INSTITUTE FOR ARTIFICIAL INTELLIGENCE, PEKING UNIVERSITY

PKU-IAI Technical Report: PKU-IAI-2023-0007

Bongard-Tool: Tool Concept Induction from Few-Shot Visual Exemplars

Guangyuan Jiang **Chuyue Tang**
Yuanpei College
Peking University
{jgy, 2000017814}@stu.pku.edu.cn

Yuyang Li **Yu Liu**
Department of Automation
Tsinghua University
{liyuyang20, liuyu20}@mails.tsinghua.edu.cn

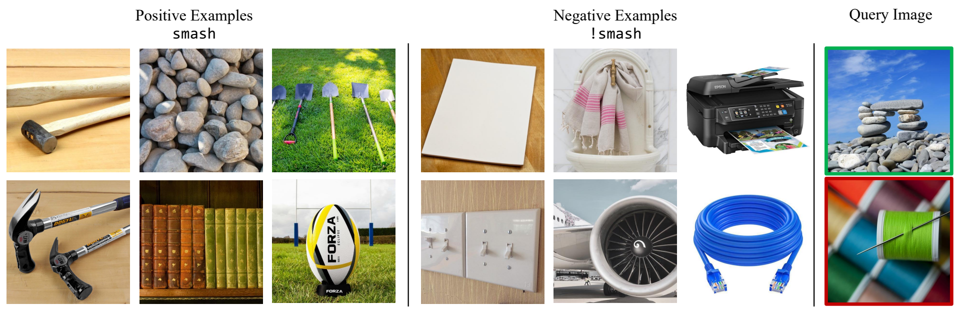


Figure 1: **An example instance for our Bongard-Tool benchmark.** Each instance includes $6 + 6 + 1 = 13$ images. The positive image set shows six tools that can afford `smash`, which is not supported by the six tools in the negative image set. The task is to predict whether the tool in the query image can afford such a function, and the answer is binary.

Abstract

There is no one-to-one mapping from objects to tool concepts. In fact, objects can support diversified tool uses, enabling compositional and flexible functionalities. These tool-like functionalities are mostly context-dependent and vary across scenes. To address this unique property of tool concepts, we propose the Bongard-Tool challenge and formulate the context-dependent tool understanding as a few-shot concept induction problem. Specifically, to build Bongard-Tool, we employ large language models for knowledge building, web crawling, and vision-language models for content retrieval and filtering. We also perform extensive experiments on recent few-shot and meta-learning methods to show the hardship of understanding compositional tool concepts from pure visual perception. We hope to shed light on future studies by introducing Bongard-Tool benchmark as a testbed for building machines that can flexibly understand and use tools. [1]

[1]See our code at `https://github.com/YuyangLee/Bongard-Tool`.

1 Introduction

The water that bears the boat is the same that swallows it up. 水能载舟,亦能覆舟.

This is an old Chinese idiom. Despite its original meaning, the idiom illustrates more than one kind of functionalities that can be leveraged from water as a tool.

What makes a physical object a tool? From one perspective, multiple objects can support the same tool use. For example, with a claw hammer, a person can smash a nail into the wood or pull it out from the wood. You can also smash with a thick book, a shovel, a rock, or even a rugby ball, all of which have their original purpose of use. On the other hand, the compositional nature of tools makes it hard to understand an object's tool-like functionalities without context. A book is a hammer for smashing, a lid covering instant noodles, fuel when making fire, and a pad for balancing an uneven table.

In other words, a tool is a composition of multiple functions, and the context determines which one comes into action. To address this unique property, we introduce the Bongard-Tool benchmark for understanding the compositional concepts of tool use.

Figure 1 shows an example of our Bongard-Tool benchmark. Inspired by the Bongard problems [3], we formulate context-dependent tool understanding as a few-shot visual reasoning task. Specifically, a Bongard-Tool instance includes three parts: (i) positive examples that share the same tool use (for example, smash), (ii) negative examples that can not afford that tool use (non-smashable), and (iii) query images for prediction.

