


# TaskCLIP: Extend Large Vision-Language Model for Task Oriented Object Detection

Hanning Chen<sup>1</sup><sup>\*</sup>, Wenjun Huang<sup>1</sup>, Yang Ni<sup>1</sup>, Sanggeon Yun<sup>1</sup>, and Yezi Liu<sup>1</sup>  
Fei Wen<sup>2</sup> Alvaro Velasquez<sup>3</sup> Hugo Latapie<sup>4</sup> Mohsen Imani<sup>1</sup>

<sup>1</sup> University of California Irvine, Irvine, CA 92697, USA

<sup>2</sup> Texas A&M University, College Station, TX 77843, USA

<sup>3</sup> University of Colorado Boulder, Boulder, CO 80303, USA

<sup>4</sup> Cisco, San Jose, CA 95134, USA

{hanningc, m.imani}@uci.edu

**Abstract.** Task-oriented object detection aims to find suitable objects for performing specific tasks. As a challenging task, it requires simultaneous visual data processing and reasoning under ambiguous semantics. Recent solutions are mainly all-in-one models. However, the object detection backbones are pre-trained without text supervision. Thus, to incorporate task requirements, their intricate models undergo extensive learning on a highly imbalanced and scarce dataset, resulting in capped performance, laborious training, and poor generalizability. In contrast, we propose TaskCLIP, a more natural two-stage design composed of general object detection and task-reasoning object selection. Particularly for the latter, we resort to the recently successful large Vision-Language Models (VLMs) as our backbone, which provides rich semantic knowledge and a uniform embedding space for images and texts. Nevertheless, the naive application of VLMs leads to suboptimal quality, due to the misalignment between embeddings of object images and their visual attributes, which are mainly adjective phrases. To this end, we design a transformer-based aligner after the pre-trained VLMs to recalibrate both embeddings. Finally, we employ a trainable score function to post-process the VLM matching results for object selection. Experimental results demonstrate that our TaskCLIP outperforms the DETR-based model TOIST in both accuracy (+6.2%) and efficiency.

**Keywords:** Vision-Language Models · Task-oriented Object Detection

## 1 Introduction

Object detection algorithms have seen tremendous progress on datasets like COCO [29] and Pascal VOC [9], where they identify object instances of pre-defined categories in a scene [13, 17, 49]. However, in real-world applications, artificial intelligence is expected to handle more specific “*task-oriented object detection*”. To this end, researchers propose a new benchmark named COCO-Tasks [41] that is much more challenging than traditional object detection tasks.

---

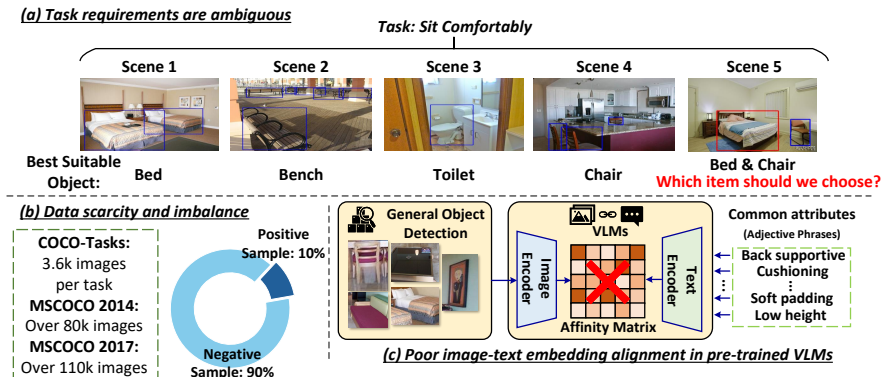
\* This work was done during an Cisco internship.

Fig. 1(a) highlights that a simple, yet practical task never confines itself to a closed and predefined set of objects (or text prompts consisting of nouns). The inputs are in the form of affordance (e.g., “*Sit Comfortably*”), conditioning the selection of objects on their capability to fulfill a task. In particular, this presents a major challenge for prior object detection algorithms. The affordability as a prompt is intrinsically ambiguous, for example, the best suitable objects range from “*Bed*” to the seemingly unrelated “*Toilet*” for the same task “*Sit Comfortably*”, depending on different scenes.

Current solutions follow either a two-stage or a single-stage design to tackle this challenge. The former starts with regular object detection, followed by task-driven object selection, whereas the latter aims to achieve both sub-tasks with one single model. The two-stage design proposed in [16, 41] applies a ranking of objects preference on top of objects extracted by traditional detectors. More recent one-stage designs [26] mainly resort to advanced object detection models such as the transformer-based DETR [3] as the backbone. Work [44] further leverages Large Language Models (LLMs) to generate intermediate visual affordance features, which close the semantic gap between affordances that are mainly verb phrases and object types that are nouns.

However, as intriguing as the all-in-one model sounds, the costs and difficulties behind it cannot be ignored. As pointed out by the previous solution [3], transformers are considered data hungry, while obtaining data sets such as COCO-Tasks remains extremely difficult. As shown in Fig. 1(b), the dataset also suffers from severe class imbalance, since only a handful of objects are relevant to the tasks concerned and the rest are labeled as negative samples. In addition, all prior works require an explicit input of task-specific information to the object detection or selection model [26, 41], fundamentally limiting their robustness against dataset domain shift or even slight changes in the affordance prompts. For instance, altering the task name to synonymous terms will cause a notable decline in the model’s performance, even if the underlying characteristics of the task remain unchanged. **This brings us to the question of whether it is optimal to train these specialized one-stage models for specific tasks from scratch, considering the limited data available in this domain. It also raises another important issue: how can we design more versatile computer vision models capable of addressing a variety of tasks?**

Our pursuit of solutions takes cues from the reasoning process of human beings, which is naturally multi-stage. Initially, they identify potential instances of objects and extract their visual features. During the process, humans parse the task, discerning the visual or functional attributes required of the candidate objects that are essential for task completion. Subsequently, they evaluated instances against these tasks-relevant attributes, selecting instances that align the most closely with the task requirements. Crucially, humans do not simply seek instances directly matching the task; instead, they engage in a step-by-step reasoning process that takes advantage of the common features of suitable objects as clues. This is similar in spirit to existing two-stage solutions. However, their

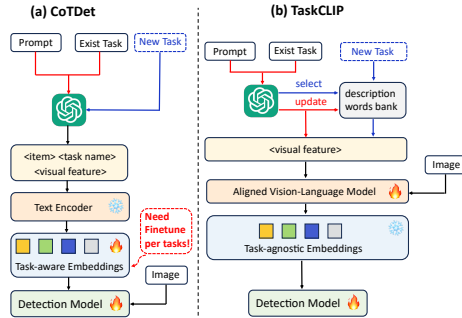


**Fig. 1:** (a) Ambiguity of task-oriented object detection. (b) Data scarcity and imbalance: suitable objects only take up a tiny portion of the total training samples. (c) Embedding misalignment when directly applying VLMs for object selection.

performance is suboptimal mainly due to the bottleneck in the second stage, i.e., the association of object visual features and the semantics in task affordance.

