasks use hand-coded experts to generate expert demonstrations. These experts use privileged state information from the simulator along with pre-specified heuristics to complete the tasks. We refer the reader to the original Transporter paper [2] for details regarding these experts. The following is a description of each language-conditioned task:

A.1 Align Rope

Example: Figure 1(a).

Task: Manipulate a deformable rope to connect its end-points between two corners of a 3-sided square. There are four possible combinations for aligning the rope: "front left tip to front right tip". "front right tip to back right corner", "front left tip to back left corner", and "back right corner to back left corner". Here 'front' and 'back' refer to canonical positions on the 3-sided square. The poses of both the rope and 3-sided square are randomized for each task instance.

Objects: All align-rope instances contain a rope with 20 articulated beads and a 3-sided square.

Success Metric: The poses of all beads match the line segments between the two correct sides.

A.2 Packing Unseen Shapes

Example: Figure 1(b).

Task: Place a specified shape in the brown box. Each task instance contains 1 shape to be picked along with 4 distractor shapes. The shape colors are randomized but have no relevance to the task. This task does not require precise placements and is mostly a test of the agent's semantic understanding of arbitrary shapes.

Objects: packing-unseen-shapes is trained with seen shapes but evaluated on unseen shapes from Figure 5.

Success Metric: The correct shape is inside the bounds of the brown box.

A.3 Assembling Kits Seq

Example: Figure 1(c).

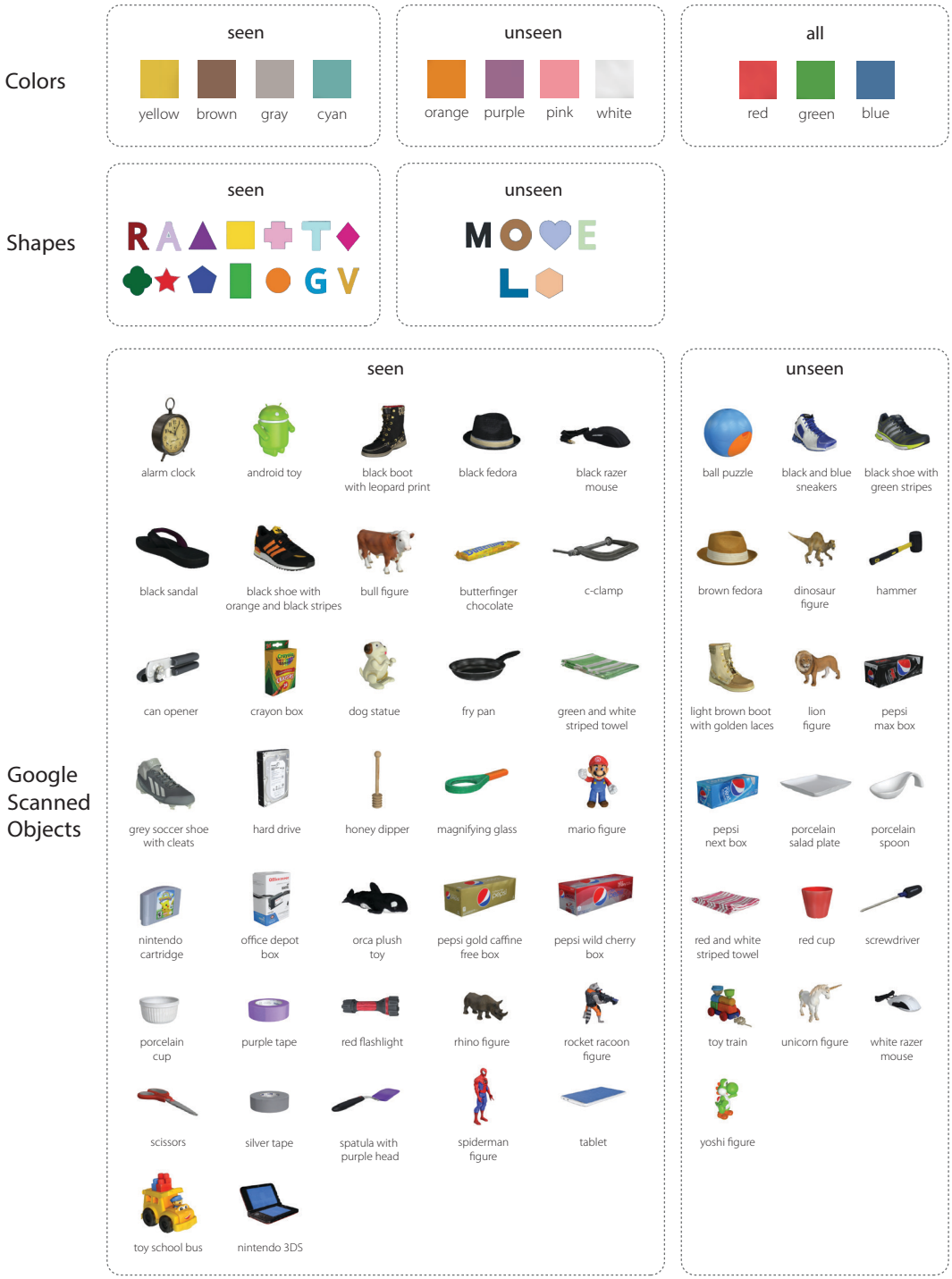

Figure 5. **Attributes and Objects**: Attributes and objects across seen and unseen splits. Shapes objects are from Transporter [2]. Other tabletop objects are from the Google Scanned Objects dataset [61]

