Visual Program Distillation: Distilling Tools and Programmatic Reasoning into Vision-Language Models

Yushi Hu[1,2*] **Otilia Stretcu**[1] **Chun-Ta Lu**[1] **Krishnamurthy Viswanathan**[1]
Kenji Hata[1] **Enming Luo**[1] **Ranjay Krishna**[2] **Ariel Fuxman**[1]
[1]Google Research [2]University of Washington
https://visual-program-distillation.github.io/

Abstract

Solving complex visual tasks such as "Who invented the musical instrument on the right?" involves a composition of skills: understanding space, recognizing instruments, and also retrieving prior knowledge. Recent work shows promise by decomposing such tasks using a large language model (LLM) into an executable program that invokes specialized vision models. However, generated programs are error-prone: they omit necessary steps, include spurious ones, and are unable to recover when the specialized models give incorrect outputs. Moreover, they require loading multiple models, incurring high latency and computation costs. We propose ***Visual Program Distillation (VPD)****, an instruction tuning framework that produces a vision-language model (VLM) capable of solving complex visual tasks with a single forward pass. VPD distills the reasoning ability of LLMs by using them to sample multiple candidate programs, which are then executed and verified to identify a correct one. It translates each correct program into a language description of the reasoning steps, which are then distilled into a VLM. Extensive experiments show that VPD improves the VLM's ability to count, understand spatial relations, and reason compositionally. Our VPD-trained PaLI-X outperforms all prior VLMs, achieving state-of-the-art performance across complex vision tasks, including MMBench, OK-VQA, A-OKVQA, TallyQA, POPE, and Hateful Memes. An evaluation with human annotators also confirms that VPD improves model response factuality and consistency. Finally, experiments on content moderation demonstrate that VPD is also helpful for adaptation to real-world applications with limited data.*

1. Introduction

Vision-language models (VLMs) have become the pre-trained backbone for many computer vision tasks [2, 4, 7, 9–11, 34, 39, 41, 60, 63, 74, 79]. Yet, all these models still fall short of solving numerous visual reasoning tasks expected of competent vision models. Even state-of-the-art (SOTA) proprietary vision-language models such as GPT-4V [49] do not perform well on tasks that involve counting and spatial reasoning [71]. They find it difficult to count (TallyQA [1]), to compositionally reason (GQA [26]), and to reason with external knowledge (OK-VQA [44], A-OKVQA [53]). Many of these tasks require VLMs to conduct compositional reasoning, which still remains an unsolved challenge. For instance, answering the question *"Who invented the musical instrument on the right?"* involves a composition of skills: identifying objects, applying spatial reasoning to locate the one on the right, recognizing the musical instrument, and accessing prior knowledge to retrieve the inventor.

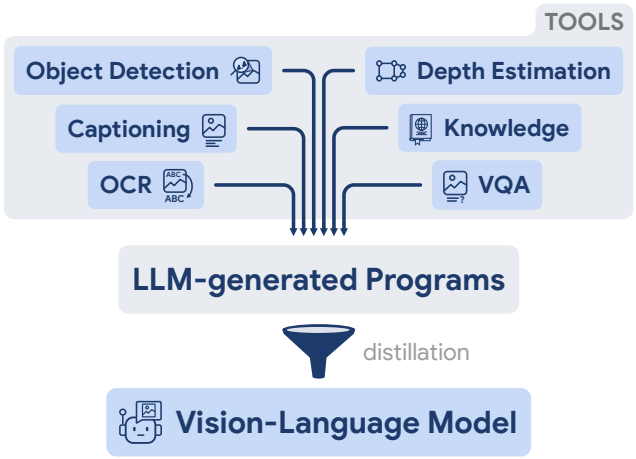

Figure 1. We introduce *Visual Program Distillation (VPD)*, a training framework which leverages LLM-generated programs that make calls to specialist models and tools to distill cross-modal reasoning abilities and specialist skills into multimodal models.

In contrast, large language models (LLMs) have demonstrated remarkable performance at generating code that solves complex and compositional tasks [3, 8, 43, 49, 69]. Several recent papers [16, 20, 25, 57] capitalize on this by prompting LLMs to generate programs where each step corresponds to a reasoning step. The programs invoke specialized "tools" (or specialized vision models) to explicitly

*Work done while interning at Google Research.

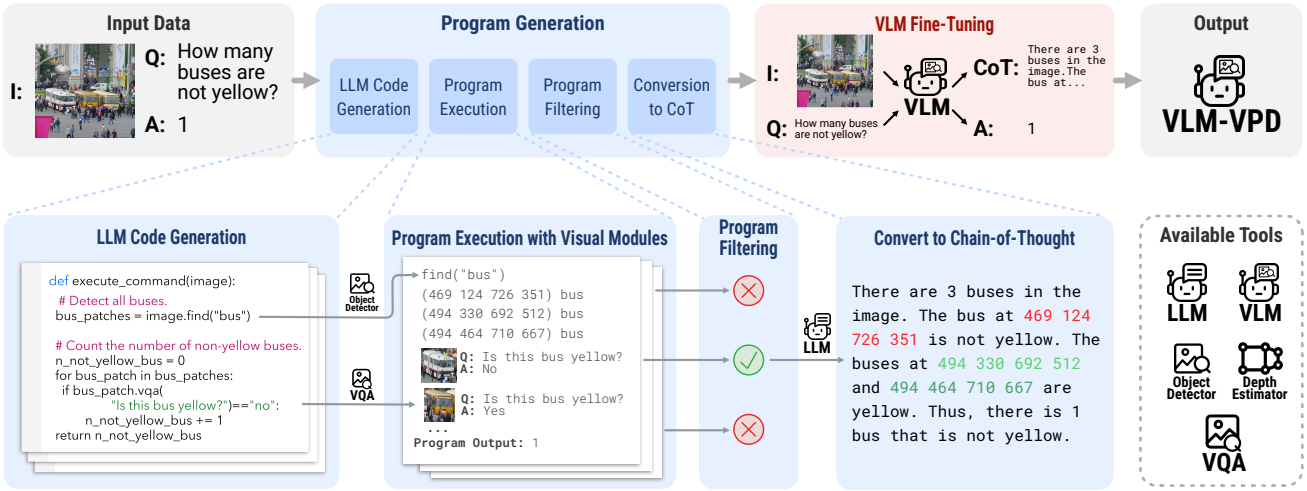

Figure 2. Overview of Visual Program Distillation (VPD). VPD uses an LLM and specialized vision tools to generate faithful chain-of-thought (CoT) training data for vision-language models (VLMs). Given a multimodal input, our 4-step data synthesis pipeline generates a CoT that answers the query. In the example above, our synthesized CoT contains a series of reasoning steps: find all buses, check if each bus is yellow, and aggregate the count into a final answer. The CoT also contains the grounding information given by object detection.

execute each reasoning step. For the question above, the program would call an "object detector" tool to identify and isolate all the objects, a "fine-grained object classification" tool to recognize the musical instrument, and a "knowledge-based question answering" tool to retrieve its inventor.

Although innovative, generating explicit programs is computationally expensive in practice, prone to errors, and still underperforms end-to-end models. Programs require loading and executing multiple tools, leading to high latency and computational cost. Moreover, generated programs may omit necessary steps or include spurious ones. Even when the program is correct, vision model invocations can produce incorrect outputs, from which the overall program cannot recover. Unfortunately, empirical results show that visual programs still fall short of end-to-end fine-tuned models [16].

