TASK2BOX: Box Embeddings for Modeling Asymmetric Task Relationships

Supplementary Material

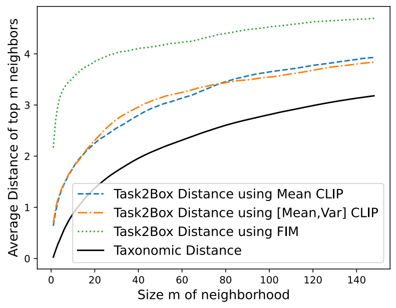

Figure 7. **Euclidean Distance between Predicted Embeddings Reflect Taxonomic Distance in iNaturalist+CUB.** TASK2BOX embeddings are projected closer for smaller taxonomic distances.

7. Further Analysis of TASK2BOX

Euclidean Distance Between TASK2BOX Embeddings Reflects Taxonomic Structure. Fig. 7 displays the average embedding and taxonomic distance within a neighborhood of the m closest datasets. Taxonomic distance is defined as the symmetric graph distance in the taxonomy tree. As the number of neighbors m being considered increases, we plot the average distance between TASK2BOX embeddings. Although the model is not directly supervised to mirror taxonomic distances, it inherently learns to position similar datasets closer together.

Average Dataset CLIP Embeddings are Effective. While employing both the mean and variance could assist in estimating the general location and spread, Table 1 indicates that using solely the mean across different data points in the dataset generally yields better or equally effective performance. This can be attributed to the unique positioning of the average embedding depending on the hierarchical level (i.e., class, order, or family in iNaturalist). Fig. 8 displays box embeddings at various hierarchical levels in iNaturalist, where the center coordinates of supersets lie beyond the bounds of the box representations of their subsets.

TASK2BOX Performs Well Even When Trained on Only 50% of Relationships Between Datasets. Table 5 demonstrates the model's ability to generalize to unseen relationships when trained with only 50% of the pairwise relationships between datasets. Despite the limited relationship data used for training, TASK2BOX can effectively generalize to other relationships. This aspect is particularly valuable in scenarios where acquiring comprehensive dataset relationship information is resource-intensive. For instance, in computing task affinities as done in Taskonomy [52], determining transfer learning gains from a source task to a target task necessitates model training. The ability to capture unseen relationships without complete information makes resource-intensive analyses more practical.

TASK2BOX has Several Advantages Over Simple Dimensionality Reduction. TASK2BOX can predict and visualize asymmetric relationships of *new tasks* in relation to a collection of other existing tasks. Instead of re-optimizing the representation each time a new task is introduced, TASK2BOX predicts an embedding that encapsulates the relationship of the new entity with existing entities. This ability to predict relationships between novel tasks demonstrates that our model can *generalize* beyond the tasks it has previously seen. While existing dimensionality reduction methods such as t-SNE [44] and UMAP [28] provide visualizations, they inherently represent each entity as a point in Euclidean space, resulting in *symmetric* visualizations. In contrast, TASK2BOX can represent *asymmetric* relationships, such as hierarchies and transferability.

TASK2BOX Can Be Used for Several Applications Related to Task Relationships. TASK2BOX serves as a tool for visualizing relationships among large collections of datasets, facilitating dataset discovery, measuring overlaps, predicting transferability, and organizing tasks for multitasking. For instance, upon encountering a new task, TASK2BOX can predict its relationship with existing tasks (see Tables 1, 2, and 3 under Novel Datasets). By utilizing task affinities, solutions for the new task can be developed by leveraging models from existing tasks with strong relationships, thereby enabling effective transfer learning. Furthermore, hierarchical relationships can identify which datasets have sufficient overlap to be utilized for further training or evaluation purposes. At present, there are limited techniques available that effectively visualize asymmetric relationships between datasets.

Training TASK2BOX is Most Effective when Utilizing All Loss Terms. Table 4 demonstrates the impact of various terms in the loss function (Eq. 4). Optimal performance is achieved by incorporating all loss terms during training.

8. Additional Visualizations

Fig. 9 and 10 present additional visualizations for iNaturalist+CUB and ImageNet, respectively. Across various subsets of datasets, TASK2BOX successfully learns and depicts the hierarchical relationships. Datasets that belong to the same parent category are shaded in identical colors. Fig. 11 provides further visualizations for source/target task relationships within the Taskonomy benchmark, effectively illustrating source and target tasks that transfer well.