In this paper, we present TaskCLIP, a two-stage task-oriented object detection algorithm that tackles challenges in prior algorithms all at once, motivated by the recent success of large-scale Vision-Language Models (VLMs) on zero-shot transfer for language-supervised vision tasks. In the first stage, we perform general object detection [12, 39] for each scene. Meanwhile, we parse the task affordance, via LLMs [2, 36], into several adjectives that describe common attributes of suitable objects. In our second stage, VLMs like CLIP [38] and Flamingo [1] serve as pre-trained backbones, which match image and text embeddings. The resulting affinity matrices then guide the selection of suitable objects. They enable the alignment of visual and text features by constructing a unified and high-quality embedding space, thereby resolving the bottleneck in current two-stage designs. More importantly, we avoid the complex customization, costly training, and poor generalizability of all-in-one models, where all parameters are learned from scratch and tuned specifically for a few tasks in COCO-Tasks with limited training samples. **Given that TaskCLIP addresses the task through the selection of items meeting visual characteristics, our model can be utilized for other tasks not included in the training group, provided they share the same fundamental attributes.**

However, incorporating VLMs like CLIP into our two-stage design is not trivial. Standard multimodal VLMs, while proficient in alignment, fall short of the reasoning capabilities required to select the most suitable objects. On one hand, these models primarily focus on aligning nouns with visual embeddings, while abstract concepts, such as **phrases with adjectives**, are poorly supported (as shown in Fig. 1(c)). On the other hand, large VLMs are trained on rich and diverse natural language annotations that contain not only nouns for objects. In response, we propose a re-alignment mechanism that blends seamlessly into the end-to-end model training. It reinstates the multimodal VLM’s ability to



**Fig. 2:** (a) CoTDet [44] workflow. (b). TaskCLIP (this work) workflow.

encode adjectives within the shared embedding space and accurately associate them with the related visual embeddings.

In short, we argue that a two-stage model is indeed a more natural, generalizable, efficient, and effective design. Our contributions are summarized as follows:

- Instead of training a multi-modal model from scratch, we propose the first VLM-based, two-stage, task-oriented object detection framework, leveraging the extensive semantic information from vision-language pretraining and its capability to support a calibrated joint vision-text embedding space.
- We design a transformer-based aligner module to recalibrate the vision and text embeddings from VLMs, ensuring good matching between object visual features and adjective phrases from object common attributes.
- To further mitigate the high false negative rate caused by imbalanced training samples, we specifically design a select-by-grouping mechanism. It ensures that all instances of the most suitable object are selected, by fully making use of the available knowledge from general object detectors.
- Our framework provides natural task-driven reasoning and high-quality object detection, while also demonstrating higher training efficiency than previous works. In contrast to earlier DETR-based models, TaskCLIP attains similar accuracy on the COCO-Tasks dataset, yet exhibits significantly improved training efficiency and better generalizability.

## 2 Related Work

### 2.1 Task-oriented Object Detection

This focuses on identifying the most suitable objects within an image to accomplish a specific task, such as serving wine. It requires the ability to reason and select objects based on a deep understanding of the task and the visual scene. Thus, task-oriented object detection is much more complex than simple visual object detection [7, 8, 30, 40] and scene understanding [5, 6, 20, 21, 42, 48]. The first task-driven object detection dataset, named COCO-Tasks, is proposed

by a study [41] based on MS COCO 2014 [29]. It also introduced a two-stage framework as a benchmark, which initially employs an existing object detection network to identify potentially suitable items, followed by gated graph neural networks (GGNN) [27] to model global inter-object relationship for object preference ranking. Subsequent works, TOIST and CoTDet [26, 44], are based on DETR [3, 24, 52] and can be considered as single-stage frameworks. Compared to GGNN, the self-attention module in DETR-based models is more effective in terms of integrating text and visual information. However, training DETR-based models for task-oriented object detection faces multiple challenges due to the complex model structure and the large number of parameters. In contrast, our TaskCLIP resorts to the pertaining vision language and offers a more efficient alternative. A further problem is that while previous DETR-based models include text modalities, they do not take into account the model’s ability to generalize. Consider CoTDet [44], as shown in Figure 2.(a). For each task, after generating task-specific visual affordance and visual feature embeddings using the LLM (ChatGPT) and the text encoder (RoBERTa [31]), the model requires fine-tuning of these embedding vectors. A change in the task name results in a significant decrease in detection accuracy.

## 2.2 Vision Language Model

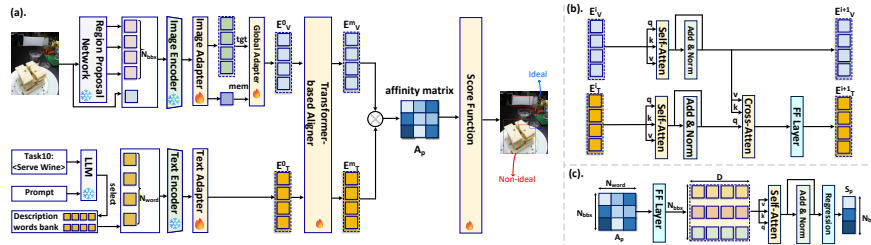
Extending pre-trained VLMs to support new applications has demonstrated their success in recent years. Among the most successful efforts, Flamingo [1], OpenAI CLIP [38], and OpenCLIP [4] have exhibited impressive performance in handling image-text matching due to their deep semantic knowledge and a comprehensive understanding of content that spans both modalities. These models have been applied successfully in downstream applications such as object detection [43], action recognition [45], image captioning [35], dense prediction [51], disease prognosis [34], and semantic segmentation [25, 28, 46]. However, existing CLIP-based algorithms primarily focus on matching image patches with nouns in text. Thus, additional calibration is necessary to facilitate the matching between image patches and visual attributes that are adjective phrases.