To solve a problem in Bongard-Tool, one needs to infer the shared tool concepts from few-shot exemplars. This constitutes a few-shot concept induction problem.

Our work makes three major contributions:

1. We formulate the context-dependent tool understanding as a few-shot concept induction problem, which can represent a broad range of compositional functionalities of tools.
2. We propose the Bongard-Tool benchmark for addressing the context-dependent tool understanding in visual reasoning. Extensive experiments on recent few-shot and meta-learning methods show the hardship of understanding compositional tool concepts from pure visual perception.
3. We demonstrate the effectiveness of fast constructing large-scale datasets by utilizing large language models for knowledge building, web crawling, and vision-language models for content retrieval and filtering.

2 Related Work

Tool Understanding Understanding and using tools are the hallmarks of high-level intelligence [21]. Humans, even crows, can infer the functionalities of daily objects to support flexible tool use [10]. Recently, computational modeling methods have tackled the problem of robotics tool use through simulation [23, 24, 1], planning [9, 19], and utilizing commonsense knowledge [20].

Our task differs from previous work as we do not aim at specific tool use but to understand how tools work from a broader conceptual perspective. In Bongard-Tool, tool concepts are induced from exemplars. And to successfully solve a problem in Bongard, one needs to understand the compositional functions of physical objects under certain scenarios.

Few-Shot Visual Reasoning Few-shot learning aims at learning from a small number of training samples. With the goal of learning general strategies from a distribution of tasks and generalizing to a new task, meta-learning has become a standard practice for few-shot learning problems. Generally speaking, meta-learning methods can be categorized into four families: (i) memory-based methods, such as SNAIL [14]; (ii) metric-based methods, such as Matching Networks [22] and ProtoNet [18]; (iii) optimization-based methods, such as MetaOptNet [13], MAML [7], and ANIL [17]; (iv) additional pre-training methods, such as Meta-Baseline [5].

These methods are usually evaluated at benchmarks like miniImageNet. However, these few-shot classification tasks still differ from our proposed benchmark since the former requires almost no high-level reasoning ability. Our Bongard-Tool requires not only object-level knowledge, i.e. what is the tool in the image, but also function-level knowledge, i.e. how can the tool in the image be used. Such common-sense knowledge is also important for efficient human-and-robot interaction. More importantly, the model needs to figure out the task-specific strategy, i.e. tools on the left side can support one common function, using human-level reasoning. Thus, we believe that our Bongard-Tool problem can not only serve as a useful benchmark for few-shot learning, but also provide a test-bed for deep learners' visual reasoning ability.

Concept Induction Concept induction is a task of learning abstract concepts from a set of examples. It is closely related to our Bongard-Tool problem, since both of them require the model to learn abstract concepts from a set of examples. However, concept induction usually focuses on learning abstract concepts from a single domain, such as shapes [12], while our Bongard-Tool requires the model to learn abstract concepts from multiple domains, such as tools and their functions. Moreover, concept induction usually requires the model to learn the concept from a single example, while our Bongard-Tool requires the model to learn the concept from multiple examples. Thus, our Bongard-Tool problem is more challenging than concept induction.

The Bongard Problem In the classical Bongard problem[3], there are two sets of images. One of them contains images with some common *properties* that none of the images in another set have. Given a query image, the target is to decide whether or not it has the property. The challenge lies in finding the right *property* by which the two sets can be separated. The following work extended such format to Human-Object Interaction (HOI)[11], abstract reasoning[15], *etc*.

3 The Bongard-Tool Benchmark

We propose the Bongard-Tool Benchmark for few-shot learning based on the conceptual compositions of tools. Following [11], we build our few-shot instances with natural images from our dataset (see Appendix A.1). Fig. 1 shows an example of such an instance for the function `smash`. Note the six positive images and six negative images, showing tools that can and cannot be used to smash, respectively.

3.1 Task Definition

Given a positive and a negative image set, as well as a query image, the task is to predict whether the tools in the query image can afford the function shown by the two contrastive image sets. Of note, the function label (*e.g.* `smash`) is not available as input data.