Another line of work is visual instruction tuning [39], which tries to distill the instruction-following ability of LLMs into VLMs. They prompt LLMs with image captions and bounding box annotations in order to generate queries and answers that are used to fine-tune VLMs [7, 23, 39, 65]. However, this approach has important limitations: image captions can miss fine-grained visual information, and LLM are prone to producing inconsistent outputs for custom vision representations like bounding boxes [18]. As a result, existing instruction-tuned VLMs still struggle with tasks that requires complex visual reasoning [7, 14, 38, 40].

In this work, we present **Visual Program Distillation (VPD)**, a novel distillation method that induces complex cross-modal reasoning capabilities into vision-language models (Fig. 2). As the name suggests, VPD combines two key insights to deliver a training paradigm that surpasses the sum of its parts: It relies on (1) advancements in visual programs that use tools [20] and (2) the recent breakthroughs in distillation through chain-of-thought reasoning [21]. Given a labeled training dataset of complex visual tasks, VPD generates a correct program, and then distills its reasoning steps into vision-language models. To avoid using programs that give the wrong answer, VPD prompts an LLM to generate multiple candidate programs, and executes every one of them. When labeled data is available, it then filters for programs that produce the correct answer upon execution. Therefore, our programs comprise multiple vision tools, are executable, and yield the correct answer when executed. Next, VPD rewrites the correct programs as natural language chain-of-thought instructions and uses step-by-step distillation [21] to inject the reasoning abilities into VLMs.

Our best instruction-tuned model, `PaLI-X-VPD (55B)`, sets a new SOTA result on 8 classical VQA tasks and 2 zero-shot multimodal benchmarks. Our models even outperform the recent SOTA established by `PaLI-X` [10]. Importantly, we conduct a quality evaluation with human raters which shows that `PaLI-X-VPD` generates more consistent and faithful rationales compared to its counterpart trained using instruction-tuning data. In addition, we experiment with `PaLI-3 (5B)`, and show that VPD also improves the performance of smaller-scale models. Finally, experiments on Hateful Memes [29] show that VPD is also helpful for adapting to new tasks, even when no labels are available.

2. Related work

VPD is a general method for improving any vision-language model, and includes automatic program generation and training with chain-of-thought data as steps of the proposed framework. We discuss each of these research areas.

Vision-language models (VLMs). Most recent generative

VLMs share a common structure: a pre-trained visual encoder, a pre-trained LLM, and a connector between the two modalities [2, 4, 9, 34, 39, 41, 60, 63, 74, 79]. The models are trained on large-scale image-text pairs from various tasks to adapt to both modalities. They are also tuned on LLM-generated visual instructions to enable models to follow versatile instructions from users [39]. Some models also include bounding boxes in the pre-training data to improve the VLM on visually grounded tasks [6, 7, 10, 41, 50, 61, 78]. The bounding boxes are usually retrieved from COCO [37] and Visual Genome [32]. Different from prior work, our method does not rely on provided dense annotations. We use LLM-generated code and specialized vision tools to generate our own visual instruction-tuning data.

Visual programming and agents. With the advancement of large language models (LLMs) [3, 5, 13, 49, 59], recent work has started using LLMs as an interface to solving complex reasoning tasks with tools [12, 42, 51, 54, 76], and also as an agent for vision tasks [23, 25, 70, 72]. From this line of work, the most relevant to us are VisProg [20] and ViperGPT [16], which leverage LLMs to generate executable programs with a sequence of invocations to specialized vision tools. They achieve SOTA zero-shot performance on various vision tasks, while being versatile and interpretable.

Training and inference with chain-of-thought. Chain-of-Thought (CoT) [66] has become a popular approach to improving LLM performance. Recent work such as Program-of-Thoughts (PoT) [8] and Faithful CoT [43] further improve this framework by splitting inference into two steps: first generate a program with LLM, then execute the program. This approach achieves better accuracy and reduces hallucinations in the reasoning steps. Moreover, CoT is also used to train language models. Distill step-by-step [21], PaD [80], and SCOTT [62] train smaller language models with CoT generated by LLMs, showing that this can improve model performance and reasoning consistency. Mammoth [68] trains an LLM with a hybrid of CoT and PoT rationales and achieves a SOTA model for math problems.

3. Visual Program Distillation (VPD)

We introduce VPD, a general model-agnostic framework that distills the reasoning power of LLM-generated programs together with the low-level image understanding abilities of vision tools into a single vision-language model (Fig. 2). At its core, VPD consists of two major steps:

1. **Program generation and verification**: Given a textual query q and a visual input i, VPD first generates a program that solves the query by making use of vision modules and tools, then converts the program execution trace into a chain-of-thought c (§3.1).
2. **Distilling step-by-step**: The visual input i, textual query q, and chain-of-thought c produced by the previous step are distilled into a vision-language model (VLM) (§3.2).

3.1. Program generation and verification

Our training data synthesis pipeline is illustrated in the blue boxes of Fig. 2, and consists of four stages. Given a sample (i, q, y) consisting of a visual input i and a textual query q about its content and, when available, its ground truth answer y, we perform the following sequence of steps:

1. *Program generation with LLM*: Given q, we generate a list of k candidate programs $\pi(q) = \{z_1, z_2, ..., z_k\}$, where π represents a program generation function.
2. *Program execution with vision modules*: We execute each program z_i with an execution engine ϕ to obtain its final result $\phi(i, z_i) = \hat{y}_i$. However, during program execution, we maintain the execution trace t_i recording all the intermediate function calls and outputs. At the end of this step, we produce a list of programs, results, and execution traces $\{(z_1, \hat{y}_1, t_1), ..., (z_k, \hat{y}_k, t_k)\}$.
3. *Program filtering*: Among the k candidate programs from the previous step, we keep a single tuple $(z, \hat{y}, t)$ with correct answer.
4. *Converting program execution traces into chain-of-thought rationales*: We rewrite t into a CoT c using an LLM.

We now discuss in detail each of the steps above.

Program generation. We adopt a similar approach with recent work [16, 20] in our program generation step, and use PaLM-2 [3] as LLM to generate candidate programs for a given query q. We prompt PaLM-2 with the same kind of text prompt as used by ViperGPT, which contains a detailed description of the available vision modules, followed by the query q (prompt in Appendix §G). The LLM is expected to directly output a Python function definition that will be executed in the following steps. However, in our experiments, we find that the success rate of top-k programs is much higher than the top-1 program. Therefore, in contrast to prior work, which only samples one program z, when the ground truth answer y is available, we set a temperature T for LLM decoding and sample a list of top-k candidate programs $\{z_1, z_2, ..., z_k\}$ from the LLM. Then, we filter out one correct program z in later steps. As shown in our ablation in §4.2, this becomes crucial to our performance gain. We use $T = 0.5$ and $k = 5$ for all our experiments. For unlabeled data, $k = 1$ and the filtering step can be skipped.

Program execution with vision modules. We use the same execution engine ϕ as ViperGPT [16]. An LLM-generated program z_i is a Python function that takes the visual input i as input. While the LLM outputs the program as a sequence of text, the execution engine ϕ is able to interpret this as a Python program and execute it, to obtain its return result $\hat{y}_i$. Additionally, ϕ also records the execution trace t_i of the program z_i, which keeps a record of all the vision module calls, their inputs, and outputs. We use the following tools for the program: PaLI-X [10] for simple visual queries, PaLI-X

detection (distilled from OWLv2 [47]) for object detection; Google Cloud Depth API [17] for depth estimation, and PaLM-2 [3] for external knowledge.

