A Task-Oriented Approach for Cost-sensitive Recognition

Roozbeh Mottaghi[1] Hannaneh Hajishirzi[2] Ali Farhadi[1,2]
[1]Allen Institute for Artificial Intelligence
[2]University of Washington

Abstract

With the recent progress in visual recognition, we have already started to see a surge of vision related real-world applications. These applications, unlike general scene understanding, are task oriented and require specific information from visual data. Considering the current growth in new sensory devices, feature designs, feature learning methods, and algorithms, the search in the space of features and models becomes combinatorial. In this paper, we propose a novel cost-sensitive task-oriented recognition method that is based on a combination of linguistic semantics and visual cues. Our task-oriented framework is able to generalize to unseen tasks for which there is no training data and outperforms state-of-the-art cost-based recognition baselines on our new task-based dataset.

1. Introduction

In recent years we have witnessed a dramatic stride in the performance of classification and detection methods. Visual recognition algorithms become more reliable and have started getting considerable traction from real world applications. Most of these applications are centered around specific tasks (e.g., autonomous navigation, autonomous vacuum cleaning, automated lawn mowing, etc.) rather than being as general as scene understanding. Considering the state of visual recognition methods, we believe, it is the right time to rethink task-oriented recognition.

Task-oriented recognition involves addressing a wide range of problems. A very first issue that needs to be addressed is to find a set of cheap but effective features for the given task. Suppose we want to design a system or a robot to efficiently perform certain *tasks* such as "Put the cup on the table" or "Walk towards the desk". The first question to answer is what features should we use? Should we use surface normal estimates? How about RGB-D data? Do we need high frequency texture information? What about object recognition? Are support surfaces useful? For a specific task, among all possible information that one could extract from visual data, only a small subset would be useful (the majority of the information extracted might be irrelevant, and some portion might be an overkill). For example, for the task of "Walk on the floor", surface normal and height features might be relevant while texture of the carpet and fine-grained object categories might not.

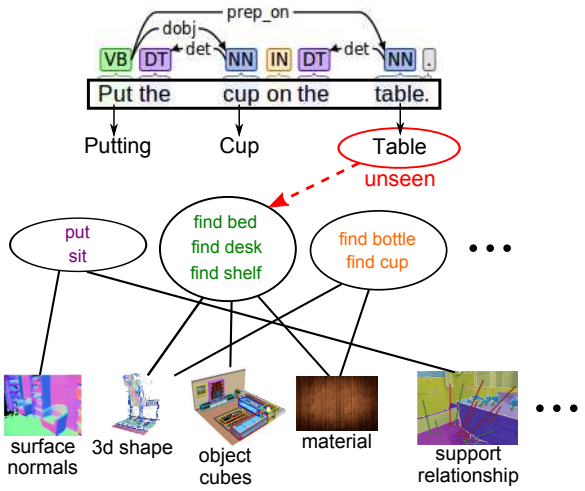


Figure 1. We create a syntactic parse of a given task to obtain part-of-speech tags (noun, verb, etc). Then, we find a mapping from each extracted noun (NN) and verb (VB) to the vocabulary of known Shared Unit Tasks (SHUTs) for which we have training data. If the noun or verb does not exist in our vocabulary (e.g., *table* in this example), we assign it to a cluster of SHUTs in our vocabulary and use the feature selection strategy of that cluster for the unseen part of the task. The bottom row shows examples of features used in our framework.

Often combination of features have shown to be effective, but picking the right combination is an exponential search. One common approach is to consider all possible information we could possibly extract and hope that our model can benefit from the right subset of it. Such an approach has the following implicit assumptions that might not hold in several application domains: (1) there exist enough training data for such a high dimensional representation; (2) there exists a high capacity learning model that can tolerate the inherent noise in the high dimensional representation; and (3) computational resources are free. A natural solution to this problem is to learn to select a subset

of features in a discriminative fashion.

Various methods have been proposed [1, 9, 24] to find a suitable set of features, however, it is not practical to find the right set of features for every task since the space of possible tasks is huge (consider the number of combinations one can make with English nouns and verbs). Therefore, it is infeasible to create a comprehensive list of possible tasks and collect training data for them. In this paper, we use a combination of visual and linguistic cues to better handle this huge space of possible tasks.

Our idea is to decompose *tasks* into a set of *Shared Unit Tasks (SHUTs)* using simple syntactic cues (extracting relevant part-of-speech tags). For example, *putting* is a SHUT that is shared between "Put the cup on the table" and "Put the bottle in the sink". So we can perform training only once for *putting* for all tasks that involve this meaning of *putting*. Although there are fewer SHUTs compared to tasks, it is still impractical to gather training data for all possible SHUTs. Hence, a practical solution should have a mechanism to handle unseen SHUTs. In this paper, we also introduce a novel cost-sensitive method to select features for *unseen* SHUTs (and consequently unseen tasks). To be able to handle both known and unseen SHUTs, we propose a discriminative co-clustering method that groups similar SHUTs based on similarities in visual features and linguistic semantics. Such a clustering allows us to assign the unseen SHUTs to a group of SHUTs and use the learned feature selection strategy of that group for the unseen SHUT. Figure 1 shows the overview of the approach.

Our experiments show that our proposed method outperforms state-of-the-art cost-based feature selection methods for known SHUTs. We additionally show that the proposed method is effective in performing feature selection for unseen SHUTs. Moreover, we show that our method obtains reasonable results for tasks when none of their constituent SHUTs is known. To train and evaluate our method we create a new dataset by augmenting NYUv2 dataset [35] with task-based annotations.

2. Related Work

Task-based computer vision. Two schools of computer vision are compared by Ikeuchi and Hebert [19]: (1) general purpose oriented, where the idea is to have a single architecture to solve all computer vision problems; (2) task oriented, which argues the architecture should change so that an optimal set of components are chosen for each different task. In this paper, we advocate the second approach and argue that instead of running a pre-defined set of components or features in a fixed order, we should learn which features or components are useful for the given task to better utilize computational resources.

