Grasp2Vec: Learning Object Representations from Self-Supervised Grasping

Eric Jang*[1], Coline Devin*†[2], Vincent Vanhoucke[1], and Sergey Levine[1,2]

[1]Google Brain
[2]Department of Electrical Engineering and Computer Sciences, UC Berkeley
`{ejang, vanhoucke, slevine}@google.com`
[1]`coline@eecs.berkeley.edu`

Abstract: Well structured visual representations can make robot learning faster and can improve generalization. In this paper, we study how we can acquire effective object-centric representations for robotic manipulation tasks without human labeling by using autonomous robot interaction with the environment. Such representation learning methods can benefit from continuous refinement of the representation as the robot collects more experience, allowing them to scale effectively without human intervention. Our representation learning approach is based on object persistence: when a robot removes an object from a scene, the representation of that scene should change according to the features of the object that was removed. We formulate an arithmetic relationship between feature vectors from this observation, and use it to learn a representation of scenes and objects that can then be used to identify object instances, localize them in the scene, and perform goal-directed grasping tasks where the robot must retrieve commanded objects from a bin. The same grasping procedure can also be used to automatically collect training data for our method, by recording images of scenes, grasping and removing an object, and recording the outcome. Our experiments demonstrate that this self-supervised approach for tasked grasping substantially outperforms direct reinforcement learning from images and prior representation learning methods.

Keywords: instance grasping, unsupervised learning, reinforcement learning

1 Introduction

Robotic learning algorithms based on reinforcement, self-supervision, and imitation can acquire end-to-end controllers from images for diverse tasks such as robotic mobility [1, 2] and object manipulation [3, 4]. These end-to-end controllers acquire perception systems that are tailored to the task, picking up on the cues that are most useful for the control problem at hand. However, if our aim is to learn generalizable robotic skills and endow robots with broad behavior repertoires, we might prefer perceptual representations that are more structured, effectively disentangling the factors of variation that underlie real-world scenes, such as the persistence of objects and their identities. A major challenge for such representation learning methods is to retain the benefit of self-supervision, which allows leveraging large amounts of experience collected autonomously, while still acquiring the structure that can enable superior generalization and interpretability for downstream tasks.

In this paper, we study a specific instance of this problem: acquiring object-centric representations through autonomous robotic interaction with the environment. By interacting with the real world, an agent can learn about the interplay of perception and action. By looking at and picking up objects, a robot can discover the relationships between image pixels and physical entities. If a robot grasps something in it's environment and lifts it out of the way, then it could conclude that anything still visible was not part of what it grasped. It can also look at its gripper and see the object from a new

*Both authors contributed equally
†Work done while author was interning at Google Brain

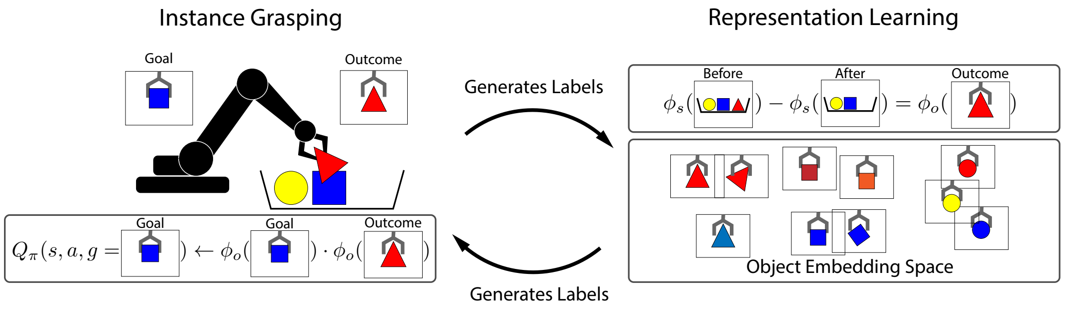


Figure 1: Instance grasping and representation learning processes generate each other's labels in a fully self-supervised manner. ***Representation learning from grasping:*** *A robot arm removes an object from the scene, and observes the resulting scene and the object in the gripper. We enforce that the difference of scene embeddings matches the object embedding.* ***Supervising grasping with learned representations:*** *We use a similarity metric between object embeddings as a reward for instance grasping, removing the need to manually label grasp outcomes.*

angle. Through active play, a robot could learn which pixels in an image are graspable objects and recognize particular objects across different poses without any human supervision.

While object-centric representations can be learned from semantically annotated data (e.g., the MSCOCO dataset [5]), this approach precludes continuous self-improvement: additional experience that the robot collects, and lacks the human annotations, is not directly used to improve the quality and robustness of the representation. In order to improve automatically, the representation must be self-supervised. In that regime, every interaction that the robot carries out with the world improves its representation.

Our representation learning method is based on object persistence: when a robot picks up an object and removes it from the scene, the representation of the scene should change in a predictable way. We can use this observation to formulate a simple condition that an object-centric representation should satisfy: the features corresponding to a scene should be approximately equal to the feature values for the same scene after an object has been removed, minus the feature value for that object (see Figure 1). We train a convolutional neural network feature extractor based on this condition, and show that it can effectively capture individual object identity and encode sets of objects in a scene without any human supervision.

Leveraging this representation, we propose learning a self-supervised grasping policy conditioned on an object feature vector or image. While labeling whether the correct object was grasped would typically require human supervision, we show that the similarity between object embeddings (learned with out method) provides an equally good reward signal.

Our main contribution is grasp2vec, an object-centric visual embedding learned with self-supervision. We demonstrate how this representation can be used for object localization, instance detection, and goal-conditioned grasping, where autonomously collected grasping experience can be relabeled with grasping goals based on our representation and used to train a policy to grasp user-specified objects. We find our method outperforms alternative unsupervised methods in a simulated goal-conditioned grasping results benchmark. Supplementary illustrations and videos are at `https://sites.google.com/site/grasp2vec/`

2 Related Work

Unsupervised representation learning. Past works on interactive learning have used egomotion of a mobile agent or poking motions [6, 7, 8, 9, 10] to provide data-efficient learning of perception and control. Our approach learns representations that abstract away position and appearance, while preserving object identity and the combinatorial structure in the world (i.e., which objects are present) via a single feature vector. Past work has also found that deep representations can exhibit intuitive linear relationships, such as in word embeddings [11], and in face attributes [12]. Wang et. al represent actions as the transformation from precondition to effect in the action recognition domain [13]. While our work shares the idea of arithmetic coherence over the course of an action, we optimize a different criterion and apply the model to learning policies rather than action recognition.

