TABLE III

OBJECT-ONLY MODEL ACCURACY FOR THE **ALL PAIRS** TASK.

	Model (M) Ego	Objects	Prediction Correct ↑ *in*	*on*
Dev		✓	$.86 \pm .04$	$.78 \pm .01$
	Baseline (MC)		$.87 \pm .00$	$.73 \pm .00$
Test		✓	$.87 \pm .02$	$.83 \pm .01$
	Baseline (MC)		$.84 \pm .00$	$.83 \pm .00$

✓indicates signal was included.

TABLE IV

OBJECT-ONLY MODEL ACCURACY FOR THE **ROBOT PAIRS** TASK.

	Model (M) Ego	Objects	Prediction Correct ↑ *in*	*on*
Dev		✓	$.85 \pm .02$	$.57 \pm .03$
		pre	$.89 \pm .03$	$.58 \pm .04$
	Baseline (MC)		$.32 \pm .00$	$.36 \pm .00$
Test		✓	$.82 \pm .04$	$.48 \pm .02$
		pre	$.87 \pm .02$	$.51 \pm .03$
	Baseline (MC)		$.20 \pm .00$	$.32 \pm .00$

✓indicates signal was included, while "pre" indicates models with object features pre-trained from **All Pairs** data.

VIII. APPENDIX

a) Additional Results: Performance of the M^{obj} models on the auxiliary task of predicting Mechanical Turk workers' annotations for whether objects can be stacked *in* or *on* one another given only object data in the form of object images and referring expressions is given in Table III for the *Test* and *Development* data folds on **All Pairs**. In most cases, the prediction model achieves matching or higher accuracy than majority class, with the exception of losing about 1% on the *Development* fold on the *in* annotation prediction task.

This auxiliary task facilitates pretraining the $M^{\text{ego+pre}}$ model examined above, but we can also classify scenes using no egocentric data at all, making predictions based on priors alone with the M^{obj} and M^{pre} models. Table IV gives the performance of these prior-based, object data only models on **Robot Pairs**. In this case, the models are *predicting* what happens to pairs of objects, as opposed to *detecting* it from available egocentric scene information.

There are two notable points. First, the M^{pre} models achieve higher success prediction accuracy than the M^{obj} models in all cases across folds and tasks, showing that pre-training on the auxiliary prediction task refines performance. Second, these prior-only models outperform the egocentric-informed models of Table I when predicting the success of *in* relations, while falling short when predicting the success of *on* relations. This relates to the discussion in Section VI regarding *on* being a wider, more general relation. In particular, static data about objects is less informative when predicting *on* relations than *in* relations, since *on* lacks indicator features like the words *tiny* and *bowl* that facilitate *in* prediction without access to egocentric scene data.

b) YCB Object Details: We make a number of small changes and omissions from the full YCB Object Set when establishing our included objects Y (Section III). In particular, we:

- exclude objects lacking camera image data used as vision information to the augmented model (Fig. 3);
- exclude `072-*_toy_airplane` parts `b-k`;
- do not split the two similar, *medium-most-sized* `063-f_cups` and `063-e_cups` objects into different folds, to evaluate more conservatively; and
- for `063-j_cups`, we sometimes use a same-sized cup that is light blue instead of yellow during robot trials.

c) Annotation Details: For *on* and *in* labels across the five trials gathered for each *Robot Pair*, the outcome label was annotated in {*Yes*, *No*, *Maybe*}. The *Maybe* annotation denotes that there was mixed success across the five trials. Similarly, for the Mechanical Turk annotations used during the auxiliary task for data augmentation, when annotators disagreed about the annotation in {*Yes*, *No*}, *Maybe* was assigned. Throughout our experiments, we round *Maybe* annotations to the *No* label for both robot trial and Mechanical Turk annotations.

For Mechanical Turk, annotators answer "yes", "no", or "yes, but only if object A is rotated." We rounded this final option down to "no" for our task, but the original label may be useful for other manipulation tasks.