Neuroscience. Task-based visual processing has been investigated by the neuroscientists as well. Wurtz et al. [39] show that the brain performs a selective reduction of the visual stimuli, which is modulated by the task and attention. Also, on the modeling side, Geisler and Kersten [12] propose a perception model that considers a task-based utility function to compute the probability of each possible state of the environment given the image formed on the retina. Also, researchers in the field of visual attention believe there is a task-dependent component that directs visual processing [41, 32, 2].

Cost-sensitive feature selection. Similar to our approach, some methods (e.g., [11, 40, 4, 14]) take feature cost into account for selecting features. Karayev et al. [22] consider the problem of 'anytime' recognition, where a reinforcement learning framework is employed to optimize feature computation cost at test time while preserving a high performance. Xu et al. [40] proposes a stage-wise regression that minimizes cpu-time during testing. None of these methods are designed for or can handle unseen categories. Moreover, our experiments show improved results compared to these state-of-the-art methods [22, 40] in selecting features for known categories.

Zero-shot learning. Conventional solutions to zero-shot learning use visual attributes [8, 26]. Direct extension of these method to our problem is not possible because we do not have visual attributes of the task; we are only given a textual name for the unseen tasks. Recent methods use a joint embedding between visual and textual cues for zero shot learning [36, 10, 33]. These methods are also not applicable to our problem because different *tasks* share the same images (many tasks can be performed in the same scene), or a single region may have multiple semantic labels (e.g., a surface can be suitable for both *sitting* and *putting* depending on the intention of the agent). Similar to our method, Elhoseiny et al. [7] learn classifiers for unseen categories purely based on textual descriptions for those categories. In contrast, we perform zero-shot learning jointly with feature selection. Moreover, the unseen SHUTs in our paper are only associated with their names which are single words, not long textual descriptions. [20] have studied a joint training approach for a group of attributes that have semantic ties. In contrast, we study the problem of task-based recognition.

Affordance. Object or scene affordances [15, 13, 21] can be useful in predicting suitable regions for some SHUTs. However, there are some SHUTs, such as *finding cup*, that cannot be effectively predicted from affordances. Most related to ours is zero-shot learning for affordances [44]. This work, however, does not consider cost or selection of features, which is the main focus of our paper.

NLP for robotics. There is a vast amount of literature on using Natural Language Processing techniques for robotics applications (e.g., [6, 31, 38, 16]). The most related work to ours is a method for grounding natural language to mobile

manipulation instructions [31]. This work relies on linguistic similarity to decompose a task description into sub-tasks. However, our goal in this paper is quite different – we study the relevance of features to different tasks, and we aim to find suitable features for unseen tasks.

3. Overview of Problem and Approach

Problem. In this paper, we address the problem of identifying the most discriminative, but least expensive set of features for task-oriented recognition. More formally, given a set of features $F = \{f_1, \ldots, f_M\}$, the cost c_i for each feature f_i, and parameter λ that specifies the trade-off between the total cost of the features and training loss, our goal is to find a subset of features $\tilde{F}$ that are most discriminative for a task t. Our assumption is that each task t can be decomposed into a disjoint set of *Shared Unit Tasks (SHUTs)* in our vocabulary of SHUTs $\mathcal{S} = \{s_1, s_2, \ldots, s_L\}$. For example, the task "Put the cup on the table" can be decomposed into three SHUTs of *putting*, *finding cup*, and *finding table*.

SHUTs can be *known* or *unseen* in the training data. Each known SHUT is annotated in the training data according to the regions that are suitable for that SHUT. For example, annotated regions for *putting* are flat surfaces that can support objects. The features capture different representations and levels of details for tasks, and are computed at different levels of run-time complexity. Some examples of features include 2D appearance features, 3D shape, material, and support relationship. We use the running time of computing features as a proxy of their cost. Our goals are (a) to decompose the task t into a set of disjoint SHUTs (possibly unseen), (b) find the most discriminative, but least expensive set of features for each known SHUT, and (c) map the unseen SHUT to a group of known SHUTs, which are grouped based on linguistic and visual characteristics among the SHUTs.

Overview of Approach. We decompose the tasks into SHUTs using a simple syntactic parse of the tasks (as shown in Figure 1). We then introduce a cost-sensitive method for finding subsets of features for known SHUTs and their corresponding weights (Section 4). Finally, we extend the cost-sensitive method to unseen SHUTs (Section 5).

4. Known SHUTs

We first assume that all the SHUTs are observed at training. Our goal is to find a subset of discriminative but least expensive features for each SHUT, and to estimate the weights of the features in that subset.

We need a framework that allows us to switch on/off the features, and can easily incorporate the feature cost. We form M groups of feature weights $\mathbf{w}_{G_1}, \ldots, \mathbf{w}_{G_M}$ that correspond to M feature types $f_1, \ldots, f_M$ in our model. For example, all of the weights for 2D appearance feature form one group. Setting a group of weights to zero means its corresponding feature will not be active.

We use a formulation similar to Group Lasso [42] penalized logistic regression to find the best set of informative features by predicting relevant image regions for each SHUT. For example, for *walking*, the formulation is to predict which regions are suitable for *walking*. The training data is available in the form of $\{\mathbf{x}_i, y_i\}_{i=1:n}$, where $\mathbf{x}_i$ represents the computed features for the i^{th} image region (concatenation of all features in the set F), y_i specifies if the region is suitable for a particular SHUT or not. Our goal is to find the best set of active features $\tilde{F}_s$ by optimizing:

$$\min_{\mathbf{w}} \sum_{i=1}^{n} \log(1+\exp(-y_i(\mathbf{w}^T\mathbf{x}_i+b))) + \lambda \sum_{m=1}^{M} \sqrt{c_m}\, \|\mathbf{w}_{G_m}\|_2 \,, \tag{1}$$

where the feature weights $\mathbf{w}$ are divided into M non-overlapping groups $\mathbf{w}_{G_1}, \ldots, \mathbf{w}_{G_M}$, c_m corresponds to the cost of the feature group f_{G_m}, and λ controls the balance between loss and total cost of selected features.