Self-supervised grasping. A recent body of work has focused on deep visuomotor policies for picking up arbitrary objects from RGB images [14, 4, 15, 16]. By automatically detecting whether some object was grasped, these methods can learn without human supervision. Ten Pas et al. combine object detection with grasping to be able to grasp target objects by providing class labeled training data for each object [17] and using grasp predictions as object proposals.

Addressing the task of instance grasping requires inferring whether the correct object was grasped. Jang et al. propose a system where the robot presents images of objects to the camera after it has grasped them and attempts to label their class given human labels. [18] Fang et al. obtain labels in simulation and use domain adaptation to generalize the policy to real-world scenes, which requires solving a simulation-to-reality transfer problem [19].

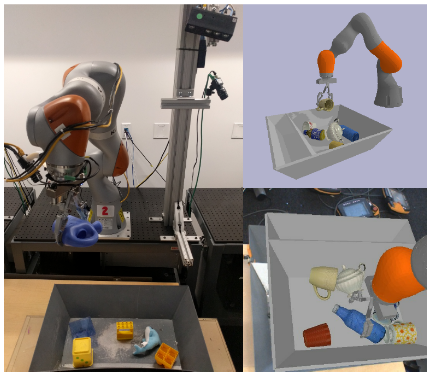


Figure 2: Self-supervised robotic setup for learning grasp2vec and goal-conditioned grasping. Left: A KUKA iiwa uses a monocular RGB camera to pick up objects from a bin. Right: The same setup in simulation. Bottom right: Example images that are fed into the Q function shown in Figure 3.

In the standard RL setting, several past works have studied labeling off-policy behavior with "unintentional rewards" [20, 21]. However, such algorithms do not address how to detect whether the desired goal was achieved, which is non-trivial outside of the simulator. Our methods circumvent the problem of labeling entirely via self-supervised representation learning. To our knowledge, this is the first work that learns the instance grasping task in a completely label-free manner.

Concurrent work by Florence et al. uses a pixelwise contrastive loss change detection in depth images of grasped objects to learn descriptors. In contrast to this work, we report quantitative results that indicate our method achieves high accuracy in identifying, localizing, and grasping objects, even when the instance is specified with a different view, under heavy occlusion, or under deformation.

3 Grasp2Vec: Representation Learning from Grasping

Our goal is to learn an object-centric embedding of images. The embeddings should represent objects via feature vectors, such that images with the same objects are close together, and those with different objects are far apart. Because labels indicating which objects are in an image are not available, we rely on a self-supervised objective. Specifically, we make use of the fact that, when a robot interacts with a scene to grasp an object, this interaction is quantized: it either picks up one or more whole objects, or nothing. When an object is picked up, we learn that the initial scene must have contained the object, and that the scene after must contain one fewer of that object. We use this concept to structure image embeddings by asking that feature difference of the scene before and after grasping is close to the representation of the grasped object.

We record grasping episodes as images triples: $(s_{\text{pre}}, s_{\text{post}}, \mathbf{o})$, where s_{pre} is an image of the scene before grasping, s_{post} is the same scene after grasping, and $\mathbf{o}$ is an image of the grasped object held up to the camera. The specific grasping setup that we use, for both simulated and real image experiments, is described in Section 5. Let $\phi_s(s_{\text{pre}})$ be a vector embedding of the input scene image (i.e., a picture of a bin that the robot might be grasping from). Let $\phi_o(\mathbf{o})$ be a vector embedding of the outcome image, such that $\phi_s(s_{\text{pre}})$ and $\phi_o(\mathbf{o})$ are the same dimensionality. We can express the logic in the previous paragraph as an arithmetic constraint on these vectors: we would like $(\phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}}))$ to be equal to $\phi_o(\mathbf{o})$. We also would like the embedding to be non-trivial, such that $(\phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}}))$ is far from the embeddings of other objects that were not grasped.

Architecture. In order to embed images of scenes and outcomes, we employ convolutional neural networks to represent both ϕ_s and ϕ_o. The two networks are based on the ResNet-50 [22] architecture followed by a ReLU (see Figure 3), and both produce 3D (HxWx1024) convolutional feature maps $\phi_{s,\text{spatial}}$ and $\phi_{g,\text{spatial}}$. Since our goal is to obtain a single vector represention of objects and sets of objects, we convert these maps into feature vectors by globally average-pooling: $\phi_s = \Sigma_{i<H} \Sigma_{j<W} \phi_{s,\text{spatial}}(X)[i][j] / H*W$ and equivalently for ϕ_o. The motivation for this architecture is that

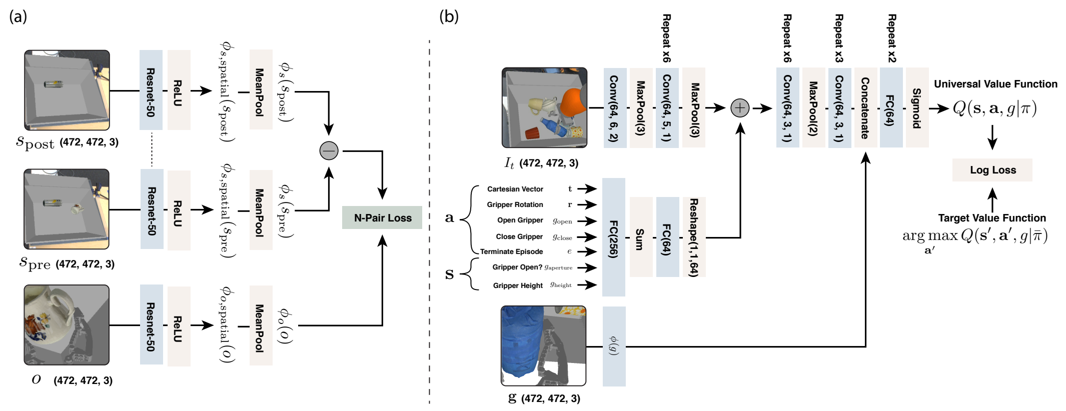

(a)
(b)
s_{post} (472, 472, 3)
s_{pre} (472, 472, 3)
o (472, 472, 3)
Resnet-50
ReLU
$\phi_{s,\text{spatial}}(s_{\text{post}})$
$\phi_{s,\text{spatial}}(s_{\text{pre}})$
$\phi_{o,\text{spatial}}(o)$
MeanPool
$\phi_s(s_{\text{post}})$
$\phi_s(s_{\text{pre}})$
$\phi_o(o)$
N-Pair Loss
I_t (472, 472, 3)
Conv(64, 6, 2)
MaxPool(3)
Repeat x6
Conv(64, 5, 1)
MaxPool(3)
Conv(64, 3, 1)
MaxPool(2)
Repeat x3
Concatenate
Repeat x2
FC(64)
Sigmoid
Universal Value Function
$Q(\mathbf{s}, \mathbf{a}, g|\pi)$
Log Loss
Target Value Function
$\arg\max_{\mathbf{a}'} Q(\mathbf{s}', \mathbf{a}', g|\bar{\pi})$
a
Cartesian Vector t
Gripper Rotation r
Open Gripper g_{open}
Close Gripper g_{close}
Terminate Episode e
s
Gripper Open? g_{aperture}
Gripper Height g_{height}
FC(256)
Sum
FC(64)
Reshape(1,1,64)
$\phi(g)$
g (472, 472, 3)