In our Bongard-Tool problem, we inherit two important features of the original Bongard problem.

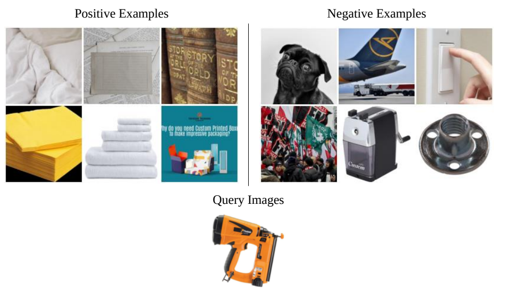
Few-Shot Classification Each Bongard-Tool problem contains six positive images and six negative images and the task is to make binary predictions on two query images. Thus, each problem can be regarded as a two-way six-shot classification task.

Context-Dependent Reasoning The label of an image is not fixed but dependent on its context (i.e. all other positive and negative images in the problem). For example, the tool `hammer` can support more than one function, such as `nail` and `tighten`. Thus, it is not a traditional whole-set classification task.

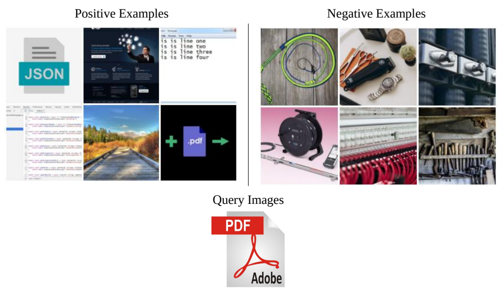
3.2 Task Instance Construction

We use a tuple to represent the concept of tool-use $u = \langle t, f \rangle$, where t denotes a tool, with $f \in \mathcal{F}_t$ being one of the functionalities that t can provide. For consider example, $t_1 =$`claw_hammer`, $t_2 =$ `rock`, with functionalities $\mathcal{F}_{t_1} = \{$ `smash`, `pull_nail`,$\dots\}$, and $\mathcal{F}_{t_2} = \{$ `decorate`, `smash`, $\dots\}$.

Given a dataset $\mathcal{D}$ consisting of a set of tools $\mathcal{T}$, all the functionalities $\mathcal{F} = \bigcup_{t\in\mathcal{T}} \mathcal{F}_t$, and all the images of the tools $\mathcal{I} = \bigcup_{t\in\mathcal{T}} \mathcal{I}_t$, Alg. 1 shows how we sample a few-shot instance from it. We first sample an anchoring functionality f and sample M images of tools that can support f and M that cannot. We also sample the query image I_q with the possibility $p = 0.5$ that it belongs to a tool that can support f, indicated by the binary label y_q. Fig. 2 shows two examples sampled from our dataset. Currently, our pipeline is still naive. The problems will be discussed in Sec. 6.2.



(a) **An example for the function `fold`.** The ground truth label is `POSITIVE`.



(b) **An example for the function `scrape`.** The ground truth label is `NEGATIVE`.

Figure 2: **Two examples of instances sampled from our dataset.** On the left is an example for the function `fold`, which is a relatively good instance. The one on the right is an example for the function `scrape`, which is a poor instance, where the function-tool matching is not reasonable.