We solve the optimization problem in Eq. 1 using standard methods for Group Lasso and find the set of active features $\tilde{F}_s$ for each SHUT s. In particular, we set $\tilde{F}_s$ to be equal to the group of features whose weights $\mathbf{w}_G$ are non-zero. By varying λ different subsets of features are activated – setting λ to a very large value results in selecting no feature, while setting λ to zero makes all features active. We store the the weights for active features for SHUT s in $\mathbf{w}_s$.

5. Generalization to Unseen SHUTs

In many scenarios, we might encounter *unseen* SHUTs for which the training data is not available. For instance, *walking* is a known SHUT in our vocabulary, but *running* is unseen. However, *running* and *walking* should have more or less the same trend for selection of feature subsets and feature weights. More specifically, for both SHUTs, we expect to see similar subsets of active features with similar weights as we change λ. The question is how we can find the most similar SHUT or group of SHUTs to the unseen SHUT (a form of zero-shot learning).

At training, we group known SHUTs with similar visual and linguistic characteristics into clusters. At inference, this allows us to assign an unseen SHUT to a group of known SHUTs and borrow their corresponding feature subset and feature weights for the unseen SHUT. Therefore, we extend Eq. 1 to take the clustering into account:

$$\min_{\mathbf{w}, z_{.a}} \underbrace{\sum_{i=1}^{n} \log(1+\exp(-y_i^z(\mathbf{w}^T\mathbf{x}_i+b))) + \lambda \sum_{m=1}^{M} \sqrt{c_m}\, \|\mathbf{w}_{G_m}\|_2}_{\text{Selecting features}} \underbrace{- \sum_{a} \sum_{s_k, s_j \in a} z_{ka} z_{ja} \Phi(s_k, s_j, \mathbf{w})}_{\text{Clustering}} \quad s.t. \sum_{a} z_{ka} = 1, \;\; \sum_{k} z_{ka} \geq 1, \tag{2}$$

walking sitting putting grasping finding trash bin finding door

Figure 2. Annotations for examples of SHUTs.

input : (1) $\{\mathbf{x}_i\}$, features for training regions
(2) $\{y_i\}$ for each SHUT
(3) Linguistic similarity function ϕ^w for all pairs of SHUTs
(4) cost c for each feature
(5) # of clusters
(6) parameter λ

output: For each SHUT cluster a:
(1) Subset of selected features $\tilde{F}^a$.
(2) Feature weights $\mathbf{w}^a$.

Initialize ϕ^f (feature similarity) ;
for $t \leftarrow 1$ **to** *# of iterations* **do**
  Perform clustering on all SHUTs based on $\Phi = \phi^w + \alpha\phi^f$;
  foreach *cluster* a **do**
    Solve Eq. 2 feature selection for the input λ, where $y_i^z = 1$ if $y_i = 1$ for at least one of the SHUTs in the cluster a;
    Update $\tilde{F}^a$ and $\mathbf{w}^a$, which are the selected subset of features and their weights for cluster a, respectively;
    foreach *SHUT* $s \in a$ **do**
      $\tilde{F}_s = \tilde{F}^a$; $\mathbf{w}_s = \mathbf{w}^a$
    end
  end
  Update visual similarity ϕ^f according to $\mathbf{w}_s$ (Section 6).
end

Algorithm 1: Learning clusters of SHUTs and feature subsets for each cluster (Eq. 2).

where, z_{ka} is an indicator variable for cluster assignments ($z_{ka} = 1$ if SHUT s_k is assigned to cluster a, and $z_{ka} = 0$ otherwise), the region label y_i^z is the union of labels for all SHUTs in the cluster, and Φ is a similarity function for a pair of SHUTs. We use a similarity function based on visual and linguistic cues (described in Section 6). The training is performed on the SHUTs in our vocabulary $\mathcal{S}$, so we have annotations to perform the optimization.

We use block coordinate descend to solve the optimization problem: for a fixed λ, (1) we solve for $\mathbf{w}$ assuming that a clustering of the SHUTs is given. (2) We solve for clustering assignment $z_{.a}$ given the updated similarity function Φ (the visual similarity function depends on $\mathbf{w}$). Then, we iterate between these two steps. The details of learning clusters of SHUTs are described in Algorithm 1.

To solve step (1), we use Group Lasso as before. The only difference with Eq. 1 is that the feature weights are learned for the cluster of SHUTs instead of individual SHUTs (i.e., the region label y_i^z is the union of labels for all SHUTs in the cluster). Step (2) is a form of spectral clustering based on similarities of the SHUTs. As the output of the optimization, we obtain a set of SHUT clusters, the subset of features that is selected for each cluster, and feature weights for different subsets. Note that there is no guarantee that this procedure obtains the optimal solution.

At inference, our goal is to select the best subset of features for a new unseen SHUT s_{new}. We compute the linguistic similarity (described in Section 6) between the new SHUT s_{new} and the SHUTs s_i in our vocabulary. We then choose the cluster a^* whose average similarity (averaged over the SHUTs in the cluster) is highest for the unseen SHUT i.e., $a^* = \arg\max_a 1/size(a) \sum_{s_j \in a} \phi^w(s_{new}, s_j)$. To classify the regions for the new SHUT s_{new}, we use the selected subset of features for the chosen cluster a^* and borrow the weights for those selected features.

6. Similarity Functions for SHUTs

We measure the similarity function Φ in Eq. 2 by incorporating both linguistic ϕ^w and visual ϕ^f characteristics of SHUTs. The linguistic similarity encodes similarities between textual names of the SHUTs, while the visual similarity encodes similarities in the visual feature space and the activation of features for each SHUT.

