Automatic Flow Engineering on LLM (action model) with LLM (plan model)
Plan is not grounded, therefore is abstract. Grounded-ness here is obtained through fine-tuning with set of functions in-consideration (for data curation process). Beautiful thing here is the simplicity of it. Data curation is 'here are functions, give me queries which could be solved by composing these functions' —> obvious problem is you never gets to the 'representation gaps' area. Second problem is temporal static plans (plan ahead once and keep on implementing them.

Octo-planner: On-device Language Model for Planner-Action Agents

Iterative plan, act, next or decompose. (Idea from a dynamic sub-tasking work from MIT) Also we would like to have 'plans', which is not sequential in the first place here. Plan is more similar to graph. And adaptive change on the graph (decomposition, etc.) is a choice we want to offer to the planning model. A separate actor model is ok, but how about just another heads or even adaptor? Or same model ?

Wei Chen*
Nexa AI & Stanford
Sunnyvale, CA 94086
alexchen@nexa4ai.com

Zhiyuan Li*
Nexa AI & Stanford
Sunnyvale, CA 94086
zack@nexa4ai.com

Zhen Guo*
MIT EECS
Cambridge, MA 02139
zguo0525@mit.edu

Yikang Shen*
MIT-IBM Watson AI Lab
Cambridge, MA 02139
yikang.shen@ibm.com

Abstract

AI agents have become increasingly significant in various domains, enabling autonomous decision-making and problem-solving. To function effectively, these agents require a planning process that determines the best course of action and then executes the planned actions. In this paper, we present an efficient on-device Planner-Action framework that separates planning and action execution into two components: a planner agent, or Octo-planner, optimized for edge devices, and an action agent using the Octopus model for function execution. Octo-planner first responds to user queries by decomposing tasks into a sequence of sub-steps, which are then executed by the Octopus action agent. To optimize performance on resource-constrained devices, we employ model fine-tuning instead of in-context learning, reducing computational costs and energy consumption while improving response times. Our approach involves using GPT-4 to generate diverse planning queries and responses based on available functions, with subsequent validations to ensure data quality. We fine-tune the Phi-3 Mini model on this curated dataset, achieving a 97% success rate in our in-domain test environment. To address multi-domain planning challenges, we developed a multi-LoRA training method that merges weights from LoRAs trained on distinct function subsets. This approach enables flexible handling of complex, multi-domain queries while maintaining computational efficiency on resource-constrained devices. To support further research, we have open-sourced our model weights at https://huggingface.co/NexaAIDev/octopus-planning. For the demo, please refer to https://www.nexa4ai.com/octo-planner#video.

We should never overlook smaller corps because OpenAI used to be a smaller corps. The grounded research they did at early stage is very valuable to us.

plan ≠ execution
↓
abstract reasoning

1 Introduction

Artificial intelligence (AI) agents [1, 2] have significantly transformed various industries by enabling autonomous decision-making and improving operational efficiencies [3, 4, 5, 6, 7]. These agents rely on a critical planning process that involves determining the optimal course of action, executing the planned actions, and summarizing the outcomes. Large Language Models (LLMs) such as Gemini-Pro [8] and GPT-4 [9] have shown potential in this domain. While these models face challenges in executing complex planning tasks at a level comparable to human performance [10, 11], they remain effective in addressing simpler tasks, thereby facilitating practical applications. One such application

*Equal contribution.

If I think of 'planning' which ultimately is not execution (therefore it's abstraction), then the query already feels equivalent to the 3 steps here ...
More interestingly, if we take the intuition of 'eval is simpler than propose', is if we ask a 'evaluator' to generate ideas on how to 'evaluate' on each step, how to supervise on the process. And we could then optimize and select the best performing path (which hopefully is also short and succinct)

Plan-ahead and act on the spot. The Problem we have with VLM is that it can't act on the spot given complex image structure... Wonder how we could've tackled that ...

I am guessing the tools here are API-based stuff, and not visual signal here ...