## 3 Method

### 3.1 Task visual attributes preparation

To close the semantic and reasoning gap between task affordance and suitable object class, we utilize an LLM to produce task-relevant visual common attributes, as is shown in Fig. 2(b). In Table 1.(a), we delineate the prompts used to guide the LLM in generating the appropriate visual attributes for each task. To illustrate this process, we consider task 10 (**Serve wine**) as an example. For task 10, the procedure unfolds as follows:

(1). We initiate the process by querying the LLM: “*Please list items to solve the task of **Serve wine**?*” The response generated by the LLM encompasses several real-life items, such as “wine glass”, “cup”, “stemware”, and others.



**Fig. 3:** TaskCLIP model architecture: (a) The overall framework of the design. (b) The architecture of the aligner module. (c) Detailed design of the score function

**Table 1:** TaskCLIP LLM prompt.

(a) Description words bank preparation	
<b>User:</b>	Please list items to solve the task of <Task Name>?
<b>LLM response:</b>	[item1, item2, ...]
<b>User:</b>	Summarize visual features of the [item1,item2,...] for <Task Name>.
<b>LLM response:</b>	[feature1, feature2, ...]
(b) Synonym task generation	
<b>User:</b>	Please generate a synonym for <Task Name>
<b>LLM response:</b>	<Task Name Synonym>
(c) Visual features selection	
<b>User:</b>	Please select 20 visual feature words from words bank for <Task Name>
<b>LLM response:</b>	[feature1, feature2, ...]

(2). Building on this initial response, we proceed to the second prompt: “*Summarize the visual features of the [wine glass, cup, stemware, ...] for **serve wine**.*” It is designed to guide the LLM in summarizing the visual characteristics of the mentioned items specifically in the context of the “serve wine” task.

(3). Ultimately, we obtain the common visual attributes of objects that can afford this task from LLM responses, such as “narrow rim” and “transparency”. In total, we generate  $N_{\text{word}} = 20$  common attributes.

(4). We save these generated common attributes in the description words bank. As depicted in Figure 2, when new tasks arise, the LLM will pick the most suitable attribute from the description words bank. Since TaskCLIP is trained by aligning item images with visual feature description words, even if the task name changes, as long as the intrinsic attributes of the new task can be articulated using the words from the bank, our model remains effective. Hence, TaskCLIP exhibits superior task generalizability compared to prior works.

### 3.2 Text and visual embedding vector generation

Meanwhile, we generate  $N_{bb,x}$  bounding boxes for all objects in the scene as is shown in Fig. 3(a). During the training process, we directly use ground truth bounding boxes, whereas, for inference, we use a pre-trained object detection network (such as Faster R-CNN [12] or YOLOv8 [23]). Notice that in the basic setting of our framework during inference, the object detection network is only responsible for generating bounding boxes for all items in the image. We will introduce a grouping mechanism in Section 3.6 to further leverage object

classification information and guide our model’s final decision. Based on the coordination of the bounding box, we extract  $N_{bbox}$  image patches. After generating  $N_{word}$  common visual attributes and  $N_{bbox}$  image patches, we pass them into pre-trained VLMs (such as OpenAI CLIP [38]) to generate text and image embeddings as is shown in Fig. 3(a). Here we use  $L_{word}$  and  $L_{bbox}$  to represent the lists of visual feature words and image patches. The lengths of the lists are  $N_{word}$  and  $N_{bbox}$ , respectively. The computation process is summarized as:

$$\mathbf{E}_V^0 = CLIP_{image}(L_{bbox}) \quad (1)$$

$$\mathbf{E}_T^0 = CLIP_{text}(L_{word}) \quad (2)$$

Suppose the embedding dimension is  $\mathbf{D}$ , the shape of the generated vision embedding matrix and text embedding matrix is  $\mathbf{N}_{bbox} \times \mathbf{D}$  and  $\mathbf{N}_{word} \times \mathbf{D}$ . To better introduce the new knowledge into the vision language models, we also add vision and text adapter [11] after the CLIP model.

$$\mathbf{E}_V^0 = (1 - \alpha) \times \mathbf{E}_V^0 + ReLU(\mathbf{E}_V^{0T} \mathbf{W}_1^v) \mathbf{W}_2^v \quad (3)$$

$$\mathbf{E}_T^0 = (1 - \beta) \times \mathbf{E}_T^0 + ReLU(\mathbf{E}_T^{0T} \mathbf{W}_1^t) \mathbf{W}_2^t \quad (4)$$

Here  $W_1^v$ ,  $W_2^v$  and  $W_1^t$ ,  $W_2^t$  are the weights of bottleneck linear layers for vision and text adapter networks.  $\alpha$  and  $\beta$  are hyper-parameters, here we choose 0.3.

### 3.3 Global attention

To gain a clearer understanding of the role of each bounding box item within the overall scene, we initially encode the input image using the CLIP vision encoder. Subsequently, we use the embedding of the bounding box list ( $E_V^0$ ) as the target and the image embedding as the memory. After the cross-attention module, we get the bounding box embedding with global information.

### 3.4 Vision and text embedding space recalibration

Although OpenAI CLIP and its derivative works [38, 50] achieve high accuracy in matching text with image patch, they focus on **noun** (such as “wine glass” and “cup”) instead of **visual attributes** (such as “transparency” and “narrow rim”). To match the image patches with task-specific visual attribute words, it is pivotal to re-calibrate the embedding space of vision and text. Therefore we design a transformer-based **aligner** module, next to the pre-trained VLM. In Fig. 3(a), we pass the vision embedding matrix and text embedding matrix into a multi-layer aligner module to generate the new re-calibrated vision and text embedding matrix,  $\mathbf{E}_V^m$  and  $\mathbf{E}_T^m$  respectively. Here  $m$  represents the number of layers in aligner module. In Fig. 3(b), we present the network architecture of each layer of the aligner module. There are multiple layers of aligner module

where each layer has two self-attention modules, one cross-attention module, and one feed-forward network layer. For the  $i^{\text{th}}$  aligner module layer, the input ( $\mathbf{E}_V^i$  and  $\mathbf{E}_T^i$ ) is the vision and text embedding matrix coming from last aligner module layer. At each layer, both vision and text embedding matrix will first go through a self-attention layer. The purpose of the self-attention layer is to learn the most optimal affordance that could be used to solve this task. For example, as is shown in Fig. 1(a) Scene 5, to solve the task of **sit comfortably** we prefer to choose chair over bed. After self-attention layer, we use a cross-attention layer to recalibrate the text embedding and vision embedding. The last part of the aligner network is a fully connected feed forward layer where the text embedding matrix will pass through. Like previous works that try to fine-tune pre-trained vision-language model [10, 15], in our aligner module, we focus on the text embedding vectors transformation. Therefore, we only pass text embedding matrix through the last feed forward layer.

