Learning secondary tool affordances from human actions using the iCub robot

Bosong Ding[1], Erhan Oztop[2], Giacomo Spigler[1], Murat Kirtay[1]

***Abstract*— Tools and other objects offer agents a range of potential actions, commonly referred to as affordances. Each tool is typically designed with a primary purpose in mind -like a hammer's function to drive nails. However, tools can also serve purposes beyond their original design. These alternative uses represent secondary affordances, extending the tool's utility beyond its primary intended function. While prior robotics research on affordance perception and learning has primarily focused on primary affordances, our work addresses the less-explored area of learning secondary tool affordances from human partners. Using the iCub robot equipped with three cameras, we observed humans performing actions on twenty objects using four different tools in ways that deviate from their primary purposes. For example, the iCub observed humans using rulers not for measuring but to push, pull, and move objects. In this setting, we constructed a dataset by taking pictures of objects before and after each action is executed. To model secondary affordance learning, we trained three neural networks (ResNet-18, ResNet-50, and ResNet-101) on three prediction tasks using these raw images as input: (1) identifying which tool was used to move an object, (2) predicting the tool with additional action category information, and (3) jointly predicting both the tool and action performed. Our results demonstrate that deep learning architectures enable the iCub robot to successfully predict secondary tool affordances, thereby paving the road for human-robot collaborative object manipulation involving complex affordances. Code and data from this study are available at https://github.com/BosongDing/second_affordance.**

I. Introduction

Both humans and animals accomplish tasks by recognizing the potential actions that tools make possible -these action possibilities are known as affordances. Tasks that require tool use can be as simple as foraging for basic survival or as complex as creating a design for an industrial application that includes human-robot collaboration [1], [2], [3]. While tools are generally crafted with a primary function in mind, they frequently possess the potential for other uses beyond their original intent. Take a ruler as an example: its primary use is to measure length, yet it can also serve to push or pull an object. The first use is an example of the tool's *primary affordance*, while the second represents one of its *secondary affordances* [1].

The concept of affordance has been explored across multiple fields, such as user interface design, human-computer interaction, and robotics. However, most of the research has focused on recognizing and utilizing primary affordances, especially in the field of robotics [2], [4], [5].

In this study, we address one of the less explored areas of affordances in robotics, focusing on the ability of a robot to learn about secondary tool functions through first-person observation of its human partners. Specifically, we ask the following research question: *How can an iCub robot with multiple cameras learn the secondary affordances of tools used by human partners?* To explore this, we designed an interactive experiment to recognize secondary affordances of tools using image data acquired from the egocentric perspective of an iCub robot via three cameras before and after operators act on various objects. It is important to note that in our study, tools are used by human partners in ways that go beyond their intended function. For example, *throwability* is a primary affordance of a boomerang, but in our experiments, a boomerang was repurposed to perform various actions such as pushing and pulling.

We assess the ability of Convolutional Neural Networks architectures to predict the secondary affordances of tools across three tasks. The first task involves identifying tools from raw images of the object captured before and after performing an action. The second task extends this by including information about the action being executed. Finally, the third task aims to independently predict the tool and action based solely on the raw images. By comparing these tasks, we assess how well the robot can learn to predict secondary affordances of the tools.

Our results show that the models achieve significant accuracy ($> 90\%$) while predicting tools and tool-action pairs across three tasks, thus suggesting that the ResNet-based architectures, particularly ResNet-50, are well suited to learn secondary affordances of objects.

This study offers the following contributions. Firstly, we conducted one of the first studies that use the iCub robot to understand secondary affordances in human interactions within a real-world environment. Secondly, we deliver extensive benchmarking results and demonstrate the effectiveness of neural networks for learning secondary tool affordances.