Program filtering. As we discussed in §1, visual programs are error-prone for a variety of reasons. The program might be wrong, and the execution process can introduce additional errors. To overcome these issues, we employ a program filtering step. We start by sampling top-k programs and attempt to execute each one. During this process, any program that fails execution is instantly discarded. We further filter the remaining programs, the strategy depending on the availability of labeled data. For tasks where human labels are available, we select a single program per input sample whose answer is correct—that is, when the program output $\hat{y}_i$ matches the human label y. In this case our visual program pipeline acts like a CoT rationale annotator. One potential problem is that some task answers are ambiguous. For example, for a question like "Where are the horses?", the answers "mountain" and "mountains" are both correct. We adopt the method in [28] and use an LLM to determine if the program output is correct (details in §G). If there is more than one correct program per question, we select the top scoring one, according to the scores provided by the program-generating LLM ϕ. If no programs pass the test, the rated sample is not wasted—we simply use the correct answer y as supervision in our fine-tuning stage, without an associated CoT. When no human-rated answer is available, we directly use the top scoring executable program[1].

Converting program execution traces into chain-of-thought rationales. After the filtering step, for each visual input i and query q, we will have selected at most one program z together with its execution result $\hat{y}$ and trace t. Since most existing VLMs have been pre-trained with text in natural language and not code, we use an LLM to rewrite the execution trace t into a natural language CoT c for our VLM distillation. Some examples are shown in §B. Concretely, similar to prior work [23, 24], we hand-craft 20 examples of how an input (q, z, t) can be converted into a CoT, and use these as few-shots for prompting PaLM-2 [3], which performs in-context learning and generates CoTs for new examples. We include a concrete example in §G.

3.2. Distilling step-by-step

In this step, we fine-tune a backbone VLM with the training data generated in §3.1, distilling the knowledge and reasoning steps of our generated programs into a single end-to-end model. We do so in a multitask fashion, where the VLM is simultaneously fine-tuned on data synthesized for multiple types tasks (e.g., free-form VQA, multiple choice).

Let f represent our VLM model. While the same VLM architecture could solve all these tasks, it needs to be prompted differently to adapt to the task. Following prior work [39], we manually design instructions for each task. For example, for free-form VQA queries, the instruction is *"Answer with a single word or phrase"*, while for multiple-choice queries we use *"Answer with the option letter from the given choices directly"*. During fine-tuning, we combine the training samples generated for all tasks into a single dataset, so to account for the different types of tasks, we augment each sample with its corresponding task-specific prompt p.

Using these instructions along with the data generated in §3.1, we put together the training dataset $\mathcal{D} = \{(i_j, q_j, \hat{y}_j, c_j, p_j)\}_{j=1}^{N}$, where N is the total number of samples, i_j is the visual input, q_j is the textual query, $\hat{y}_i$ is the visual program output[2], and c_j is the CoT rationale.

We train f to minimize a loss for predicting both the label and the rationale. As shown in the red box of Fig. 2, similar to [21], we treat predicting the output label $\hat{y}_j$ and the rationale c_j as two separate optimization goals. However, since VLMs are open-ended text generation models, they need additional prompting to indicate whether we want a short answer or a long answer that includes a rationale. Therefore, we append the suffix s_c ="*Explain the rationale to answer the question*" at the end of the prompt for generating a CoT, and use the task instruction p_j for short answers. Our optimization objective is:

$$\mathcal{L} = \mathcal{L}_{label} + \mathcal{L}_{rationale} \tag{1}$$

$$= \sum_{j=1}^{N} \ell(f(i_j, q_j, p_j), \hat{y}_j) + \ell(f(i_j, q_j, s_c), c_j) \tag{2}$$

Here ℓ is the cross entropy-loss normalized by sequence length to ensure that the labels (typically short) and rationales (typically long) have similar weights. $\mathcal{L}_{rationale}$ both teaches the VLM to generate faithful reasoning steps similar to program execution traces, and carries more information beyond the label $\hat{y}$ that also helps the VLM in better predicting the label. During test time, the rationale generation is not required. We can adjust the task instruction to directly get the short output label using p, and using s_c to get the human-interpretable reasoning steps if needed.

4. Experiments

In this section, we demonstrate VPD's effectiveness by using it to train a generalist VLM. We attempt this for two VLMs with different scales, `PaLI-3 (5B)` [11] and `PaLI-X (55B)` [10]. Detailed experimental setups are given in §4.1. Qualitatively, models fine-tuned with VPD exhibit the ability to reason step-by-step like a program, as illustrated in Fig. 3, and supported by human evaluation results (§4.3). Quantitative results show that our `PaLI-3-VPD` and `PaLI-X-VPD`

[1] Studying ways to approximate program correctness strategies for unlabeled data is an interesting direction for future work.

[2] When labeled data is available, this is equivalent to the ground truth label y due to our filtering strategy.

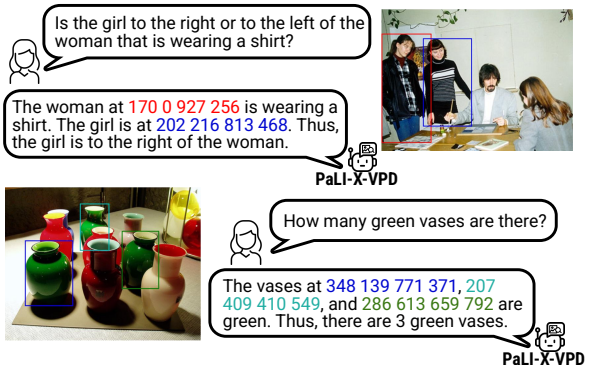

Figure 3. Outputs of PaLI-X trained with VPD. Training with execution traces of filtered programs improves the model's ability to count, understand spatial relations, and reason compositionally. The images are from Visual Genome [32].

achieve new SOTA on a broad range of tasks, on both generalist and per-task fine-tuning settings (§4.2). We also conduct a detailed analysis of the source of performance gain (§4.2). Finally, we conduct human evaluation on the quality of the rationales generated by our models, and compare it with rationales generated by an instruction-tuned model trained without VPD (§4.3).

4.1. Experimental setup

Backbone models. We use two state-of-the-art VLMs, `PaLI-3 (5B)` [11] and `PaLI-X (55B)` [10] as our base models. Both take images and text as input, and generate text as output. For simplicity, we further refer to them as "PaLI model" when we discuss steps performed with each of them individually.

Data for Generalist Models. We fine-tune the pre-trained PaLI model on two types of datasets to make it a *generalist* VLM—that is, a model that performs relatively well on any task without further training on that specific task:

(1) *Multimodal Instruction-Tuning (MMIT) tasks*, created in the spirit of Self-Instruct [64]. An LLM is prompted with image captions, and generates task inputs and the desired outputs about the corresponding image. Details of MMIT tasks are covered in [65]. Note that these tasks cover a wide variety of common use cases, but do not include the specific in-domain tasks used in this work.