Algorithm 1: Sampling a few-shot instance from the dataset.

```
Require: Dataset $\mathcal{D} = \langle \mathcal{T}, \mathcal{F}, \mathcal{I} \rangle$, $M$
  $n \leftarrow 0$
  $f \sim \mathcal{F}$
  $\mathcal{T}^P \leftarrow \{t \in \mathcal{T} : f \in \mathcal{F}_t\}, \mathcal{T}^N \leftarrow \{t \in \mathcal{T} : f \notin \mathcal{F}_t\}$
  $\mathbf{I}^P \leftarrow \emptyset, \mathbf{I}^N \leftarrow \emptyset$
  repeat
    Sample $t^P \sim \mathcal{T}^P, t^N \sim \mathcal{T}^N$
    Sample $I^P \sim \mathcal{I}_{t^P} / \mathbf{I}^P, I^N \sim \mathcal{I}_{t^N} / \mathbf{I}^N$
    $\mathbf{I}^P \leftarrow \mathbf{I}^P \cup \{I^P\}, \mathbf{I}^N \leftarrow \mathbf{I}^N \cup \{I^N\}$
    $n \leftarrow n + 1$
  until $n = M$
  $\mathcal{I}^P \leftarrow \bigcup_{t \in \mathcal{T}^P} I_t, \mathcal{I}^N \leftarrow \bigcup_{t \in \mathcal{T}^N} I_t$
  Sample $\rho \sim U[0, 1]$
  if $\rho < 0.5$ then
    Sample $I_q \sim \mathcal{I}^P$
    $y_q \leftarrow 1$
  else
    Sample $I_q \sim \mathcal{I}^N$
    $y_q \leftarrow 0$
  end if
  return $\mathbf{I}^P, \mathbf{I}^N, I_q, y_q$
```

3.2.1 Benchmark Formulation

We describe the formal formulation for our benchmark. Following Sec. 3.2 we sample the positive and the negative image sets $\mathbf{I}^P = \{I_i^P\}_{i=1}^M, \mathbf{I}^N = \{I_i^N\}_{i=1}^M, I_q, y_q$ with $M = 6$. Each task instance can be represented as $\mathcal{S} = \mathcal{P} \cup \mathcal{N} = \left\{(I_1^P, 1), \cdots, (I_M^P, 1), (I_1^N, 0), \cdots, (I_M^N, 0)\right\}$. The task performance is evaluated on the query image (I_q, y_q). The label y_q indicates whether the query image I_q contains the common feature possessed by images in the positive set $\mathcal{P}$.

4 Methods

4.1 Baselines

In this part we briefly discuss different pipelines on the baseline benchmark. We evaluate several state-of-the-art (SOTA) few-shot learning approaches on our Bongard-Tool benchmark. Also, we investigate whether there is a need for additional representation learning, e.g. pre-training, and from which the representation is learned.

Following the tradition in relevant works, Bongard-LOGO [15] and Bongard-HOI [11], we mainly evaluate two kinds of baselines, meta-learning methods and non-episodic methods, which differ in how they regard the training set. In non-episodic methods, each task is treated as a single training sample $(I_q \cup_{i=1}^{2M} I_i, y_q)$. In meta-learning methods, each task is treated as a sequence of training samples (I, y).

4.1.1 Non-Episodic Methods

For this part, we consider two different ways to encode each task.

CNN-Baseline In each task, a query image is concatenated with both the positive set and the negative set, producing two stacked images of $(6 + 1) * 3 = 21$ input channels. By changing the number of input channels, the few-shot learning problem is transformed into a traditional binary classification problem. Both the positive and negative stacked images are then fed into ResNet encoder for feature extraction. The two features are concatenated and passed to the MLP layer to produce the final logit. The sigmoid function is applied to the logit and used for binary classification.

WReN-Bongard WReN [2] is a new architecture designed for the Raven Progressive Matrices (RPMs) to encourage reasoning. WReN forms relational representations by pairing features between the context and the query, and between context images. To use WReN in our task, ResNet is first used to extract features for each image. Then, the relation network module produces a relational representation by pairing seven image features (six context features and one query feature). The two representations are used in the same way as CNN-Baseline to produce the binary classification result.

4.2 Meta-Learning Methods

Various meta-learning algorithms have been a standard framework for few-shot classification problems. We consider the following categories: (i) memory-based method: SNAIL; (ii) metric-based method: ProtoNet; (iii) optimization-based method: MetaOptNet and ANIL; (iv) additional pre-training method: Meta-Baseline.

