CLIPGraphs: Multimodal Graph Networks to Infer Object-Room Affinities

Ayush Agrawal*[1], Raghav Arora*[1], Ahana Datta[1], Snehasis Banerjee[1,2], Brojeshwar Bhowmick[2], Krishna Murthy Jatavallabhula[3], Mohan Sridharan[4], Madhava Krishna[1]

***Abstract*—This work focuses on improving upon pre-trained feature representations for learning functional and semantic priors for embodied AI tasks. Specifically, we propose a GCN-based training pipeline that fine-tunes the CLIP embeddings to effectively estimate object-room affinities. Our approach, CLIPGraphs, efficiently combines human commonsense domain knowledge, multimodal information from language and vision inputs(leveraging the strengths of CLIP); and a Graph Network to encode these functional/semantic priors. We experimentally demonstrate the effectiveness of our approach on a benchmark dataset of object categories, showing a significant improvement over state-of-the-art baselines. The learned embeddings from our approach can be used as priors in downstream embodied AI tasks such as object navigation and scene rearrangement, demonstrating the broad applicability of our method.**

I. Introduction

Imagine a robot being given a task of tidying up an unfamiliar house. This task is a variant of the *scene rearrangement* challenge for embodied AI [1]. To perform this task, the robot must first determine what tidying up means in this specific house, which requires constructing a representation of the current state of the house and inferring a possible goal state (i.e., a configuration in which the house is deemed as *tidy*). Any errors in this step can affect downstream planning and control, resulting in irrecoverable failure. Computing the most appropriate room location for specific object categories is thus critical to the successful completion of such tasks.

Human-inhabited environments such as homes and offices are designed to be functional and aesthetically pleasing. A key characteristic of such environments is the semantic organization, i.e., objects are placed in locations based on their purpose. This enables humans to adapt efficiently to new environments designed to serve the same purpose. For example, when a person enters a new home and wants to find sugar to make a cup of coffee, they instinctively look in the kitchen or pantry. We leverage this semantic organization to enable robots to predict the likely locations of any given object. Specifically, we leverage recent developments in vision-language representation learning to propose a flexible approach for learning *object-room affinities*, i.e., the relative likelihood of any given object belonging to a particular room in a house, based on image and text input.

*Denotes equal contribution
[1]Robotics Research Center, IIIT Hyderabad, India
[2]TCS Research, Tata Consultancy Services, India
[3]CSAIL, Massachusetts Institute of Technology, USA
[4]Intelligent Robotics Lab, University of Birmingham, UK

State-of-the-art methods have used Large Language Models (LLMs) as *commonsense* reasoning machinery for the *tidy up* task [2]. These methods are limited to textual descriptors, which can be challenging to ground to a specific scene. Moreover, they use ground truth object labels for generating object-room affinities, which limits their operation outside of the training data distribution. Others have used reinforcement learning (RL) to compute policies for related tasks such as visual semantic navigation [3]–[6], and Multi-Object Navigation [7]–[9], but do not fully leverage knowledge from *different sources* in the learning process.

Our framework, *CLIPGraphs*, seeks to leverage the complementary strengths of commonsense knowledge, data-driven methods, and multimodal embeddings to estimate object-room affinities with high precision. It does so by incorporating:

(1) A *knowledge graph* that encodes human preferences of the room location of objects in home environments;

(2) Joint embeddings of image and text features [10] to support multimodal learning and queries involving either object images or object name labels; and

(3) A graph network that learns object-room affinities over a dataset of common household objects based on latent embeddings of the knowledge graph that includes the image and text feature embeddings.

We evaluate our framework's ability to combine these components and correctly estimate the best room location for any given object, the key step in scene rearrangement. We do so using a dataset of around 8000 image-text pairs that we created by extracting images from the Web for 268 benchmark object categories [2]. We show experimentally that our framework substantially improves performance in comparison to state-of-the-art baselines comprising LLMs and language embeddings encoding commonsense knowledge of the location of objects.

II. Related Work

To train embodied agents to perform human-like activities, many common tasks have been explored recently like image goal navigation [11]–[14], object navigation [3], [5], [7], [15], embodied Question Answering [16]–[18], and scene rearrangement [1], [19], [20]. ALFRED [21], TEACh [22], and [23] studies the ability of agents to perform actions based on natural language instructions, and [24]–[26] use knowledge graphs for visual classification and object detection. While these works include explicit specification of the goal state

by a human agent, recent works [2], [27], [28] have started the inclusion of reasoning with commonsense knowledge to enable agents to perform these tasks intelligently.

III. Problem Formulation and Framework

To perform tidying up or other scene rearrangement tasks, a robot needs the key ability to accurately compute the appropriate location for any given object. To explore this ability, we created the *Images for Room-Object Nexus through Annotations* (IRONA) dataset of 30 RGB images from the Web for each of the 268 categories of household objects used by Housekeep [2] (See Appendix 2.2). For any such image, the robot had to compute the likelihood that the object in the image belongs to each of 17 room categories.

Our framework, called CLIPGraphs, trains a Graph Convolutional Network (GCN) [29] to compute embeddings that are used to estimate these object-room affinities. Figure 2 shows the training pipeline. It uses a knowledge graph to encode existing information of human preferences (of room location of objects) for the object categories [2], and incorporates a modified contrastive loss function to compute better latent embeddings of the image and language encoder features provided by CLIP [30] for the nodes of the knowledge graph. The resultant node embeddings model the information about the room location of various objects in the latent space. During inference, the CLIP features generated for any (test) RGB images are processed by the GCN, with the cosine similarity between the embeddings of the rooms and the image providing the desired estimate of object-room affinities. We describe individual components of our framework below.

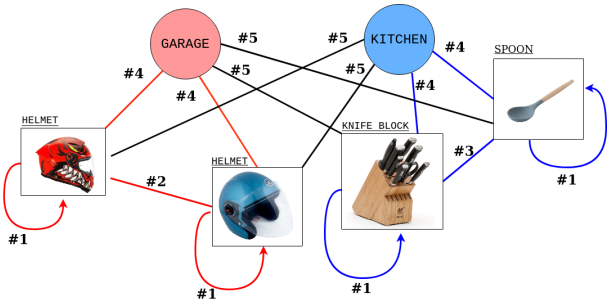


Fig. 1: An illustration of the five types of edges in our knowledge graph. The colored edges denote positive edge weights whereas black ones denote negative weights. The number on the edge denotes the type of edge.