Task: Precisely place each specified shape in the specified hole following the order prescribed in the language instruction at each timestep. This is one of the hardest tasks in the benchmark requiring precise placements of unseen shapes of unseen colors and grounding spatial relationships like "the middle square hole" or "the bottom letter R hole". Each task instance contains 5 shapes and a kit with randomized poses.

Objects: Both assembling-kits-seq-seen-colors and assembling-kits-seq-unseen-colors are trained on seen shapes but evaluated on unseen shapes from Figure 5. However for color randomization, assembling-kits-seq-seen-colors is trained and evaluated on seen colors, and assembling-kits-seq-unseen-colors is trained with seen colors but evaluated on unseen colors from Figure 5.

Success Metric: The pose of each shape matches the specified hole at the correct timestep. The final score is the total number of shapes that were placed in the correct pose at the correct timestep, divided by the total number of shapes in the scene (always 5).

A.4 Put Blocks in Bowl

Example: Figure 1(d).

Task: Place all blocks of a specified color in a bowl of specified color. Each bowl fits just one block and all scenes contain enough bowls achieve the goal. Each task instance contains several distractor blocks and bowls with randomized colors. The solutions to this task are multi-modal in that there could be several ways to place the blocks specified in the language goal. This task does not require precise placements and mostly tests an agent's ability to ground color attributes.

Objects: put-blocks-in-bowl-seen-colors is trained and evaluated on seen colors from Figure 5 for both blocks and bowls. put-blocks-in-bowl-unseen-colors is trained on seen colors but evaluated on unseen colors from Figure 5 for both blocks and bowls.

Success Metric: All blocks of the specified color are within the bounds a bowl of the specified color. The final score is the total number of correct blocks in the correct bowls, divided by the total number of relevant color blocks in the scene.

A.5 Packing Box Pairs

Example: Figure 1(e).

Task: Tightly pack all the boxes of two specified colors inside the brown box. All scenes contain the exact number of relevant color blocks to fill the box completely, but also contain some distractor boxes of irrelevant colors. The sizes of the boxes and the brown box are randomized. The distractor objects have equivalent sizes to the relevant objects to make the task more difficult. Sometimes the scene only contains one of the two specified specified colors and the agent has to actively ignore the missing color. Overall, this task requires both semantic understanding of colors and precise spatial reasoning for tightly packing boxes of unknown sizes.

Objects: Boxes with randomized widths and lengths and a brown box. packing-box-pairs-seen-colors is trained and evaluated on seen color boxes from Figure 5. packing-box-pairs-unseen-colors is trained on seen color boxes but evaluated on unseen color boxes from Figure 5.

Success Metric: All blocks of the two specified colors are tightly packed inside the bounds of the brown box. The final score is the total volume of the correct color blocks inside the box, divided by the total volume of the relevant color blocks in the scene.

A.6 Packing Google Objects Seq

Example: Figure 1(f).

Task: Place the specified objects in the brown box following the order prescribed in the language instruction at each timestep. This task does not require precise placements and mostly evaluates an agent's ability to ground semantic object descriptions. All objects in a scene are unique without any duplicates. The poses of the objects and the box are randomized for each scene.

Objects: packing-seen-google-objects-seq is trained and evaluated on all 56 objects in Figure 5. packing-unseen-google-objects-seq is trained on 37 seen objects but evaluated on 19 unseen objects in Figure 5.

Success Metric: Each specified object is within the bounds of the brown box at the correct timestep. The final score is the total volume of the correct objects placed inside the box at the correct timestep, divided by the total volume of the relevant objects.

A.7 Packing Google Objects Group

Example: Figure 1(g).

Task: Place all objects of the specified category in the brown box. This task does not require precise placements or following a specific action sequence. Each scene contains objects of multiple categories with each category containing at least 2 duplicates. The task cannot be solved by counting the number of objects since there are distractor objects, each with 2 or more duplicates.

Objects: packing-seen-google-objects-group is trained and evaluated on all 56 objects in Figure 5. packing-unseen-google-objects-group is trained on 37 seen objects but evaluated on 19 unseen objects in Figure 5.

Success Metric: All specified objects of a category are within the bounds of the brown box. The final score is the total volume of the correct objects in the box, divided by the total volume of the relevant objects of the specified category in the scene.