Figure 3: (a) The Grasp2Vec architecture. We use the first 3 blocks of Resnet-50 V2 to form ϕ_s and ϕ_o, which are randomly initialized. $\phi_s(s_{pre})$ and $\phi_s(s_{post})$ have tied weights. The output of the third resnet block is passed through a ReLu to yield a spatial feature map we call $\phi_{spatial}$. This is then mean pooled globally to output the single vector embddings ϕ_s and ϕ_o. The n-pairs loss is applied as described in Section 3. (b) The instance grasping architecture is conditioned on Grasp2Vec goal embeddings of goal images, and is trained via Q-learning.

it allows $\phi_{s,\text{spatial}}$ to encode object position by which receptive fields produce which features. By applying a ReLU non-linearity to the spatial feature vectors, we constrain object representations to be non-negative. This ensures that a set of objects can only grow as more objects are added; one object cannot be the inverse of another.

Objective. The problem is formalized as metric learning, where the desired metric places $(\phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}}))$ close to $\phi_o(\mathbf{o})$ and far from other embeddings. Many metric learning losses use the concept of an "anchor" embedding and a "positive" embedding, where the positive is brought closer to the anchor and farther from the other "negative" embeddings. One way to optimize this kind of objective is to use the n-pairs loss [23] to train the encoders ϕ_s and ϕ_o, such that paired examples (i.e., $(\phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}}))$ and $\phi_o(\mathbf{o})$ are pushed together, and unpaired examples are pushed apart. Rather than processing explicit (anchor, positive, negative) triplets, the n-pairs loss treats all *other* positives in a minibatch as negatives for an (anchor, positive) pair. Let i index into the anchors a of a minibatch and let j index into the positives p. The objective is to maximize $a_i \top p_i$ while minimizing $a_i \top p_{j \neq i}$. The loss is the the sum of softmax cross-entropy losses for each anchor i accross all positives p.

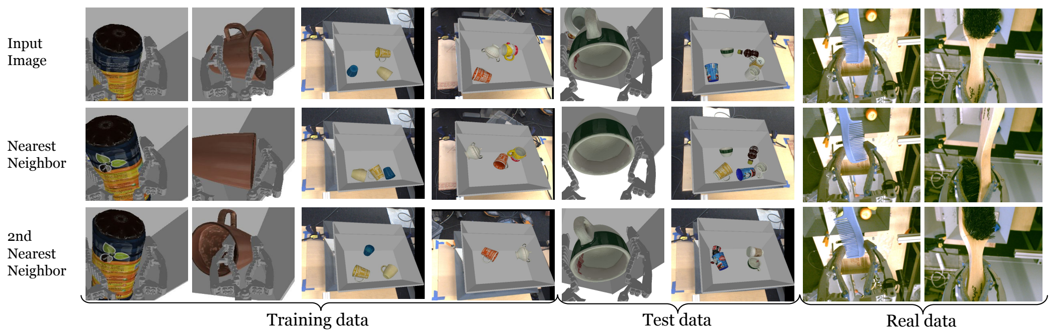
$$\text{NPairs}(a, p) = \sum_{i<B} -\log\left(\frac{e^{a_i \top p_i}}{\sum_{j<B} e^{a_i, p_j}}\right) + \lambda(||a_i||_2^2 + ||p_i||_2^2).$$

The hyperparameter λ regularizes the embdding magnitudes and B is the batch size. In our experiments, $\lambda = 0.0005$ and $B = 16$. This loss is asymmetric for anchors and positives, so we evaluate with the embeddings in both orders, such that our final training objective is:

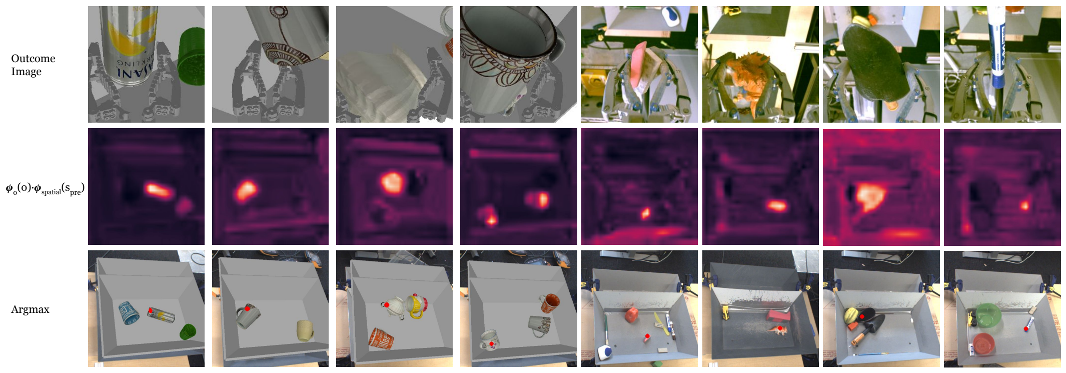
$$\mathcal{L}_{\text{Grasp2Vec}} = \text{NPairs}((\phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}})), \phi_o(\mathbf{o})) + \text{NPairs}(\phi_o(\mathbf{o}), (\phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}}))).$$

4 Self-Supervised Goal-Conditioned Grasping

The Grasp2Vec representation can enable effective goal-conditioned grasping, where a robot must grasp an object matching a user-provided query. In this setup, the same grasping system can both collect data for training the representation and utilize this representation for fulfilling specified goals. The grasping task is formulated as a Markov decision process (MDP), similar to the indiscriminate grasping system proposed by Kalashnikov et al. [24]. The actions $\mathbf{a}$ correspond to Cartesian gripper motion and gripper opening and closing commands, and the state $\mathbf{s}$ includes the current image and a representation of the goal $\mathbf{g}$. We aim to learn the function $Q_\pi(\mathbf{s}, \mathbf{a}, \mathbf{g})$ under the following reward function: grasping the object specified by $\mathbf{g}$ yields a terminal reward of $\mathbf{r} = 1$, and $\mathbf{r} = 0$ for all other time steps. The architecture of Q_π is shown in Figure 3. Learning this Q-function presents two challenges that are unique to self-supervised goal-conditioned grasping: we must find a way to train the policy when, in the early stages of learning, it is extremely unlikely to pick up the right object, and we must also extract the reward from the episodes without ground truth object labels.