A. Knowledge Graph

Our framework's first step uses the human-annotated preferences included in the Housekeep data [2]. For every object-room pair, 10 human annotators ranked the receptacles in that room based on the likelihood of the object being placed there correctly or incorrectly. For each object-room-receptacle tuple, there are thus 10 opinions that could be positive, negative, or zero. We filter the dataset to ensure good agreement amongst annotators. We calculate the positive (negative) soft scores as the mean of the positive (negative) reciprocal preference of all the receptacles for a given object-room pair. To establish ground truth object-room mappings, we use the object-room-receptacle scores, i.e., we select the room with the highest positive-scored receptacle. Every other room in the domain is assigned the mean negative soft score of receptacles in that room. (See Appendix 3.2)

To use the available annotated information to populate a knowledge graph, we partitioned the IRONA web-scraped dataset into training, validation, and test sets in a ratio of 15:5:10 images per object category. The knowledge graph is instantiated with each image of the training set as a node, along with room names, i.e., there are 268*15 + 17 = 4037 nodes. We then considered five types of edges connecting nodes (see Figure 1): (1) image self edge (edge weight=1); (2) edge between images of same object (edge weight=1); (3) edge between two objects in the same ground truth room; (4) edge between object and its correct room node; and (5) edge between object and its incorrect room nodes. Next, we assigned weights for each type of edge. Weights for edges of type 4 and 5 were based on the object-room soft scores. Edges of type 3 were given a randomly chosen weight between 0.5 to 0.7, and edges of type 1 and 2 were assigned a weight of 1.

Once the basic knowledge graph is created, we initialize the graph's nodes using the pretrained CLIP model's high-dimensional embeddings. Specifically, each object node is initialized with the corresponding CLIP image encoder embedding, and each room node is initialized with the corresponding CLIP language encoder embedding. This is because we want to capture the appearance of the objects and the known association between objects and rooms (based on the large dataset used to train CLIP embeddings). In particular, we considered three pretrained architectures of CLIP in our experiments: Vision Transformer (ViT), ResNet-50, and ConvNeXt. ViT-H/14 [31] is trained on LAION-2B, which is a 2.3 billion subset of the LAION-5B [32] dataset with English captions. ResNet-50 [33] uses OpenAI's pretrained weights [30], and ConvNeXt base [34] is pretrained on LAION-400m [35], which contains 400 million image-text pairs. Once we have associated CLIP embeddings with our knowledge graph's nodes, we move to the next steps of our training pipeline.

B. GCN Training

The next step of training feeds these node embeddings, each of 512 or 1024 dimensions based on the CLIP architecture chosen, and the adjacency matrix (of knowledge graph structure) to a Graph Convolutional Network (GCN) [29] to learn better latent space embeddings of our knowledge graph. GCNs are able to capture non-linear relationships between nodes, and learn from both local and global structures in a graph. As a result, nodes that are more similar are mapped to points that are closer in the latent embeddings space, whereas nodes that are dissimilar are mapped to points further away in the latent space. For example, the output 128-dimensional GCN (object) embedding for a microwave will have a higher cosine similarity with the output 128-dimensional GCN (language) embedding for the kitchen.

An important design decision during training is the choice of the loss function. Prior work has devoted much attention to

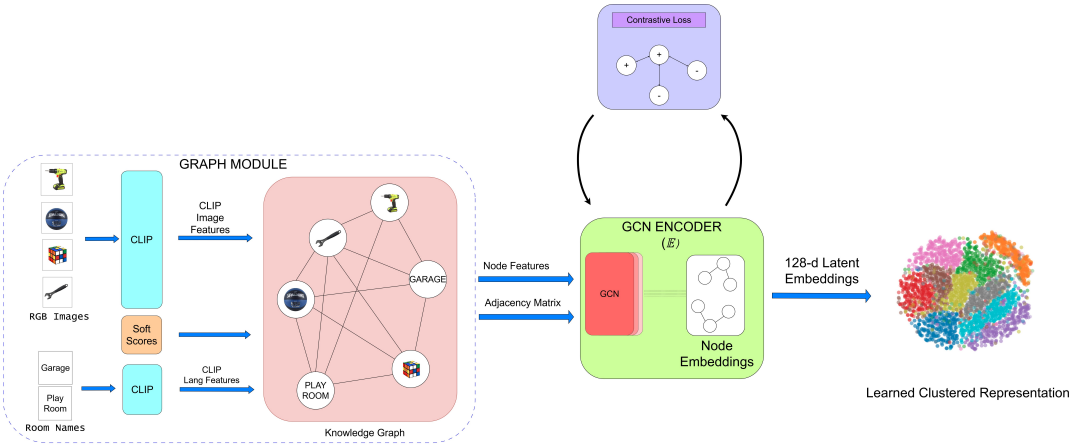


Fig. 2: *CLIPGraphs* constructs a graph module (bottom-left) using CLIP encoders and passes that to a GCN Encoder($\mathbb{E}$) module. The encoder is trained using contrastive loss to create better node embeddings that bring similar embeddings closer. Visualization of final layer activations confirms the formation of well-defined node clusters.

functions such as contrastive loss [36], triplet loss [37], and multi-class N-pair loss [38]. Recent work has demonstrated the benefits of using the loss function introduced in the CLIP-Fields method [39]. We modify this loss function to further leverage the knowledge graph created using the IRONA dataset and human preference annotations.

Loss Function. We train our GCN using a contrastive loss function similar to that described in the CLIP-Fields method [39] with the objective of clustering similar embeddings closer in the latent space and mapping dissimilar embeddings to points that are further away in the latent space. We adapt the basic loss function to our problem formulation and use the additional information of edge weights.

$$L = -e^{-weight_{+\bullet}} \log \left(\frac{e^{(sim_{+\bullet}/T)}}{\sum_{i=1}^{K} e^{(sim_{-\bullet,i}/T)}} \right) \quad (1)$$

where $weight_{+.}$ is the edge weight between the positive node and the anchor node, $sim_{+\bullet}$ is the cosine similarity between the anchor and a positive node embedding, and $sim_{-\bullet,i}$ is the cosine similarity between anchor node embedding and i^{th} negative node embedding. T is a temperature term that is tuned over a validation set. We randomly select one of the 17 rooms as our anchor node, then choose a positive node (for numerator in Equation 1) by picking an object within that room at random, and finally sample k negative nodes for the denominator of the loss function from objects located outside the room. This formulation of the loss function minimizes the distance between the anchor node and the positive node while maximizing the distance with each of the negative nodes, leading to distinct clusters in the graph embeddings. As stated before, the training pipeline is outlined in Figure 2.

C. Testing

Once the GCN has been trained, the pipeline used for testing (i.e., inference) is shown in Figure 3. Similar to the process of training, we compute the CLIP image encoder embedding for the test image, and the CLIP language encoder embedding for the possible rooms. These embeddings are passed to the GCN with only self-edges (in the absence of a knowledge graph) to obtain the output (latent space) embedding for the test image and the possible rooms. Next, similarity scores are calculated between each image node $\vec{x}$ and each of the room(s) $\vec{y}$ using the cosine similarity function. We then average the similarity scores over different images of each object category to get the affinity score between that object category and each of the candidate rooms.