(2) *Academic task-oriented VQA tasks*. Since image captions contain only a coarse description of visual information in the image, and may miss details that are important for solving the task. Additionaly, LLM are likely to hallucinate during this data curation process. To further boost the accuracy of our VLMs, we also fine-tune the PaLI models with academic task-oriented VQA tasks. The data mixture covers subsets of a wide variety of VQA tasks, including general VQA (VQAv2 [19]), optical character recognition (OCRVQA [48]), compositional questions and reasoning (GQA [26]), counting (TallyQA [1]), and VQA that involves external knowledge (OK-VQA [45] and A-OKVQA [52]). The tasks contain a textual query and a short expected label. We use the pipeline in §3.1 to synthesize CoT reasoning steps for these labels, and tune the PaLI model with these the loss in Equation 1. Notice that sometimes the pipeline fails to find a program that generates the correct answer. In that case, we set $\mathcal{L}_{rationale}$ to 0 and only keep $\mathcal{L}_{label}$. §4.2 shows how many programs are kept after the filtering stage.

Data for specialist models. While fine-tuning the generalist model with VPD, we only use a subset of each task's training data. To evaluate our model's ability on each individual task, we continue fine-tuning `PaLI-3-VPD` and `PaLI-X-VPD` on each individual task on the training splits.

Training setup. Both PaLI-3 and PaLI-X follow an encoder-decoder architecture, where images are encoded into visual tokens via a visual encoder and then passed into a UL2 model. Due to resource constraint, we use LoRA [22] to fine-tune both PaLI models. Specifically, we add LoRA weights on each linear layer in the attention blocks and the MLP blocks for both the encoder and decoder in the UL2 transformer. More training details are in §C.

Evaluation setup. We evaluate our models on a wide range of tasks, including various VQA tasks and recent zero-shot VLM benchmarks. Noted that A-OKVQA contains two kinds of questions, multiple-choice (MC) and direct answer (DA). We report results on both. TallyQA [1] contains complex counting questions that involve object relationships, attribute identification, and reasoning to get the correct answer. It contains two evaluation partitions, simple and complex. TextVQA [55] focuses reading texts. We include it to evaluate our models on zero-shot tasks. In addition to VQA tasks, we also test our models on two popular VLM benchmarks. POPE [36] focuses on VLM hallucination, containing binary questions of whether or not an object exists in the image. MMBench [40] is a robust and comprehensive VLM benchmark testing a range of fine-grained abilities (e.g., object localization, attribute recognition, spatial relationship). Details of the evaluation sets and metrics are in §C.

Baselines. We refer to the two PaLI models fine-tuned with VPD as `PaLI-3-VPD` and `PaLI-X-VPD`, and compare them with various baselines. To evaluate the effectiveness of VPD, we experiment with removing the synthesized CoT from our data mixture, and train the PaLI models with the exact same hyper-parameters and steps. We call these models as `PaLI-3 Instruct` and `PaLI-X Instruct`. For a fair comparison, these models are also trained with the same supervised loss for predicting the ground truth answers, on the same images and textual queries as our VPD variant. Moreover, we also compare with the most recent SOTA vision-language models. These VLMs are initialized with pre-trained visual encoders and LLMs, and then trained with image-text pairs, LLM-generated data, and academic tasks.

Generalist Models	VQAv2	GQA	OK-VQA	A-OKVQA MC	A-OKVQA DA	TallyQA Simp.	TallyQA Comp.	TextVQA	POPE	MMB
Prior generalist VLMs										
Flamingo (80B) [2]	82.0	-	57.8*	-	-	-	-	57.1	-	-
MiniGPT-4 (Vicuna-13B) [79]	-	43.5	-	67.2	-	-	-	-	-	42.3
InstructBLIP (Vicuna-13B) [14]	-	49.5*	-	-	-	75.2*	57.5*	50.7*	78.9	44.0
Shikra (Vicuna-13B) [7]	77.4	-	47.2	-	-	-	-	-	84.7	58.8
Qwen-VL (9.7B) [4]	78.8	59.3	58.6	-	-	82.6*	65.8*	63.8	-	38.2
Qwen-VL-Chat (9.7B) [4]	78.2	57.5	56.6	-	-	81.1*	64.0*	61.5	-	60.6
mPLUG-Owl2 (8.2B) [74]	79.4	56.1	57.7	-	-	-	-	54.3*	86.2	64.5
LLaVA-1.5 (Vicuna-13B) [38]	80.0	63.3	-	-	-	76.9*	65.4*	61.3*	85.9	67.7
Our instruction-tuned baselines										
PaLI-3-Instruct (5B)	79.9	59.7	56.7	78.3	57.6	81.9	70.4	63.3*	87.7	68.6
PaLI-X-Instruct (55B)	83.6	63.3	64.3	84.1	61.5	85.5	75.4	65.0*	**88.9**	75.0
Our visual program distillation models										
PaLI-3-VPD (5B)	80.4	61.3	57.5	78.5	56.5	83.1	70.9	63.7*	88.6	69.0
PaLI-X-VPD (55B)	**83.9**	**64.9**	**64.6**	**84.5**	**62.7**	**86.2**	**76.6**	**65.4***	88.8	**76.2**

Table 1. Comparison with SOTA generalist VLMs. PaLI-X-VPD outperforms all prior models. VPD improves performance on $8/9$ tasks for both PaLI-3 and PaLI-X, and is particularly effective on counting questions (TallyQA), compostional questions (GQA), and the comprehensive benchmark (MMBench). Underline indicates when PaLI-3-VPD outperforms the Instruct version. * marks tasks unseen during training. POPE and MMBench are zero-shot benchmarks for all models.

4.2. Quantitative results

Generalist model. Table 1 compares VPD with the baselines discussed in §4.1. We infer all answers by open-ended generation with the prompt *"Answer with a single word or phrase."*, using greedy decoding without any constraint on the model's output space. For multiple-choice questions, we run inference with the prompt *"Answer with the option letter from the given choices directly."* and generate the option letter. As Table 1 shows, **PaLI-X-VPD sets the new state-of-the-art on all benchmarks**. Compared with prior generalist VLMs, it achieves significant improvement on MMBench (+8.5), TallyQA complex (+9.8), and A-OKVQA(+9.5 compared with specialist SOTA[14]). PaLI-3-VPD, despite its small architecture size (5B), outperforms all prior models on VQAv2, TallyQA, POPE, and MMBench, including much larger models that use Vicuna-13B as backbones [7, 14, 38]. However, its performance on knowledge-based VQA tasks does not outperform the prior SOTA. Our hypothesis for this is that its language model (3B UL2) is too small to contain all the necessary knowledge needed for correctly solving these tasks.

When compared to their Instruct variants, both PaLI-3-VPD and PaLI-X-VPD outperform on $11/12$ tasks. Specifically, for PaLI-X, VPD obtains a $+1.6$ improvement on GQA (which is heavily focused on compositional questions, spatial relationship, and localization), $+1.2$ on TallyQA for complex questions, and $+1.2$ on MMBench. These results suggest that **VPD is a more effective method for creating instruction-tuning data which enables VLMs to improve their visual reasoning ability**.

Per-Task Fine-Tuning (Specialist). The results for the specialist models are shown in Table 2. **PaLI-X-VPD sets a new SOTA on all benchmarks**. Note how the specialist models tend to have higher scores than the generalist one. We propose several hypotheses for this performance gain: (1) As shown in Table 2, the score gap tends to be larger on free-form VQA tasks. This may be due the fact that human annotations on these datasets have ambiguities. For example, for the question *"Who is looking up?"*, GQA [26] labels are *"man"* or *"woman"* while OK-VQA [45] have more detailed labels, for example, *"worker"* or *"cook"*. Per-task fine-tuning alleviates this annotation ambiguity and lets the model focus on the annotation style of that task. For multiple-choice and counting tasks, the answer has less ambiguity, and the score gaps are much smaller. (2) The performance gap between PaLI-3 (5B) *specialist* and *generalist* is larger than that of PaLI-X. We hypothesize that this shows models with larger scale has more multi-task capacity; (3) Per-task fine-tuning adds more in-domain training data, which generally improves performance.