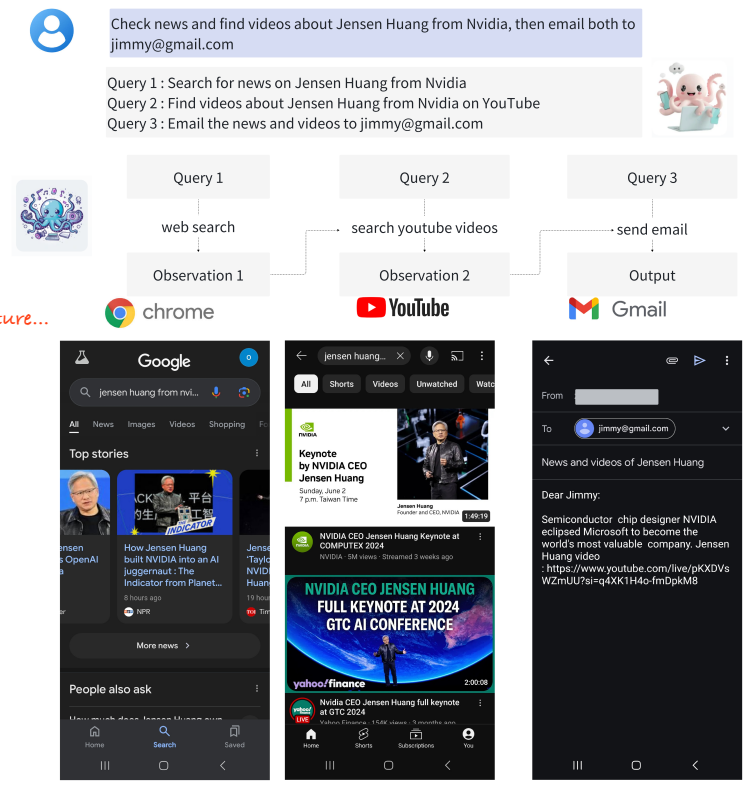

Figure 1: Planner-Action Agent in smartphone using Octopus models

Method-wise, given a 3B sized model, it is Definitely ok to have 2 of them taking care of different aspects of task to reach on-par performance to bigger models. (Nothing wrong with that, too!)

is the emergence of AI assistant tools from companies like MultiOn [12], Simular AI [13], and Adept AI [14], which leverage the capabilities of LLMs to provide intelligent assistance across various domains. Additionally, consumer-oriented AI hardware products, such as Rabbit R1 [15], Humane AI Pin [16], and Limitless Pendant [17], integrate LLMs into user-friendly devices, making intelligent assistance more accessible and driving significant traction.

The success of AI agents depends on the performance of the underlying LLMs. Agents using pre-trained models without fine-tuning on task demonstrations have relatively low success rates, ranging from 12% on desktop applications [18] to 46% on mobile applications [19], while those leveraging fine-tuned models can achieve up to 80% success rate on tasks similar to their training data [20, 21]. However, using LLMs for AI agents is costly due to high computational demands and infrastructure expenses, limiting widespread adoption. The lack of on-device AI agents restricts applications requiring real-time processing, offline functionality, or enhanced privacy.

unable to slot 'tool' into long-term implicit memory leads to exhausting context length with function descriptions. Escaping from that problem results in separation of 2-models, where you would have double context-length to do the same calculations.....

On-device AI agents offer advantages including reduced latency, offline operation, lower costs, and improved data security [22, 23, 24, 25]. While action models like Octopus V2 achieve over 95% accuracy for function calling [26], an on-device planning model is still missing. General agent frameworks use single-model in-context learning, requiring lengthy function descriptions and planning instructions in each prompt. This approach is impractical for on-device models with limited context lengths, causing high latency and battery consumption on edge devices.

Not satisfactory

In this paper, we introduce Octo-planner, an on-device planning agent that addresses the key challenges of efficiency, adaptability, and resource constraints. Our Planner-Action framework separates planning and action execution into two components: a planner agent, or Octo-planner, optimized for edge devices, and an action agent using the Octopus model for function execution. By prioritizing fine-tuning over few-shot prompting, we reduce computational costs and minimize key-value (KV) cache requirements. Our approach uses GPT-4 to generate and validate planning data, which is then used to fine-tune Phi-3 Mini for on-device deployment. In-domain tests demonstrate that this fine-tuning improves planning success rates to 97%. To address multi-domain planning challenges, we developed a multi-LoRA training method that merges weights from LoRAs trained on distinct

function subsets. This enables flexible handling of complex, multi-domain queries while maintaining computational efficiency on resource-constrained devices. By focusing on pre-defined functions for simpler tasks and leveraging fine-tuning, we aim to make AI agents more practical, accessible, and cost-effective for real-world applications.