SNAIL [14] SNAIL proposes a class of simple and generic meta-learner architectures that use a novel combination of temporal convolutions and soft attention. In Bongard-Tool problem, it receives as input a random sequence of context image-label pairs $(I_1^P, 1), \cdots, (I_M^P, 1), (I_1^N, 0), \cdots, (I_M^N, 0)$ for timesteps $1, \cdots, 2M$, followed by the unlabeled example I_q. It then outputs its prediction for I_q at timestep $2M + 1$ based on the positive and negative examples it has seen.

ProtoNet [18] ProtoNet is built on a simple assumption that there exists an embedding space where points belonging to the same class cluster around a single prototype representation. In Bongard-Tool problem, the positive (negative) prototype representation is the average of six positive (negative) embeddings. Classification is performed by finding the nearest prototype for the query embedding.

MetaOptNet [13] MetaOptNet proposes that a linear support vector machine (SVM) classifier can outperform simple nearest-neighbor classifier used in ProtoNet. It has the advantage of being explicitly solved by convex optimization and thus more efficient.

ANIL [17] ANIL is a simplified version of MAML. The original MAML algorithm tries to land on an initialization of neural networks so that new tasks can be easily learned with few-shot examples via two loops. The outer loop updates the initialization and the inner loop performs a few gradient updates on few-shot examples. ANIL analyses this algorithm in detail and finds that inner loop adaption only to the head of the network is enough to perform effective classification.

Meta-Baseline [5] Meta-Baseline adopts a two-stage approach. First, a classifier baseline is trained on a labeled dataset of base classes C_{base}. Second, the previously trained encoder is used in the meta-learning stage to adapt to novel concepts C_{novel}. The meta-learning stage is similar to ProtoNet.

To study the effect of the first pre-training stage, we consider its two variants: (i) Meta-SC, where we meta-train the model from scratch; (ii) Meta-IN, where we use an ImageNet pre-trained ResNet from PyTorch as the backbone encoder and then apply meta-training.

5 Experiments

5.1 Train/Test Split

To evaluate the models' ability at different levels, we consider two ways to split the dataset.

Normal Split (NS) We sample all Bongard-Tool instances from the whole dataset, and then we randomly select 80% of instances for training and 20% for testing. In this setting, the models have seen all concepts in the training process.

Conceptual Generalization Split (CGS) We sample 80% of tool concepts for training and 20% for testing, and then we sample Bongard-Tool instances from training concepts and testing concepts respectively. In this setting, the models have never seen the testing concepts so it's more difficult for models to generalize to new concepts.

5.2 Results

To test the concept induction ability of the models introduced in Section 4, we benchmark them on Bongard-Tool. We show our experimental results in Table 1. We observe that meta-learning methods generally outperform non-episodic methods. Meanwhile, relational modules can also enjoy performance gain on our dataset. However, does this imply that tool concept induction can be done with only visual representations? Our results on the conceptual generalization split (CGS) prove that these models fail to generalize to novel concepts; instead, they may find some shortcuts to our Bongard-Tool tasks.

Split	Non-Episodic		Meta-Learning Methods					
	CNN	WReN	SNAIL	ProtoNet	MetaOptNet	ANIL	Meta-SC	Meta-IN
NS	52.97	74.10	63.91	**75.18**	75.13	63.03	76.54	**88.82**
CGS	51.35	55.90	**64.42**	61.83	61.02	54.29	61.37	**69.78**

Table 1: **Test accuracy on baseline methods.** We report the test accuracy of baseline methods on our benchmark. Bold fonts indicate the best result in the group.

6 Discussion

We discuss about several problems we encounter in this section.