	$\mathcal{L}_E + \mathcal{L}_D + \mathcal{L}_R$	$-\mathcal{L}_E$	$-\mathcal{L}_D$	$-\mathcal{L}_R$
iNat+CUB (F1 Score)	84.67%	4.60%	84.56%	84.21%
ImageNet (F1 Score)	90.58%	3.25%	89.02%	89.76%
Taskonomy (ρ)	0.94	0.26	0.62	0.93

Table 4. **Effect of Various Loss Terms on Training TASK2BOX.** (Refer to Eq. 4). $\mathcal{L}_E$ accounts for overlap, $\mathcal{L}_D$ the distance, and $\mathcal{L}_R$ the regularization. 5D box embeddings were predicted using the base representation $\mu CLIP$ for iNaturalist (iNat), CUB, and ImageNet datasets; attribute-based representations were used for Taskonomy.

	ImageNet			iNaturalist + CUB			Taskonomy
	μ_{CLIP}	$[\mu, \sigma^2]_{CLIP}$	FIM	μ_{CLIP}	$[\mu, \sigma^2]_{CLIP}$	FIM	Spearman's ρ
TASK2BOX (2D)	51.94%	51.07%	27.12%	53.52%	51.78%	27.21%	0.82 $\pm$ 0.06
TASK2BOX (3D)	<u>67.26%</u>	<u>66.68%</u>	<u>41.23%</u>	<u>63.78%</u>	<u>65.38%</u>	<u>48.19%</u>	<u>0.90 $\pm$ 0.02</u>
TASK2BOX (5D)	**74.09%**	**72.89%**	**53.66%**	**68.11%**	**68.60%**	**62.61%**	**0.92 $\pm$ 0.03**
MLP	45.44%	52.24%	31.44%	43.42%	55.08%	29.91%	0.77 $\pm$ 0.06
Linear	12.75%	10.47%	25.90%	3.53%	5.63%	8.98%	0.70 $\pm$ 0.05

Table 5. **Performance when Trained on only 50% of Relationships.** Even with limited data on relationships between existing datasets, TASK2BOX shows strong performance.

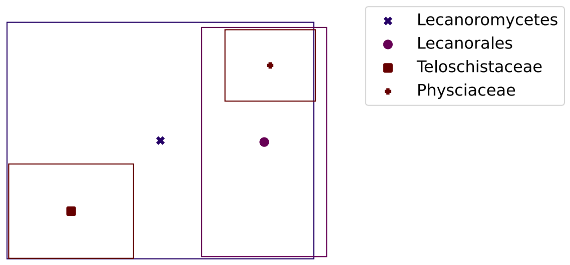

Figure 8. **Average embeddings (center box coordinates) of superset datasets are outside the bounds of the subset datasets**. Due to the distinct location of the mean, simply providing the average embedding to the model can be sufficient for training.

9. Implementation Details

Below, we detail the configurations and hyperparameters used for extracting the base representations, training TASK2BOX, and visualizing public datasets.

9.1. Base Task Representations

9.1.1 CLIP Embeddings

The CLIP base embeddings were generated using a ViT-H/14 network pretrained on LAION-2B. Each image was preprocessed by the provided transforms in the OpenCLIP library [20]. The text labels for the corresponding images were processed to be in the form "A photo of [CLS]", tokenized with OpenCLIP, and encoded using the pretrained network. For iNaturalist+CUB, the labels correspond to the species of the organism in the image, whereas for ImageNet, the labels correspond to the name of the object in the image. The embeddings are taken across all images and processed as discussed in § 3. To have a consistent setup across different base representations for iNaturalist+CUB and ImageNet, we excluded datasets that only have a single class (since TASK2VEC only extracts embeddings for those that have multiple classes). In addition, similar to the TASK2VEC setup, we also exclude taxa that have conflicting taxonomy between iNaturalist and CUB.

9.1.2 TASK2VEC Embeddings

For each dataset, we generated embeddings using TASK2VEC with their recommended settings, on the pretrained ResNet34 network, using the Adam optimizer over 10 epochs with $LR = 4 \times 10^{-4}$ and weight decay of $\lambda = 10^{-4}$. We used a maximum dataset size of 10,000 and 100 batches in order to generate task representations with the montecarlo method. For a given hierarchical task, we generated a dataset using a subset of classes which fell within the given hierarchy for training. For instance, if there were 20 species in a given family, our task representation for that family was the corresponding vector from TASK2VEC trained on the 20 species within that family. We combined CUB and iNaturalist into a single dataset by merging overlapping species classes. Due to discrepancies in the taxonomy between the two datasets, we simply excluded taxa which conflicted in our experiments. Finally, since TASK2VEC requires the training of a model, we did not generate embeddings for groupings which only contained a single element.