We compute the linguistic similarity $\phi^w(s_k, s_j)$ between pairs of SHUTs using syntagmatic (association) and paradigmatic (similarity) relations between SHUT textual names. For example, *washing* and *dish* are *associated* as they tend to occur together, while *dish* and *plate* are *similar* as they tend to occur in similar contexts. To capture *similarity*, we use Word2Vec [30] (trained on Google News dataset), which computes continuous vector representation for words. The similarity of two words is defined as the cosine distance of their vectors. To capture the degree of *association* between two words, we use Pointwise Mutual Information (PMI), which is a measure of the strength of co-occurrence between two words [5]. We compute PMI on the Wumpus corpus [3]. We compute ϕ^w as a linear combination of these measures for pairs of SHUTs s_k and s_j in our vocabulary $\mathcal{S}$. We envision using dependency-based word similarity [18] can improve the linguistic similarity.

We compute the visual similarity based on the activation *ordering* of features for each SHUT as well as the *weights* of the features for each SHUT. Two SHUTs are visually similar if the order of activation of features for those two SHUTs is highly correlated. We compute the activation order of features by iteratively optimizing Eq. 1 (or first part

of Eq. 2) when the number of selected feature groups is increased at every iteration. For example, Table 2 shows the order of feature activation for two SHUTs *putting* and *grasping*. We compute the similarity between SHUTs by measuring the correlation between the order of activation of SHUTs. More specifically, we use Kendall rank correlation [23]. The order of activation for all SHUTs are shown in the supplementary material.

Additionally, two SHUTs are visually similar if their feature weights are similar. We compute the Euclidean distance between the feature weights of two SHUTs when all of the features can be switched on. We use a logistic function to normalize the distances between 0 and 1 and convert distances to similarity. Finally, the visual similarity $\phi^f(s_k, s_j)$ is computed based on the linear combination of the rank correlation term and the weight similarity term.

We compute Φ in Eq. 2 as a linear of combination of visual and linguistic similarities $\Phi(s_k, s_j) = \phi^w(s_k, s_j) + \alpha\phi^f(s_k, s_j)$. We use $\alpha = 4$ in our experiments. For the linguistic similarity ϕ^w, the weights are 0.2 and 0.8 for PMI and Word2Vec. For visual similarity ϕ^f, the weights are 0.3 and 0.7 for weight similarity and ordering, respectively.

7. SHUTs and Features

The vocabulary $\mathcal{S}$ includes 25 types of SHUTs: *walking*, *sitting*, *putting*, *grasping*, and *finding X*, where X corresponds to 21 most frequent object categories in NYUv2 dataset [35]. The annotation of SHUTs is performed by labeling *regions* that are suitable for a particular SHUT.

Regions. The features are defined on regions that span a volume in 3D, and correspond to a set of pixels in the 2D image. In this paper, we use RGB-D data and employ the region generation method of [35].

7.1. Annotating SHUTs

We augment NYUv2 dataset [35] with our task-based annotations by annotating regions relevant to SHUTs. Some example annotations are shown in Figure 2.

- **Walking.** Suitable regions include *floor*, *carpet*, *rug*, the flat part of *treadmill* and so on. We label all pixels that belong to any of these categories as *walking* regions.
- **Sitting.** Sitting can be performed on the flat regions of *sofas*, *chairs*, *beds*, etc. For annotating *sitting*, we automatically find regions in those categories whose surface normal points upward.
- **Putting.** Suitable regions for placing objects include flat surfaces e.g., *tables*, *counters*, and *shelves*. We label all the pixels that belong to these surfaces as *putting* regions.
- **Grasping.** Most of the objects can be grasped with some exceptions such as *floor*, *wall* or *stairs* (64 categories of the NYUv2 dataset [35] cannot be grasped). We label all objects that can be grasped as regions for *grasping*.
- **Finding X.** The goal is to look for objects of a certain category X. We use 21 most frequent categories in the NYUv2 dataset [35] (in terms of the numbers of regions).

7.2. Features

Our features capture different representations and levels of detail for objects' appearance or context with different levels of computational complexity. The features range from low-level features such as height, which can be obtained by simple processing of the output of an RGB-D sensor to higher level features such as support relationship, which is computed based on more complex reasoning.

The cost associated with each feature is the average time for computing that feature for an image. Running time is an important factor in various applications such as robotics or autonomous driving. However, it can be easily replaced with other notions of cost such as the usage of memory, the battery usage, etc. We provide a brief description of features here. For more implementation details and the feature costs, refer to the supplementary material. These are just representatives of commonly used features in the recognition frameworks. Our framework can be adapted to use any other features.

Height. Region height is useful for some SHUTs, e.g., it is unlikely to *sit* on surfaces with more than a certain height.

Surface Normals. The surface normal feature is useful for some SHUTs (e.g., *putting*). We compute the histogram of surface normals for the pixels of a region.

Material Attributes. Material of the regions is important for some SHUTs (e.g., *glass* is not used in a *sitting* surface). For computing this feature, we train an attribute classifier using the material annotations of [43]: *wood*, *painted*, *cotton*, *glass*, *glossy*, *plastic*, *shiny*, and *textured*.

2D Appearance. We capture the 2D appearance or texture of the regions using the descriptors of [34].

3D Shape. 3D shape features play an important role in most human tasks which are performed in 3D environments, and 2D images alone are ambiguous in that the entire 3D structure of the scene is projected onto a 2D image. We compute 3D shape features using four different 3D cues introduced in [37]: point density, scatter-ness, linear-ness, and 3D normals.

Distance to *any* Object. This feature is informative for SHUTs (e.g., *walking*) that only require the distance to the surrounding objects, but do not require the actual appearance of the surrounding objects. For example, the rough estimate of the distance to nearby objects, trees, and buildings is important for *walking*, but their detailed appearance can be ignored. We compute this feature by estimating distances between the region and a set of hypotheses cuboids that are generated by the method of [27]. Figure 3 illustrates the cubes and their distance to a region.