The structure of this paper is as follows. Section II reviews the literature on affordance learning in robotics. Section III details the dataset specifics used in our research. The tasks we investigate are described in Section IV. Our methods

[1]Bosong Ding is with the department of Cognitive Science and Artificial Intelligence, Tilburg University, Tilburg, Netherlands. {b.ding_3}@tilburguniversity.edu

[2]Erhan Oztop is with Symbiotic Intelligent Systems Research Center, Institute for Open and Transdisciplinary, Research Initiatives, Osaka University, Japan. He is also affiliated with Ozyegin University, Istanbul, Turkey. {erhan.oztop}@otri.osaka-u.ac.jp

[1]Giacomo Spigler is with the department of Cognitive Science and Artificial Intelligence, Tilburg University, Tilburg, Netherlands. {g.spigler}@tilburguniversity.edu

[1]Murat Kirtay is with the department of Cognitive Science and Artificial Intelligence, Tilburg University, Tilburg, Netherlands. {m.kirtay}@tilburguniversity.edu

for developing a secondary affordance learning framework are explained in Section V. Section VI presents the results. Section VII summarizes our conclusions and outlines potential directions for future work. Lastly, Section VIII provides access to the online GitHub repository to reproduce our results.

II. Related Work

Tool use is an active area of research in robotics [6], [7]. Due to its broad scope, here, we review the studies on tool affordance prediction and learning studies in robotics. We also note that most studies in robotics focus on primary rather than secondary affordances [2], [8], [9].

Varadarajan and Vince [10] present the extension of affordance computing initiative (AfNet) to robotics applications, AfRob, to employ vision data to recognize object affordances, such as containability, rollability, grab supportability, etc. Although the motivation of the study was framed for social robots in a domestic environment, the authors did not provide an application for a real robot. Chu et al. [11] employ a manipulator with RGB-D sensor to detect and rank objects' multi-affordances: primary (e.g., grasping a hammer from handle) and non-primary (e.g., grasping a hammer from head) affordances of objects. The authors designed a convolutional neural network (CNN) architecture with an RGB-D data processing pipeline to detect affordances by labeling each image pixel for different affordance types. Lai et al. [12] used a deep learning architecture to determine the regions of the different objects that are functionally the same (i.e., functional correspondence problem) in performing a task, such as the body of a bottle and the front part of a shoe can be used in a task of pounding. Here, the authors draw parallels between the secondary affordances of the objects and functional correspondence. To realize their approach, the authors presented a dataset of RGB images for different objects with the same affordances (i.e., actions). The CNN architecture was adopted to detect functionally corresponding points in images. Do et al. [13] proposed a convolutional network-based primary affordance and object detection framework. The results show that the implementation outperforms some state-of-the-art models on affordance detection tasks using RGB images from off-the-shelf datasets. The authors provide a robotic application to offer the same approach that can be employed on a real robot platform to perform a task that requires grasping affordances, e.g., pouring a bottle.

Based on the studies introduced above, we conclude that our work offers the following differences. On the one hand, unlike existing studies that enable robots to detect object affordances, our study employs the iCub robot to learn human secondary affordances using visual data. On the other hand, we achieve the results in a noisy real-world setting where the iCub robot is physically co-located with human partners.

III. Dataset Acquisition Setup and Specifications

Inthis section, we describe our procedue to collect a dataset for learning the secondary affordances of objects from human partners, using the iCub humanoid robot [14].

Our experimental setup consists of the upper body of an iCub robot that has three color cameras (two Dragonfly cameras, one in each eye, and an Intel Realsense d435i camera, placed on the robot's forhead). Figure 1(a) shows a photo from one of the experiments in which a right-handed operator uses a ruler as a tool to perform an action that is associated with its secondary affordance, i.e. pulling a wooden cube with a ruler. In this setting, we designed