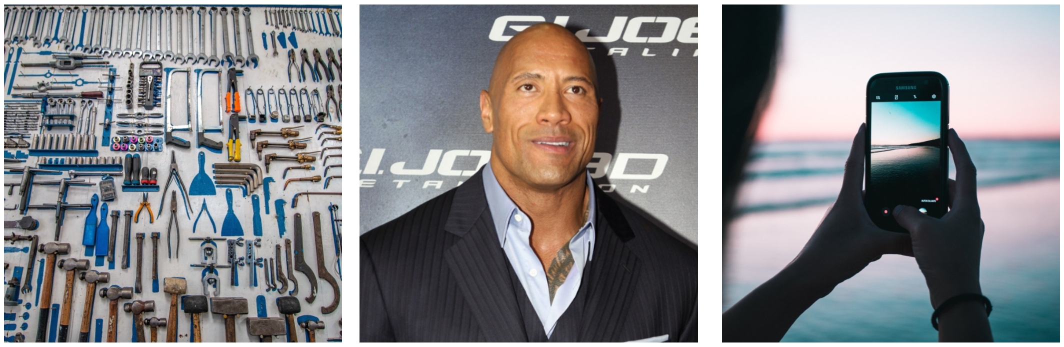
6.1 Problem Formulation

In our proposed benchmark, we only take the form of Bongard [3]. Below perception, all Bongard benchmarks take the form of visual few-shot learning and induction on two contrastive exemplars as visual input, predicting the binary result for the query. Beyond perception, however, different benchmarks are based on different knowledge. While Bongard-HOI [11] is built on the spatial pattern of human-object interaction, our benchmark is built on the affordance and functionalities of tools.

However, unlike previous Bongard benchmarks, our benchmark requires way more knowledge than the information provided by the images (*e.g.* basic algorithmic rules and bike riding). To enable explainable human-level induction, many aspects should be taken into consideration, including priors of physical properties of tools, experiences of using tools, *etc*. As a result, although baseline methods show acceptable performance, it is more likely that they find shortcuts between tool functionalities and their image patterns, rather than really understanding tools.

6.2 Dataset Construction

The process we construct our dataset leverages knowledge in GPT-3, text labels of internet images, as well as the vision-language knowledge of the CLIP model. In this process, not only does the



(a) **Tool cluster**. (b) **Text polysemy.** (c) **Grounding disagreement.**

Figure 3: **Examples of mismatch between the search query and downloaded image.** Photo credit: (a) Cesar Carlevarino Aragon on Unsplash, (b) Cyriac Jannel on Unsplash, and (c) Eva Rinaldi on Flickr.

knowledge limit the range of tools and functionalities, but the bias and ambiguity also produce non-negligible errors in tool-function and text-image matching.

6.2.1 Tool-Functionality Mismatch

It is hard for machine learning models to understand tool concepts, even for the largest model available, GPT-3. We try to ask GPT-3 to give many tool categories. However, we find that GPT-3 has a problem distinguishing physical tools (*e.g*., hammer) and virtual tools (*e.g*., software). Meanwhile, given a tool category name, it also fails to understand what specific tool it may refer to if the tool word has polysemies.

6.2.2 Text-Image Mismatch

We report the text-image mismatch problems mainly in four ways.

Tool Cluster (*Many-to-One*) In some images, there is a tool cluster including different types of tools (*e.g*. in Fig. 3a). *Many* text labels are related to the same *one* image, which introduces ambiguity to reference. This can be remedied by using big N_t to distribute the normalized probability prediction to other candidates, so that tool cluster images will have a lower probability for the query.

Text Polysemy (*One-to-Many*) Many nouns are associated with multiple senses, which may cause problems where *one* text label has images of *many* kinds of objects (not only just tools). Fig. 3b shows an interesting example where photos of Dwayne Johnson (also known by his ring name the *Rock*) occur in the images for `rock`. More examples include photos of Cane Corso dogs and sugar canes for `cane` (walking stick as a tool),

Grounding Disagreement In various images, the query may be grounded in various things, which causes disagreement. For example, in Fig. 3c, the query *camera* should be referring to a photo camera, but it refers to the camera app in the photo. Some grounding disagreement problems are also caused by Text Polysemy. We find that images from search engines tend to have fewer such disagreement, mainly because it contains way more images to barely demonstrate the object referred to in the query, while stock images are mostly for artistic purposes.

Rare / Abstract Tools Some tools are abstract or so rare that there are no adequate photos on the three websites. As a result, some of the downloaded images do not include the tool. As an example, we cannot find any photo of a *brick hammer*, which refers to a kind of hammer with a thin tail that can split bricks cleanly. It is important to manually filter the tool keywords.