IV. Experimental Setup and Results

A. Experimental Setup

Object-room affinities have predominantly been determined by language-based embeddings or human input in prior work. Since our work combines prior knowledge and multimodal (vision, language) inputs, our chosen baselines were off-shelf language encoders and the GPT-3 LLM. We experimentally evaluated the following hypothesis:

H1: CLIP language embeddings result in better performance than other language encoder embeddings;

H2: Multimodal CLIP embeddings, by themselves, do not perform better than language-based embeddings;

H3: Our framework leads to better performance than (i) the underlying CLIP embedding, (ii) just the language-based encodings, and (iii) the GPT-3 LLM;

H4: Our framework provides robustness to previously unseen noisy backgrounds.

We evaluated **H1-H3** quantitatively and evaluated **H4** qualitatively (see Appendix 7). The performance task was to compute estimates of object-room affinities for all 268 object categories and 17 rooms in the test split of the IRONA dataset. We considered two performance measures: **mAP** (Mean Average Precision) [40], and **Top** k **Hit Ratio**.(See Appendix 4)

B. Quantitative Results

To evaluate **H1**, we first compared two existing language encoder embeddings (RoBerta [41], GloVE [42]) with just the CLIP-based language embeddings with each of the three CLIP architectures. As shown in Table I, the CLIP-based language embeddings (particularly the ViT architecture) resulted in better performance, supporting **H1**.

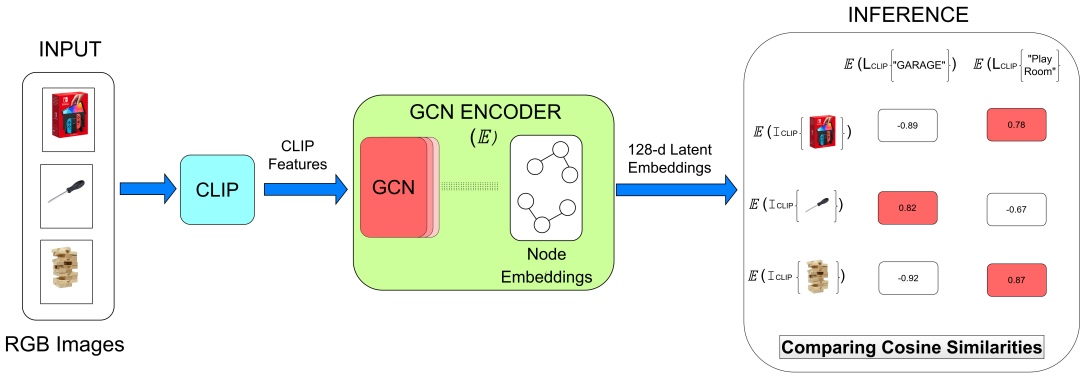


Fig. 3: Our inference pipeline processes input RGB images to generate CLIP image embeddings. These embeddings are processed by the GCN Encoder to produce latent image embeddings. Cosine similarity between these latent embeddings and previously learned room embeddings determines object-room affinities.

Lang Model	Test mAP ⇑	Hit-Ratio ⇑		
		Top-1	Top-3	Top-5
ConvNeXt	0.405	0.223	0.472	0.632
ViT	**0.456**	0.256	**0.576**	**0.710**
RN50	0.453	**0.275**	0.546	0.643
RoBerta	0.417	0.238	0.491	0.636
GloVE	0.148	0.123	0.208	0.278

TABLE I: CLIP-based language embeddings perform better than other popular language encoders; results support **H1**.

UnTuned-CLIP	Test mAP ⇑	Hit-Ratio ⇑		
		Top-1	Top-3	Top-5
ConvNeXt	0.41	0.24	0.46	0.62
ViT	**0.42**	**0.25**	**0.49**	**0.65**
RN50	0.39	0.19	0.45	0.67

TABLE II: Multimodal CLIP embeddings, by themselves, do not improve performance in comparison to CLIP-based language embeddings (see Table I). Results support **H2**.

Next, we compared the performance of the multimodal (vision, language) CLIP embeddings for each of the three CLIP architectures. As shown in Table II, performance is comparable but slightly worse than that in Table I. These results support **H2** and motivate the use of GCNs.

Next, we computed the performance of our architecture, i.e., with GCNs trained using the contrastive loss function and the underlying multimodal CLIP embeddings, with the corresponding results shown in Table III. The best performance was (once again) with the ViT version of the CLIP architecture. Also, performance was substantially better than with the multimodal CLIP embeddings (Table II) or CLIP's language encoder embeddings (Table I). For example, there is an $\approx 40\%$ increase in mAP score compared with not using the GCNs. These results partially support **H3**.

To further explore the benefits of a multimodal CLIP representation, we conducted experiments with our framework, but with GCN embeddings of only the language-based encoding of CLIP. The results reported in Table IV show the benefits of using the multimodal CLIP embeddings.

GCN-CLIP	Test mAP ⇑	Hit-Ratio ⇑		
		Top-1	Top-3	Top-5
ConvNeXt	0.73	0.62	0.81	0.88
ViT	**0.85**	**0.76**	**0.93**	**0.97**
RN50	0.66	0.53	0.75	0.81

TABLE III: CLIPGraphs use of GCN embeddings of multimodal CLIP features and commonsense knowledge results in substantially better performance compared with just the CLIP embeddings in Tables I and II. Results support **H3**.

GCN-CLIP[Lang]	Test mAP ⇑	Hit-Ratio ⇑		
		Top-1	Top-3	Top-5
ConvNeXt	0.64	0.53	0.69	0.76
ViT	**0.77**	**0.68**	**0.77**	**0.83**
RN50	0.59	0.46	0.63	0.74

TABLE IV: Using our GCN-based embedding with just the underlying language-based CLIP encoding results in better performance than in the absence of the GCN embedding, but performance is not as good as when GCNs are used with the multimodal CLIP embeddings (in Table III).

The next experiment compared our framework's performance with the GPT-3 LLM and a state of the art language encoder that provided the best performance among language-based encoders. The results summarized in Table V show that our framework provides substantially better performance by fully leveraging prior commonsense knowledge and multimodal CLIP embeddings. These results strongly support **H3**.

	Test mAP ⇑	Hit-Ratio ⇑		
		Top-1	Top-3	Top-5
Our[GCN-CLIP] [III]	**0.85**	**0.76**	**0.93**	**0.97**
GPT-3	0.66	0.52	0.76	0.81
Best Lang Encoder[I]	0.456	0.275	0.576	0.71

TABLE V: Our framework, with GCN and underlying multimodal CLIP embeddings, substantially improves performance compared with standalone GPT-3 LLM and language-based encoders; hence, the results strongly support **H3**.

V. Conclusion

In conclusion, our proposed framework, CLIPGraphs, provides a novel approach for accurately estimating object-room affinities by leveraging the complementary strengths of commonsense knowledge, data-driven methods, and multimodal embeddings. We demonstrated significant improvement in affinity estimation compared to state-of-the-art methods. Our approach also provides robustness to previously unseen noisy backgrounds.

References

[1] D. Batra, A. X. Chang, S. Chernova, A. J. Davison, J. Deng, V. Koltun, S. Levine, J. Malik, I. Mordatch, R. Mottaghi, M. Savva, and H. Su, "Rearrangement: A challenge for embodied ai," 2020.