### 3.5 Score function and training

After  $m$  transformer aligner layers' processing, we get the recalibrated image vision embedding ( $\mathbf{E}_V^m$ ) and text embedding ( $\mathbf{E}_T^m$ ). In Fig. 3(a), we perform matrix to matrix multiplication between  $\mathbf{E}_T^m$  and  $\mathbf{E}_V^m$  to generate the predicted affinity matrix  $\mathbf{A}_p$ . The computation could be summarized as:

$$\mathbf{A}_p = \mathbf{E}_V^{mT} \times \mathbf{E}_T^m \quad (5)$$

The shape of the affinity matrix  $\mathbf{A}_p$  is  $N_{bbox} \times N_{word}$ . In traditional CLIP, to perform the zero-shot image classification, we will directly apply softmax function over affinity matrix to get the predict label. In task-oriented object detection, since we need to find the most-optimal items, further processing is necessary. Regarding the meaning of  $\mathbf{A}_p[i][j]$ , it represents the score that bounding box  $i$  have visual attribute  $j$ . To make the final decision of choosing the most optimal item that could be treated as the ideal task affordance, we pass the affinity matrix into a score function. In Fig. 3(c), we present the score function model architecture. It will first encode each bounding box's visual feature score vector into a higher dimensional vector. In Fig. 3(c), we represented the encoded new matrix as  $\mathbf{H}_p$  whose dimension is  $N_{bbox} \times D'$ . This encoding process is:

$$\mathbf{H}_p[j] = MLP(\mathbf{A}_p[j]) \quad \text{where } j \in [0, N_{bbox}) \quad (6)$$

Then we pass  $\mathbf{H}_p$  into a self-attention layer. Here, we incorporate a self-attention layer because we aim to learn the relative importance of different items, enabling us to select the most suitable items for this task. Following the self-attention layers, we employ multiple fully connected layers for regression, ultimately producing the score vector, ( $\mathbf{S}_p$ ). The shape of  $\mathbf{S}_p$  is  $1 \times N_{bbox}$  while each element of  $\mathbf{S}_p$  has value ranging from 0 to 1. For inference, we will set a threshold value ( $\delta$ ) and we have:

$$\mathbf{P}[i] = \begin{cases} 1 & \text{if } \mathbf{S}_p[i] \geq \delta, \\ 0 & \text{else.} \end{cases} \quad (7)$$



Here  $\mathbf{P}[i]$  is the prediction of whether the item of bounding box  $i$  is an ideal affordance of this task. During training, we will have the ground truth score vector  $\mathbf{S}_g$ . For the  $i^{th}$  element of  $\mathbf{S}_g$ , if it can solve the task then its value is 1 otherwise its value is 0. The loss  $\mathbf{L}$  during the back propagation is:

$$\mathbf{L} = MSELoss(\mathbf{S}_g, \mathbf{S}_p) \quad (8)$$

Here  $MSELoss$  is the mean squared error loss function [19]. After getting the loss, we perform the back propagation and conduct end to end training of CLIP adapter, global attention layer, transformer aligner module, and score function as is shown in Fig. 3(a). Unlike end-to-end training style of DETR-based works, our framework will not touch object detection network and vision-language model. Reuse vision-language pretraining knowledge make our framework has higher training efficiency when compared with previous works, as is shown in section 4.

**Table 2:** Original COCO-Tasks dataset task name and corresponding synonym task name from ChatGPT. The synonym task name is in the parentheses.

task1: step on (trend on)	task2: sit comfortably (loungue easily)	task3: place flowers (arrange blooms)
task4: get potatoes out of fire (retrieve tubers from heat)	task5: water plant (irrigate vegetation)	task6: get lemon out of tea (remove citrus from brew)
task7: dig hole (excavate pit)	task 8: open bottle of beer (uncap ale)	task9: open parcel (unwrap package)
task10: serve wine (serve a cocktail)	task11: pour sugar (dispense sweetener)	task12: smear butter (spread cream)
task13: extinguish fire (douse flames)	task 14: pound carpet (beat rug)	

### 3.6 Select-by-grouping mechanism

When using existing object detection network, such as Faster R-CNN, we not just get the bounding box (bbox) information but also the bbox COCO class. To assist us find all suitable affordance, here we propose the select-by-grouping mechanism. Suppose in an image, we find a total of  $N$  bbox. After the score function, we find that the  $i^{th}$  bbox’s regression score is higher than this task’s threshold. Considering the bbox quality, it is possible that we omit some items which are the same COCO class of the  $i^{th}$  bbox. In most cases, the same class items show a high chance to be ideal task affordance. To compensate this false-negative drawback, our select-by-grouping mechanism will check all other bbox and see if the following scenario satisfied:

$$\begin{aligned} \mathbf{P}[j] = 1 \quad & \text{if } \mathbf{class}[j] = \mathbf{class}[i] \\ & \text{and } \mathbf{conf}[j] > \beta \\ & \text{and } \mathbf{conf}[i] > \beta \end{aligned} \quad (9)$$

Here  $\mathbf{P}$  is the final prediction to see if bbox  $i$  is an ideal affordance. The shape of  $\mathbf{P}$  is  $1 \times N$ . The  $\mathbf{class}$  and  $\mathbf{conf}$  are also two  $1 \times N$  vectors which represent the bbox COCO class type and class confidence.  $\beta$  is a hyperparameter that identifies whether the front-end detection network has sufficient confidence in the bounding box classification result. In our experiment, we set it to 0.8. The

**Table 3:** Experiment setup and model comparison.

Algorithm	Platform	Training epoch	Vision-Language Pretraining	# Trainable Parameters
DETR-based methods [26, 44]	×8 A100 GPUs	30 epochs	✗	Large
TaskCLIP	×1 RTX4090 GPU	20 epochs	✓	Small

meaning of equation 9 is that suppose we found the  $j^{th}$  bbox has the same COCO class with the  $i^{th}$  bbox with high confidence, we also set the  $j^{th}$  bbox item as an ideal affordance. As we shown in Fig. 1(b), since the samples of category 1 items are much less than category 0, our original model has high true positive and false negative rate. The select-by-grouping mechanism therefore will reduce the false negative prediction.