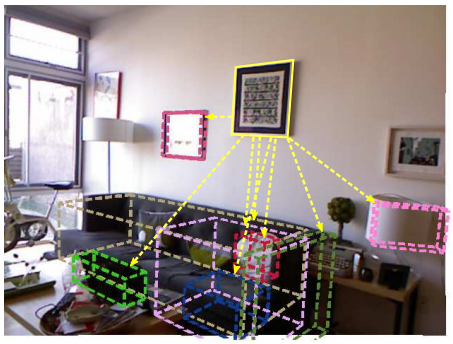


Figure 3. We compute the distance between all of the regions to the cubes generated by [27].

Distance to Instances of a Particular Category. In computing the previous feature, we ignored the category of the object, but for some SHUTs the category of the surrounding objects (contextual information) is very informative for a task. For instance, a surface next to a *cup* is most likely to be a suitable surface for *putting*. For this feature we again use the method of [27] to detect cuboids and use their object classification algorithm to detect their categories. Although this feature is more informative than the previous distance feature, it is more computationally expensive.

Support Relations. Support relationships are important for task-oriented reasoning (e.g., a *supporter* surface for an object is a good candidate for *putting*). Following [35], we compute this feature from the four types of support relationships between pairs of regions: supported from behind, from below, by a hidden object, or not supported.

Object Size. Object size is important for task-oriented recognition (e.g., a large object such as *bed* cannot be *grasped*). We approximate the object size by computing the volume of the object cuboids generated by [27] (used above) that has the largest overlap with the object region. The overlap is defined as the size of the intersection of the region and the cuboid in 3D divided by the region size. The feature for each region is the cuboid volume, the area of the largest surface and the area of the smallest surface of the cuboid.

8. Experiments

8.1. Experimental Setup

Dataset. We use NYUv2 [35] RGB-D dataset for our experiments. This dataset contains 1449 RGB-D images. We use the same split as NYUv2 for training and test that includes 795 training images and 654 test images, containing 76,837 and 60,872 regions, respectively. We provide additional annotations for the dataset in terms of 25 types of SHUTs using the procedure described in Section 7.1.

Evaluation and Implementation Details. We evaluate our task-oriented recognition framework in three parts: (1) known SHUTs for which we have training data (2) zero-shot learning and generalization to unseen SHUTs (3) handling known and unseen *tasks*.

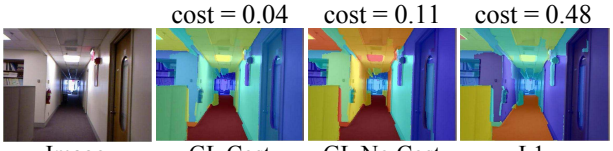


Figure 4. A qualitative example showing predictions of some of the methods for *walking*. These heat maps show the probability of the prediction. On top of each image, the fraction of total cost for making the prediction is shown.

For known SHUTs, our method called GL COST optimizes Eq. 1; for learning the weights, we use the SLEP implementation [28] for Group Lasso. For inference, we apply the learned weights $\mathbf{w}$ in Eq. 1 to the features computed for regions of a test image and predict the score of regions. For unseen SHUTs, our method called ZERO SHOT first optimizes Eq. 2 on known SHUTs to find the clusters and feature selections for clusters. For inference for an unseen SHUT, we use the linguistic similarity function ϕ^w to find the most similar cluster a to the unseen SHUT and borrow the features of a. For zero-shot learning, we set the number of clusters to 20. For tasks, our method decomposes tasks into SHUTs by deriving the part-of-speech tags (nouns, verbs, etc.) for the given task description using Stanford CoreNLP [29]. For experiments, we use the regions generated in the 5th level of the hierarchical segmentation since they provide a reasonable overlap with object and non-object instances. Further details can be found in the supplementary material.

Evaluation Metric. We use an evaluation metric (called area under AP-cost curve) that measures the classification accuracy vs. the cost of selected features. For each λ, a subset of features get activated by optimizing Eqs. 1 and 2. We compute the classification accuracy for that subset and we have the total cost of features in that subset. We plot curves whose x axes correspond to region classification Average Precision (AP) and y axes correspond to $1 - \frac{C_{sel}}{C_{tot}}$, where C_{sel} is the cost of the selected features and C_{tot} is the total cost of all features. We plot these curves by varying λ, where each λ corresponds to one point on the curve.

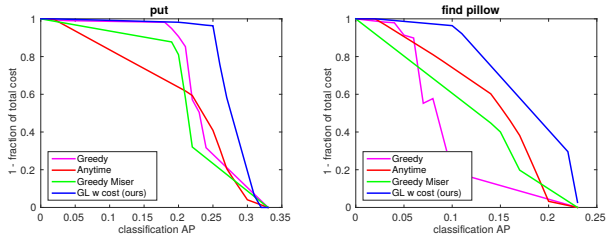
8.2. Results for Known SHUTs

We evaluate our method, GL COST, for cost-sensitive feature selection for a known SHUT and compare it with the previous work and baselines.