(a) Experiment setup

(b) Objects

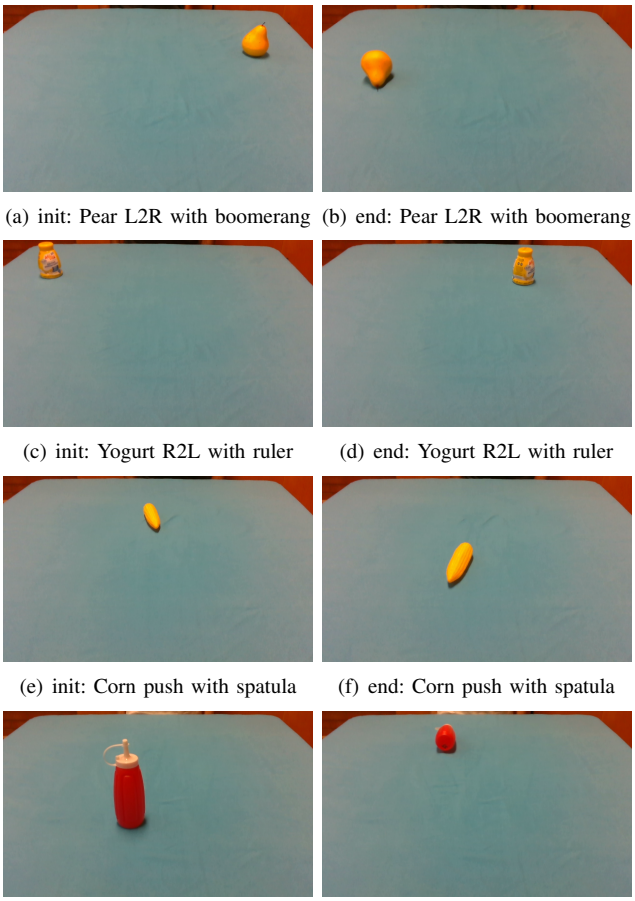
(c) Tools

Fig. 1. The experimental setup (a), where an operator performs a pull action on a wooden cube with a ruler as a tool, the objects (b) and tools (c) that were employed to construct the dataset.

experiments in which the iCub robot becomes an action observer, and four different human partners (i.e., operators) are assigned to be action performers. To construct the dataset, we used 20 objects of different material, color, size, shape, etc.

As shown in Figure 1(a), the operator performs predefined actions (namely, push, pull, left to right, and right to left) on each object using a tool. The objects and tools (boomerang, ruler, slingshot, and spatula) used in this experimental setup are shown in Figure 1(b), 1(c), respectively. We note that the operators do not perform primary affordance of the tools. For example, see Figure 1(a), the operator did not measure the size of an object with the ruler as a tool; instead, the ruler was used to perform predefined actions: pulling, pushing, moving an object from left to right, and moving an object from right to left.

We captured images using all three cameras to record the initial and final poses of objects before and after actions were performed. Figure 2 shows examples of the initial and final pose of objects manipulated with different tools, as



(a) init: Pear L2R with boomerang (b) end: Pear L2R with boomerang

(c) init: Yogurt R2L with ruler (d) end: Yogurt R2L with ruler

(e) init: Corn push with spatula (f) end: Corn push with spatula

(g) init: Ketchup pull with slingshot (h) end: Ketchup pull with slingshot

Fig. 2. Demonstration of tool-object interactions showing initial states (left column) and resulting effects (right column) for various items using different tools.

observed from the robot's central camera. In the dataset, each object was subject to an action using a specific tool, and this process was repeated 10 times for every combination of object, action, and tool. After the experiment, we acquired a total of 3200 samples, each consisting of 6 color images with a resolution of 640×480 pixels (i.e., each sample one of all combinations of 20 objects, 4 tools, 4 actions, 10 repetitions, and consist of initial and final pairs of images for each of the three cameras).

We note that the robot's role in this setting is to observe human partners' actions using a set of tools. However, the results presented in this work can be used as a foundation for employing the iCub robot in a human-robot collaboration task where the robot actively interacts with the human partner and performs secondary affordances on the tools.

IV. Tasks

This paper focuses on the following tasks:

A. Tool Recognition

For this task, the model is given an initial image (object position before manipulation), a final image (object position after manipulation), and optionally the action label used for manipulating the object. The objective is to predict the correct tool. That is, we want the model to identify which tool was used solely by analyzing the observed transformations (and possibly the action label). In our experiments, we also test a setting that removes the explicit action label from the input (i.e., "tools no actions"), to see how the model performs without that hint.

B. Action and Tool Recognition

In the second task, the model again receives initial and final images to predict both the *action* and the *tool* that caused the transition from the initial to the final state of the object. We consider four distinct actions (push, pull, left-to-right, right-to-left) and four possible tools (boomerang, ruler, slingshot, spatula). Hence, the challenge is to simultaneously identify the correct action and tool combination from $4 \times 4 = 16$ possibilities.

V. Methods

In this section, we describe the pre-processing pipeline that transforms our raw dataset into suitable input-output pairs, followed by details on the network architectures used for learning tool affordances. (Note that we provide links to our dataset and GitHub repository for reproducibility in Section VIII, including the scripts, trained models, hyperparameter configurations, and additional figures.)