A.8 Stack Block Pyramid

Example: Figure 1(h).

Task: Build a pyramid of colored blocks in a color sequence specified through the step-by-step language instructions. Each task contains 6 blocks with randomized colors and 1 rectangular base, all initially placed at random poses.

Objects: 6 blocks and 1 rectangular base. stack-block-pyramid-seq-seen-colors is trained and evaluated on seen color blocks from Figure 5. stack-block-pyramid-seq-unseen-colors is trained on seen color blocks but evaluated on unseen color blocks from Figure 5.

Success Metric: The pose of each block at the corresponding timestep matches the specified location. The final score is the total number of blocks in the correct pose at the correct timestep, divided by the total number of blocks (always 6).

A.9 Separating Piles

Example: Figure 1(i).

Task: Sweep the pile of blocks into the specified zone. Each scene contains two square zones: one relevant to the task, another as a distractor. The pile and zones are placed at random poses on the table.

Objects: A pile of colored blocks and two squares. separating-piles-seen-colors is trained and evaluated on seen colors from Figure 5 for all blocks and squares. separating-piles-unseen-colors is trained on seen colors but evaluated on unseen colors from Figure 5 for all blocks and squares.

Success Metric: All blocks are inside the bounds of the specified zone. The final score is the total number of blocks inside the correct zone, divided by the total number of blocks in the scene.

A.10 Towers of Hanoi Seq

Example: Figure 1(j).

Task: Move the ring to the specified peg in the language instruction at each timestep. The sequence of ring placements is always the same, i.e. the perfect solution to three-ring Towers of Hanoi. This task can be solved without using colors by just observing the ring sizes. However, it tests the agent's ability to ignore irrelevant concepts to the task (color in this case). The task involves precise pick and place actions for moving the rings from peg to peg.

Objects: 1 peg base and 3 rings (small, medium, and big). towers-of-hanoi-seen-colors is trained and evaluated on seen ring colors from Figure 5. towers-of-hanoi-unseen-colors is trained on seen ring colors but evaluated on unseen ring colors from Figure 5.

Success Metric: The pose of each ring at the corresponding timestep matches the specified peg location. The final score is the total number of correct ring placements, divided by total steps in the perfect solution (7 for three-ring Towers of Hanoi).