7 Conclusion

This project proposes Bongard-Tool, a benchmark for few-shot concept learning based on conceptual compositions of tools. Inspired by the compositional and flexible nature of tools, we formulate the context-dependent tool understanding as a few-shot concept induction problem, which can represent a broad range of compositional functionalities of tools.

To construct the Bongard-Tool dataset, we employ large language models (*i.e.*, GPT-3)'s knowledge to generate the tool categories and conceptual knowledge of flexible tool use. We use web crawling and CLIP-assisted image-text similarity filtering to find images of specific tools. Our method demonstrates a novel, fast and automatic dataset-building pipeline.

We test state-of-the-art meta-learning and visual reasoning models, and their deficiency in the conceptual generalization split shows some of the visual reasoning methods can not perform the real tool concept induction but instead learn some shortcuts through the training samples.

In conclusion, this project investigates how tool concepts are induced under the diversified context of tool use scenarios by proposing the Bongard-Tool benchmark. We hope our project may shed light on better understanding the contextual basis of tool concepts from visual perception.

References

[1] Kelsey R Allen, Kevin A Smith, and Joshua B Tenenbaum. Rapid trial-and-error learning with simulation supports flexible tool use and physical reasoning. *Proceedings of the National Academy of Sciences*, 117(47):29302–29310, 2020. 2

[2] David Barrett, Felix Hill, Adam Santoro, Ari Morcos, and Timothy Lillicrap. Measuring abstract reasoning in neural networks. In *International Conference on Machine Learning (ICML)*, pages 511–520. PMLR, 2018. 5

[3] Michail M. Bongard, M. Bongard, and Michail M. Bongard. *Pattern Recognition*. Spartan Books, New York, 1970. ISBN 978-0-87671-118-7. 2, 3, 6

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 33: 1877–1901, 2020. 10

[5] Yinbo Chen, Xiaolong Wang, Zhuang Liu, Huijuan Xu, and Trevor Darrell. A new meta-baseline for few-shot learning. 2020. 2, 5, 10

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *Computer Vision and Pattern Recognition*, 2009. 10

[7] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, pages 1126–1135. PMLR, 2017. 2

[8] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *Cornell University - arXiv*, 2015. 10

[9] Rachel Holladay, Tomás Lozano-Pérez, and Alberto Rodriguez. Force-and-motion constrained planning for tool use. In *International Conference on Intelligent Robots and Systems (IROS)*, pages 7409–7416. IEEE, 2019. 2

[10] Gavin R Hunt. Manufacture and use of hook-tools by new caledonian crows. *Nature*, 379(6562): 249–251, 1996. 2

[11] Huaizu Jiang, Xiaojian Ma, Weili Nie, Zhiding Yu, Yuke Zhu, and Anima Anandkumar. Bongard-hoi: Benchmarking few-shot visual reasoning for human-object interactions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19056–19065, 2022. 3, 5, 6

[12] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015. 3

[13] Kwonjoon Lee, Subhransu Maji, Avinash Ravichandran, and Stefano Soatto. Meta-learning with differentiable convex optimization. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10657–10665, 2019. 2, 5

[14] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. *arXiv preprint arXiv:1707.03141*, 2017. 2, 5

[15] Weili Nie, Zhiding Yu, Lei Mao, Ankit B Patel, Yuke Zhu, and Anima Anandkumar. Bongard-logo: A new benchmark for human-level concept learning and reasoning. *Advances in Neural Information Processing Systems (NeurIPS)*, 33:16468–16480, 2020. 3, 5

[16] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, pages 8748–8763. PMLR, 2021. 10

[17] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. *arXiv preprint arXiv:1909.09157*, 2019. 2, 5

[18] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 30, 2017. 2, 5

[19] Marc A Toussaint, Kelsey Rebecca Allen, Kevin A Smith, and Joshua B Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. *Robotics: Science and Systems (RSS)*, 2018. 2