Analysis: Sampling multiple programs is key to good data generation. Fig. 5 shows the success rate of finding at least one program that passes the filtering stage, when the LLM generates the top-1 or top-5 programs, respectively. There is a dramatic increase in success rate from 1 program to 5: $+45\%$ on GQA and A-OKVQA, $+33\%$ on OK-VQA, and $+10\%$ on TallyQA. This design choice greatly improves our data synthesis efficiency, and, as a consequence, adding more CoT data that requires complex reasoning in our training set. We also conduct an analysis in §E to compare the performance of VPD models with that of the visual programs they are distilled from.

4.3. Human evaluation on rationales

In this section, we focus on the quality of model outputs. We performed this analysis using human annotators, who are asked to evaluate both the correctness of the final answer and the quality of the rationale behind it.

Models. We compare PaLI-X-VPD with PaLI-X Instruct. Among possible baselines, we chose PaLI-X

Specialist Models	GQA	OK-VQA	A-OKVQA				TallyQA	
			Multi-choice		Direct Answer		Simple	Complex
	test-dev	val	val	test	val	test	test	test
InstructBLIP (Vicuna-7B) [14]	-	62.1	75.7	73.4	64.0	62.1	-	-
PaLI-3 (5B) [11]	-	60.1	-	-	-	-	83.3	70.5
PaLI (17B) [9]	-	64.5	-	-	-	-	81.7	70.9
CogVLM (17B) [63]	65.2	64.7	-	-	-	-	-	-
PaLI-X (55B) [10]	-	66.1	-	-	-	-	86.0	75.6
PaLI-3-VPD (5B) *generalist*	61.3	57.5	78.5	-	56.5	-	83.1	70.9
PaLI-3-VPD (5B) *specialist*	64.7	60.3	79.7	76.5	65.5	63.6	83.3	70.8
PaLI-X-VPD (55B) *generalist*	64.9	64.6	84.5	-	62.7	-	**86.2**	**76.6**
PaLI-X-VPD (55B) *specialist*	**67.3**	**66.8**	**85.2**	**80.4**	**71.1**	**68.2**	**86.2**	76.4

Table 2. Comparison of per-task fine-tuning results of specialist models. PaLI-X-VPD sets a new SOTA for all the tasks.

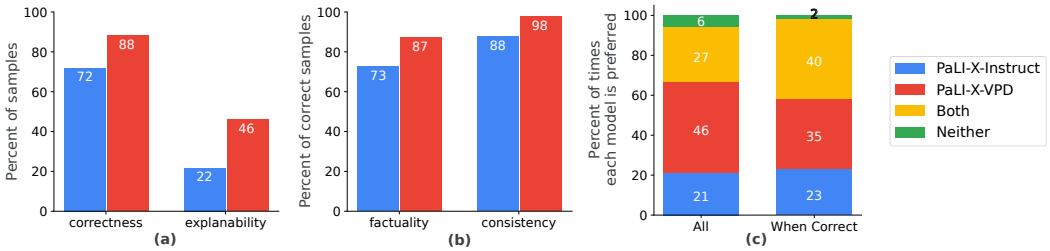

Figure 4. Human evaluation results assessing answer and rational quality for PaLI-X Instruct and PaLI-X VPD: **(a)** percentage of answers that are correct and have explanations; **(b)** rationale factuality and consistency for the answers that contain an explanation; **(c)** preference between the two models, when aggregating across all samples ("All") and across those with correct answers ("When correct").

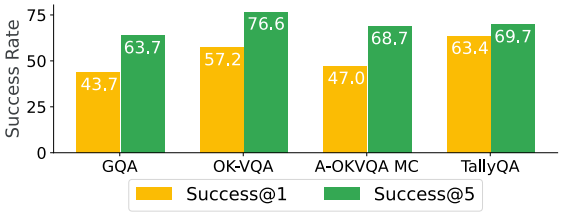

Figure 5. Success rate of top-1 program and top-5 programs on the training set during our data synthesis process.

Instruct because it is trained to generate long-form answers [65], which allows us to assess if it is the quality of PaLI-X-VPD's answers that the annotators prefer, rather than its length. Since PaLI-X Instruct is also instruction-tuned, it can be prompted to provide long answers to alleviate this confounder.

Annotation protocol. We run inference with each of the two models on a combination of 600 samples from GQA (test-dev), A-OKVQA (val), and TallyQA (Simple and Complex) and record their answers and rationales. We then ask 3 human annotators to evaluate each model answer. We use prior work from natural language processing (NLP) as inspiration, and build upon it for selecting the evaluation criteria [75]. Given an image and a query, for each model-generated rationale, we ask human annotators to score the model answers along the following criteria: (1) **correctness**—is the final answer correct? (2) **explainability**—does the model explain its rationale for reaching the final answer? (3) **factuality**—is every step in the rationale factually correct (with respect to the image and external knowledge)? (4) **consistency**—does step and final answer logically follow from the previous ones? Note that a rationale may have the wrong answer while being consistent. We also conduct a **side-by-side comparison**, and ask annotators which of the two answers—the one provided by PaLI-X-VPD or by PaLI-X Instruct—they prefer in terms of quality of the answer and explanation. More details about the annotation protocol are in §D.

Human evaluation results. The results of the evaluation are first averaged among the human raters per sample, then aggregated across samples. **Our PaLI-X-VPD model far outperforms PaLI-X Instruct along all criteria**. Fig. 4 (a) shows the correctness of the final answer provided by each model (i.e. accuracy), and the proportion of the samples where the model provides a CoT rationale to explain its answer. PaLI-X-VPD's gain in accuracy of +16.7% is even more impressive than in evaluations in §4.2 based on benchmark labels—this is because human annotators are better able to assess correctness for ambiguous questions with different possible interpretations (e.g., the model's answer *"In the living room."* to the question *"Where is the couch located?"* was considered correct by annotators, even when the benchmark answer was *"On the right."*). Moreover, the explainability results confirm that our model is able to explain its own answer on +24% more samples than the instruction tuned model. Additionally, Fig. 4 (b) shows **impressive rationale quality**: among the samples where

an explanation is provided, PaLI-X-VPD's rationales are **factual 87.2% of times**, and **consistent 97.8% of times**, a gain of +14.6% and +10%, respectively, when compared to PaLI-X Instruct.

We also asked the annotators which of the two answers given by the two models they prefer. We aggregated these results in two ways: (1) across all samples, (2) across the samples where both models answered the question correctly. The results in Fig. 4 (c) show that PaLI-X-VPD is preferred on 25% more samples than PaLI-X Instruct in the "All" case, and on 12% more samples when both are correct. This suggests that even when PaLI-X-VPD makes mistakes, it still provides a better answer. Such examples are shown in §D. These results confirm that a model fine-tuned with VPD leads to more faithful and consistent answers.