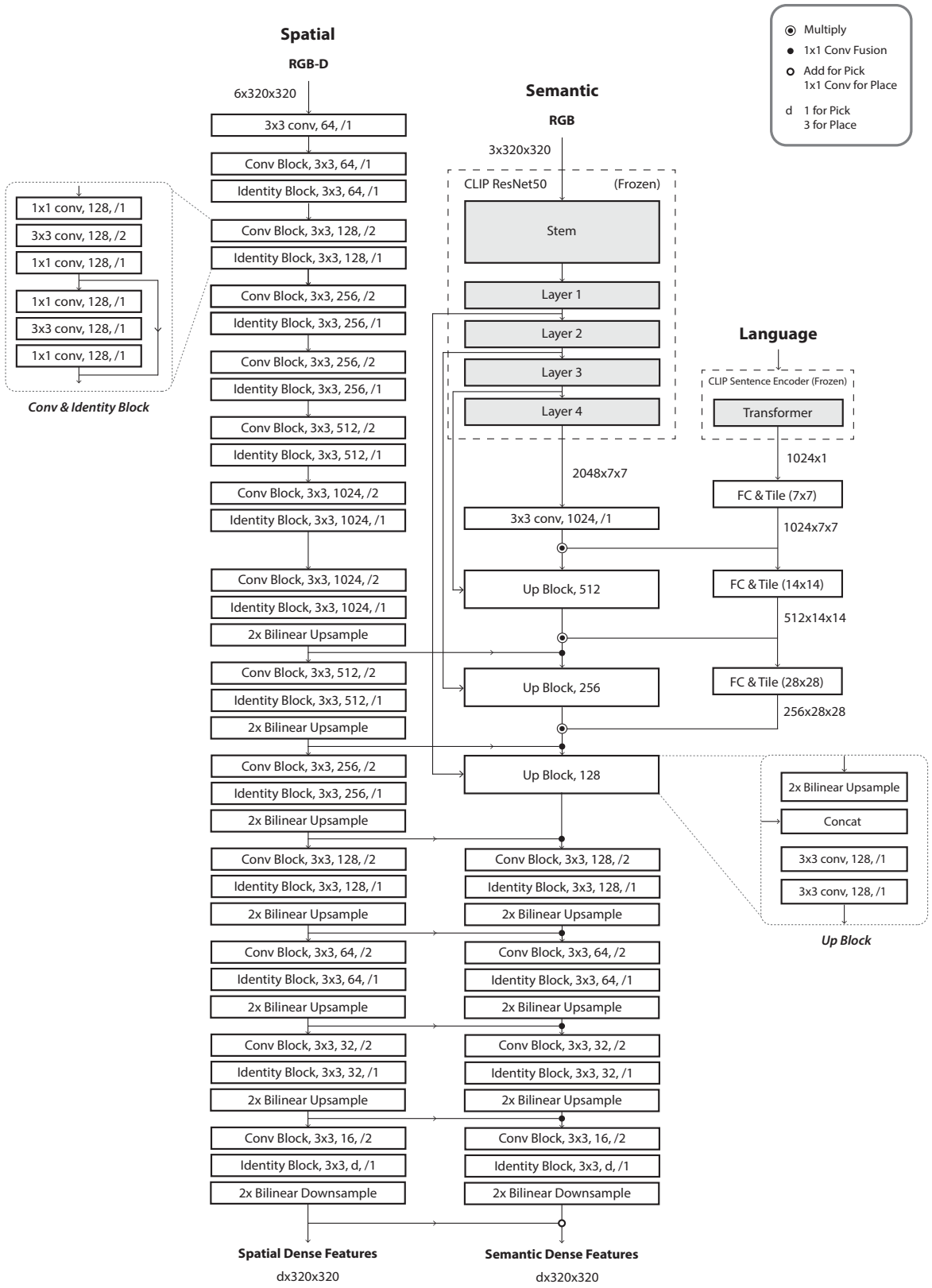

Figure 6. **CLIPORT Two-Stream Architecture**: A detailed architecture diagram of the semantic and spatial pathways.

B Two Stream Architecture Details

Figure 6 provides a detailed architecture diagram of CLIPORT's two-stream design. We use ReLU activations after each `conv` and `identity` blocks without any Batch Normalization. Note that we repeat the depth input to match the dimensions of the RGB image $\mathbb{R}^{H\times W\times 1} \rightarrow \mathbb{R}^{H\times W\times 3}$ following Transporter [2]. All models were implemented in PyTorch [64]. For CLIP, we use the implementation and pre-trained checkpoint released by the authors[2].

C Robot Setup

Hardware Setup. All real-robot experiments were conducted on a Franka Panda robot with a parallel-gripper. For perception, we use a Kinect-2 RGB-D camera mounted on a tripod, tilted down looking at the table. Although the Kinect-2 provides images at a resolution of 1280×720, we use downsampled 960×540 images for a faster user-interface. The extrinsic calibration between the camera and the robot base-frame is computed with an AR Marker through ARUCO ROS[3]. See Figure 7 for an overview of the setup.

Figure 7. Real-Robot Experimental Setup.

Demonstrations and Execution. For collecting demonstrations with the Franka Panda, we developed a 2D interactive tool that uses the top-down RGB view from the Kinect-2 to specify pick-and-place locations. The user first selects a 2D bounding box on the live RGB feed, and then picks a discrete rotation angle by clicking around the bounding box. For grasping, we use a simple heuristic to determine the height at which to close the fingers. First we segment the pointcloud encapsulated by the bounding box, then we vertically crop the pointcloud up to the height of the gripper fingers, and then compute a 3D centroid of the selected points by taking an average. This 3D centroid is used to plan a path for the end-effector with an RRT* motion-planner to execute a predefined sequence – go down, open/close the gripper, raise up. For executing a trained CLIPORT model, a similar grasping approach is used, but instead of the user-specified bounding box, we take 32×32 crops centered around the pick and place predictions (i.e. affordance argmax) to compute 3D centroids from the pointcloud. Only the sweeping and folding actions are different in that the end-effector does not raise up after grasping.

Pick Rotations for Parallel Grippers. The suction gripper used in simulation does not require a pick rotation since the grasps are specified as pin-point locations. However, with the Franka Panda, the parallel gripper requires a specific yaw rotation at which to grasp an object. To handle this, we separate the pick module $\mathcal{Q}_{\text{pick}}$ into two components: locator and rotator. The locator predicts a pixel location (u, v) given the full observation and language input. The rotator takes a 64×64 crop of the observation at (u, v) along with the language input and predicts a discrete rotation angle by selecting from one of the k rotated crops. We use $k = 36$ in all our hardware experiments. While it's possible to predict both the location and rotation with a single module, this decoupled approach allows us to fit the model on a single GPU (NVIDIA P100) with reduced memory usage from cropped rotations.

D Data Augmentation

Following common practice and the original Transporter implementation [2], we augment the training samples by applying random $\mathbf{SE}(2)$ transformations. Augmentations where $\mathcal{T}_{\text{pick}}$ or $\mathcal{T}_{\text{place}}$ are out of frame after the transformation are discarded. These augmentations are particular important for learning spatially-equivariant representations with FCNs without overfitting to images from limited training demonstrations.