Comparisons. L1 is a baseline that uses L1 regularization for linear SVM. In other baselines, the weights for the entire feature are switched on/off, but for the L1 baseline, a subset of the dimensions of the weights might be switched off. GREEDY is a baseline, which selects the features in the

	walk	sit	put	grasp	find bag	find bed	find blinds	find book	find bottle	find box	find cabinet	find clothes	find cup	find desk	find door	find trash bin	find lamp	find light	find paper	find picture	find pillow	find shelves	find sink	find sofa	find window	**Avg.**
	Results for Known SHUTs																									
L1	48.8	4.9	6.1	52.0	0.3	19.6	7.1	0.2	1.4	0.5	9.9	0.8	0.0	3.6	3.1	0.5	1.9	3.2	1.5	7.5	6.4	12.4	4.5	13.6	6.4	8.64
GREEDY	65.9	10.0	21.9	75.8	0.7	20.4	7.7	0.8	2.7	0.0	9.5	1.0	0.8	5.3	5.5	0.8	1.4	10.4	2.9	10.7	7.5	11.0	4.2	15.7	11.0	12.16
ANYTIME [22]	56.9	12.5	19.9	76.5	0.8	33.9	17.1	0.8	3.7	0.1	16.4	1.1	0.6	5.2	4.7	0.4	1.6	12.4	3.2	13.9	13.2	21.4	8.7	24.5	10.5	14.41
GREEDY MISER [40]	57.1	**25.3**	19.5	70.6	0.3	**40.2**	**23.9**	0.9	3.3	0.2	**19.7**	0.8	0.0	3.6	4.6	0.4	1.5	7.2	2.4	17.2	10.7	**24.6**	1.1	**33.0**	15.2	15.34
GL NO COST	62.1	13.1	22.8	76.4	0.9	29.2	12.8	1.0	3.7	1.1	9.1	2.1	0.8	6.9	6.6	1.1	3.5	14.7	5.1	14.3	12.5	19.4	12.8	18.5	15.3	14.63
GL COST (OURS)	**66.2**	15.6	**27.0**	**77.4**	**1.0**	38.5	20.1	**2.2**	**8.4**	**1.7**	13.0	**3.7**	**2.2**	**7.0**	**7.4**	**2.0**	**8.6**	**19.5**	**8.7**	**17.7**	**17.2**	21.5	**19.7**	22.6	**22.6**	**18.06**
	Results for Unseen SHUTs																									
RANDOM	5.5	**1.2**	4.4	**65.0**	**0.4**	3.9	1.5	0.9	0.7	0.9	3.7	1.1	0.0	2.3	0.5	**0.8**	0.4	**1.0**	0.6	1.1	3.5	**6.9**	0.1	4.2	1.9	4.50
NO TEXT	**19.6**	1.0	19.6	63.5	0.0	5.4	3.2	**1.6**	**3.9**	**1.0**	**3.9**	1.0	0.5	**2.9**	1.4	0.0	**1.0**	0.0	**2.9**	**4.8**	3.3	3.3	**1.3**	9.1	4.7	6.37
NO VISUAL	17.6	1.0	**19.7**	64.0	0.0	8.9	4.6	**1.6**	0.9	**1.0**	3.3	1.0	1.0	2.8	**2.9**	**0.8**	**1.0**	**1.0**	2.8	4.3	3.5	6.8	0.0	**9.6**	**5.4**	6.60
ZERO SHOT (OURS)	**19.6**	1.0	19.6	63.5	0.0	**23.0**	**4.9**	**1.6**	1.9	**1.0**	**3.9**	**1.7**	**1.5**	**2.9**	**2.9**	0.0	**1.0**	0.0	**2.9**	**4.8**	**6.9**	3.9	**1.3**	9.2	4.9	**7.46**
ORACLE BEST	17.5	6.5	19.7	72.0	1.8	16.6	4.5	3.0	3.9	1.0	5.0	2.0	1.0	3.4	2.9	1.0	1.9	1.1	2.9	7.8	7.6	7.0	1.0	9.6	9.4	8.41
ORACLE CLUSTER	19.6	10.8	19.6	71.6	1.0	23.0	4.9	1.6	4.0	1.0	4.0	2.0	1.5	2.9	3.0	0.5	1.0	9.8	3.0	6.8	6.9	5.9	1.3	9.2	6.9	8.87

Table 1. The results for known SHUTs (*top*) and unseen SHUTS (*bottom*). The evaluation metric is the area under the classification AP vs. (1-fraction of total cost) curves.



(a) (b)

Figure 5. classification AP vs. (1-fraction of total cost) curves for some example known SHUTs.

order of their cost (from low to high). ANYTIME [22] and GREEDY MISER [40] are the two state-of-the-art cost-based feature selection methods. We use their publicly available implementation and train (and tune) them on our data (more details in supplementary material). Both methods gradually add one feature to the existing selection of features, while our method sometimes removes a selected feature and adds a new feature (as we vary λ). GL NO COST is a cost-agnostic variation of our method, in which the cost is replaced by the feature length similar to the original formulation of Group Lasso.

Results. Figure 5 shows the AP-cost curves for a few SHUTs. The supplementary material includes the rest of the curves. Table 1 shows the the area under AP-cost curves for all the SHUTs. Our method outperforms the baselines and the state of the art across most of the known SHUTs. As shown in the curves, given a fixed cost, GL COST achieves the highest AP compared to the other methods. Similarly, for a fixed AP, GL COST selects the features with the least cost. As shown on the curves, the region classification AP usually increases as more features are selected. The point on the x axis shows the region classification AP when all the features are selected.

Qualitative Examples. Figure 4 shows a few qualitative examples for predicting *walking*. GL COST achieves the best result for *walking* with much lower cost, confirming that we do not always need complex and costly features to achieve the best performance. In Table 2, we show the order of activation of features for some example SHUTs. For example, *support relationship* feature appears earlier in the ordering of *putting* compared to the ordering for *grasping*. On the other hand, *3D Shape* feature appears earlier in the ordering of *grasping*.

	Feature IDs								
putting	(2)	(1)	(3)	(8)	(5)	(6)	(9)	(7)	(4)
grasping	(1)	(3)	(2)	(5)	(7)	(4)	(8)	(9)	(6)

Table 2. The order feature activations for two SHUTs (from left to right). (1) Height, (2) Surface Normal, (3) Material, (4) 2D appearance, (5) 3D Shape, (6) Distance to any cube, (7) Distance to cubes of certain categories, (8) Support relationship, (9) Object size. The ID assignment is based on feature cost.