5. Experiments on content moderation

Prior experiments focus on training generalist VLMs. Here, we explore the effectiveness of VPD at quickly adapting models to real-world applications from a different domain than the training data [e.g., 56]. We experiment on Hateful Memes [29], a content moderation dataset where the task is to classify if a meme contains hateful content. The target labels are "yes" or "no", and models are evaluated in terms of classification accuracy and AUC-ROC. We experiment with two settings: *supervised*, in which the models are trained on the provided training set with 8,500 labels, and *unsupervised*, in which no human labels are provided.

Model	Acc	AUC-ROC
Unsupervised / Zero-Shot Methods		
Generated Programs **(ours)**	69.7	70.1
Supervised Methods		
VisualBERT [35]	69.5	75.4
Flamingo (80B) [2]	-	86.6
Previous SOTA [46]	78.8	86.7
PaLI-X-VPD (label-only FT)	77.6	88.0
PaLI-X-VPD (specialist w/ CoT)	**80.8**	**89.2**
Human [29]	84.7	82.7

Table 3. Results on Hateful Memes [29] seen test set. We improve SOTA for both zero-shot and supervised settings. Unsupervised PaLI-X-VPD outperforms strong supervised baselines.

5.1. Unsupervised setting

We experiment with three methods:

1. Generated Program: The program generated by PaLM-2 [3] for solving this task (shown in §F) consists of following main steps: (1) get image description with PaLI-X [10]; (2) use an OCR tool to extract embedded texts; (3) given the image description and OCR texts, ask PaLM-2 to explain if this meme is hateful.
2. PaLI-X-VPD (generalist): In a zero-shot setting, we directly prompt our PaLI-X-VPD (generalist) with: *"The text is <OCR text>. Is this a hateful meme?"* We compute the probability of PaLI-X-VPD generating "yes" or "no" and measure accuracy and AUC-ROC.
3. PaLI-X-VPD (specialist with zero-shot CoT): We follow our VPD pipeline and convert the execution traces of the program into CoTs, and then fine-tune PaLI-X-VPD to output these CoTs. Since no groundtruth labels are available, no filtering is done during the process.

VPD significantly improves performance even when no labels are available. As shown in Table 3, PaLI-X-VPD (generalist) outperforms all other VLMs (61.4%). Interestingly, the generated programs themselves get much higher accuracy (69.7%) than on the previous datasets, perhaps because PaLM-2 is better suited at analyzing a meme than typical VQA datasets. Moreover, our PaLI-X-VPD (specialist) sets a new zero-shot SOTA on Hateful Memes, and even outperforms supervised VisualBERT [35]. Manual inspection of model outputs show a great deal of similarity with the generated code. See examples in §F.

5.2. Supervised setting

We fine-tune PaLI-X-VPD in two ways:

1. Label-only fine-tuning: To establish a strong baseline, we first experiment with the traditional supervised setting, where we fine-tune the model to output "yes" or "no".
2. PaLI-X-VPD (specialist): We use the complete VPD pipeline to train this model. We select an execution trace that leads to the correct label, and use it to tune our VLM.

Results. Our PaLI-X-VPD (specialist) **sets a new SOTA for this task**, with an accuracy of 80.8% and AUC-ROC of 89.2%. It outperforms the label-only fine-tuning baseline and significantly improves the SOTA metrics, **achieving nearly human-level accuracy, and superhuman AUC-ROC**.

6. Conclusion, limitations, and future work

In this paper, we introduced VPD, a framework for distilling the reasoning abilities of LLMs along with the capabilities of vision tools into VLMs. VPD synthesizes training data for VLMs by generating programs that can leverage external tools. We used this technique to fine-tune some of the best existing VLMs (PaLI-3 and PaLI-X) and established new SOTA results on 8 classical VQA and 2 zero-shot multimodal benchmarks. According to human evaluations, VPD-tuned models provide more accurate answers and better rationales. Experiments on the Hateful Memes dataset show how VPD can also adapt models to new real-world domains, even when no labeled data is available, also establishing a new SOTA on this dataset. From our experiments we learnt that both a strength and a weakness of VPD is the quality of the generated programs. Better programs lead to bigger gains with VPD. We discuss multiple ideas on how to improve the program generation in Appendix H, including using additional tools, agents, and fact-checking for multimodal CoTs.

References

[1] Manoj Acharya, Kushal Kafle, and Christopher Kanan. Tallyqa: Answering complex counting questions. In *Proceedings of the AAAI conference on artificial intelligence*, pages 8076–8084, 2019. 1, 5, 14, 15

[2] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, Roman Ring, Eliza Rutherford, Serkan Cabi, Tengda Han, Zhitao Gong, Sina Samangooei, Marianne Monteiro, Jacob Menick, Sebastian Borgeaud, Andrew Brock, Aida Nematzadeh, Sahand Sharifzadeh, Mikolaj Binkowski, Ricardo Barreira, Oriol Vinyals, Andrew Zisserman, and Karen Simonyan. Flamingo: a visual language model for few-shot learning, 2022. 1, 3, 6, 8

[3] Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, Eric Chu, Jonathan H. Clark, Laurent El Shafey, Yanping Huang, Kathy Meier-Hellstern, Gaurav Mishra, Erica Moreira, Mark Omernick, Kevin Robinson, Sebastian Ruder, Yi Tay, Kefan Xiao, Yuanzhong Xu, Yujing Zhang, Gustavo Hernandez Abrego, Junwhan Ahn, Jacob Austin, Paul Barham, Jan Botha, James Bradbury, Siddhartha Brahma, Kevin Brooks, Michele Catasta, Yong Cheng, Colin Cherry, Christopher A. Choquette-Choo, Aakanksha Chowdhery, Clément Crepy, Shachi Dave, Mostafa Dehghani, Sunipa Dev, Jacob Devlin, Mark Díaz, Nan Du, Ethan Dyer, Vlad Feinberg, Fangxiaoyu Feng, Vlad Fienber, Markus Freitag, Xavier Garcia, Sebastian Gehrmann, Lucas Gonzalez, Guy Gur-Ari, Steven Hand, Hadi Hashemi, Le Hou, Joshua Howland, Andrea Hu, Jeffrey Hui, Jeremy Hurwitz, Michael Isard, Abe Ittycheriah, Matthew Jagielski, Wenhao Jia, Kathleen Kenealy, Maxim Krikun, Sneha Kudugunta, Chang Lan, Katherine Lee, Benjamin Lee, Eric Li, Music Li, Wei Li, YaGuang Li, Jian Li, Hyeontaek Lim, Hanzhao Lin, Zhongtao Liu, Frederick Liu, Marcello Maggioni, Aroma Mahendru, Joshua Maynez, Vedant Misra, Maysam Moussalem, Zachary Nado, John Nham, Eric Ni, Andrew Nystrom, Alicia Parrish, Marie Pellat, Martin Polacek, Alex Polozov, Reiner Pope, Siyuan Qiao, Emily Reif, Bryan Richter, Parker Riley, Alex Castro Ros, Aurko Roy, Brennan Saeta, Rajkumar Samuel, Renee Shelby, Ambrose Slone, Daniel Smilkov, David R. So, Daniel Sohn, Simon Tokumine, Dasha Valter, Vijay Vasudevan, Kiran Vodrahalli, Xuezhi Wang, Pidong Wang, Zirui Wang, Tao Wang, John Wieting, Yuhuai Wu, Kelvin Xu, Yunhan Xu, Linting Xue, Pengcheng Yin, Jiahui Yu, Qiao Zhang, Steven Zheng, Ce Zheng, Weikang Zhou, Denny Zhou, Slav Petrov, and Yonghui Wu. Palm 2 technical report, 2023. 1, 3, 4, 8, 13