## 4 Experiments

### 4.1 Datasets and Metric

**Dataset** We conduct experiments on the COCO-Tasks dataset [41], which is derived from MS COCO2014 [29]. The main difference between COCO-Tasks dataset and MS COCO2014 lies in the labeling of each item. In COCO-Tasks, the objective is to detect the most suitable items applicable to solve specific tasks. The COCO-Tasks dataset comprises a total of 14 tasks, each consisting of 3600 training images and 900 test images. Within each image, the labels of bounding boxes can either be categorized as category 0 (not ideal affordance) or category 1 (ideal affordance). It’s important to note that the same item (e.g., a bed) may be classified into opposite categories (ideal affordance vs. not ideal affordance) for solving the same task (e.g., sitting comfortably) across different images, as illustrated in Fig. 1(c).

**Task-agnostic experiment** To evaluate the model’s ability to generalize, we not only used the 14 original tasks from the COCO-Tasks dataset but also performed a task-agnostic experiment. Initially, referencing Tab. 1.(b), for each original task in the COCO-Tasks dataset, we requested ChatGPT to generate synonymous task words, as displayed in Tab. 2. For instance, the synonym task for *Serve wine?* is *Serve a cocktail?*. These two tasks share common affordances (such as wine glasses), so an effective generalization model trained on the original COCO-Tasks dataset should be capable of addressing the new task.

**Metric** We use the AP@0.5 metric for task-oriented detection. Then we compute the mAP@0.5 by averaging the AP@0.5 accuracy across all 14 tasks. To keep consistency with previous studies [26, 41], we only report accuracy results for predictions related to suitable affordances (category 1).

### 4.2 Implementation Details

For the object detection network, we experiment with both Faster R-CNN [40] and YOLOv8 [22]. The implementation of Faster R-CNN utilizes the X101-FPN

**Table 4:** Comparison with previous methods. Values in parentheses indicate results from task-agnostic experiments.

Task (AP@0.5(%))	GGNN [41]	TOIST [26]	TOIST† [26]	CoTDet [44]	TaskCLIP	TaskCLIP*
task1	36.6 (-)	44.0 (-)	45.8 (-)	58.9 (56.2)	48.4 (47.6)	52.1 (50.9)
task2	29.8 (-)	39.5 (-)	40.0 (-)	55.0 (53.1)	44.6 (44.1)	47.4 (46.6)
task3	40.5 (-)	46.7 (-)	49.4 (-)	51.2 (48.2)	52.4 (52.4)	54.1 (54.1)
task4	37.6 (-)	43.1 (-)	49.6 (-)	68.5 (57.3)	60.6 (60.6)	62.5 (62.5)
task5	41.0 (-)	53.6 (-)	53.4 (-)	60.5 (54.9)	54.4 (54.4)	56.9 (56.9)
task6	17.2 (-)	23.5 (-)	26.9 (-)	47.7 (35.0)	31.5 (31.5)	34.0 (34.0)
task7	43.6 (-)	52.8 (-)	58.3 (-)	76.9 (50.5)	67.7 (67.7)	69.9 (69.9)
task8	17.9 (-)	21.3 (-)	22.6 (-)	40.7 (3.6)	18.6 (18.6)	19.2 (19.2)
task9	21.0 (-)	23.0 (-)	32.5 (-)	47.4 (20.1)	39.4 (39.4)	41.4 (41.4)
task10	40.6 (-)	46.3 (-)	50.0 (-)	66.5 (58.5)	55.7 (55.7)	58.1 (58.1)
task11	22.3 (-)	33.1 (-)	35.5 (-)	41.9 (30.6)	38.1 (38.1)	39.8 (39.8)
task12	28.4 (-)	41.7 (-)	43.7 (-)	48.3 (47.3)	47.6 (47.6)	49.8 (49.8)
task13	39.1 (-)	48.1 (-)	52.8 (-)	61.7 (9.4)	46.3 (46.3)	48.7 (48.7)
task14	40.7 (-)	52.9 (-)	56.2 (-)	71.4 (58.4)	67.4 (67.4)	69.8 (69.8)
mean (mAP@0.5(%))	32.6 (-)	41.3 (-)	44.1 (-)	56.9 (40.9)	48.1 (47.9)	50.3 (50.1)

\*TOIST† represents the model with noun-pronoun distillation.

\*TaskCLIP\* is our method with a select-by-grouping mechanism, where we also apply different optimized thresholds for each task.

configuration in Detectron2 [47]. As for YOLOv8, we employ the YOLOv8x configuration. Regarding the large vision language model (VLM), we employ OpenCLIP [4, 38]. Specifically, the vision transformer encoder configuration is ViT/H and the text encoder is RoBERTa [31]. We implement transformer aligner module and score function using PyTorch [37]. The model was trained for 20 epochs with an initial learning rate of 1e-6, employing AdamW [32] as the optimizer. We generated visual feature attributes for each task using OpenAI ChatGPT [33].

In Table 3, we present a high-level comparison of experimental settings between TaskCLIP and previous DETR-based models, including TOIST and CotDet [26, 44]. Compared to transformer-based approaches, our training model is significantly smaller as we efficiently leverage pre-trained large VLM. Unlike training everything from scratch, our main training emphasis lies in recalibrating the vision and text embedding space. Without employing any additional training assistance methods, such as knowledge distillation [14], our entire framework can be trained and deployed on a **single NVIDIA RTX 4090 GPU**.

### 4.3 Comparisons with Previous Works

In Table 4, we present a comparison of TaskCLIP’s performance with previous works on the COCO-Task dataset. Here we present results for both original COCO-Task dataset tasks’ results and synonym tasks’ result (in the parentheses). We utilize Faster R-CNN as the front-end object detection network to identify bounding boxes (bbox) for each item. The transformer aligner module between OpenAI CLIP and the score function comprises 8 layers, with each layer’s self-attention and cross-attention having an attention head size ( $N_{Head}$ ) of 4. We set the threshold for each score function output to 0.15 for all tasks as the baseline TaskCLIP result. Additionally, we provide TaskCLIP’s performance with various optimizations in Table 4. In Section 4.4, we will delve into the effects of the transformer aligner module’s layer size ( $N_{Layer}$ ), attention head size ( $N_{Head}$ ), and threshold value ( $\delta$ ) on the detection accuracy (AP@0.5).