(a) Nearest neighbors of goal and scene images.



(b) Localization results. The heatmap is defined as $\phi_o(\mathbf{o})\top\phi_{s,\text{spatial}}(s_{\text{pre}})$, resulting in a $H \times W \times 1$ array. Note that the embeddings were never trained on this task, nor was any detection supervision provided. The fourth column shows a failure case, where two different mugs have similar features and the argmax is on the wrong mug. The right half shows results on real images. The representations trained on real grasping results are able to localize objects with high success.

Figure 4: An analysis of our learned embeddings. Examples shown were chosen at random from the dataset.

Algorithm 1 Goal-conditioned policy learning

```
1:  Initialize goal set $G$ with 10 present images
2:  Initialize $Q_\pi$ and replay buffer $B$
3:  **while** $\pi$ not converged **do**
4:      $g, g' \leftarrow$ sample($G$)
5:      $(\mathbf{s}, \mathbf{a}, \mathbf{r}_{\text{any}}, \mathbf{o}) \leftarrow$ ExecuteGrasp($\pi, \mathbf{g}$)
6:      // Posthoc Labeling (PL)
7:      **if** $\mathbf{r}_{\text{any}} = 1$ **then**
8:          $B \leftarrow B \bigcup (\mathbf{s}, \mathbf{a}, [\mathbf{o}, 1])$ (outcome is successful goal)
9:      **else**
10:         $B \leftarrow B \bigcup (\mathbf{s}, \mathbf{a}, [\mathbf{g}, 0])$ (desired goal is a failure)
11:     **end if**
12:     // Embedding Similarity (ES)
13:     **if** $\mathbf{r}_{\text{any}} = 1$ **then**
14:         $B \leftarrow B \bigcup (\mathbf{s}, \mathbf{a}, [\mathbf{g}, \hat{\phi}_o(\mathbf{o}) \cdot \hat{\phi}_o(\mathbf{g})])$
15:         // Auxiliary Goal (AUX)
16:         $B \leftarrow B \bigcup (\mathbf{s}, \mathbf{a}, [\mathbf{g}', \hat{\phi}_o(\mathbf{o}) \cdot \hat{\phi}_o(\mathbf{g}')])$
17:     **end if**
18:     $(\mathbf{s}, \mathbf{a}, [\mathbf{g}, \mathbf{r}], \mathbf{s}') \leftarrow$ sample($B$)
19:     $\pi \leftarrow \pi - \alpha \nabla_\pi (Q_\pi(\mathbf{s}, \mathbf{a}, \mathbf{g}) - (\mathbf{r} + \gamma V_\pi(\mathbf{s}', \mathbf{g})))^2$
20: **end while**
```

We assume that the grasping system can determine automatically whether it successfully grasped an object, but not which object was grasped. For example, the robot could check whether the gripper is fully closed to determine if it is holding something. We will use $\mathbf{r}_{\text{any}}$ to denote the indiscriminate reward function, which is 1 at the last time step if an object was grasped, and 0 otherwise. Q-learning can learn from any valid tuple of the form $(\mathbf{s}, \mathbf{a}, \mathbf{r}, \mathbf{g})$, so we use the $\mathbf{r}_{\text{any}}$ to generate these tuples without object labels. We utilize three distinct techniques to automatically augment the training data for Q-learning, making it practical to learn goal-conditioned grasping:

Embedding Similarity (ES) A general goal labeling system would label rewards based on a notion of similarity between what was commanded, $\mathbf{g}$, and what was achieved, $\mathbf{o}$, approximating the true on-policy reward function. If the Grasp2Vec representations capture this similarity between objects, setting $\mathbf{r} = \hat{\phi}_o(\mathbf{g}) \cdot \hat{\phi}_o(\mathbf{o})$ would enable policy learning for instance grasping.

Posthoc Labeling (PL) Embedding similarity will give close to correct rewards to the Q function, but if the policy never grasps the right object there will be no signal to learn from. We use a data augmentation approach similar to the hindsight experience replay technique proposed by Andrychowicz et al. [20]. If an episode grasps any object, we can treat $\mathbf{o}$ as a correct goal for that episode's states and actions, and add the transition $(\mathbf{s}, \mathbf{a}, \mathbf{o}, \mathbf{r} = 1)$ to the buffer. We refer to this as the posthoc label.

Auxiliary Goal Augmentation (AUX) We can augment the replay buffer even further by relabling transitions with unacheived goals. Instead of sampling a single goal, we sample a pair of goals $(\mathbf{g}, \mathbf{g}')$ from the goal set G without replacement If $\mathbf{r}_{\text{any}} == 1$ after executing the policy conditioned on $\mathbf{g}$, we add the transition $(\mathbf{s}, \mathbf{a}, \mathbf{g}', \mathbf{r} = \hat{\phi}_o(\mathbf{g}') \cdot \hat{\phi}_o(\mathbf{o}))$ to the replay buffer. In baselines that do not use embeddings, the reward is replaced with 0 under the assumption that $\mathbf{g}'$ is unlikely to be the grasped object.

Pseudocode for the goal-reward relabeling schemes, along with the self-supervised instance grasping routine, are summarized in Algorithm 1.

5 Experiments

In our experiments, we evaluate our representation both in terms of its ability to localize and detect object instances, and in terms of its ability to enable goal-conditioned grasping by supplying an effective reward function and goal representation. Illustrations and videos are at `https://sites.google.com/site/grasp2vec/`

5.1 Experimental setup and data collection. Real-world data collection for evaluating the representation on real images was conducted using KUKA LBR iiwa robots with 2-finger grippers, grasping objects with various visual attributes and varying sizes from a bin. Monocular RGB observations are provided by an over-the-shoulder camera. Actions $\mathbf{a}$ are parameterized by a Cartesian displacement vector, vertical wrist rotation, and binary commands to open and close the gripper. The simulated environment is a copy of the real environment in Bullet [25]. Figure 2 depicts the simulated and real-world setup, along with the observations available to the robot.

We train and evaluate grasp2vec embeddings on both the real and simulated environments across 3 tasks: object recall from scene differences, object localization within a scene, and use as a goal and/or reward function for instance grasping.