[20] Shreshth Tuli, Rajas Bansal, Rohan Paul, et al. Tango: Commonsense generalization in predicting tool interactions for mobile manipulators. *arXiv preprint arXiv:2105.04556*, 2021. 2

[21] Krist Vaesen. The cognitive bases of human tool use. *Behavioral and brain sciences*, 35(4): 203–218, 2012. 2

[22] Oriol Vinyals, Charles Blundell, Timothy P. Lillicrap, Koray Kavukcuoglu, and Daan Wierstra. Matching networks for one shot learning. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. 2

[23] Yixin Zhu, Yibiao Zhao, and Song Chun Zhu. Understanding tools: Task-oriented object modeling, learning and recognition. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2855–2864, 2015. 2

[24] Yixin Zhu, Tao Gao, Lifeng Fan, Siyuan Huang, Mark Edmonds, Hangxin Liu, Feng Gao, Chi Zhang, Siyuan Qi, Ying Nian Wu, et al. Dark, beyond deep: A paradigm shift to cognitive ai with humanlike common sense. *Engineering*, 6(3):310–345, 2020. 2

A Appendix

A.1 Data Collection

We collect a large-scale image dataset for our benchmark. It includes 40K natural images for 635 various tools, labeled with 113 functionalities in total.

A.1.1 Tool List Generation

We first prompt GPT-3 [4] to generate 100 various purposes of using tools or tool-related concepts, like *tap*, *carve*, and *hammer*. For each of them, we further prompt the model to generate 10 tools that can support the purpose. For example, *nail, screw, anvil, stake, stump, rock, log, brick, concrete block,* and *steel plate* are object instances that can support the tool related-concept *hammer*. After careful selection, we have 113 tool concepts and 635 object instances.

To obtain real photos in high-definition, we use the tool name as the keyword to download images from Unsplash, Pexels, Flickr, and Bing Image.

A.1.2 Data Processing

However, directly downloaded images require processing and filtering. We resize our images to the size of 512×512 pixels by cropping at their centers with bicubic interpolation. To clean up the images, we first use pre-trained CLIP [16] to filter our data. We feed the model with the candidate images with N_t different text prompts, including the image search query and three randomly selected queries. The prompts take the form of `A photo of` **`QUERY`**. We accept the image if the label probability for its query exceeds the threshold τ. In our practice, we use $\tau = 0.70$

A.1.3 Copyright

We stress that our dataset does not own the copyright of the images, but only provides images from the Internet with a list of tools with their functionalities. We only release their metadata with image URLs for use with non-commercial research and/or educational purposes.

A.2 Baseline Experiments Setup

A.2.1 Bongard-Tool Tasks

For each function, we sample 100 Bongard-Tool tasks and obtain 11300 instances. We finally generate 9040/2260 tasks for training/testing in Normal Split (NS) and 9000/2300 tasks for training/testing in Conceptual Generalization Split (CGS).

A.2.2 Implementation Details

We use ResNet50 [8] as the backbone to extract features from the input images. For the Meta-Baseline [5] method, we consider pre-training the ResNet50 on ImageNet [6] dataset (Meta-IN) or training from scratch (Meta-SC). We use the SGD optimizer with a learning rate of 0.001 and train all the models with a batch size of 16 for 150 epochs. For Meta-SC and WReN, we train them for 50 additional epochs because both are not convergent after 150 epochs.

A.3 Contributions

All four authors contributed hard to this project. They all participated in the long, struggling process of topic choosing. Guangyuan and Yuyang mainly contributed to the dataset. Chuyue and Yu Liu carried out baseline experiments with existing methods. The authors managed to come up with a pipeline that can handle this task, but they are still working on a proper way to solve the task, beyond barely manipulating the image features. Guangyuan and Chuyue mainly wrote Secs. 1, 2 and 7, Yuyang mainly wrote Secs. 3 and 6 and the Appendix, Yu Liu and Chuyue mainly wrote Secs. 3 and 4.