Among the previous works focusing on task-oriented object detection, TOIST and CoTDet are based on DETR, while the work [41] employs a pre-trained object detection network as the front-end. However, in the back-end, work [41] use Gated Graph Neural Network (GGNN) [27] to select objects that could potentially serve as the ideal affordance for specific tasks. Given that both TOIST and GGNN require human involvement to prepare words for specific tasks, they are **task-dependent in an unnatural manner**, making it challenging to apply them to other tasks. Therefore, for a task-agnostic evaluation, we only consider CoTDet and TaskCLIP. To assess CoTDet’s performance on the synonym task, we adhere to its ChatGPT prompt and utilize RoBERTa to produce new item and visual feature text embeddings. For TaskCLIP, we reference Tab. 1.(c), requesting ChatGPT to choose visual description words from the visual description word banks maintained during the training phase. These newly selected description words are then directly used to enable TaskCLIP to perform object detection.

As indicated by Tab. 4, TaskCLIP surpasses both GGNN and TOIST on the original COCO-Tasks dataset. Although TaskCLIP’s accuracy is lower than CoTDet’s on the original COCO-Tasks dataset, it outperforms CoTDet by over **9%** on the new synonyms dataset. This is because CoTDet’s training involves direct training of descriptive words and affordance text embedding vectors. Consequently, the accuracy of CoTDet significantly declines when new tasks arise and ChatGPT changes its responses. Conversely, TaskCLIP concentrates on learning the visual features of tasks, which results in better generalizability.

As shown in Table 3, TaskCLIP’s training efficiency surpasses that of TOIST and CoTDet due to TaskCLIP’s utilization of existing pre-trained vision-semantic information rather than recalibrating them from scratch.

#### 4.4 Ablation Study

**Table 5:** Transformer aligner model architecture search.

	threshold = 0.2								threshold = 0.4															
	$N_{Layer} = 4$		$N_{Layer} = 6$		$N_{Layer} = 8$		$N_{Layer} = 10$		$N_{Layer} = 4$		$N_{Layer} = 6$		$N_{Layer} = 8$		$N_{Layer} = 10$									
$N_{Head}$	4	8	16	4	8	16	4	8	16	4	8	16	4	8	16	4	8	16						
mAP@0.5(%)	33.5	43.5	42.6	43.6	43.4	42.7	<b>44.4</b>	43.0	<b>44.2</b>	43.4	42.6	<b>44.5</b>	30.2	42.4	41.7	42.7	42.6	41.6	<b>43.6</b>	42.2	<b>43.4</b>	42.7	41.8	<b>43.8</b>

**Hyperparameter Tuning** In this section, we focus on three hyperparameters: the number of hidden layers of the transformer aligner module ( $N_{Layer}$ ), the number of attention heads in each aligner layer ( $N_{Head}$ ), and the score function threshold ( $\delta$ ). In Table 5, we present the effects of  $N_{Layer}$  and  $N_{Head}$  on mAP@0.5 across 14 tasks, under two different  $\delta$  values, 0.2 and 0.4. We choose Faster R-CNN as the object detection network. As noted by previous studies [41], the occurrence of suitable affordances for each task in the COCO-Tasks dataset is much lower than that of not ideal affordances, with an average ratio of  $\sim 1 : 15$ .

To address this, we set the threshold below 0.5 for our ablation study of the transformer aligner architecture. Table 5 demonstrates that the top three configurations yielding the best accuracy results are (8,4), (8,16), and (10,16) in the format of  $(N_{Layer}, N_{Head})$ . Among these, we believe the combination  $(N_{Layer}: 8, N_{Head}: 4)$  strikes the best balance between model size and accuracy. In the investigation of threshold effects, we consider the data imbalance of COCO-Tasks. Here, we employ g-means [18] to determine each task’s optimal threshold. The average threshold ( $\delta$ ) over all 14 tasks for aligner configuration (8,4) is 0.15, which is also the  $\delta$  value used in Table 4.

**Table 6:** Ablation study of different model components’ effect.

AP@0.5 (%)	task1	task2	task3	task4	task5	task6	task7	task8	task9	task10	task11	task12	task13	task14	Mean(%)
(a) without aligner	25.3	35.3	30.1	26.5	24.9	21.4	40.3	4.6	16.9	36.2	22.0	22.9	20.7	36.7	26.0 (+0.0)
(b) F-RCNN	48.4	44.6	52.4	60.6	54.4	31.5	67.7	18.6	39.4	55.7	38.1	47.6	46.3	67.4	48.1 (+22.1)
(c) F-RCNN+diff $\delta$	48.4	44.8	52.4	60.6	54.5	31.5	67.6	18.8	39.4	55.9	38.1	47.6	46.3	67.4	48.6 (+22.6)
(d) F-RCNN+SG	52.0	47.4	54.1	62.5	56.7	34.0	69.9	19.2	41.5	58.1	39.8	49.8	48.7	69.8	50.1 (+24.1)
(e) F-RCNN+both	52.1	47.4	54.1	62.5	56.9	34.0	69.9	19.2	41.4	58.1	39.8	49.8	48.7	69.8	50.3 (+24.3)
(f) YOLOv8	48.7	46.6	45.6	57.1	50.1	30.9	62.5	22.0	38.0	53.9	36.9	40.0	42.6	65.4	47.5 (+21.5)
(g) YOLOv8+diff $\delta$	48.7	45.4	45.6	57.1	50.1	31.1	61.0	23.1	38.2	55.6	36.3	40.5	43.1	65.3	47.7 (+21.7)
(h) YOLOv8+SG	48.6	46.6	45.9	57.1	50.4	31.1	62.5	22.2	38.3	54.3	37.1	40.0	42.6	64.4	47.8 (+21.8)
(i) YOLOv8+both	49.0	46.2	46.0	57.5	50.4	31.3	61.5	23.2	38.5	55.6	36.6	40.5	43.3	66.9	49.0 (+23.0)
(j) ground truth bounding box	61.2	73.0	56.3	76.6	56.6	47.4	81.5	28.7	49.9	67.7	47.9	56.0	52.1	81.5	62.5

\*Here we use F R-CNN to represent Faster R-CNN.