5.2 Grasp2Vec embedding analysis. We train the goal and scene embeddings on successful grasps. We train on 15k successful grasps for the simulated results and 437k for the real world results. The objective pushes embeddings of outcome images to be close to embedding difference of their respective scene images, and far from each other. By representing scenes as the sum of their objects, we expect the scene embedding space to be structured by object presence and not by object location. This is validated by the nearest neighbors of scenes images shown in Figure 4a. The nearest neighbors of scene embeddings correspond to scenes containing the same objects, and the nearest neighbors of outcome images lie nearest to other images of the same objects (see Figure 11b).

	sim seen	sim novel	real seen	real novel
Retrieval (ours)	88%	64%	89%	88%
Outcome Neighbor (ImageNet)	—	—	23%	22%
Localization (ours)	96%	77%	83%	81%
Localization (ImageNet)	—	—	18%	15%

Table 1: Quantitative study of Grasp2Vec embeddings. Object retrieval for a grasp is calculated by finding the outcome image whose embedding is closest to $\phi_s(s_{pre}) - \phi_s(s_{post})$ in the dataset. The object retrieval is counted as correct if the nearest neighbor image contains the same object as the one grasped from s_{pre}. As we cannot expect weights trained on ImageNet to exhibit this property, we evaluate the nearest neighbors between outcome images. The accuracy measures how often top nearest neighbor for an outcome image contains the same object. Object localization is calculated by multiplying $\phi_o(\mathbf{o})$ with each feature vector in $\phi_{s,spatial}(s_{pre})$ to obtain a heatmap. An object is localized if the maximum point of the heatmap lies on the outcome object in the scene. See Figure 4b for examples heatmaps and Appendix A for example retrievals.

We can evaluate how well the scene and outcome feature spaces are in correspondence by verifying that the the difference in scene features $\phi_s(s_{\text{pre}}) - \phi_s(s_{\text{post}})$ is near the embedding of the object that was grasped $\phi_o(\mathbf{o})$. As many outcome images of the same object will have similar features, we measure the retrieval by checking whether the nearest neighbor outcome image contains the same object as the groundtruth outcome. Appendix A shows example successes and failures. As indicated in Table 1, retrieval accuracy is high for the training simulated objects, but low otherwise. Retrieving never-before-seen objects is difficult as it may required knowing what a unseen surface of the object looks like (e.g. differentiating mugs by the writing on the bottom if they are upside-down).

The grasp2vec architecture and objective enables our method to localize objects without any spatial supervision. By embedding scenes and single objects (outcomes) into the same space, we can use the outcome embeddings to localize that object within a scene. As shown in Figure 4b, we compute the dot produce of $\phi_o(o)$ with each pixel of $\phi_{s,spatial}(s_{\text{pre}})$ to obtain a heatmap over the image corresponding to the affinity between the query object and each pixel's receptive field. A localization is considered correct only if the maximum point of the heatmap lies on the object in the scene. As shown in Table 1, grasp2vec embeddings perform localization at almost 80% accuracy on objects that were never seen during training, and without receiving any position labels. The simulated objects seen during training are localized at even higher accuracy. We expect that such a method could be used to provide goals for pick and place or pushing task where a particular object position is desired. For this localization evaluation, we compare grasp2vec embeddings against the same ResNet50-based architecture used in the embeddings, but trained on ImageNet [26]. This network is only able to localize the objects at 15% accuracy, because the features of an object in the gripper are not necessary similar to the features of that same object in the bin.

5.3 Instance grasping experiments. We perform ablation studies in simulation and analyze how choices in model architecture and goal-reward relabeling affect instance grasp performance and generalization to unseen test objects. In each episode of the data collection and learning loop, an indiscriminate grasping policy collects 10 objects from the bin and saves their images to a goal set G. Afterwards, the data collection protocol switches to on-policy instance grasping, using previously grasped images $\mathbf{o}^{(1)}...\mathbf{o}^{(10)}$ as subsequent goals. Exploration policy hyperparameters are as described in [24] and we parallelize data collection across 1000 simulated robots for each experiment.

Overall instance grasp success is reported in Table 2. Our best model, described in Expt. 5, achieves 80% instance success rate on seen objects and 59% success on unseen objects. Furthermore, our experiments yield the following conclusions:

Posthoc Labeling enables a 30 percent point increase of instance grasping success when no object similarity metrics are available as shown in Experiments $1 and 3$.

Aux is beneficial when complemented by Embedding Similarity, perhaps increasing generalization by providing more diverse feedback to the policy. However, when used without ES, the auxialiary goal's reward may be incorrect, which decreases performance in Experiment 4.

Grasp2Vec Embedding Similarity can provide both reward labels and goal representations when learning to perform instance grasping, without requiring ground-truth object labels. As the grasp2vec embeddings perform closely to the Oracle labels, we are confident that the embeddings capture the object identities to provide a good reward label. In comparison, a ϕ_o trained via an autoencoding loss on outcome images (Experiment 5) fails to capture similarity between goal and outcome embeddings, resulting in poor performance.

Goal encoding is not necessary for good performance, as shown by Experimetn 11. Both learning Q values form a raw image and from the grasp2vec embeddings work for instance grasping. The raw image version uses the convolutional architecture proposed in [19], and learn this function's parameters jointly with the Q function.

Composite goals. The additive compositionality of Grasp2Vec embeddings enables users to freely manipulate embeddings interactively to enable rich behavior at test time, without the policy ever having been explicitly trained to handle such goals. We show in Table 3 that policies conditioned on ϕ_o can grasp one of two simultaneously commanded goal embeddings that are averaged together. This generalizes due to the additive semantics of grasp2vec embeddings.

#	Goal Conditioning	Reward Labels	Train Objects	Test Objects
1	Indiscriminate Grasping	N/A	18.3	21.9
2	Raw Image CNN	Oracle Labels	83.9	43.7
3	Raw Image CNN	PL	50.4	41.1
4	Raw Image CNN	PL + Aux(0)	22.1	19.0
5	Raw Image CNN	PL + ES (autoencoder)	18.7	20.9
9	Raw Image CNN	PL + ES	**80.1**	53.9
10	Raw Image CNN	PL + Aux + ES	78.0	**58.9**
11	$\phi_o(g)$	PL + ES	78.4	45.4

Table 2: Evaluation and ablation studies on a simulated instance grasping task, averaged over 700 trials. In simulation, the scene graph is accessed to evaluate ground-truth performance, but it is withheld from our learning algorithms. Performance is reported as percentage of grasps that picked up the user-specified object. Table reports early stopping scores for instance grasping on training objects and evaluation for the same checkpoint on test objects. Best numbers (for unsupervised approaches) in bold font.