[4] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-vl: A versatile vision-language model for understanding, localization, text reading, and beyond, 2023. 1, 3, 6

[5] 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*, 33:1877–1901, 2020. 3

[6] Jun Chen, Deyao Zhu, Xiaoqian Shen, Xiang Li, Zechun Liu, Pengchuan Zhang, Raghuraman Krishnamoorthi, Vikas Chandra, Yunyang Xiong, and Mohamed Elhoseiny. Minigpt-v2: large language model as a unified interface for vision-language multi-task learning, 2023. 3

[7] Keqin Chen, Zhao Zhang, Weili Zeng, Richong Zhang, Feng Zhu, and Rui Zhao. Shikra: Unleashing multimodal llm's referential dialogue magic, 2023. 1, 2, 3, 6

[8] Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2022. 1, 3

[9] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish Thapliyal, James Bradbury, Weicheng Kuo, Mojtaba Seyedhosseini, Chao Jia, Burcu Karagol Ayan, Carlos Riquelme, Andreas Steiner, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali: A jointly-scaled multilingual language-image model, 2022. 1, 3, 7

[10] Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, Siamak Shakeri, Mostafa Dehghani, Daniel Salz, Mario Lucic, Michael Tschannen, Arsha Nagrani, Hexiang Hu, Mandar Joshi, Bo Pang, Ceslee Montgomery, Paulina Pietrzyk, Marvin Ritter, AJ Piergiovanni, Matthias Minderer, Filip Pavetic, Austin Waters, Gang Li, Ibrahim Alabdulmohsin, Lucas Beyer, Julien Amelot, Kenton Lee, Andreas Peter Steiner, Yang Li, Daniel Keysers, Anurag Arnab, Yuanzhong Xu, Keran Rong, Alexander Kolesnikov, Mojtaba Seyedhosseini, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali-x: On scaling up a multilingual vision and language model, 2023. 2, 3, 4, 5, 7, 8, 15, 16

[11] Xi Chen, Xiao Wang, Lucas Beyer, Alexander Kolesnikov, Jialin Wu, Paul Voigtlaender, Basil Mustafa, Sebastian Goodman, Ibrahim Alabdulmohsin, Piotr Padlewski, Daniel Salz, Xi Xiong, Daniel Vlasic, Filip Pavetic, Keran Rong, Tianli Yu, Daniel Keysers, Xiaohua Zhai, and Radu Soricut. Pali-3 vision language models: Smaller, faster, stronger, 2023. 1, 4, 5, 7, 15, 16

[12] Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, et al. Binding language models in symbolic languages. *arXiv preprint arXiv:2210.02875*, 2022. 3

[13] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robin-

son, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022. 3

[14] Wenliang Dai, Junnan Li, Dongxu Li, Anthony Meng Huat Tiong, Junqi Zhao, Weisheng Wang, Boyang Li, Pascale Fung, and Steven Hoi. Instructblip: Towards general-purpose vision-language models with instruction tuning, 2023. 2, 6, 7

[15] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Patrick Collier, Alexey Gritsenko, Vighnesh Birodkar, Cristina Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetić, Dustin Tran, Thomas Kipf, Mario Lučić, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters, 2023. 15

[16] Surís Dídac, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*, 2023. 1, 2, 3, 20, 24

[17] Ruofei Du, Eric Turner, Maksym Dzitsiuk, Luca Prasso, Ivo Duarte, Jason Dourgarian, Joao Afonso, Jose Pascoal, Josh Gladstone, Nuno Cruces, Shahram Izadi, Adarsh Kowdle, Konstantine Tsotsos, and David Kim. DepthLab: Real-Time 3D Interaction With Depth Maps for Mobile Augmented Reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology*. ACM, 2020. 4

[18] Weixi Feng, Wanrong Zhu, Tsu-Jui Fu, Varun Jampani, Arjun Reddy Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. Layoutgpt: Compositional visual planning and generation with large language models. *ArXiv*, abs/2305.15393, 2023. 2

[19] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 5, 15

[20] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. 1, 2, 3

[21] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes, 2023. 2, 3, 4

[22] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models, 2021. 5, 16

[23] Yushi* Hu, Hang* Hua, Zhengyuan Yang, Weijia Shi, Noah A Smith, and Jiebo Luo. Promptcap: Prompt-guided task-aware image captioning. *arXiv preprint arXiv:2211.09699*, 2022. 2, 3, 4

[24] Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. In-context learning for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643, Abu Dhabi, United Arab Emirates, 2022. Association for Computational Linguistics. 4

[25] Ziniu Hu, Ahmet Iscen, Chen Sun, Kai-Wei Chang, Yizhou Sun, David A Ross, Cordelia Schmid, and Alireza Fathi. Avis: Autonomous visual information seeking with large language model agent, 2023. 1, 3

[26] Drew A Hudson and Christopher D Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6700–6709, 2019. 1, 5, 6, 14, 15

[27] Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, Pierre-luc Cantin, Clifford Chao, Chris Clark, Jeremy Coriell, Mike Daley, Matt Dau, Jeffrey Dean, Ben Gelb, Tara Vazir Ghaemmaghami, Rajendra Gottipati, William Gulland, Robert Hagmann, C. Richard Ho, Doug Hogberg, John Hu, Robert Hundt, Dan Hurt, Julian Ibarz, Aaron Jaffey, Alek Jaworski, Alexander Kaplan, Harshit Khaitan, Daniel Killebrew, Andy Koch, Naveen Kumar, Steve Lacy, James Laudon, James Law, Diemthu Le, Chris Leary, Zhuyuan Liu, Kyle Lucke, Alan Lundin, Gordon MacKean, Adriana Maggiore, Maire Mahony, Kieran Miller, Rahul Nagarajan, Ravi Narayanaswami, Ray Ni, Kathy Nix, Thomas Norrie, Mark Omernick, Narayana Penukonda, Andy Phelps, Jonathan Ross, Matt Ross, Amir Salek, Emad Samadiani, Chris Severn, Gregory Sizikov, Matthew Snelham, Jed Souter, Dan Steinberg, Andy Swing, Mercedes Tan, Gregory Thorson, Bo Tian, Horia Toma, Erick Tuttle, Vijay Vasudevan, Richard Walter, Walter Wang, Eric Wilcox, and Doe Hyun Yoon. In-datacenter performance analysis of a tensor processing unit. In *Proceedings of the 44th Annual International Symposium on Computer Architecture*. ACM, 2017. 16

[28] Ehsan Kamalloo, Nouha Dziri, Charles L. A. Clarke, and Davood Rafiei. Evaluating open-domain question answering in the era of large language models, 2023. 4, 23

[29] Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes, 2020. 2, 8, 18, 19

[30] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2015. 16

[31] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything, 2023. 24

[32] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations, 2016. 3, 5, 15

[33] Xin Lai, Zhuotao Tian, Yukang Chen, Yanwei Li, Yuhui Yuan, Shu Liu, and Jiaya Jia. Lisa: Reasoning segmentation via large language model, 2023. 24

[34] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *ArXiv*, abs/2301.12597, 2023. 1, 3

[35] Liunian Harold Li, Mark Yatskar, Da Yin, Cho-Jui Hsieh, and Kai-Wei Chang. VisualBERT: A Simple and Performant Baseline for Vision and Language. *ArXiv*, abs/1908.03557, 2019. 8