A. Data Pre-processing

We collected data using four different tools: boomerang, ruler, slingshot, and spatula. Together with four actions: push, pull, left-to-right, and right-to-left. These actions were applied to a set of 20 objects. Each data sample includes initial and final RGB images of the object before and after manipulation, as well as corresponding action and tool labels. Each image was resized to 128×128 pixels and normalized before use. The dataset was divided into train, validation, and test splits in a 6:2:2 ratio, ensuring the splits were based on unique action-tool repetitions for each object to maintain consistency.

For **tool recognition with actions** provided as input, a one-hot encoding of four actions is fed into the model together the initial and final images. For the **tool recognition without actions**, the action vector is omitted. Finally, for **action and tool recognition**, the model receives initial and final images and produces two distinct predictions, one for the action and one for the tool.

B. Network Architectures

We use ResNet [15] architectures to learn whether the observed initial and final images (with optional action encoding) correspond to a particular tool or tool–action pair. Here, we summarize two key architectural choices where C indicates number of camera input and N denotes the network type.

1) **Baseline (3C-1N, Stacked-channels):** We *stack* the initial and final images from all three cameras (center, left, right) into a single input tensor with $6 \times$

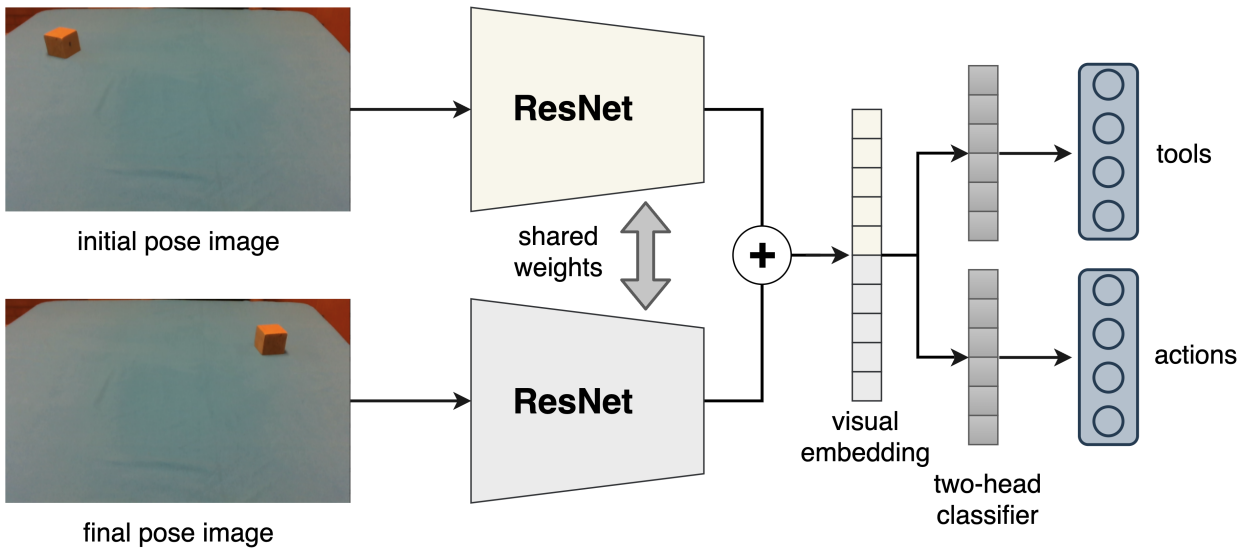


Fig. 3. Schematic representation of the ResNet-50 architecture for tool-action pair prediction described in Section IV-B. The architecture uses paired initial and final images processed through ResNet with *shared* weights. The feature maps from the initial and final images are concatenated and passed to a two-headed classifier to simultaneously predict tools and actions.

$3 = 18$ channels (since each image is RGB). A single ResNet50 then processes these 18-channel inputs to produce a final embedding. This embedding is passed to the classification head(s) to predict tool or tool–action classes. We note that this stacking is a common approach to handle multiple images but may dilute the network's ability to focus on *differences* between initial and final frames.