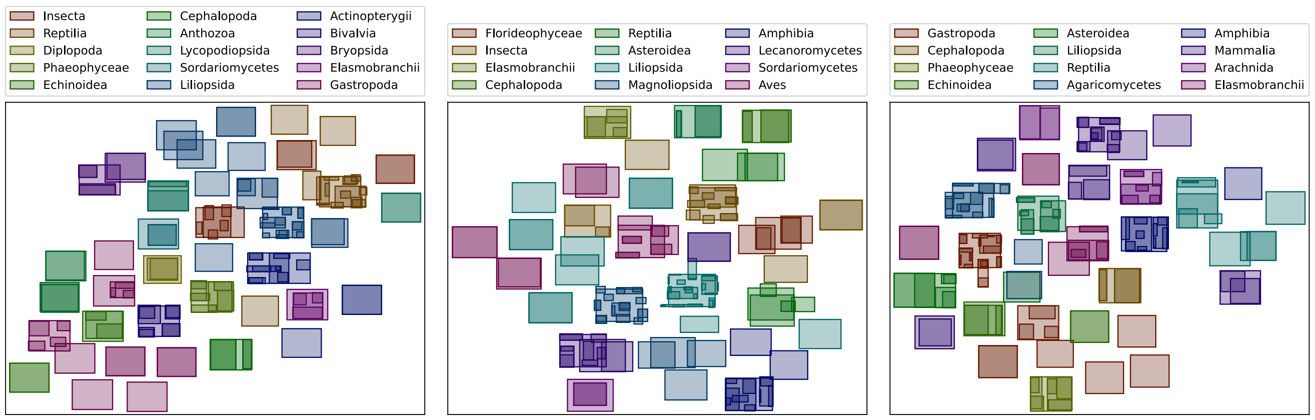

Figure 9. **Learned embeddings for additional iNaturalist+CUB test cases**. Visualization of various test cases to evaluate TASK2BOX. Similar to previous results, groups belonging to the same classes naturally form a hierarchy and are positioned close to each other.

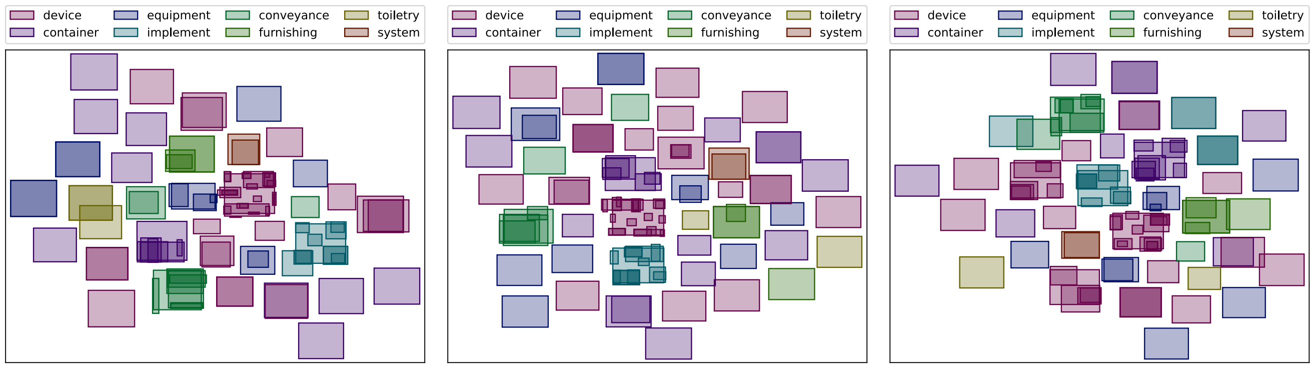

Figure 10. **Learned embeddings for additional ImageNet test cases**. TASK2BOX is shown to learn hierarchies in the various groups. In addition, similar objects are placed closer to each in the embedding space.