[2] Y. Kant, A. Ramachandran, S. Yenamandra, I. Gilitschenski, D. Batra, A. Szot, and H. Agrawal, "Housekeep: Tidying virtual households using commonsense reasoning," in *European Conference on Computer Vision*, 2022.

[3] D. S. Chaplot, D. Gandhi, A. K. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *ArXiv*, vol. abs/2007.00643, 2020.

[4] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, "Zson: Zero-shot object-goal navigation using multimodal goal embeddings," *ArXiv*, vol. abs/2206.12403, 2022.

[5] N. Gireesh, D. A. S. Kiran, S. Banerjee, M. Sridharan, B. Bhowmick, and M. Krishna, "Object goal navigation using data regularized q-learning," *International Conference on Automation Science and Engineering*, pp. 1092–1097, 2022.

[6] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "Clip on wheels: Zero-shot object navigation as object localization and exploration," *ArXiv*, vol. abs/2203.10421, 2022.

[7] N. Gireesh, A. Agrawal, A. Datta, S. Banerjee, M. Sridharan, B. Bhowmick, and M. Krishna, "Sequence-agnostic multi-object navigation," in *IEEE International Conference on Robotics and Automation*, 2023, (to be published).

[8] K. Ellis, D. Hadjivelichkov, V. Modugno, D. Stoyanov, and D. Kanoulas, "Navigation among movable obstacles via multi-object pushing into storage zones," *IEEE Access*, vol. 11, pp. 3174–3183, 2023.

[9] P. Marza, L. Matignon, O. Simonin, and C. Wolf, "Teaching agents how to map: Spatial reasoning for multi-object navigation," *International Conference on Intelligent Robots and Systems*, pp. 1725–1732, 2021.

[10] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon, C. Schuhmann, L. Schmidt, and J. Jitsev, "Reproducible scaling laws for contrastive language-image learning," *ArXiv*, vol. abs/2212.07143, 2022.

[11] E. Wijmans, A. Kadian, A. S. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra, "Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames," in *International Conference on Learning Representations*, 2019.

[12] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On evaluation of embodied navigation agents," *ArXiv*, vol. abs/1807.06757, 2018.

[13] N. Kim, O. Kwon, H. Yoo, Y. Choi, J. Park, and S. H. Oh, "Topological semantic graph memory for image-goal navigation," in *Conference on Robot Learning*, 2022.

[14] O. Kwon and S. Oh, "Image-goal navigation algorithm using viewpoint estimation," *2021 21st International Conference on Control, Automation and Systems (ICCAS)*, pp. 689–692, 2021.

[15] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, M. Savva, A. Toshev, and E. Wijmans, "Objectnav revisited: On evaluation of embodied agents navigating to objects," *ArXiv*, vol. abs/2006.13171, 2020.

[16] A. Das, S. Datta, G. Gkioxari, S. Lee, D. Parikh, and D. Batra, "Embodied question answering," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2135–213 509, 2017.

[17] D. Gordon, A. Kembhavi, M. Rastegari, J. Redmon, D. Fox, and A. Farhadi, "Iqa: Visual question answering in interactive environments," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 4089–4098, 2017.

[18] C. Cangea, E. Belilovsky, P. Lio', and A. C. Courville, "Videonavqa: Bridging the gap between visual and embodied question answering," in *British Machine Vision Conference*, 2019.

[19] L. Weihs, M. Deitke, A. Kembhavi, and R. Mottaghi, "Visual room rearrangement," *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 5918–5927, 2021.

[20] B. Trabucco, G. Sigurdsson, R. Piramuthu, G. S. Sukhatme, and R. Salakhutdinov, "A simple approach for visual rearrangement: 3d mapping and semantic search," 2022.

[21] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10 737–10 746, 2019.

[22] A. Padmakumar, J. Thomason, A. Shrivastava, P. Lange, A. Narayan-Chen, S. Gella, R. Piramithu, G. Tur, and D. Z. Hakkani-Tür, "Teach: Task-driven embodied agents that chat," in *AAAI Conference on Artificial Intelligence*, 2021.

[23] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. D. Reid, S. Gould, and A. van den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3674–3683, 2017.

[24] X. Chen, L.-J. Li, L. Fei-Fei, and A. K. Gupta, "Iterative visual reasoning beyond convolutions," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7239–7248, 2018.

[25] K. Marino, R. Salakhutdinov, and A. K. Gupta, "The more you know: Using knowledge graphs for image classification," *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 20–28, 2016.

[26] X. Wang, Y. Ye, and A. K. Gupta, "Zero-shot recognition via semantic embeddings and knowledge graphs," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 6857–6866, 2018.

[27] G. Sarch, Z. Fang, A. W. Harley, P. Schydlo, M. J. Tarr, S. Gupta, and K. Fragkiadaki, "Tidee: Tidying up novel rooms using visuo-semantic commonsense priors," in *European Conference on Computer Vision*, 2022.

[28] S. Y. Gadre, K. Ehsani, S. Song, and R. Mottaghi, "Continuous scene representations for embodied ai," *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14 829–14 839, 2022.

[29] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.

[30] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, 2021.

[31] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," 2021.

[32] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, P. Schramowski, S. Kundurthy, K. Crowson, L. Schmidt, R. Kaczmarczyk, and J. Jitsev, "Laion-5b: An open large-scale dataset for training next generation image-text models," 2022.

[33] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 770–778, 2015.

[34] Z. Liu, H. Mao, C.-Y. Wu, C. Feichtenhofer, T. Darrell, and S. Xie, "A convnet for the 2020s," in *CVPR*, 2022.

[35] C. Schuhmann, R. Vencu, R. Beaumont, R. Kaczmarczyk, C. Mullis, A. Katta, T. Coombes, J. Jitsev, and A. Komatsuzaki, "Laion-400m: Open dataset of clip-filtered 400 million image-text pairs," 2021.

[36] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine, "Time-contrastive networks: Self-supervised learning from video," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141, 2017.

[37] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[38] K. Sohn, "Improved deep metric learning with multi-class n-pair loss objective," *Advances in neural information processing systems*, vol. 29, 2016.

[39] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. D. Szlam, "Clip-fields: Weakly supervised semantic fields for robotic memory," *ArXiv*, vol. abs/2210.05663, 2022.

[40] S. M. Beitzel, E. C. Jensen, and O. Frieder, *MAP*. Boston, MA: Springer US, 2009, pp. 1691–1692. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_492

[41] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized BERT pretraining approach," *CoRR*, vol. abs/1907.11692, 2019. [Online]. Available: http://arxiv.org/abs/1907.11692

[42] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Conference on Empirical Methods in Natural Language Processing*, 2014.

CLIPGraphs: Appendix

Contents

1 Introduction

Humans have always been fond of arranging stuff. Over the course of the development of society, the norms of how we arrange our houses have changed. Still, there are traces of common associations that we as the Human race have managed to stick to.