2) **Proposed Method (1C-1N, Shared-central):** We only use the *central* camera view, and we do not stack the initial and final images. Instead, a single ResNet50 is applied *twice*, once to the initial image and once to the final image, sharing the same weights. The two resulting embeddings are concatenated, then passed to the final classification heads. The intuition is that shared weights highlight changes specific to the object's transformation, rather than mixing multiple viewpoints into a single stacked input. The architectural diagram of the proposed method is illustrated in Figure 3.

In both methods, for the **tool+action** tasks, the network has *two* output heads, one for tool classification and one for action classification. In an ablation study, we also tried a single 16-class output layer representing all tool–action combinations, but this degraded performance compared to splitting them into separate heads.

When *only* tool prediction is required, these second heads are either removed, or they receive the action one-hot vector directly for the "tools with actions" input.

C. Evaluation

All models are trained from scratch using cross-entropy loss for up to 150 epochs, with a grid search over learning rates $(1 \times 10^{-3}, 5 \times 10^{-4}, 1 \times 10^{-4})$, batch sizes (16, 32, 64, 128), and first-layer kernel sizes $(3 \times 3, 5 \times 5, 7 \times 7)$. We identify the best hyperparameters via validation set accuracy and then retrain each model with 5 different random seeds to compute a mean test accuracy and a 95% confidence interval (CI).

VI. Results

We show a summary of the results across three tasks in Table I: predicting tools as a 4-class classification problem, predicting tools without any action input, and jointly predicting tools and actions using a two-headed setup with separate 4-class outputs for tools and actions. In all cases, our proposed **1C-1N** architecture (shared-central) consistently outperforms the baseline **3C-1N** (stacked-channels) model across all tasks. We observe that stacking multiple camera views introduces visual clutter, whereas focusing on a single central camera and sharing weights between initial and final images encourages the network to isolate and learn meaningful changes more effectively.

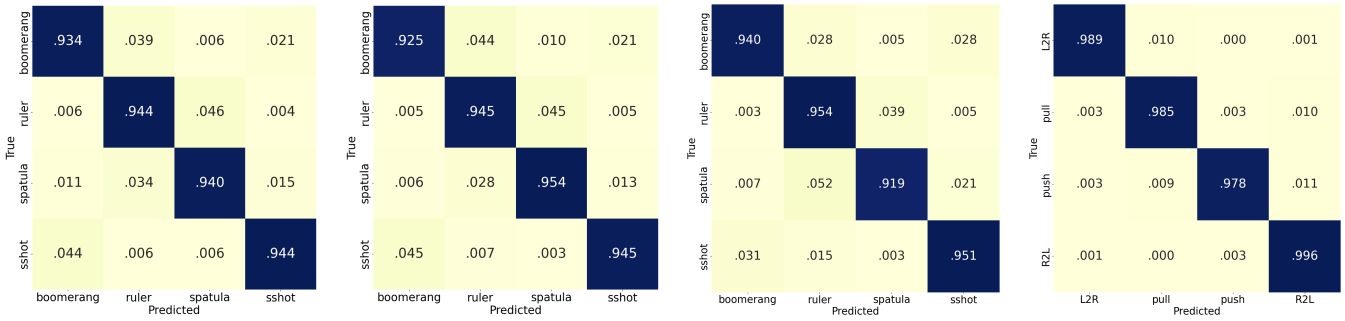
TABLE I

Performance comparison across tasks: **Baseline (3C-1N)** vs. **Best (1C-1N)**

Metric	Baseline (3C-1N)	Proposed (1C-1N)
Tools (%)	88.44 ± 1.34	94.03 ± 1.23
Tools, no action (%)	89.38 ± 2.41	94.21 ± 0.92
Tools+Actions (%)	80.62 ± 2.70	90.78 ± 4.18

A. Ablation and Comparative Study

To better understand the architectural components that influence the models' accuracy, we conducted a comprehensive ablation study. Specifically, we investigated the effect on performance of (1) stacking input images versus processing them separately, (2) weights-sharing versus keeping separate networks for the embedding of each image (i.e., initial vs. final, and camera views), and (3) network depth (ResNet18



(a) Tool recognition with action

(b) Tool recognition without action input

(c) 'Tool' output head (tool+action prediction network)

(d) 'Action' output head (tool+action prediction network)