Goal Network	Reward Label	Train Objects	Test Objects
$\frac{\phi_o(g1)+\phi_o(g2)}{2}$	PL + ES(Grasp2Vec)	51.9%	42.9%

*Table 3: **Instance grasp success rate on composed goals:** We average two grasp2vec vectors at test time and evaluate whether the policy can grasp either of the requested objects.*

6 Discussion

We presented grasp2vec, a representation learning approach that learns to represent scenes as sets of objects, admitting basic manipulations such as removing and adding objects as arithmetic operations in the learned embedding space. Our method is supervised entirely with data that can be collected autonomously by a robot, and we demonstrate that the learned representation can be used to localize objects, recognize instances, and also supervise a goal-conditioned grasping method that can learn via goal relabeling to pick up user-commanded objects. Importantly, the same grasping system that collects data for our representation learning approach can also utilize it to become better at fulfilling grasping goals, resulting in an autonomous, self-improving visual representation learning method. Our work suggests a number of promising directions for future research: incorporating semantic information into the representation (e.g., object class), leveraging the learned representations for spatial, object-centric relational reasoning tasks (e.g., [27]), and further exploring the compositionality in the representation to enable planning compound skills in the embedding space.

References

[1] S. Ross, N. Melik-Barkhudarov, K. S. Shankar, A. Wendel, D. Dey, J. A. Bagnell, and M. Hebert. Learning monocular reactive uav control in cluttered natural environments. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 1765–1772. IEEE, 2013.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

[3] A. Ghadirzadeh, A. Maki, D. Kragic, and M. Björkman. Deep predictive policy training using reinforcement learning. In *Intelligent Robots and Systems (IROS), 2017 IEEE/RSJ International Conference on*, pages 2351–2358. IEEE, 2017.

[4] S. Levine, P. Pastor, A. Krizhevsky, and D. Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. In *International Symposium on Experimental Robotics (ISER 2016)*, 2016.

[5] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[6] P. Agrawal, A. V. Nair, P. Abbeel, J. Malik, and S. Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, pages 5074–5082, 2016.

[7] F. Ebert, C. Finn, A. X. Lee, and S. Levine. Self-supervised visual planning with temporal skip connections. *arXiv preprint arXiv:1710.05268*, 2017.

[8] L. Pinto, D. Gandhi, Y. Han, Y.-L. Park, and A. Gupta. The curious robot: Learning visual representations via physical interactions. In *European Conference on Computer Vision*, pages 3–18. Springer, 2016.

[9] D. Jayaraman and K. Grauman. Learning image representations tied to ego-motion. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1413–1421, 2015.

[10] R. Jonschkowski and O. Brock. Learning state representations with robotic priors. *Autonomous Robots*, 39(3):407–428, 2015.

[11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[12] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[13] X. Wang, A. Farhadi, and A. Gupta. Actions˜ transformations. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 2658–2667, 2016.

[14] L. Pinto and A. Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *Robotics and Automation (ICRA), 2016 IEEE International Conference on*, pages 3406–3413. IEEE, 2016.

[15] I. Lenz, H. Lee, and A. Saxena. Deep learning for detecting robotic grasps. *The International Journal of Robotics Research*, 34(4-5):705–724, 2015.

[16] 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 *Robotics and Automation (ICRA), 2017 IEEE International Conference on*, pages 1386–1383. IEEE, 2017.

[17] A. ten Pas, M. Gualtieri, K. Saenko, and R. Platt. Grasp pose detection in point clouds. *The International Journal of Robotics Research*, page 0278364917735594, 2017.

[18] E. Jang, S. Vijaynarasimhan, P. Pastor, J. Ibarz, and S. Levine. End-to-end learning of semantic grasping. *arXiv preprint arXiv:1707.01932*, 2017.

[19] K. Fang, Y. Bai, S. Hinterstoisser, and M. Kalakrishnan. Multi-task domain adaptation for deep learning of instance grasping from simulation. *arXiv preprint arXiv:1710.06422*, 2017.

[20] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. Hindsight experience replay. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5048–5058. Curran Associates, Inc., 2017.

[21] S. Cabi, S. G. Colmenarejo, M. W. Hoffman, M. Denil, Z. Wang, and N. De Freitas. The intentional unintentional agent: Learning to solve many continuous control tasks simultaneously. *arXiv preprint arXiv:1707.03300*, 2017.

[22] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[23] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 1857–1865. Curran Associates, Inc., 2016.

[24] 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. *arXiv preprint arXiv:1806.10293*, 2018.

[25] E. Coumans and Y. Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. `http://pybullet.org`, 2016–2018.

[26] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015.

[27] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. Clevr: A diagnostic dataset for compositional language and elementary visual reasoning. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pages 1988–1997. IEEE, 2017.