When a human being is tasked to find us "Headphones" they would intuitively look for it in places such as "Home Office", "Living Room", and "Entertainment Room". This small thought experiment gives us a good idea that subconsciously we have generated some associations between the objects and the rooms they are supposed to be in. Let's dig a bit deeper.

1.1 Traces of Utility in our decision-making process:

For some objects this mapping can be guided by "utility", for example: if asked to find "an apple" in an unknown house, you know it's a perishable food item, and food items are meant to be consumed but also stored appropriately. Thus you would first look for Kitchen, Pantry, or Dining Room (Whatever is available in your house)
More familiar the object, the less the decision-making time to infer the associated room: One way we could further comprehend our inference is by breaking down our thought process into steps. Let's take an example where the object to be found is an unknown name, you don't know if it's a stationery item. a skin care product? or a species of whale? How would you react in such a situation? a logical way to go about this is by asking questions. Your first thought while encountering an object category you don't know is to get to know its utility. Is it a consumable item? Is it cosmetic? Is it somewhat related to electronics? and so on.. Once you are known of its utility, you have an estimate of where different known objects belonging to similar utility are found. This gives you a reasonable amount of confidence in the room you then choose to go search for your unknown object.
This thought experiment establishes 2 facts:

- By successfully answering some of these questions we humans can infer where that object should be found in its correct orientation in *any* house.
- We humans not only have commendable Object-Room mappings in our brains, but we also leverage the object-object relationships developed on the basis of the co-occurrence of these objects, which is again influenced by their common utility.

In this work, we aim to use a graph network to cluster commonly occurring household objects into room categories. The basic intuition behind using a Graph Convolutional Network is to make use of Object-Room Relationships as well as Object-Object Relationships to generate latent representations that are indicative of the categorization of objects into rooms on the basis of their common feature(utility).

Graph Convolutional Networks not only use node features but also use the message passing mechanism between neighboring nodes to generate node representations that take into account the information of neighboring nodes in addition to the node features.

We make use of the Human Preference Dataset [1] to get to know what are some most prevalent human-accepted correct object-room mappings and further use it along with multimodal CLIP [2] features to learn such utility-based clustering using a graph network.

2 Datasets

We used object,room categories and Human annotations for those as provided by Housekeep [1]. Further we curated a visual counterpart to the 268 object categories.

2.1 Human Preference Dataset

This dataset was curated by Housekeep [1] to understand how humans prefer to put everyday household objects in an organized and disorganized house. They ran a study on Amazon MTurk with 372 participants. Here, for a given object-room pair, they asked each participant to classify the available receptacles of that room into 3 categories:

- *misplaced*: subset of receptacles where the object is found in *un-tidy* houses
- *correct*: subset of receptacles where the object is found in *tidy* houses
- *implausible*: subset of receptacles where the object is unlikely to be found either in a *tidy* or *un-tidy* house

They further asked them to rank all the receptacles present in that room. For how they collected this data, filtered it out, and used it for Scene Rearrangement, we guide the interested readers to their paper.

2.2 IRONA Dataset

We created a new dataset by scraping 30 images per object category. These were white background catalog images of the same 268 object categories that Housekeep [1] used to benchmark the Scene-Rearrangement task.

2.2.1 Rooms

We consider the same 17 room categories as used by Housekeep [1]:

Rooms		
bathroom	bedroom	childs_room
closet	corridor	dining_room
exercise_room	garage	home_office
kitchen	living_room	lobby
pantry_room	playroom	storage_room
television_room	utility_room	

2.2.2 Object Categories

The list of 268 object categories we used can be accessed here

3 Knowledge Graph Creation

An important step in learning Object-Room Affinities using our proposed multimodal model was knowledge graph creation using our IRONA Dataset and Housekeep Human Preference Dataset [1]

Figure 1: Representative Image Of Our Web Scraped Dataset ; 1 image per object category

3.1 Nodes

There are currently 2 types of nodes in our knowledge graph

- *Room Nodes:* Since we have 17 room categories, there are 17 such nodes. For our proposed method, the node features for these nodes are generated using the CLIP language encoders.
- *Object Nodes:* For each of the 268 object categories, 15 images are chosen for training, and for each of the 268*15 nodes, we generate node features using the CLIP image encoders[2]. (In our proposed method) However, for baseline comparisons, we also create CLIP Language Embeddings for these 268 Object Categories.

3.2 Edge Weights

For assigning edge weights to various edges between our nodes, we use the ranks provided by Housekeep Human Preference Dataset for each *object-room-receptacle*[1] combination. We further calculate soft scores using algorithm 1 which takes these ranks as input.

[1]"receptacle" was a categorization used by Housekeep [1] to define 128 flat horizontal surfaces in a household where objects can be found - misplaced or correctly placed

For each object-room-receptacle combination, 10 annotators were given just 3 categories to classify the combinations. Therefore for each object-room-receptacle, there could either be a majority of *positive ranks*, *negative ranks*, or *implausible ranks*:

1. For any object-room-receptacle combination if a majority of annotators (> 5) gave positive ranks[2]; we calculate a *positive score*

2. For any object-room-receptacle combination if a majority of annotators (> 5) gave negative ranks[3]; we calculate a *negative score*

3. However, if *implausible* ranks were in majority then we need to modify those a bit. According to Housekeep [1] an *implausible* combination was such a combination that could neither occur in an *untidy* house nor in a*tidy* house. Therefore it accounts for the maximum negative object-room-receptacle affinity, thus we assign -1 i.e. max possible negative score to such object-room-receptacle pairs.

Thus finally we would have a dictionary that records whether the majority opinion for every object-room-receptacle was *positive*, *negative* or *implausible*(-1)

Algorithm 1 Generating ORR Positive Soft Scores

```
for each object in objects do
    for each room in rooms do
        ranks ← object − room − receptacle − ranks
        score_dict ← {}
        for each receptacle in the room do
            combination_score ← []
            for each rank given for the combination do
                if rank > 0 then
                    combination_score.append(1/rank)
                end if
            end for
            if len(combination_score) >= 5 then
                score_dict[combination] = sum(combination_score)/max_len_pos[4]
            else
                score_dict[combination] = 0
            end if
        end for
    end for
end for
```

3.2.1 Correct Object-Room Mappings

To get the correct object-room mappings, we compare for a given object which object-room-receptacle has the highest *positive* score. We assign that room as the correct object-room mapping.[5]

[2]A receptacle with "+1" rank is a more appropriate for an object as compared to a receptacle with a "+2" rank for the same object-room pair.

[3]A receptacle with "-1" rank is a receptacle where humans are more prone to keep objects in an *untidy* state of the house as compared to a receptacle with a "-2" rank for the same object-room pair.

[4]For a given object, max_len_pos/neg/imp is the maximum number of annotators that gave positive/negative/implausible ranks across all room-receptacle pairs.