**TaskCLIP Component Analysis** Table 6 illustrates the contribution of each component of TaskCLIP to the final accuracy, including the object detection network, transformer aligner module, and select-by-grouping mechanism. We compare the performance with the baseline, which directly selects the objects based on the affinity matrix without any re-calibration and alignment. For assessing the effect of the transformer aligner, we present the accuracy of the combination of OpenCLIP and the score function ( $f_{score}$ ). The original OpenAI CLIP and OpenCLIP models were primarily designed to match images with nouns, making it challenging for them to accurately match each bounding box item image with its corresponding visual feature attributes. For Table 6 (b-e) and Table 6 (f-i), we depict the effects of various optimizations on TaskCLIP’s final performance. TaskCLIP effectively recalibrates the item embeddings (vision space) with task-related attribute embeddings (text embedding). We report the accuracy obtained with ground truth bounding boxes in Table 6 (j).

#### 4.5 Visualization and Discussion

Fig. 4 presents the predictions of our model alongside the ground truth in various image samples. Specifically, Fig. 4 (a) showcases examples where our model performs well. The predictions exhibit great overlap with the ground truth across various tasks, demonstrating the robust generalizability of our model. However, Fig. 4 (b) highlights two examples where our model’s performance is sub-optimal. These examples illustrate two common challenges faced by our model. Firstly, in the upper panel of Fig. 4 (b), there are two chairs depicted, but only the distant



**Fig. 4:** Visualization for prediction results of the TaskCLIP (dash blue rectangle) and ground truth (solid red rectangle) (a). Examples with good performance. (b). Examples of unsatisfactory performance.

chair can be utilized to open a bottle of beer due to a person occupying the nearby chair. While the ground truth accurately captures this subtle distinction, our model erroneously identifies all objects resembling chairs based solely on visual features. Furthermore, in the lower panel of Fig. 4 (b), our model erroneously selects both a cup and a tie to extinguish fire. This misprediction arises from the tie’s visual features closely resembling those provided by LLM for solving this task, such as a sturdy handle and a bucket shape. Consequently, our model mistakenly identifies the tie due to its misleading appearance.

## 5 Conclusion

In this study, we introduce TaskCLIP, a novel framework for task-oriented object detection. TaskCLIP leverages pre-trained knowledge and vision-language associations from the frozen CLIP model in an efficient manner, distinguishing itself from prior research efforts. Comparative analysis with prior DETR-based approaches demonstrates TaskCLIP’s superiority in terms of both task generability, accuracy, and training efficiency.

## Acknowledgements

This work was supported in part by the DARPA Young Faculty Award, the National Science Foundation (NSF) under Grants #2127780, #2319198, #2321840, #2312517, and #2235472, the Semiconductor Research Corporation (SRC), the Office of Naval Research through the Young Investigator Program Award, and Grants #N00014-21-1-2225 and N00014-22-1-2067. Additionally, support was provided by the Air Force Office of Scientific Research under Award #FA9550-22-1-0253, along with generous gifts from Xilinx and Cisco.

## References

1. Alayrac, J.B., Donahue, J., Luc, P., Miech, A., Barr, I., Hasson, Y., Lenc, K., Mensch, A., Millican, K., Reynolds, M., et al.: Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems* **35**, 23716–23736 (2022) [3](#), [5](#)
2. Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J.D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al.: Language models are few-shot learners. *Advances in neural information processing systems* **33**, 1877–1901 (2020) [3](#)
3. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: *European conference on computer vision*. pp. 213–229. Springer (2020) [2](#), [5](#)
4. Cherti, M., Beaumont, R., Wightman, R., Wortsman, M., Ilharco, G., Gordon, C., Schuhmann, C., Schmidt, L., Jitsev, J.: Reproducible scaling laws for contrastive language-image learning. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 2818–2829 (2023) [5](#), [11](#)
5. Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., Schiele, B.: The cityscapes dataset for semantic urban scene understanding. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 3213–3223 (2016) [4](#)
6. Deng, J., Yang, Z., Chen, T., Zhou, W., Li, H.: Transvg: End-to-end visual grounding with transformers. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 1769–1779 (2021) [4](#)
7. Du, D., Zhu, P., Wen, L., Bian, X., Lin, H., Hu, Q., Peng, T., Zheng, J., Wang, X., Zhang, Y., et al.: Visdrone-det2019: The vision meets drone object detection in image challenge results. In: *Proceedings of the IEEE/CVF international conference on computer vision workshops*. pp. 0–0 (2019) [4](#)
8. Dvornik, N., Mairal, J., Schmid, C.: Modeling visual context is key to augmenting object detection datasets. In: *Proceedings of the European Conference on Computer Vision (ECCV)*. pp. 364–380 (2018) [4](#)
9. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. *International journal of computer vision* **88**, 303–338 (2010) [1](#)
10. Gao, P., Geng, S., Zhang, R., Ma, T., Fang, R., Zhang, Y., Li, H., Qiao, Y.: Clip-adapter: Better vision-language models with feature adapters. *International Journal of Computer Vision* pp. 1–15 (2023) [8](#)
11. Gao, P., Geng, S., Zhang, R., Ma, T., Fang, R., Zhang, Y., Li, H., Qiao, Y.: Clip-adapter: Better vision-language models with feature adapters. *International Journal of Computer Vision* **132**(2), 581–595 (2024) [7](#)
12. Girshick, R.: Fast r-cnn. In: *Proceedings of the IEEE international conference on computer vision*. pp. 1440–1448 (2015) [3](#), [6](#)
13. Girshick, R., Donahue, J., Darrell, T., Malik, J.: Rich feature hierarchies for accurate object detection and semantic segmentation. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 580–587 (2014) [1](#)
14. Gou, J., Yu, B., Maybank, S.J., Tao, D.: Knowledge distillation: A survey. *International Journal of Computer Vision* **129**, 1789–1819 (2021) [11](#)
15. Goyal, S., Kumar, A., Garg, S., Kolter, Z., Raghunathan, A.: Finetune like you pretrain: Improved finetuning of zero-shot vision models. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 19338–19347 (2023) [8](#)