8.3. Results for Unseen SHUTs (Zero-shot)

We evaluate how well our method (ZERO SHOT) generalizes to unseen SHUTs and compare the results with baselines and ablations. We report results in Table 1 (*bottom*). For this experiment, we remove one of the SHUTs from the SHUT vocabulary and measure how well our method selects features for the removed SHUT (which is now unseen) using the remaining SHUTs.

Comparisons. RANDOM selects the features of one of the randomly chosen remaining SHUTs to make predictions for the unseen SHUT. The results reported in the table are the average of 5 trials. NO TEXT is a variant of our method that ignores the linguistic information for clustering. Note that it is impossible to remove linguistic information from the inference step since the only available information for an unseen SHUT is its name. NO VISUAL is a variant of

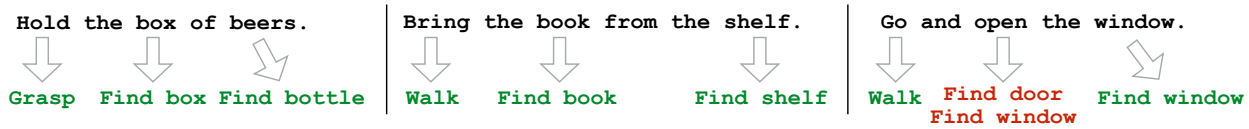


Figure 6. Example results of mapping unseen *tasks* to the SHUTs in the vocabulary.

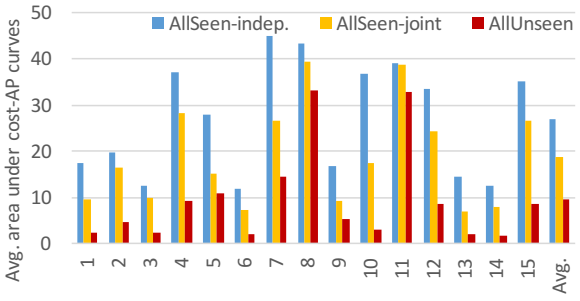


Figure 7. Task results. The x axis denotes the 15 defined tasks.

our method that ignores the visual similarity ϕ^f. ORACLE BEST is an oracle zero shot method; it selects the features of one of the remaining SHUTs which produces the best results for the unseen SHUT.

Results. Table 1 shows the comparisons in terms of the area under AP-cost curves. Our ZERO SHOT method outperforms its variants, NO TEXT and NO VISUAL, and the RANDOM baseline for most of the SHUTs. This shows the importance of both visual and textual similarities in clustering and optimizing Eq. 2. A drop in performance for a few SHUTs is due to the fact that we have used a fixed parameter for the number of clusters for all the SHUTs, while using different numbers of clusters for different SHUTs improve results. For example, the performance for *sitting* improves dramatically when the number of clusters is changed to 10. The ORACLE CLUSTER row in Table 1 shows the results for the case that an oracle chooses the best number of clusters for each SHUT. Supplementary material includes the results for different number of clusters.

It is interesting to observe that for some SHUTs e.g., *finding book* and *finding cup*, our ZERO SHOT method (Table 1(*bottom*)) outperforms some of the supervised methods in Table 1(*top*). We conjecture that this is due to lack of training examples in the supervised setting. The zero-shot method takes advantage of more training examples by clustering SHUTs together.

8.4. Results for Tasks

We evaluate how well our method performs on tasks, which can be decomposed into a set of SHUTs. We design a set of 15 tasks (such as "Put the box next to the cabinet" and "Sit on the sofa") – the complete list can be found in the supplementary material. In order to quantitatively evaluate these tasks, we define them such that they can be decomposed into the SHUTs in our vocabulary. The participating SHUTs are extracted nouns and verbs appearing in the task description, derived from the part-of-speech tags (nouns, verbs, etc.) for the given description.

Quantitative Results. We experiment in three settings: ALL SEEN-INDEP.: If each SHUT appearing in the task description can be aligned to a SHUT in our vocabulary $\mathcal{S}$ and we train for each SHUT independently. ALL SEEN-JOINT: We train for SHUTs in a task jointly i.e., the labels are the union (logical OR) of the constituent SHUT labels. ALL UNSEEN: If None of the SHUTs appearing in the task description are known. As before, for unseen SHUTs, we use the features of the most similar cluster. Figure 7 reports the average of area under AP-cost curve for all the SHUTs in each task. It is interesting to observe that our zero shot method for ALL UNSEEN performs reasonably well compared to ALL SEEN cases. For example, the results for $8th$ and $11th$ tasks, are about 80% of results of ALL SEEN - INDEP., which is a supervised case.

Qualitative Examples. The above tasks are decomposed into SHUTs in the vocabulary. However, for some tasks (e.g., "Bring the book from the shelf') some of the extracted nouns or verbs (*bring*) do not exist in the vocabulary, and hence the quantitative evaluation is not feasible. Figure 6 shows interesting qualitative examples of mappings using our method; *hold* has been mapped to *grasp* or *bring* has been mapped to *walk*. There are, however, some mistakes in finding similarities – e.g., the verb *open* is mapped to the cluster of *find door* and *find window*, while a better mapping could be *grasp*.

9. Conclusion

We addressed the problem of task-driven recognition. Collecting training data for all tasks is impractical, so we decomposed the tasks into simpler unit tasks that can be shared among tasks. We introduced a cost-sensitive method for selecting appropriate features/representations for the unit tasks. We showed that our method generalizes to unseen unit tasks which are not available in training data.

This work only addresses the problem of selecting features and representations for task-driven recognition. Handling other aspects of task-driven recognition (e.g., searching over the models) will be our future work. Additionally, we plan to extend the same framework for videos. Finally, we plan to explore more sophisticated language processing techniques [17, 25] to decompose task descriptions.

Acknowledgments: This research was partially supported by ONR N00014-13-1-0720, NSF IIS-1338054, NSF IIS-1352249, and Allen Distinguished Investigator Award.