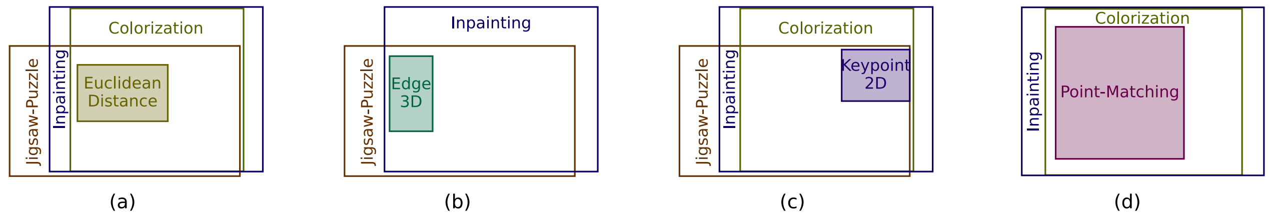

Figure 11. **Additonal visualizations of tasks in Taskonomy showing source tasks that transfer well to specific target tasks (shaded)**. (a) shows that Colorization, Jigsaw Puzzle, and Inpainting are all good candidates for transfer learning onto Euclidean distance. (b), (c), and (d) show similar source task relationships for Edge-3D, Keypoints-2D, and Point Matching, respectively. It can also be observed that Inpainting transfers well to Colorization from observing subfigures (a), (c), (d).

9.1.3 Attribute-based Embeddings

Due to the various modalities present in Taskonomy tasks that have different types of inputs/outputs, we were unable to represent them via CLIP in a straightforward and consistent manner. As a result, we constructed a table of 15 attributes for each task which can be answered with a *yes* or *no*. In order to construct unique embeddings for each task, the i-th attribute corresponds to the i-th dimension of the base representation embedding e_i, where $e_i = 1$ if the task satisfies the attribute (i.e., when considering a specific task, the i-th attribute can be answered with a *yes*) and $e_i = 0$ otherwise. While this representation is not necessarily exhaustive, we show that TASK2BOX can effectively generalize on unseen relationships and tasks. Future work can consider looking into available datasheets [15] or dataset descriptions to construct attribute-based embeddings. Table 6 enumerates the attributes we considered for representing each task in Taskonomy. Table 7 shows the corresponding base representations for each Taskonomy task based on the enumerated attributes.

9.2. Task2Box Training Details

The box embedding library from [8] was used to create instances of boxes. To compute the loss in Eq. 5, we use a volume temperature of 0.1, and intersection temperature of 0.0001. The model for Task2Box uses a 3-layer MLP with two linear layer heads, and is trained using the loss in Eq. 4. The Adam optimizer is used with $LR = 1 \times 10^{-3}$.

For hierarchical datasets, we train all models (including baselines) on at least 150 datasets at a time, with 18,000 relationships used for training, 2,250 for validation, and the remaining 2,250 relationships for evaluation. An unseen collection of 100 datasets (15,000 relationships with existing datasets) are used as novel dataset evaluations. The performance is averaged over multiple instances of training and testing from randomly sampled datasets and relationships.

For the Taskonomy benchmark, models are also trained on a subset of randomly sampled relationships $\mathcal{R}$ within existing datasets $\mathcal{D}_E$. Eq. 4 is optimized such that relationships between low dimension embeddings z match the task affinities provided by Taskonomy [52]. Since Taskonomy is limited to 25 vision tasks, only 3 could be used for evaluating the performance on novel datasets. The rest of the vision tasks were used for training and validation. Similar to the hierarchical datasets, multiple instances were sampled for performance evaluation. At the same time, while task affinities from [52] were already normalized in the range $[0, 1]$, the distribution is skewed to the right. The task affinities are re-scaled using Eq. 8 to normalize the distribution where x is the task affinity value, x' is the re-scaled task affinity value, and k is a hyperparameter we set to 50. We train all models using the re-scaled task affinities, and convert it back to the original scale for evaluation.

$$x' = \frac{\exp(kx) - \exp(-kx)}{\exp(kx) + \exp(-kx)} \tag{8}$$

9.3. Public Dataset Visualization

Since no ground truth relation labels are available for training the model on public datasets, we use the following "soft" labels for defining relationships between pairs of datasets. The soft labels are defined as the asymmetric overlap (similar to Eq. 5) between the base representation of datasets. For the Hugging Face visualization, we use CLIP embeddings as the base representation: for each dataset, we sample N image-label pairs where $N \leq 10{,}000$, then we embed the images and the labels using CLIP to produce N embeddings per dataset. To get the soft overlap value between two datasets $so(\mathcal{D}_i, \mathcal{D}_j)$, we use Eq. 9 where $\mathcal{E}_i = \{e_i^{(1)}, e_i^{(2)}, \ldots e_i^{(N)}\}$ is the set of image-label embeddings for dataset $\mathcal{D}_i$, $co(\mathcal{D}_i, \mathcal{D}_j)$ is the count of overlapping embeddings of $\mathcal{D}_i$ with respect to $\mathcal{D}_j$, and $|\mathcal{E}_i|$ is the number of embeddings in $\mathcal{E}_i$.