16. Huang, W., Chen, H., Ni, Y., Rezvani, A., Yun, S., Jeon, S., Pedley, E., Imani, M.: Ecosense: Energy-efficient intelligent sensing for in-shore ship detection through edge-cloud collaboration. arXiv preprint arXiv:2403.14027 (2024) [2](#)
17. Huang, W., Rezvani, A., Chen, H., Ni, Y., Yun, S., Jeong, S., Imani, M.: A plug-in tiny ai module for intelligent and selective sensor data transmission. arXiv preprint arXiv:2402.02043 (2024) [1](#)
18. Jain, P., Garibaldi, J.M., Hirst, J.D.: Supervised machine learning algorithms for protein structure classification. *Computational biology and chemistry* **33**(3), 216–223 (2009) [13](#)
19. James, W., Stein, C.: Estimation with quadratic loss. In: *Breakthroughs in statistics: Foundations and basic theory*, pp. 443–460. Springer (1992) [9](#)
20. Jin, X., Pei, K., Won, J.Y., Lin, Z.: Symlm: Predicting function names in stripped binaries via context-sensitive execution-aware code embeddings. In: *2022 ACM SIGSAC Conference on Computer and Communications Security*. ACM (2022) [4](#)
21. Jin, X., Wang, Y.: Understand legal documents with contextualized large language models (2023) [4](#)
22. Jocher, G., Chaurasia, A., Qiu, J.: Ultralytics YOLO (Jan 2023), <https://github.com/ultralytics/ultralytics> [10](#)
23. Jocher, G., Stoken, A., Borovec, J., Changyu, L., Hogan, A., Diaconu, L., Poznaniski, J., Yu, L., Rai, P., Ferriday, R., et al.: ultralytics/yolov5: v3. 0. Zenodo (2020) [6](#)
24. Kamath, A., Singh, M., LeCun, Y., Synnaeve, G., Misra, I., Carion, N.: Mdetmodulated detection for end-to-end multi-modal understanding. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 1780–1790 (2021) [5](#)
25. Li, P., Lin, Y., Schultz-Fellenz, E.: Contextual hourglass network for semantic segmentation of high resolution aerial imagery (2023) [5](#)
26. Li, P., Tian, B., Shi, Y., Chen, X., Zhao, H., Zhou, G., Zhang, Y.Q.: Toist: Task oriented instance segmentation transformer with noun-pronoun distillation. *Advances in Neural Information Processing Systems* **35**, 17597–17611 (2022) [2](#), [5](#), [10](#), [11](#)
27. Li, Y., Tarlow, D., Brockschmidt, M., Zemel, R.: Gated graph sequence neural networks. arXiv preprint arXiv:1511.05493 (2015) [5](#), [12](#)
28. Liang, F., Wu, B., Dai, X., Li, K., Zhao, Y., Zhang, H., Zhang, P., Vajda, P., Marculescu, D.: Open-vocabulary semantic segmentation with mask-adapted clip. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 7061–7070 (2023) [5](#)
29. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6–12, 2014, Proceedings, Part V 13*. pp. 740–755. Springer (2014) [1](#), [5](#), [10](#)
30. Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.Y., Berg, A.C.: Ssd: Single shot multibox detector. In: *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part I 14*. pp. 21–37. Springer (2016) [4](#)
31. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: Roberta: A robustly optimized bert pretraining approach. arXiv preprint arXiv:1907.11692 (2019) [5](#), [11](#)
32. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. arXiv preprint arXiv:1711.05101 (2017) [11](#)



33. Lund, B.D., Wang, T.: Chatting about chatgpt: how may ai and gpt impact academia and libraries? *Library Hi Tech News* **40**(3), 26–29 (2023) [11](#)
34. Lyu, W., Dong, X., Wong, R., Zheng, S., Abell-Hart, K., Wang, F., Chen, C.: A multimodal transformer: Fusing clinical notes with structured ehr data for interpretable in-hospital mortality prediction. *AMIA Annu Symp Proc.* (2022) [5](#)
35. Mokady, R., Hertz, A., Bermano, A.H.: Clipcap: Clip prefix for image captioning. *arXiv preprint arXiv:2111.09734* (2021) [5](#)
36. Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al.: Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems* **35**, 27730–27744 (2022) [3](#)
37. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* **32** (2019) [11](#)
38. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: *International conference on machine learning*. pp. 8748–8763. PMLR (2021) [3](#), [5](#), [7](#), [11](#)
39. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. pp. 779–788 (2016) [3](#)
40. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* **28** (2015) [4](#), [10](#)
41. Sawatzky, J., Souri, Y., Grund, C., Gall, J.: What object should i use?-task driven object detection. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 7605–7614 (2019) [1](#), [2](#), [5](#), [10](#), [11](#), [12](#)
42. Schön, M., Buchholz, M., Dietmayer, K.: Mgnnet: Monocular geometric scene understanding for autonomous driving. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 15804–15815 (2021) [4](#)
43. Shi, H., Hayat, M., Wu, Y., Cai, J.: Proposalclip: Unsupervised open-category object proposal generation via exploiting clip cues. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. pp. 9611–9620 (2022) [5](#)
44. Tang, J., Zheng, G., Yu, J., Yang, S.: Cotdet: Affordance knowledge prompting for task driven object detection. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. pp. 3068–3078 (2023) [2](#), [4](#), [5](#), [10](#), [11](#)
45. Wang, M., Xing, J., Liu, Y.: Actionclip: A new paradigm for video action recognition. *arXiv preprint arXiv:2109.08472* (2021) [5](#)
46. Wang, Z., Li, T., Zheng, J.Q., Huang, B.: When cnn meet with vit: Towards semi-supervised learning for multi-class medical image semantic segmentation. In: *ECCV 2022 Workshops*. Springer Nature Switzerland (2023) [5](#)
47. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. <https://github.com/facebookresearch/detectron2> (2019) [11](#)
48. Xiao, T., Liu, Y., Zhou, B., Jiang, Y., Sun, J.: Unified perceptual parsing for scene understanding. In: *Proceedings of the European conference on computer vision (ECCV)*. pp. 418–434 (2018) [4](#)
49. Yun, S., Chen, H., Masukawa, R., Errahmouni Barkam, H., Ding, A., Huang, W., Rezvani, A., Angizi, S., Imani, M.: Hypersense: Hyperdimensional intelligent sensing for energy-efficient sparse data processing. *Advanced Intelligent Systems* p. 2400228 (2024) [1](#)

50. Zhong, Y., Yang, J., Zhang, P., Li, C., Codella, N., Li, L.H., Zhou, L., Dai, X., Yuan, L., Li, Y., et al.: Regionclip: Region-based language-image pretraining. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16793–16803 (2022) [7](#)
51. Zhou, Z., Lei, Y., Zhang, B., Liu, L., Liu, Y.: Zegclip: Towards adapting clip for zero-shot semantic segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11175–11185 (2023) [5](#)
52. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020) [5](#)