[36] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models, 2023. 5, 15

[37] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 3

[38] Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. Improved baselines with visual instruction tuning, 2023. 2, 6

[39] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023. 1, 2, 3, 4

[40] Yuan Liu, Haodong Duan, Yuanhan Zhang, Bo Li, Songyang Zhang, Wangbo Zhao, Yike Yuan, Jiaqi Wang, Conghui He, Ziwei Liu, Kai Chen, and Dahua Lin. Mmbench: Is your multi-modal model an all-around player?, 2023. 2, 5, 15

[41] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-io: A unified model for vision, language, and multi-modal tasks. *ArXiv*, abs/2206.08916, 2022. 1, 3

[42] Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. Chameleon: Plug-and-play compositional reasoning with large language models, 2023. 3

[43] Qing Lyu, Shreya Havaldar, Adam Stein, Li Zhang, Delip Rao, Eric Wong, Marianna Apidianaki, and Chris Callison-Burch. Faithful chain-of-thought reasoning, 2023. 1, 3

[44] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. In *Proceedings of the IEEE/cvf conference on computer vision and pattern recognition*, pages 3195–3204, 2019. 1

[45] Kenneth Marino, Mohammad Rastegari, Ali Farhadi, and Roozbeh Mottaghi. Ok-vqa: A visual question answering benchmark requiring external knowledge. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 5, 6, 15

[46] Jingbiao Mei, Jinghong Chen, Weizhe Lin, Bill Byrne, and Marcus Tomalin. Improving hateful memes detection via learning hatefulness-aware embedding space through retrieval-guided contrastive learning, 2023. 8

[47] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection, 2023. 4

[48] Anand Mishra, Shashank Shekhar, Ajeet Kumar Singh, and Anirban Chakraborty. Ocr-vqa: Visual question answering by reading text in images. In *ICDAR*, 2019. 5, 15

[49] OpenAI. Gpt-4 technical report, 2023. 1, 3

[50] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world, 2023. 3

[51] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools, 2023. 3

[52] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge, 2022. 5, 14, 15

[53] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-okvqa: A benchmark for visual question answering using world knowledge. *arXiv preprint arXiv:2206.01718*, 2022. 1

[54] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face, 2023. 3

[55] Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8317–8326, 2019. 5, 15

[56] Otilia Stretcu, Edward Vendrow, Kenji Hata, Krishnamurthy Viswanathan, Vittorio Ferrari, Sasan Tavakkol, Wenlei Zhou, Aditya Avinash, Emming Luo, Neil Gordon Alldrin, et al. Agile modeling: From concept to classifier in minutes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 22323–22334, 2023. 8

[57] Sanjay Subramanian, Medhini Narasimhan, Kushal Khangaonkar, Kevin Yang, Arsha Nagrani, Cordelia Schmid, Andy Zeng, Trevor Darrell, and Dan Klein. Modular visual question answering via code generation, 2023. 1

[58] Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Siamak Shakeri, Dara Bahri, Tal Schuster, Huaixiu Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. UL2: Unifying Language Learning Paradigms, 2022. 15

[59] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian,

Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. 3

[60] Jianfeng Wang, Zhengyuan Yang, Xiaowei Hu, Linjie Li, Kevin Lin, Zhe Gan, Zicheng Liu, Ce Liu, and Lijuan Wang. Git: A generative image-to-text transformer for vision and language. *arXiv preprint arXiv:2205.14100*, 2022. 1, 3

[61] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. *CoRR*, abs/2202.03052, 2022. 3

[62] Peifeng Wang, Zhengyang Wang, Zheng Li, Yifan Gao, Bing Yin, and Xiang Ren. Scott: Self-consistent chain-of-thought distillation, 2023. 3

[63] Weihan Wang, Qingsong Lv, Wenmeng Yu, Wenyi Hong, Ji Qi, Yan Wang, Junhui Ji, Zhuoyi Yang, Lei Zhao, Xixuan Song, Jiazheng Xu, Bin Xu, Juanzi Li, Yuxiao Dong, Ming Ding, and Jie Tang. Cogvlm: Visual expert for pretrained language models, 2023. 1, 3, 7

[64] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions, 2022. 5

[65] Yaqing Wang, Jialin Wu, Tanmaya Dabral, Jiageng Zhang, Geoff Brown, Chun-Ta Lu, Frederick Liu, Yi Liang, Bo Pang, Michael Bendersky, and Radu Soricut. Non-intrusive adaptation: Input-centric parameter-efficient fine-tuning for versatile multimodal modeling, 2023. 2, 5, 7

[66] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2022. 3

[67] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation. 2023. 24

[68] Ge Zhang Yao Fu Wenhao Huang Huan Sun Yu Su Wenhu Chen Xiang Yue, Xingwei Qu. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023. 3

[69] Yiheng Xu, Hongjin Su, Chen Xing, Boyu Mi, Qian Liu, Weijia Shi, Binyuan Hui, Fan Zhou, Yitao Liu, Tianbao Xie, Zhoujun Cheng, Siheng Zhao, Lingpeng Kong, Bailin Wang, Caiming Xiong, and Tao Yu. Lemur: Harmonizing natural language and code for language agents, 2023. 1

[70] Zhengyuan Yang, Zhe Gan, Jianfeng Wang, Xiaowei Hu, Yumao Lu, Zicheng Liu, and Lijuan Wang. An empirical study of gpt-3 for few-shot knowledge-based vqa. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3081–3089, 2022. 3

[71] Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9, 2023. 1

[72] Zhengyuan Yang, Linjie Li, Jianfeng Wang, Kevin Lin, Ehsan Azarnasab, Faisal Ahmed, Zicheng Liu, Ce Liu, Michael Zeng, and Lijuan Wang. Mm-react: Prompting chatgpt for multimodal reasoning and action, 2023. 3

[73] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*, 2022. 24

[74] Qinghao Ye, Haiyang Xu, Jiabo Ye, Ming Yan, Haowei Liu, Qi Qian, Ji Zhang, Fei Huang, and Jingren Zhou. mplug-owl2: Revolutionizing multi-modal large language model with modality collaboration, 2023. 1, 3, 6

[75] Xi Ye and Greg Durrett. The unreliability of explanations in few-shot prompting for textual reasoning. In *Advances in Neural Information Processing Systems*, pages 30378–30392. Curran Associates, Inc., 2022. 7

[76] Andy Zeng, Adrian Wong, Stefan Welker, Krzysztof Choromanski, Federico Tombari, Aveek Purohit, Michael Ryoo, Vikas Sindhwani, Johnny Lee, Vincent Vanhoucke, et al. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv preprint arXiv:2204.00598*, 2022. 3

[77] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training, 2023. 15

[78] Haotian Zhang, Pengchuan Zhang, Xiaowei Hu, Yen-Chun Chen, Liunian Harold Li, Xiyang Dai, Lijuan Wang, Lu Yuan, Jenq-Neng Hwang, and Jianfeng Gao. Glipv2: Unifying localization and vision-language understanding, 2022. 3

[79] Deyao Zhu, Jun Chen, Xiaoqian Shen, Xiang Li, and Mohamed Elhoseiny. Minigpt-4: Enhancing vision-language understanding with advanced large language models, 2023. 1, 3, 6

[80] Xuekai Zhu, Biqing Qi, Kaiyan Zhang, Xingwei Long, and Bowen Zhou. Pad: Program-aided distillation specializes large models in reasoning, 2023. 3