$$so(\mathcal{D}_i, \mathcal{D}_j) = \frac{co(\mathcal{D}_i, \mathcal{D}_j)}{|\mathcal{E}_i|} \tag{9}$$

In Eq. 9, $co(\mathcal{D}_i, \mathcal{D}_j)$ counts the number of embeddings $e_i^{(k)} \in \mathcal{E}_i$ that satisfy $d_{ij}^{(k)} < t_j$, where $d_{ij}^{(k)}$ is the minimum euclidean distance of $e_i^{(k)}$ among all embeddings $e_j^{(k)} \in \mathcal{E}_j$ of dataset $\mathcal{D}_j$, and t_j is the average euclidean distance $d_{euc}(\cdot, \cdot)$ between any two embeddings in $\mathcal{D}_j$. Eq. 10 shows how $co(\mathcal{D}_i, \mathcal{D}_j)$ is computed where $[\cdot]$ is an indicator function that evaluates to 1 if the expression inside the brackets is true, and 0 otherwise. $N = |\mathcal{E}_i|$ which is the number of image-label embeddings in dataset $\mathcal{D}_i$.

$$co(\mathcal{D}_i, \mathcal{D}_j) = \sum_{k=1}^{N} [d_{ij}^{(k)} < t_j] \tag{10}$$

Eq. 11 and Eq. 12 show how $d_{ij}^{(k)}$ and t_j are computed. $M = |\mathcal{E}_j|$ which is the number of image-label embeddings in dataset $\mathcal{D}_j$.

$$d_{ij}^{(k)} = \min_{e_j^{(l)} \in \mathcal{E}_j} d_{euc}\left(e_i^{(k)}, e_j^{(l)}\right) \tag{11}$$

$$t_j = \frac{2}{M(M-1)} \sum_{u=1}^{M} \sum_{v=u+1}^{M} d_{euc}\left(e_j^{(u)}, e_j^{(v)}\right) \tag{12}$$

The soft overlaps between all pairs of datasets are computed and used as supervision to train Task2Box. Note that similar to Eq. 5, Eq. 9 is also an asymmetric measure of similarity between two datasets. The input to Task2Box uses the average CLIP embedding per dataset (Eq. 1), and the model is trained using Eq. 4. We additionally include a loss term $\mathcal{L}_A$ that encourages the area of the box embedding z_i to correspond to the size of the dataset $|\mathcal{D}_i|$ (i.e., the number of samples available in the dataset). Eq. 13 shows how the loss is computed. $\mathcal{L}_A$ is added to the objective function discussed in Eq. 4.

$$\mathcal{L}_A = \mathrm{MSE}\left(\mathrm{vol}(z_i), |\mathcal{D}_i|\right) \tag{13}$$

Results on image classification datasets from Hugging Face are shown in Fig. 5. The same method can also be applied to other datasets where only images and corresponding labels are available.

Dimension	Corresponding Task Attribute (answered with yes/no)
0	Does the task input consist of multiple images depicting different scenes?
1	Are the output spatial dimensions (height and width) the same as the input?
2	Does the task output contain geometric information?
3	Does the task output contain classification of objects?
4	Does the task require 3D knowledge?
5	Is the task a generative task?
6	Is the task output a single channel output?
7	Does the task have more than one input?
8	Does the task have more than two inputs?
9	Is the task output the result of a first order operation (e.g. edges as opposed to curvature)?
10	Does the task require generating new information?
11	Does the task require knowledge of objects?
12	Does the task require knowledge of colors/light?
13	Does the task involve camera pose estimation?
14	Does the task involve pixel alignment?

Table 6. **List of attributes used to represent each task in Taskonomy and their corresponding dimension**. We did this to generate a unique representation of each task. While not necessarily exhaustive, we show TASK2BOX can generalize using these attributes.