[2] https://github.com/openai/CLIP
[3] https://github.com/pal-robotics/aruco_ros

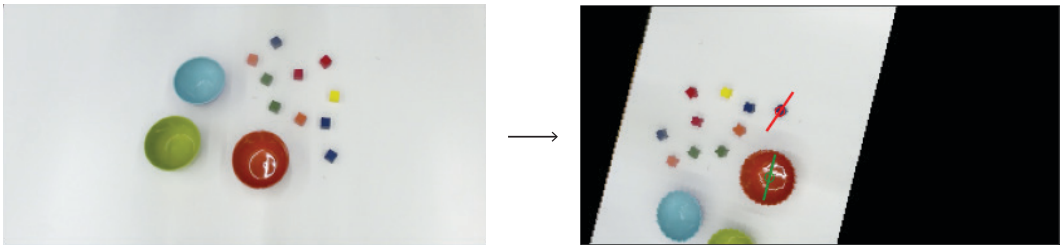

Figure 8. **Data Augmentation**: $\mathbf{SE}(2)$ transform applied to RGB-D input. The left image shows the original input, and the right image shows the transformed input along with expert $\mathcal{T}_{\text{pick}}$ (red) and $\mathcal{T}_{\text{place}}$ (green) actions.

E Ablations and Baselines

	stack-block-pyramid seq-seen-colors				stack-block-pyramid seq-unseen-colors				packing-seen-google object-seq				packing-unseen-google object-seq			
Method	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
One-Stream Transporter-only	6.8	4.0	6.8	5.7	4.8	5.3	5.0	5.0	29.0	39.1	39.5	41.5	25.9	35.1	41.7	37.9
One-Stream CLIP-only	3.8	30.0	58.7	54.7	2.0	16.3	21.3	14.8	29.3	**76.0**	86.4	**97.0**	36.5	46.2	58.8	58.0
One-Stream Language Transporter	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.1	0.0	0.1	0.1	1.7	0.2	0.1	0.2
One-Stream Image-Goal Transporter	8.5	2.5	8.5	7.7	3.7	6.0	6.8	6.2	**58.8**	63.7	78.3	84.0	**41.3**	**59.2**	**78.4**	**79.7**
Two-Stream CLIP-Transporter w/o skips	0.2	4.7	4.3	6.8	3.8	4.7	5.7	3.5	25.5	36.3	37.8	42.0	24.8	40.6	40.2	36.5
Two-Stream Untrained-Sem-Transporter	3.3	13.3	59.7	49.5	3.7	9.0	15.0	13.8	32.4	42.6	70.6	83.0	25.6	37.9	39.5	42.0
Two-Stream RN50-BERT-Transporter	5.7	35.5	94.0	98.0	5.2	10.5	19.7	33.8	27.9	48.4	87.9	94.6	37.7	40.5	55.8	56.3
Two-Stream CLIP-Transporter (ours)	**29.0**	**68.8**	**95.0**	**99.3**	**15.8**	**29.0**	**32.7**	**41.8**	15.3	60.3	**92.9**	96.5	25.9	52.0	62.5	61.6

Table 4. **Ablations and Baselines.** Evaluation scores (mean %) for stack-block-pyramid-seq and packing-google-objects-seq tasks from 100 evaluation runs. Stacking block pyramids involves both semantic and precise spatial reasoning, whereas packing objects mostly involves semantic grounding without precise placements.

Table 4 presents various baselines and ablations from our simulated experiments. The following is a description of each model:

One-Stream Transporter-only is the original Transporter [2] with RGB-D input, or equivalently, the **spatial** stream of CLIPORT. For all experiments, we implemented our own version of Transporter in PyTorch and did not use the modeling code provided with the original paper. Our Transporter models are also trained for 200K iterations instead of 40k iterations.

One-Stream CLIP-only is the **semantic** stream of CLIPORT with RGB and language input.

One-Stream Language Transporter is Transporter [2], but the bottleneck features are conditioned with CLIP language features in a similar fashion to the **semantic** stream in CLIPORT. This model performs very poorly because the high-level language features corrupt the low-level spatial features necessary for precise pick-and-place actions.

One-Stream Image-Goal Transporter is a goal-conditioned version of Transporter [6] which receives a goal-image as input. For sequential tasks with a specific order (indicated with seq in their name), we provide the goal-image from the next timestep, and for non-sequential tasks we provide the goal-image from the final timestep. The implementation follows the goal-conditioned Transporter proposed in [6], except we found that element-wise addition worked better than element-wise product for combining goal-image features with $\mathcal{Q}_{\text{place}}$ features.

Two-Stream CLIP-Transporter w/o skips is a variant of the CLIPORT model without skip connections from the CLIP-ResNet encoder to the decoder layers. The results in Table 4 show that these skip connections are particularly important for good performance. We hypothesize that utilizing different levels of semantic information from the visual encoder – patterns, shapes, parts, objects, and high-level concepts, is crucial for conditioning the **semantic** stream decoders.