Fig. 4. Normalized Confusion Matrices for ResNet50-based 1C-1N Architecture: (a) Tool-Only Recognition with Action Reference, (b) Tool-Only Recognition without Action Reference, (c) Tool Recognition Output Head, and (d) Action Recognition Output Head. The visualization of these matrices demonstrates the model's high accuracy in different scenarios, with the action recognition component achieving near-perfect accuracy across all actions and tool recognition showing robust performance.

vs. ResNet101) on performance. We explored the following configurations in addition to the baseline and proposed model:

1) **Separate (3C-6N)**: Each of the six images (initial/final from three cameras) is processed by its own ResNet (no weight sharing). Embeddings are concatenated before the final layer.
2) **Shared (3C-3N)**: Three ResNets (one per camera) handle initial/final images with *shared* weights for that camera. Resulting embeddings are concatenated.
3) **Separate-central (1C-2N)**: Use only the central camera but separate ResNets (one for the initial image, another for the final).
4) **ResNet18 and ResNet101**: We tested both shallower and deeper backbones in the above configurations to see if increased capacity boosts change-detection accuracy.

Overall, these configurations can be grouped into three broad categories. The *stacked input* model (3C-1N) serves as a straightforward baseline, as it is the simplest model. The *shared-weights* models (1C-1N, 3C-3N) assume that using the same parameters for the networks embedding initial and final images helps the network to emphasize changes between them. The *separate-networks* models (1C-2N, 3C-6N) use separate networks to embed each picture (initial and final images for each camera), thus having more parameters and potentially able to produce more expressive representations.

As shown in Table II, the proposed **shared-central** (1C-1N) architecture with a ResNet50 backbone achieves the highest overall accuracy. Interestingly, we find that using fewer cameras can improve performance. Adding the stereo cameras may introduce noise or redundant context.

Weight-sharing across initial and final images also improves detecting changes between frames. Architectures like 1C-1N and 3C-3N, which reuse the same weights across input images, consistently outperform their separate-weight counterparts. This suggests that weight-sharing helps the model to learn differences in the scene rather than redundantly relearning the same features at each step. When comparing backbone networks, ResNet50 offers the best performance. It consistently outperforms both the shallower ResNet18 and the deeper ResNet101.

TABLE II

NETWORK ARCHITECTURES' ACCURACY AND 95% CONFIDENCE INTERVAL ON ACTION AND TOOL RECOGNITION TASK

Architectures	**ResNet18**	**ResNet50**	**ResNet101**
Stacked-channels (3C-1N)	76.68 ± 1.31	80.62 ± 2.70	79.00 ± 1.79
Separate (3C-6N)	81.68 ± 3.35	73.90 ± 1.30	71.28 ±1.39
Shared (3C-3N)	81.56 ± 3.54	86.09 ± 3.02	80.87 ± 1.01
Separate-central (1C-2N)	78.06 ± 2.57	76.31 ± 5.10	72.43 ± 1.99
Shared-central (1C-1N)	**86.06 ± 2.05**	**90.78 ± 4.18**	**85.15 ± 5.24**

We also observe that separating the final layer into two output heads —one for tool classification and one for action recognition- improves performance compared to a single 16-class classifier. This two-headed approach likely reduces interference between the tasks and allows each head to specialize more effectively.

Figure 4 (d) illustrates that action recognition is almost trivial across all architectures, with accuracies often exceeding 99%. This is likely due to strong visual cues in the final object poses that indicate action direction—such as whether an object was pushed, pulled, or moved from left to right. Tool classification, on the other hand, remains more challenging. Certain tools, like the *boomerang* and *slingshot*, can produce visually similar outcomes, requiring the model to pick up on subtle features such as tool curvature or orientation.

Taken together, our results highlight the importance of network architectural design, particularly in tasks where visual distinctions are subtle. While action recognition seems

largely unaffected by architectural choices, tool classification is much more sensitive, especially in scenarios where every tool can perform every action. This overlap places a greater demand on the model's precision and its ability to capture fine-grained visual cues.

VII. Conclusions

We presented a systematic exploration of ResNet-based deep neural architectures for modeling tool–action relationships in a human–robot collaboration context. We demonstrated that these models enable the iCub humanoid robot to learn secondary affordances of tools from human demonstrations -specifically, by observing the position of objects before and after manipulation.