A Qualitative Detection Results

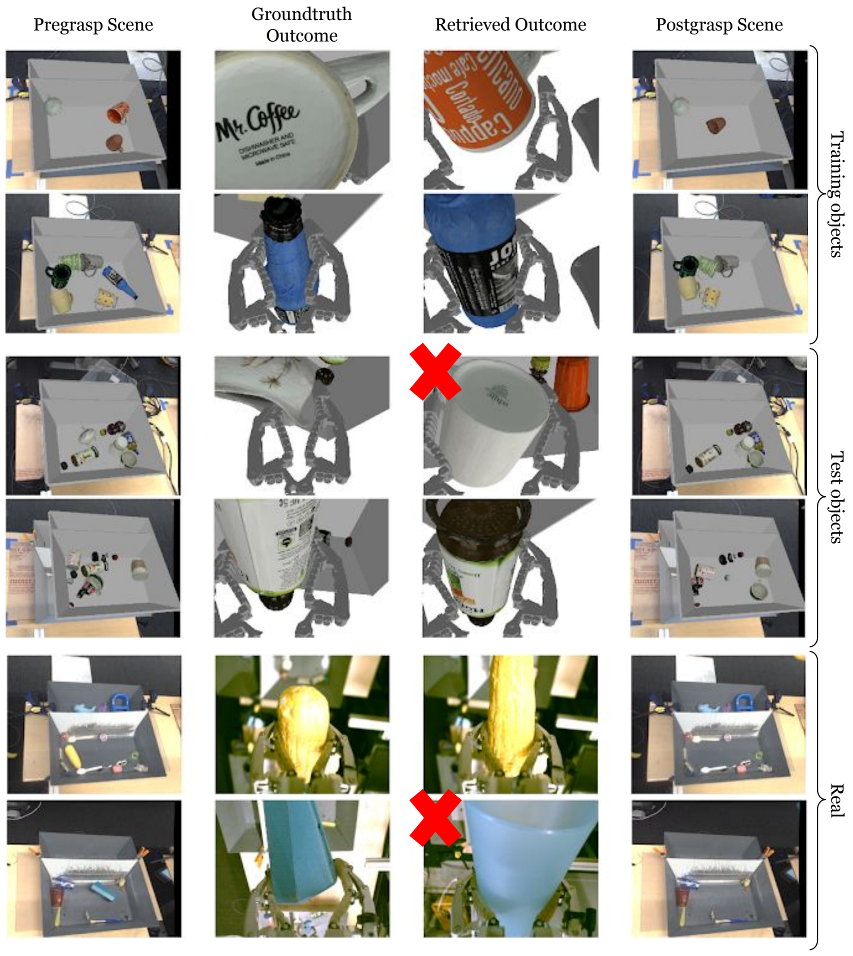


Figure 5: This table illustrates the object recall property. From left to right: The scene before grasping, the grasped object, the outcome image retrieved by subtracting the postgrasp scene embedding from the pregrasp scene embedding, and lastly the postgrasp scene. We show example successes and failures (the later are marked with a red X). Failures occur in the test object set because multiple white objects had similar embeddings. Failures occur in the real data because the diversity of objects is very high, which likely makes the embddings less partitioned between object instances.

Success Cases Figures 6, 7, 8 depict examples from the real-world localization task in which grasp2vec demonstrates surprisingly effective localization capabilities.

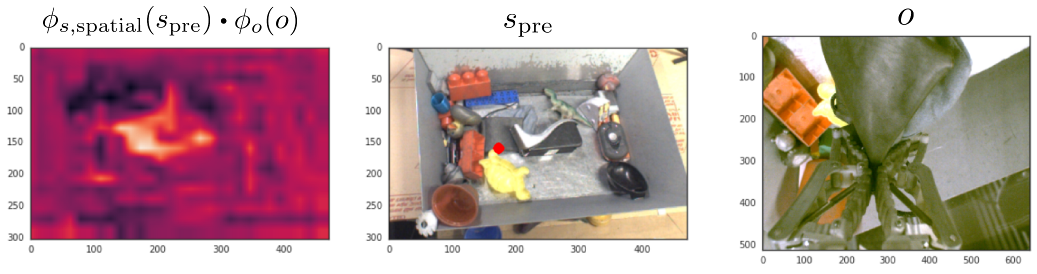


Figure 6: Training objects. Recognize deformable object (bag) in a large amount of clutter.

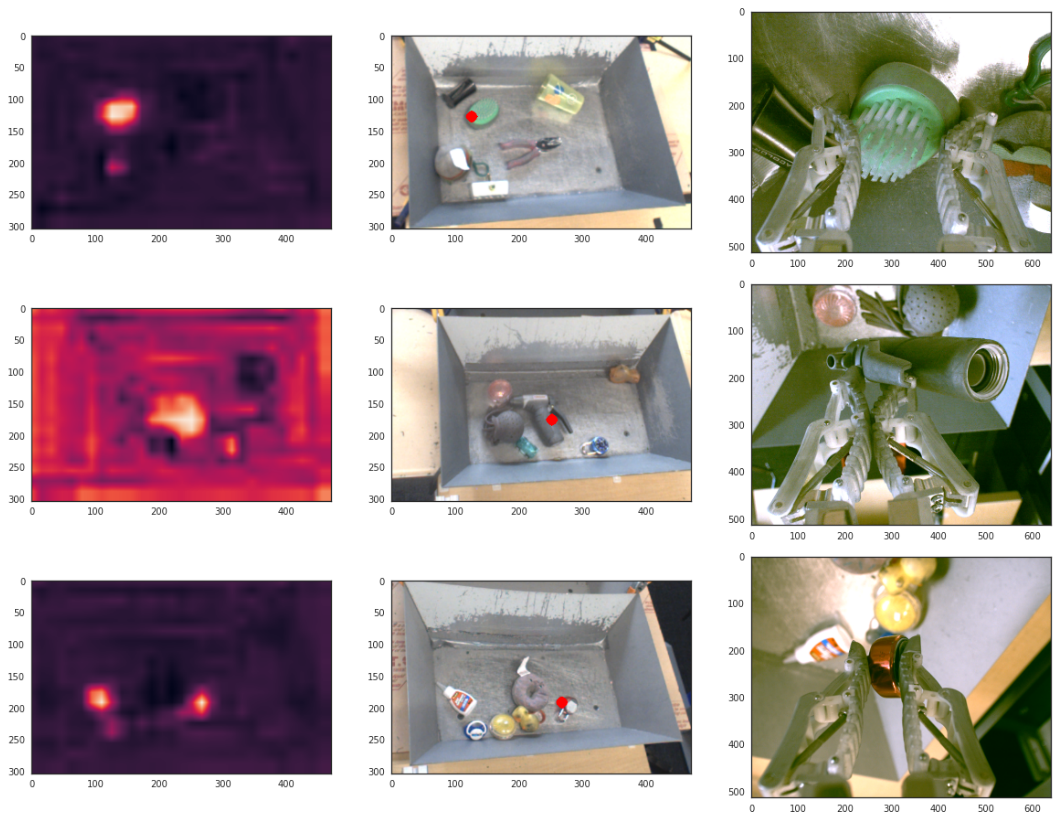


Figure 7: Testing objects. Recognizes correct object, even when goal is presented from a different pose than the object's pose in the bin.

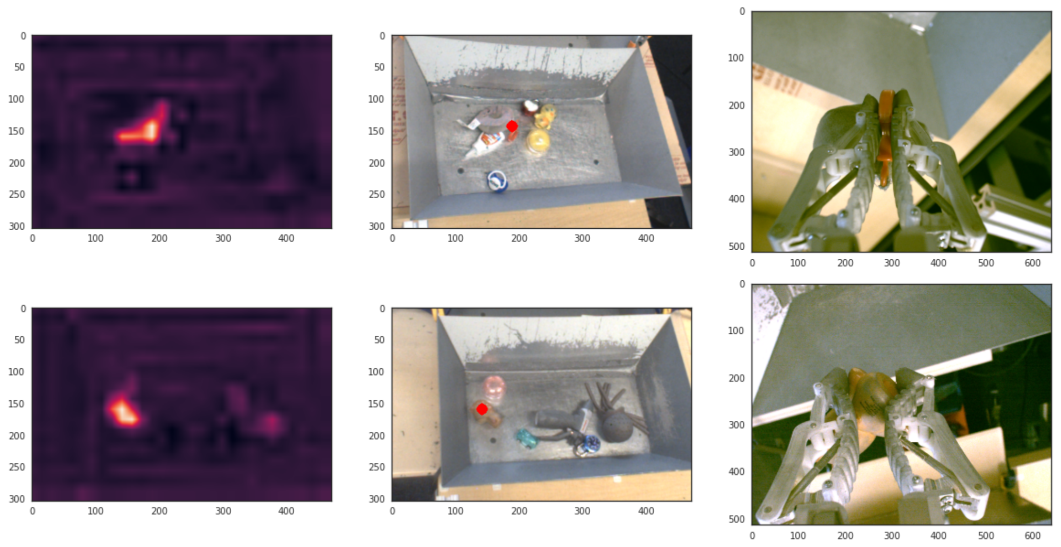


Figure 8: Testing objects. Recognizes objects by parts when the goal object is occluded by the gripper.