[5]Our future extension would be to extend this mapping to top-K correct rooms

Algorithm 2 Generating ORR Negative Soft Scores

for each object in objects **do**
  for each room in rooms **do**
    $ranks \leftarrow object - room - receptacle - ranks$
    $score\_dict \leftarrow \{\}$
    for each receptacle in the room **do**
      $combination\_score \leftarrow []$
      $min\_neg\_rank = min(all\ ranks\ for\ the\ ORR\_combination)$
      for each rank given for the combination **do**
        if $rank < 0$ **then**
          $rank = rank + min\_neg\_rank + 1$
          $combination\_score.append(1/rank)$
        end if
      end for
      if $len(combination\_score) >= 5$ **then**
        $score\_dict[combination] = sum(combination\_score)/max\_len\_neg$[4]
      else
        $score\_dict[combination] = 0$
      end if
    end for
  end for
end for

Algorithm 3 Generating ORR Implausible Soft Scores

for each object in objects **do**
  for each room in rooms **do**
    $ranks \leftarrow object - room - receptacle - ranks$
    $score\_dict \leftarrow \{\}$
    for each receptacle in the room **do**
      $combination\_score \leftarrow []$
      for each rank given for the combination **do**
        if $rank == 0$ **then**
          $combination\_score.append(-1)$
        end if
      end for
      if $len(combination\_score) >= 5$ **then**
        $score\_dict[combination] = sum(combination\_score)/max\_len\_imp$[4]
      else
        $score\_dict[combination] = 0$
      end if
    end for
  end for
end for

3.2.2 Incorrect Object-Room Mappings

Once we've created the correct object-room GT mapping, it still leaves the other 16 rooms' edge weights to be allotted. So for each such object-room pair; there can be 3 cases:

1. **All/majority positive receptacles**:we assign $-\epsilon$ as edge weights
2. **All/majority negative receptacles**: we assign the mean of all the negative scored receptacles
3. **All/majority implausible receptacles**: we assign the mean of all the negative scored receptacles. (since we had already assigned implausible receptacles with -1)

For example, for a particular object-room-receptacle combination, the calculation of positive soft scores is shown:

$$\begin{aligned}
\text{knife-bottom_cabinet ranks:} &\quad -1, -3, 0, 1, 3, 2, -2, 5, -1, 4 \\
\text{Filter the negative ranks:} &\quad 1, 3, 2, 5, 4 \\
\text{Take the reciprocal of ranks:} &\quad \frac{1}{1} + \frac{1}{3} + \frac{1}{2} + \frac{1}{5} + \frac{1}{4} = 2.367 \\
\text{Calculate the mean of reciprocal ranks:} &\quad 2.283/5 = 0.45
\end{aligned}$$

For each object, the ground-truth room is decided by choosing the room containing the highest-positively scored receptacle for that object. Every other room in the domain is assigned the mean negative soft score(for a given object-room pair) of all the receptacles present in that room.

3.3 Overall

Table 1 summarizes various information about our knowledge graph that is being used for training purposes.

Statistics About our Knowledge Graph	**Value**
#Nodes	4020 Object Images Nodes & 17 Room Nodes
#Edges	7,66,649
Self Loops	Yes
Train Images	268*15 Images
Test Images	268*10 Images
Val Images	268*5 Images
Types of edges	weighted undirected

Table 1: Statistics for the Knowledge Graph created using the web scraped dataset

3.4 Comments on node features

For our proposed method we used 3 architectures of CLIP as given by OpenCLIP [3]: ViT-H/14, and RN50 have dimensionality of 1024, and ConvNeXt-base has a dimensionality of 512 features. These were chosen taking into consideration the datasets they were trained on and their performance in other embodied AI tasks.

4 Loss Function Ablations

We considered two performance measures:

1. **mAP:** The mean average precision (mAP) is the average of precision scores at different recall values for each instance of an object category, and the mean over all the object categories. For a given object, Average Precision is calculated by:

$$AP = \sum_{n} (R_n - R_{n-1}) P_n$$

   Where, P_n and R_n are Precision and Recall values at the n^{th} threshold. Taking a mean over all the objects, gives us the final mean Average Precision.

2. **Top k Hit Ratio:** The average fraction of object categories for which the ground truth correct room was among the Top k estimates from our framework.:

$$\text{Top-k HR} = \frac{1}{|O|} \sum_{o \in O} \mathbb{1}(R_o \cap T_o \neq \emptyset)$$

   where O is the set of objects, R_o is the set of top-k rooms recommended for object o, T_o is the ground truth room for object o, and $\mathbb{1}$ is the indicator function that returns 1 if the condition is true and 0 otherwise.

4.1 vs existing loss functions

Loss Fn	mAP
Margin[4]	0.371
Triplet[5]	0.51
Ours	**0.85**

4.2 Hyperparameters

We chose a modified version of the loss function used by [6]. The effectiveness of our loss function depended on our sampling technique. We tuned

1. Batch Size
2. Number of negative nodes sampled per batch for each epoch.

4.2.1 Batch Size

We experiment on the number of batches of anchor, positive and negative nodes sampled per epoch for loss computation. As evident from the figure, we achieve the best performance at batch size = 15. Thus we continue our experiments with this value. The comparison is shown in Figure 3

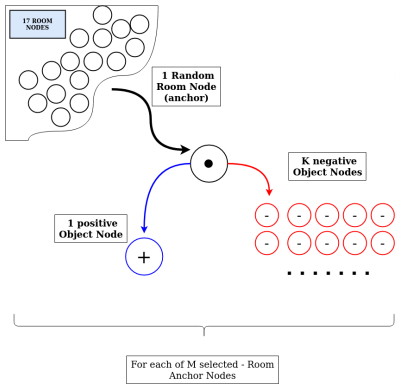


Figure 2

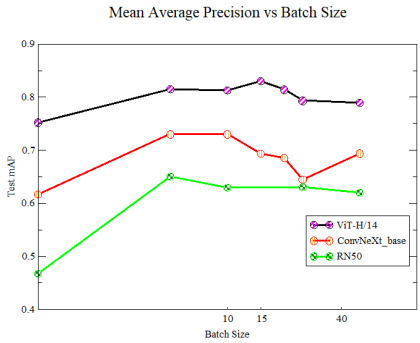


Figure 3: Variation of testing metrics with number of batches used for contrastive loss

4.2.2 Number of negatives

We compare the performance of our model by changing the number of negative nodes sampled per anchor point for computing the contrastive loss. The results are compiled in figure 4. As evident from the figure, the best performance was observed at negative samples = 40. Thus we fix the total number of negatives per anchor as 40.

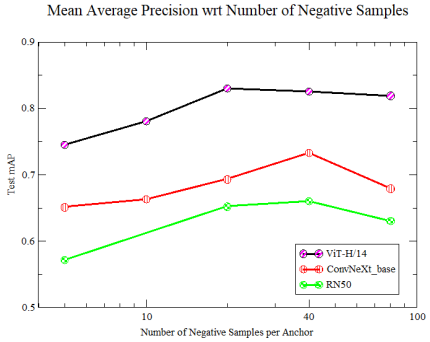


Figure 4: Variation of testing metrics with different numbers of negative samples used per anchor.