Our proposed shared-central (1C-1N) architecture, when combined with a ResNet50 backbone and two-headed output design, achieved the best overall performance. This configuration not only reduced noise from irrelevant views but also focused the model's capacity on scene transitions critical to tool understanding.

Notably, we observed that while action classification was largely trivial due to consistent visual cues, tool classification required greater nuance. The uniform pairing of all tools with all actions in our dataset created overlapping visual outcomes, pushing the models to learn more fine-grained representations. This finding underscores the importance of architectural decisions in high-ambiguity settings and validates our approach of using separate output heads for multi-task learning.

Finally, our results support the broader vision that social robots can acquire meaningful understanding of tool use through observation, enabling collaborative capabilities in real-world tasks such as furniture assembly or kitchen assistance. In future work, we plan to extend our system with multimodal sensory inputs -including audio and proprioception- to infer human intent and action in shared environments.

VIII. Reproducibility of the Study

The dataset used in this study is available freely at https://www.crossvalidate.me/datasets.html. All code, trained model checkpoints, and detailed hyperparameter settings are available in our public GitHub repository: https://github.com/BosongDing/second_affordance. This includes ablation scripts, generated figures, and versioned dependencies, so others can replicate our results exactly or build upon our methodology.

References

[1] L. Ye, W. Cardwell, and L. S. Mark, "Perceiving multiple affordances for objects," *Ecological Psychology*, vol. 21, no. 3, pp. 185–217, 2009.

[2] L. Jamone, E. Ugur, A. Cangelosi, L. Fadiga, A. Bernardino, J. Piater, and J. Santos-Victor, "Affordances in psychology, neuroscience, and robotics: A survey," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 10, no. 1, pp. 4–25, 2016.

[3] F. Stramandinoli, A. Roncone, O. Mangin, F. Nori, and B. Scassellati, "An affordance-based action planner for on-line and concurrent human-robot collaborative assembly," in *2nd ICRA International Workshop on Computational Models of Affordance in Robotics*, 2019.

[4] D. Norman, *The design of everyday things: Revised and expanded edition*. Basic books, 2013.

[5] N. Masoudi, G. M. Fadel, C. C. Pagano, and M. V. Elena, "A review of affordances and affordance-based design to address usability," in *Proceedings of the Design Society: International Conference on Engineering Design*, vol. 1, no. 1. Cambridge University Press, 2019, pp. 1353–1362.

[6] K. P. Tee, J. Li, L. T. P. Chen, K. W. Wan, and G. Ganesh, "Towards emergence of tool use in robots: Automatic tool recognition and use without prior tool learning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6439–6446.

[7] S. Brown and C. Sammut, "Tool use learning in robots," in *2011 AAAI Fall Symposium Series*, 2011.

[8] P. Zech, S. Haller, S. R. Lakani, B. Ridge, E. Ugur, and J. Piater, "Computational models of affordance in robotics: a taxonomy and systematic classification," *Adaptive Behavior*, vol. 25, no. 5, pp. 235–271, 2017.

[9] H. Min, R. Luo, J. Zhu, S. Bi *et al.*, "Affordance research in developmental robotics: A survey," *IEEE Transactions on Cognitive and Developmental Systems*, vol. 8, no. 4, pp. 237–255, 2016.

[10] K. M. Varadarajan and M. Vincze, "Afrob: The affordance network ontology for robots," in *2012 IEEE/RSJ international conference on intelligent robots and systems*. IEEE, 2012, pp. 1343–1350.

[11] F.-J. Chu, R. Xu, L. Seguin, and P. A. Vela, "Toward affordance detection and ranking on novel objects for real-world robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4070–4077, 2019.

[12] Z. Lai, S. Purushwalkam, and A. Gupta, "The functional correspondence problem," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 15 772–15 781.

[13] T.-T. Do, A. Nguyen, and I. Reid, "Affordancenet: An end-to-end deep learning approach for object affordance detection," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 5882–5889.

[14] M. Kirtay, U. Albanese, L. Vannucci, G. Schillaci, C. Laschi, and E. Falotico, "The icub multisensor datasets for robot and computer vision applications," in *Proceedings of the 2020 International Conference on Multimodal Interaction*, 2020, pp. 685–688.

[15] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.