Two-Stream RN50-BERT-Transporter is the same two-stream architecture as CLIPORT, except instead of the CLIP ResNet50, we use a standard ResNet50 [62] pre-trained on ImageNet classification. And instead of the CLIP sentence encoder, we use a pretrained DistilBERT model [65] to extract language embeddings. CLIP offers the benefit of multi-modal alignment between vision and language features while not being restricted to instance segmentation or bounding box detection pipelines.

Two-Stream Untrained-Sem-Transporter uses an untrained ResNet50 and Transformer language encoder for the **semantic** stream. Even without any pre-training, the random features from the **semantic** stream somewhat help in conditioning policies. However, the performances are substantially worse than models with pre-trained multimodal features, which enable better generalization.

F Performance on Demo-Conditioned Tasks

	block-insertion				place-red-in-green				towers-of-hanoi				align-box-corner				stack-block-pyramid			
Method	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
Transporter [2]	97.0	**100**	**100**	**100**	**100**	**100**	**100**	**100**	52.3	90.3	98.7	**100**	**69.0**	85.0	**100**	97.0	51.7	74.8	**96.8**	**99.3**
CLIPORT w/o Lang	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**100**	**88.7**	**99.0**	**99.7**	**100**	59.0	98.0	99.0	99.0	**71.0**	**92.0**	95.3	97.8
Transporter (multi) [2]	98.0	99.0	**100**	**100**	91.5	99.5	**100**	**100**	49.6	79.6	96.3	92.9	50.0	**99.0**	99.0	**100**	16.3	37.3	36.0	26.7
CLIPORT w/o Lang (multi)	0.0	99.0	**100**	**100**	0.0	94.7	**100**	92.5	0.0	57.6	85.9	75.3	0.0	86.0	98.0	**100**	0.0	66.0	80.8	77.7

	palletizing-boxes				assembling-kits				packing-boxes				manipulating-rope				sweeping-piles			
	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000	1	10	100	1000
Transporter [2]	**91.6**	**99.0**	**99.9**	**99.9**	33.2	67.4	**98.2**	**100**	88.6	96.0	98.2	**100**	62.7	78.5	93.7	97.8	98.8	100	99.9	99.8
CLIPORT w/o Lang	89.4	98.6	99.6	99.4	**52.8**	83.2	92.8	97.8	**96.9**	**99.5**	**100**	**100**	**69.4**	**93.6**	**97.9**	**100**	**99.2**	**100**	**100**	**100**
Transporter (multi) [2]	90.7	98.7	99.7	99.1	22.6	58.6	66.8	68.8	93.4	96.6	**100**	**100**	34.3	68.7	87.2	83.7	92.5	97.0	95.6	97.3
CLIPORT w/o Lang (multi)	0.0	61.1	94.9	86.4	0.0	**86.6**	95.2	89.0	0.4	98.8	99.3	**100**	0.4	90.0	85.2	93.2	6.5	99.8	**100**	**100**

Table 5. **Demo-Conditioned Tasks.** Task success scores (mean %) from 100 evaluation instances vs. # of demonstration episodes (1, 10, 100, or 1000) used in training.

To investigate if our framework can be applied to demo-conditioned tasks that do not require language instructions, we run evaluations on the original Transporter tasks [2]. Table 5 compares our two-stream architecture without language conditioning to Transporter. Our method outperforms Transporter in $30/40 = 75\%$ of the evaluations in Table 5, especially in low-data regimes with 100 demonstrations or less. Particularly for the assembling-kits and manipulating-rope tasks, the two-stream architecture shows significant performance gains. We hypothesize that this is because the CLIP-ResNet model provides a strong visual prior on object representations for learning generalizable policies.

G Affordance Prediction Examples

Figure 9 showcases more examples of affordance predictions from trained CLIPORT (multi) models. Traditional object-centric representations like pose and instance segmentation generally struggle to represent piles of beans or squares on a chessboard. In such cases, a single detector would have to be trained (with supervision data) to detect every bean and square on the chessboard, which is often infeasible, especially in multi-task settings.