References

[1] A. Blum and P. Langley. Selection of relevant features and examples in machine learning. *AI*, 1997. 2

[2] A. Borji, D. N. Sihite, and L. Itti. Learning of task-specific visual attention. In *CVPR*, 2012. 2

[3] S. Büttcher and C. L. A. Clarke. Efficiency vs. effectiveness in terabyte-scale information retrieval. In *TREC*, 2005. 4

[4] M. Chen, Z. Xu, K. Q. Weinberger, O. Chapelle, and D. Kedem. Classifier cascade for minimizing feature evaluation cost. In *AISTATS*, 2012. 2

[5] K. W. Church and P. Hanks. Word association norms, mutual information and lexicography. In *ACL*, 1989. 4

[6] F. Duvallet, T. Kollar, and A. Stentz. Imitation learning for natural language direction following through unknown environments. In *ICRA*, 2013. 2

[7] M. Elhoseiny, B. Saleh, and A. Elgammal. Write a classifier: Zero-shot learning using purely textual descriptions. In *ICCV*, 2013. 2

[8] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing objects by their attributes. In *CVPR*, 2009. 2

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *EuroCOLT*, 1995. 2

[10] A. Frome, G. S. Corrado, J. Shlens, S. Bengio, J. Dean, M. Ranzato, and T. Mikolov. Devise: A deep visual-semantic embedding model. In *NIPS*, 2013. 2

[11] T. Gao and D. Koller. Active classification based on value of classifier. In *NIPS*, 2011. 2

[12] W. S. Geisler and D. Kersten. Illusions, perception and bayes. *Nature Neuroscience*, 2002. 2

[13] H. Grabner, J. Gall, and L. V. Gool. What makes a chair a chair? In *CVPR*, 2011. 2

[14] A. Grubb and J. A. Bagnell. Speedboost: Anytime prediction with uniform near-optimality. In *AISTATS*, 2012. 2

[15] A. Gupta, S. Satkin, A. A. Efros, and M. Hebert. From 3d scene geometry to human workspace. In *CVPR*, 2011. 2

[16] H. Hajishirzi, J. Hockenmaier, E. T. Mueller, and E. Amir. Reasoning about robocup soccer narratives. In *UAI*, 2011. 2

[17] H. Hajishirzi, M. Rastegari, A. Farhadi, and J. K. Hodgins. Semantic understanding of professional soccer commentaries. In *UAI*, 2012. 8

[18] W. Hwang, H. Hajishirzi, M. Ostendorf, and W. Wu. Aligning sentences from standard wikipedia to simple wikipedia. In *NAACL*, 2015. 4

[19] K. Ikeuchi and M. Hebert. Task oriented vision. In *IROS*, 1991. 2

[20] D. Jayaraman, F. Sha, and K. Grauman. Decorrelating semantic visual attributes by resisting the urge to share. In *CVPR*, 2014. 2

[21] Y. Jiang, H. S. Koppula, and A. Saxena. Hallucinated humans as the hidden context for labeling 3d scenes. In *CVPR*, 2013. 2

[22] S. Karayev, M. Fritz, and T. Darrell. Anytime recognition of objects and scenes. In *CVPR*, 2014. 2, 7

[23] M. G. Kendall. *Rank Correlation Methods*. 1970. 5

[24] D. Koller and M. Sahami. Toward optimal feature selection. In *ICML*, 1996. 2

[25] R. Koncel-Kedziorski, H. Hajishirzi, and A. Farhadi. Semantic understanding of professional soccer commentaries. In *EMNLP*, 2014. 8

[26] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 2

[27] D. Lin, S. Fidler, and R. Urtasun. Holistic scene understanding for 3d object detection with rgbd cameras. In *ICCV*, 2013. 5, 6

[28] J. Liu, S. Ji, and J. Ye. *SLEP: Sparse Learning with Efficient Projections*, 2009. 6

[29] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky. The Stanford CoreNLP natural language processing toolkit. In *ACL*, 2014. 6

[30] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. In *ICLR Workshop*, 2013. 4

[31] D. K. Misra, J. Sung, K. Lee, and A. Saxena. Tell me dave: Context-sensitive grounding of natural language to mobile manipulation instructions. In *RSS*, 2014. 2, 3

[32] V. Navalpakkam and L. Itti. Modeling the influence of task on attention. *Vision Research*, 2005. 2

[33] M. Norouzi, T. Mikolov, S. Bengio, Y. Singer, J. Shlens, A. Frome, G. S. Corrado, and J. Dean. Zero-shot learning by convex combination of semantic embeddings. In *ICLR*, 2014. 2

[34] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *CVPR*, 2012. 5

[35] N. Silberman, D. Hoiem, P. Kohli, and R. Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 5, 6

[36] R. Socher, M. Ganjoo, C. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *NIPS*, 2013. 2

[37] S. Song and J. Xiao. Sliding shapes for 3d object detection in depth images. In *ECCV*, 2014. 5

[38] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. Roy. Imitation learning for natural language direction following through unknown environments. In *AAAI*, 2011. 2

[39] R. H. Wurtz, M. E. Goldberg, and D. L. Robinson. Behavioral modulation of visual responses in the monkey: Stimulus selection for attention and movement. *Progress in Psychobiology and Physiological Psychology*, 1980. 2

[40] Z. Xu, K. Weinberger, and O. Chapelle. The greedy miser: Learning under test-time budgets. In *ICML*, 2012. 2, 7

[41] A. Yarbus. *Eye Movements and Vision*. 1967. 2

[42] M. Yuan and Y. Lin. Model selection and estimation in regression with grouped variables. *Journal of the Royal Statistical Society, Series B*, 2007. 3

[43] S. Zheng, M.-M. Cheng, J. Warrell, P. Sturgess, V. Vineet, C. Rother, and P. H. Torr. Dense semantic image segmentation with objects and attributes. In *CVPR*, 2014. 5

[44] Y. Zhu, A. Fathi, and L. Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *ECCV*, 2014. 2