5 Our Model: GCN

Table 2 shows the hyperparameters used for the model.

#	**Hyperparameter**	**Value**
1	Node Feature Size	512(ConvNeXt) / 1024 (Others)
2	Output Node Embedding	128
3	GCN[7] Layers	3
4	Learning Rate	10^{-3}
5	Learning Rate Schedule	StepLR: step size=1000 , $\gamma = 0.25$
6	Temperature	0.01
7	Batch Size [Loss]	15
8	Negatives Per Batch	40
9	Epochs	5k

Table 2: Hyperparameter choices for our Graph Based Network to learn latent representations of CLIP Visual Encoder Features

6 Baseline Predictions

6.1 GPT-3

To obtain baseline results, we query GPT-3 to rank the 17 rooms for each object like this:

> Which of the following rooms would you expect to find a **knife block** in? Please rank in decreasing order of likelihood: bathroom, bedroom, child room, closet, corridor, dining room, exercise room, garage, home office, kitchen, living room, lobby, pantry room, playroom, storage room, television room, utility room.

We use such queries to obtain room rankings for each of 268 objects and use that to obtain baseline mAP and hit ratio for GPT-3, and the results are shown in Table 3

	Test mAP ⇑	Hit-Ratio ⇑		
		Top-1	Top-3	Top-5
GPT-3	0.66	0.52	0.76	0.81

Table 3: Results for object-room mappings based on queries to GPT-3

The room rankings for each object category by querying GPT-3 is compiled in the file: gpt_pred.txt

6.2 Language Encoders

The code for generating the statistic metrics as well as the object-room mappings corresponding the each language baseline is available at CLIPGraphs GitHub Repository.

6.3 Our Predictions

Similar rankings for every object category based on our model can be generated by running the script available at: CLIPGraphs GitHub Repository

7 Qualitative Results

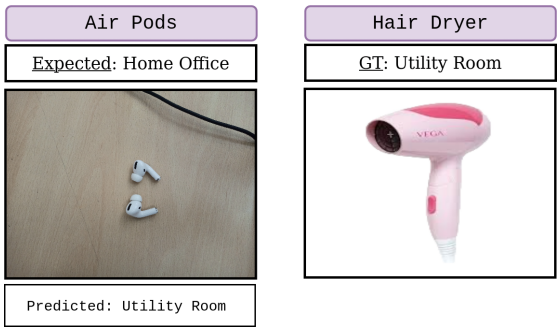
Our model was trained on clean white background images to predict the most appropriate room. To test its performance on real-world images we ran a few runs on a mobile phone by clicking a photograph and feeding it to our model. We fed our model images of objects out of the training set to see its generalization capabilities, we report some success and failure cases for the same.

7.1 Success Cases

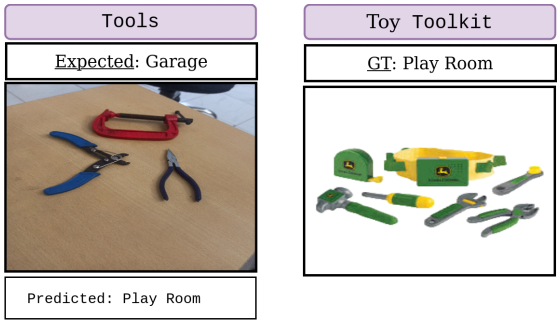
Figure 5: Success of the model on unseen object category images (absent in the training set)[6]

7.2 Failure Cases

Figure 6 shows some specific limitations of our models. In Figure 6a when we input the image of earpods (not a part of the training set) to the model, the top output prediction is utility room. The model confuses the images of earpods to similar image of hair dryer, since it does not contain knowledge of scale. Figure 6b shows the limitation of the model in identifying real objects with similar category of toys.



(a) Failure to determine correct room for object category *earpods* (not in our train set) because it was structurally similar to *hair dryer* category that was in our training set



(b) Failure with composite object categories; *tools* was not a category in our training set, but they were incorrectly associated with the *play room* because they were structurally similar to the *toy toolkit* that was in the training set.

Figure 6: Failure cases of our model

[6]Since these categories were unseen thus we didn't have any ground truth available, thus we mention the *expected* room on the basis of our commonsense

Apart from testing our model on unseen categories, we also try our model's generalization capabilities on real-world noisy images. For this experiment, we generated 4 different scenarios. The representative results for 3 object categories are shown below:

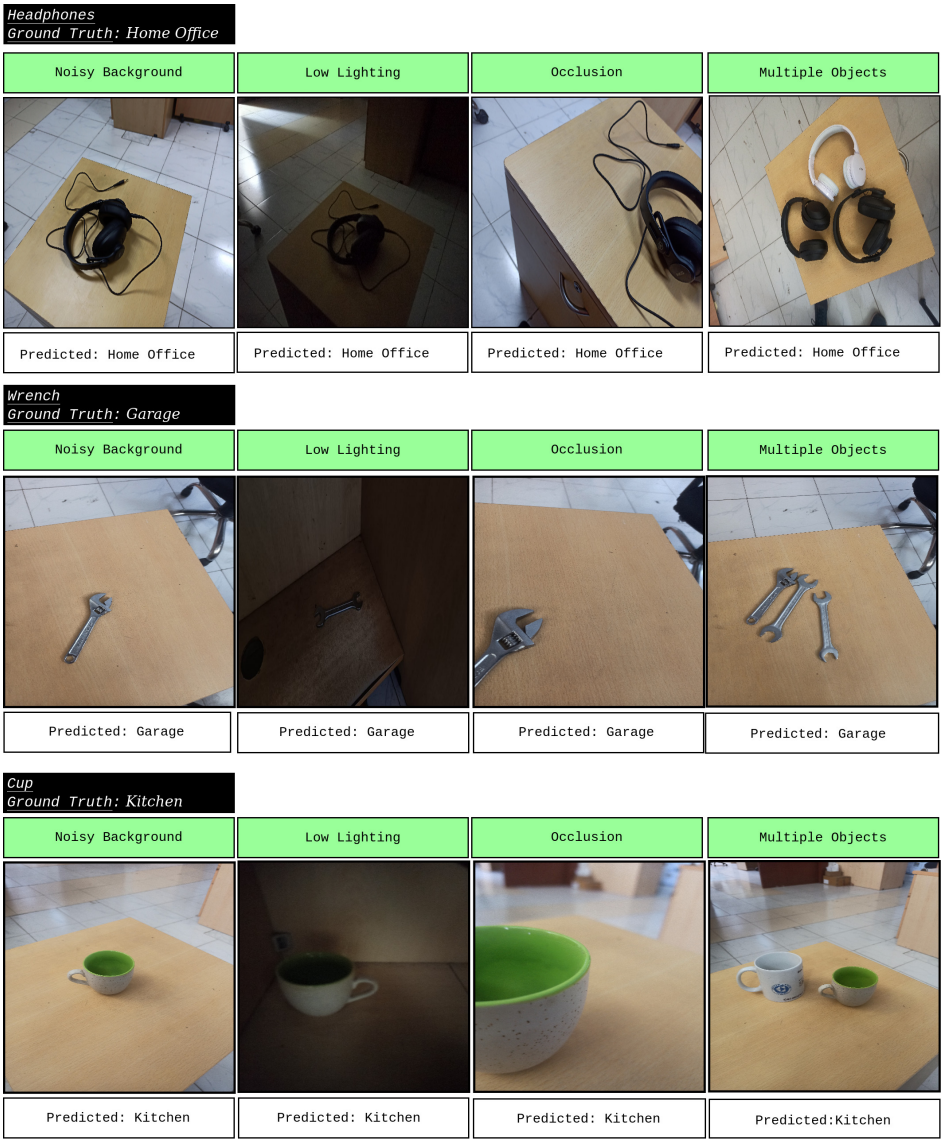


Figure 7: Qualitative result of using our framework with images of previously *seen* objects but in noisy backgrounds. In each case, the object's room association was estimated correctly demonstrating broad applicability of our method

8 Some additional Plots

8.1 t-SNE

Using t-SNE to visualize the high-dimensional embeddings, we observe initial random nodes in Figure 8, where each color represents objects of a unique room. As the model trains, we observe clustering in the embedding space to cluster objects belonging to the same room [Figure 9]

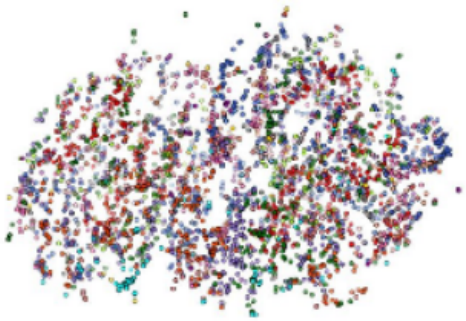


Figure 8: Untrained TSNE

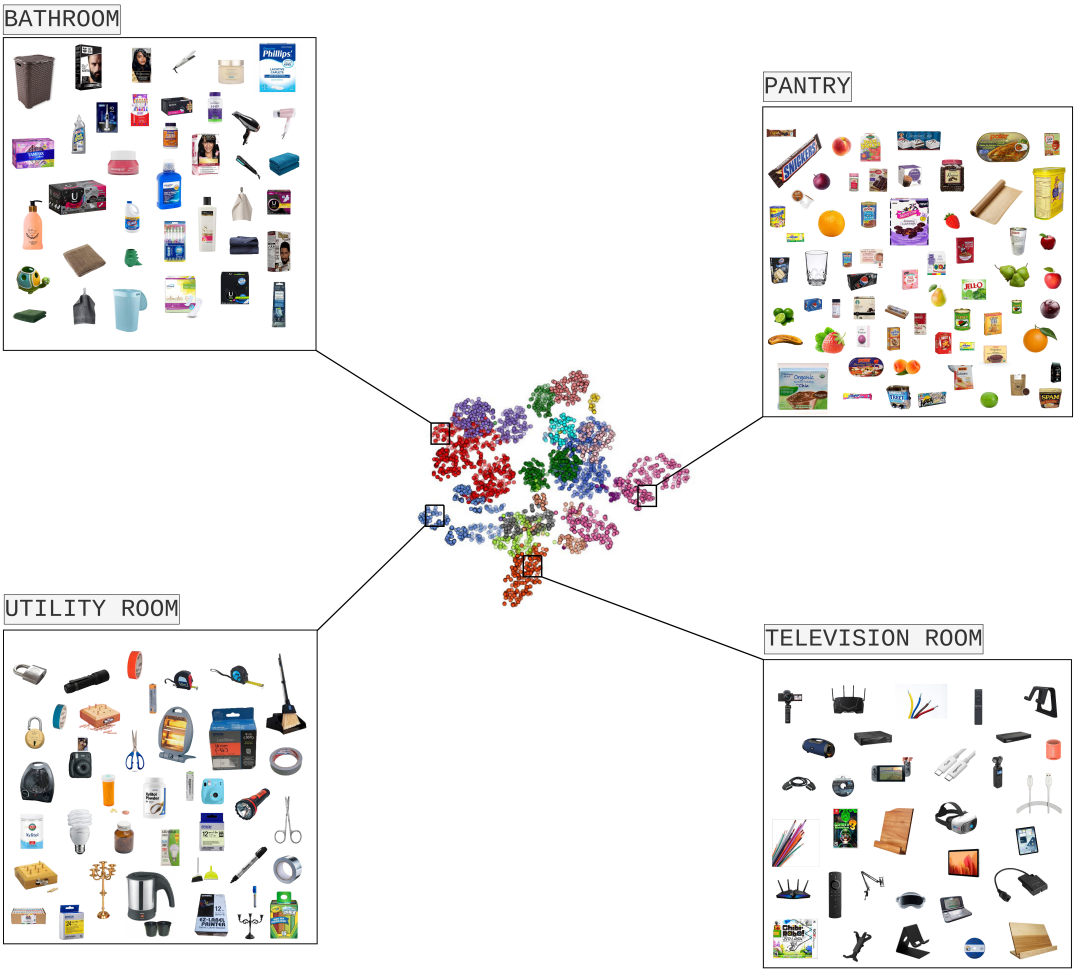


Figure 9: t-SNE visualization of our embeddings on the test split of the Web Scraped Dataset. The boxes show images of objects belonging to the same rooms getting clustered [a]

[a] For a more interactive view of this figure, check out our website: https://clipgraphs.github.io

Figure 10: Image showing objects that got clustered in the t-SNE corresponding to "Bathroom" room category

Figure 11: Image showing objects that got clustered in the t-SNE corresponding to "Pantry" room category

References

[1] Y. Kant, A. Ramachandran, S. Yenamandra, I. Gilitschenski, D. Batra, A. Szot, and H. Agrawal, "Housekeep: Tidying virtual households using commonsense reasoning," in *European Conference on Computer Vision*, 2022.

[2] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever, "Learning transferable visual models from natural language supervision," in *International Conference on Machine Learning*, 2021.

[3] M. Cherti, R. Beaumont, R. Wightman, M. Wortsman, G. Ilharco, C. Gordon, C. Schuhmann, L. Schmidt, and J. Jitsev, "Reproducible scaling laws for contrastive language-image learning," *ArXiv*, vol. abs/2212.07143, 2022.

[4] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 815–823, 2015.

[5] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, and S. Levine, "Time-contrastive networks: Self-supervised learning from video," *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1134–1141, 2017.

[6] N. M. M. Shafiullah, C. Paxton, L. Pinto, S. Chintala, and A. D. Szlam, "Clip-fields: Weakly supervised semantic fields for robotic memory," *ArXiv*, vol. abs/2210.05663, 2022.

[7] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," in *International Conference on Learning Representations (ICLR)*, 2017.