Task Name	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Autoencoder	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
Colorization	0	1	0	0	0	1	0	0	0	0	1	0	1	0	0
Curvatures	0	1	1	0	1	0	0	0	0	1	0	0	0	0	0
Denoising-Autoencoder	0	1	0	0	0	1	0	0	0	0	1	0	0	0	0
Depth	0	1	1	0	1	0	1	0	0	0	0	0	0	1	0
Edge-2D	0	1	1	0	0	0	1	0	0	0	0	0	0	0	0
Edge-3D	0	1	1	0	1	0	1	0	0	0	0	1	0	0	0
Euclidean-Distance	0	1	1	0	1	0	1	0	0	0	0	0	0	0	0
Inpainting	0	1	0	0	0	1	0	0	0	0	1	1	1	0	0
Jigsaw-Puzzle	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
Keypoint-2D	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0
Keypoint-3D	0	1	0	0	1	0	1	0	0	0	0	0	0	0	0
Object-Classification	0	0	0	1	0	0	1	0	0	0	0	1	0	0	0
Reshading	0	1	1	0	1	0	1	0	0	0	0	0	1	0	0
Room-Layout	0	0	1	0	1	0	1	0	0	0	0	0	0	1	0
Scene-Classification	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0
Segmentation-2D	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
Segmentation-3D	0	1	0	0	1	0	0	1	1	0	0	0	0	0	0
Segmentation-Semantic	0	1	0	1	0	0	0	0	0	1	0	1	0	0	0
Surface-Normal	0	1	1	0	1	0	0	0	0	0	0	0	0	0	0
Vanishing-Point	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0
Pairwise-Nonfixated-Camera-Pose	1	0	1	0	1	0	1	1	0	0	0	0	0	1	0
Pairwise-Fixated-Camera-Pose	1	0	1	0	1	0	1	1	0	0	0	0	0	1	1
Triplet-Fixated-Camera-Pose	1	0	1	0	1	0	1	1	1	0	0	0	0	1	1
Point-Matching	1	0	0	0	0	0	1	1	0	0	0	0	0	0	1

Table 7. **Base representation used for each task for the attribute-based embedding**. The i-th dimension signifies a yes/no response to the question in Table 6, where $e_i = 1$ for yes, and $e_i = 0$ otherwise.

10. Descriptions of Taskonomy Tasks

Below, we provide a brief description of each task. For a more complete definition, we refer to Taskonomy [52].

1. Autoencoder: Using an encoder-decoder neural network that takes an input image, distills it to a single vector representation, then reconstructs the image.
2. Colorization: Selecting pixel color assignments for a black and white image.
3. Curvatures: Given an image, identify the degree of cur-

vature of the physical object on each pixel.

4. Denoising-Autoencoder: Denoise an image using an encoder-decoder structure.
5. Depth: Find the z-buffer depth of objects in every pixel of an image.
6. Edge-2D: Identify strong boundaries in the image.
7. Edge-3D: Find occlusion edges, where an object in the foreground obscures things behind it.
8. Euclidean-Distance: For each pixel, find the distance of the object to the camera.
9. Inpainting: Given a part of an image, reconstruct the rest.
10. Jigsaw-Puzzle: Given different parts of an image, reassemble the parts in order.
11. Keypoint-2D: Find good pixels in the image which are distinctive for feature descriptors.
12. Keypoint-3D: Find good pixels like in Keypoint-2D, but using 3D data and ignoring distracting features such as textures.
13. Object-Classification: Assign each image to an object category.
14. Reshading: Given an image, generate a reshaded image which results from a single point light at the origin.
15. Room-Layout: Given an image, estimate the 3D layout of the room.
16. Scene-Classification: Assign a scene category to each image.
17. Segmentation-2D: Group pixels within an image, based on similar-looking areas.
18. Segmentation-3D: Group pixels within an image, based on both the image and depth image and surface normals.
19. Segmentation-Semantic: Assign each pixel to an object category.
20. Surface-Normal: Assign each pixel a vector representing the surface normal.
21. Vanishing-Point: Identify the vanishing points within an image.
22. Pairwise-Nonfixated-Camera-Pose: Identify the 6 degrees of freedom relative camera pose between two images.
23. Pairwise-Fixated-Camera-Pose: Identify the 5 degrees of freedom relative camera pose between two images which share a pixel center.
24. Triplet-Fixated-Camera-Pose: Identify the relative camera poses between three images.
25. Point-Matching: Given two images and one point, identify the matching point in the other image.