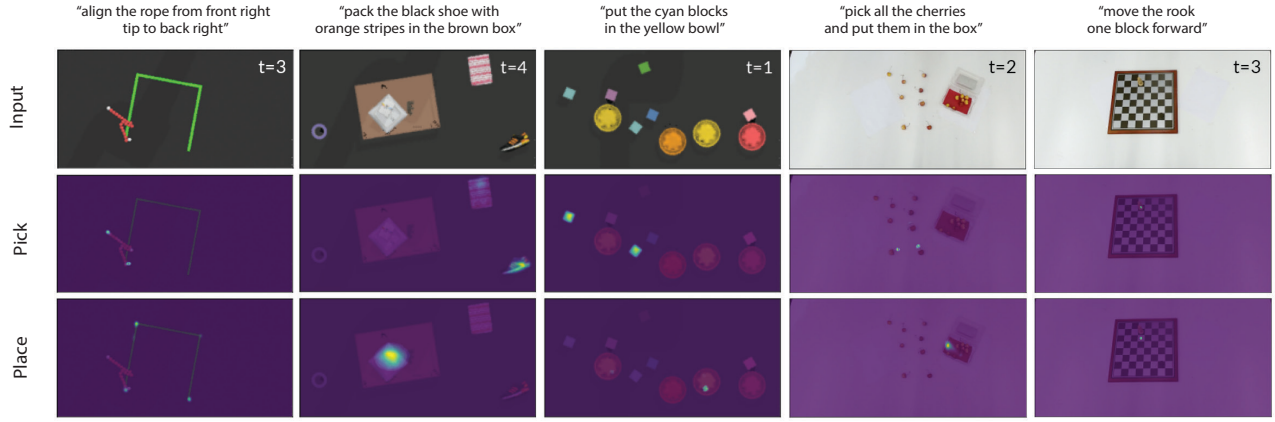

Figure 9. More examples of pick and place affordance predictions from CLIPORT (multi). The left three columns are from simulated tasks, and the right two columns are from real-world tasks.

H Limitations and Risks

While CLIPORT is highly capable, it is not without issues. In the following sections we discuss various limitations and risks of using CLIPORT for real-world manipulation.

Balanced Datasets. CLIPORT can learn generalizable policies from very few demonstrations, but it relies heavily on a balanced training dataset with a good converge of expected skills and invariances. As discussed in Section 4.3, the model will exploit any bias, e.g. always place "yellow blocks" inside 'blue bowls" if that is the only example of yellow blocks that it's provided with. Sometimes these biases can be hard to spot since everything (from perception to action) is trained end-to-end through demonstrations. During our real-world experiments we ended up iteratively refining some datasets after finding such biases during execution.

Hand-Eye Calibration and Closed-Loop Control. The execution of policies is sensitive to the accuracy of the hand-eye calibration. The action-space of CLIPORT is 2D pixels with yaw-rotations. Translating these pixel coordinates to end-effector poses relies on carefully calibrated extrinsics between the robot's base frame and the RGB-D camera. Further, while the framework takes closed-loop actions across discrete pick-and-place timesteps, the execution of each pick and place primitive itself is open-loop. This restricts usage to mostly quasi-static tasks and leads to issues if objects move while the robot is executing a pick or place primitive. Future works could incorporate a separate visuo-servoing mechanism for more robust grasping.

Dexterous Manipulation. Extending CLIPORT's action-space to 6-DOF or N-DOF control for dexterous non-quasi-static manipulation is non-trivial. The $\mathbf{SE}(2)$ action-space is one of the key factors that make Transporter and CLIPORT highly data efficient. Since the actual end-effector control is abstracted away, the model can easily reason about high-level affordances at discrete timesteps, but at the price of loosing dexterity. Similarly, extending $\mathbf{SE}(2)$ equivariance to $\mathbf{SE}(3)$ equivariance is also non-trivial. Cross-correlating in voxelized 3D spaces might be expensive and slow.

Grasping Novel Objects. CLIPORT has some limited capacity in grasping unseen instances of objects in one-shot or few-shot settings. While CLIP is a pure vision-language model with no understanding of affordances, actions, or physical properties, in CLIPORT we fine-tune CLIP's visual representations in the **semantic** decoder layers to produce visual affordance predictions – like grasping pliers by the handle. We illustrate this in Figure 10 with an example of one-shot learning. Despite having seen just a single training example with pliers, CLIPORT is able to correctly grasp the handles of 2/3 unseen pliers of different shapes, sizes, and colors. The model fails in Test 3 where the instance is significantly outside the training distribution. But even so, the model is able to correctly localize the pliers among the distractor objects, and with a few more training examples it might be able to correctly grasp the instance. In contrast, RN50-BERT struggles to identify pliers with just a single example since pliers are not part of the 1000 ImageNet classes [66]. Further, without the appropriate language goal to condition the policy, e.g. when provided with a nonsensical object name like "dax", the model falls back to the most familiar object seen during training.

Grounding Complex Object Relationships. In general, CLIPORT struggles with complex object-relationships that require reasoning about several objects. The model performs poorly on assembling-kits-seq tasks that involve grounding spatial relationships like "middle" with unseen shapes and language. The model's capacity to infer these relationships purely from dense global features might be limited. Also, CLIPORT cannot count objects since it does not maintain a history or belief across timesteps, thus limiting instructions to 'any' or 'all' quantifiers. Future works could explore neuro-symbolic [67] or attention-based [68] methods for better generalization to novel object-relationships.

Scope of Language Grounding. CLIPORT's understanding of verb-noun phrases is tightly grounded in the demonstrations and tasks seen during training. For instance, an user could have used "sort out all the Mars bars from the pile and put them in the yellow bin" while demonstrating a task. Here the model only understands 'sort' in the context of separating something from the pile and putting it in a bin, and not in the most generic sense that is applicable in any context, like sorting numbered blocks in descending order.