Failure Cases Figures 9, 10 depict examples where Grasp2Vec embeddings make mistakes in localization.

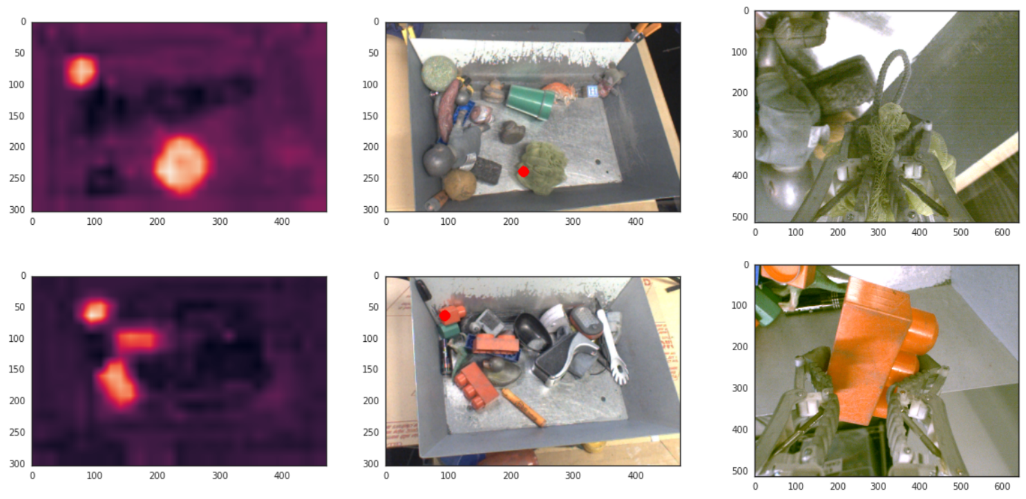


Figure 9: Training objects. Embedding localizes objects with the correct colors but wrong (though similar) shape.

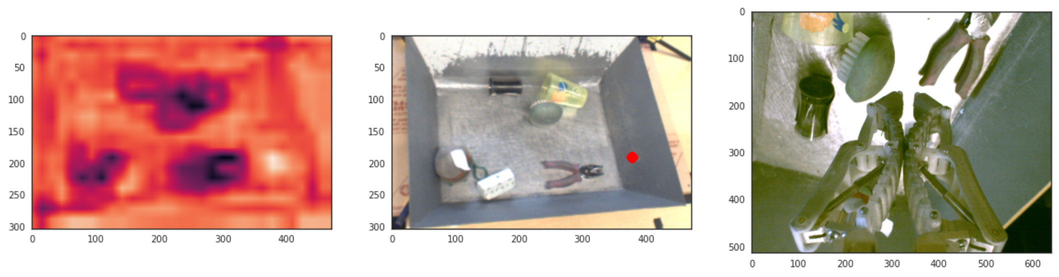
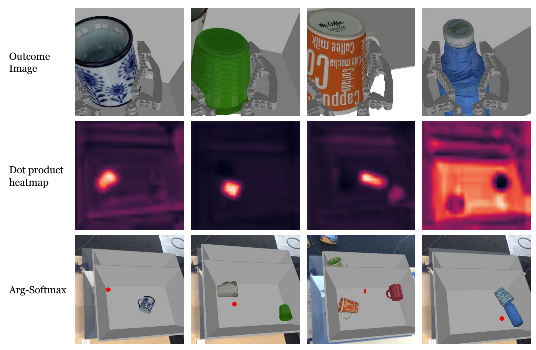
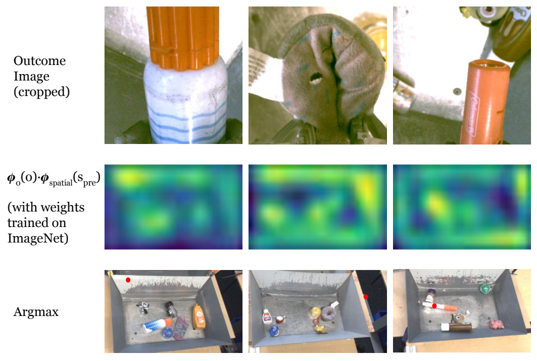


Figure 10: Test objects. Failed grasp incorrectly labeled as successful, resulting in an empty (meaningless) goal for localization.



(a) Localization using untrained weights in simulation.


Outcome Image (cropped)
$\varphi_o(o)\cdot\varphi_{spatial}(s_{pre})$
(with weights trained on ImageNet)
Argmax

(b) Localization using weights trained on imagenet.

Figure 11: (a)The detection analysis with an untrained model. This verifies that our loss, rather than the architecture on it's own, enables the detection property. (b) The failure of localization indicates that Imagenet features are not consistent between scene and outcome images, probably because of resolution and pose differences.

B Simulation Experiment Details

The simulated experiments use the Bullet [25] simulator with a model of a 7-DoF Kuka arm. We use 44 unique objects for training and 15 unique objects for evaluation. The training and evaluation objects are mutually exclusive, and each object has a unique mesh and texture. All objects are scans of mugs, bottles, cups, and bowls.

For data collection and evaluation, a particular scene has up to 6 objects sampled from the total objects without replacements; this means that no scene has multiple copies of a particular object. The objects are dropped into the scene at a random pose, and the objects may bounce onto each other.

C Real-World Experiment Details

We use roughly 500 objects for training and 42 unseen objects for evaluation. Objects are not restricted to object categories. For data collection and evaluation, 6 objects are placed randomly into the bin. After grasping an object, the robot drops it back into the bin to continue. The objects in a bin are switched out about twice a day and 6-7 robots are used in parallel, each with its own bin.