Task Completion. CLIPORT relies on an expert to indicate task-completion. For real-world tasks, this means the model keeps taking actions until an user stops the execution. Future works can address this issue by training a success classifier [2] to predict task completion from RGB-D observations.

Risks from Pre-Trained Models. CLIP was trained with massive amounts of "in-the-wild" image-caption pairs from the internet. This makes it prone to unchecked biases and associations [59, 69] that can be harmful to certain individuals and communities. These issues are further exacerbated by the fact that we use CLIP's representations to take actions with a physical robot. The end-to-end framework is also vulnerable to adversarial attacks [59] that try to maliciously affect the model's behavior. For safe deployment in the real-world, keeping humans in the loop – both during the training phase and while instructing the robot, might help in mitigating some of these issues and potential risks.

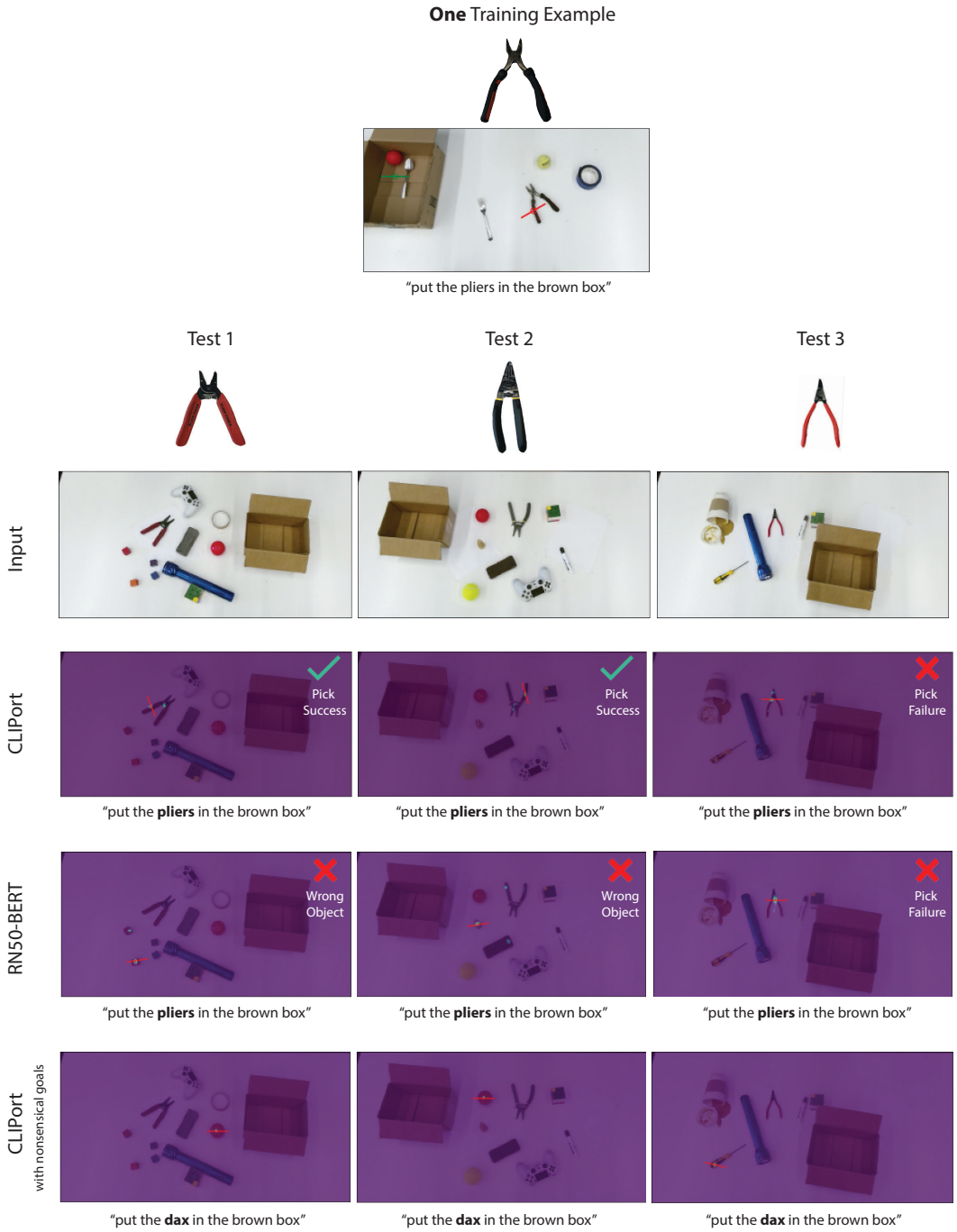

Figure 10. **One-Shot Learning**. Selected examples of grasping pliers with CLIPORT, RN50-BERT, and CLIPORT with nonsensical goals.