This work aims to contribute to the ongoing efforts to make AI more accessible and practical for everyday use. By bridging the gap between AI agent potential and edge computing constraints, we seek to facilitate the adoption of intelligent, on-device assistants across various domains. Through open-sourcing our approach, we hope to inspire further innovations in on-device AI, expanding the reach of advanced planning capabilities to a broader range of applications.

2 Related works

Planner agent Language models have become essential in planning agent systems. Proprietary models like OpenAI's assistant API [27] excel in generating strategies based on user queries and available functions. Recent advancements have further expanded the capabilities of language models in planning. The ReAct framework [28] integrates planning and acting for limited action spaces, while research from Alibaba Group [29] highlights the effectiveness of separate planning and action models for complex tasks. In robotics, language models are also increasingly applied to task-level planning [30, 31]. Notable examples include SayCan [32], which uses LLMs to break high-level tasks into concrete sub-tasks, and Video Language Planning (VLP) [33], which enhances long-horizon planning through a text-to-video dynamics model. This broad application of language models in planning systems, from general strategies to specific robotics tasks, underscores their growing importance and adaptability in decision-making processes across diverse domains.

Fine-tuning to replace long context Fine-tuning language models to internalize specific prompts or context information reduces input length and improves efficiency [34, 35]. This approach involves training models on carefully curated, task-specific datasets. For models with limited context windows, this technique is particularly valuable as it enables more efficient query processing without sacrificing response quality. The success of fine-tuning largely depends on the use of diverse, high-quality datasets, which ensure the model can generalize across various prompt phrasings [36, 37, 38, 39]. When implemented effectively, fine-tuning streamlines application-specific interactions, addressing both context length limitations and computational challenges in practical deployments.

Super important to keep on experimenting. We only grow with experimentation as a researcher. Just do the lab works.

LoRA and Multi-LoRA Low-Rank Adaptation (LoRA) efficiently adapts pre-trained language models to specific tasks [40]. Unlike fine-tuning, which updates all parameters, LoRA freezes pre-trained weights and adds trainable low-rank matrices to each layer, significantly reducing trainable parameters and computational demands. Multi-LoRA extends this concept by enabling multiple task-specific adapters to be trained, combined, or switched during inference, allowing a single base model to handle various tasks efficiently [41]. Building on these approaches, researchers have developed several related variants to address different aspects of model adaptation: LoRA+ optimizes learning rates [42], VeRA uses random projections [43], AdaLoRA implements adaptive rank [44], DoRA decomposes weights [45], and Delta-LoRA updates pretrained weights [46]. These variations aim to further refine efficiency or performance in specific scenarios.

3 Method

This section presents our framework for on-device Planner-Action agents. We first describe the integration of planning and action agents for efficient problem-solving. We then detail our approach to dataset design and the training process for the planning agent, including support for extensive functions and a plug-and-play capability for additional function sets. Finally, we outline our benchmark used to evaluate agent performance.

3.1 Planner and action agents framework

Our Planner-Action approach distinguishes itself from general agent frameworks by separating the planning and action execution processes into two components. This separation improves modularity and allows for specialized optimization of each component. The framework operates as follows:

So here, the planning model is grounded on the available functions. Fine-tune move external memory into internal memory state within the planner model. Yet the grounding is partial. Since only the actor model takes care of 'look at output from last step and implement the current step's planned function with corresponding arguments'

Tokenize plans to compress these stuff again.

Planner Phase: Given a user query q, our planning model π_{plan} decomposes the task into a sequence of sub-steps. Formally:

$$\{\tau_1, \tau_2, ..., \tau_n\} = \pi_{plan}(q; F), \tag{1}$$

where F is the set of available function descriptions, and τ_i is the i^{th} execution step. π_{plan} internalizes F during instruction fine-tuning.

Action Phase: For each step in the execution sequence, we employ an action model π_{action}. At step i, given the observation of the current state O_i, the action model performs:

$$O_{i+1} = \pi_{action}(\tau_i, O_i), \tag{2}$$

where O_{i+1} and τ_{i+1} are passed to the next step for continued execution. This iterative process ensures a coherent progression through the task's substeps.

For the action model, we utilize the Octopus model, which is specifically designed for on-device function calling. Figure 2 illustrates the difference between our Planner-Action framework and the single-model approach for LLM agents.

Plans have topology, sequential plan collapses the more interesting bit of it away...

One reason the planning bit is quite disappointing is due to the limited number of steps they took (Bottlenecked by their compute I suppose)

$n < 6$

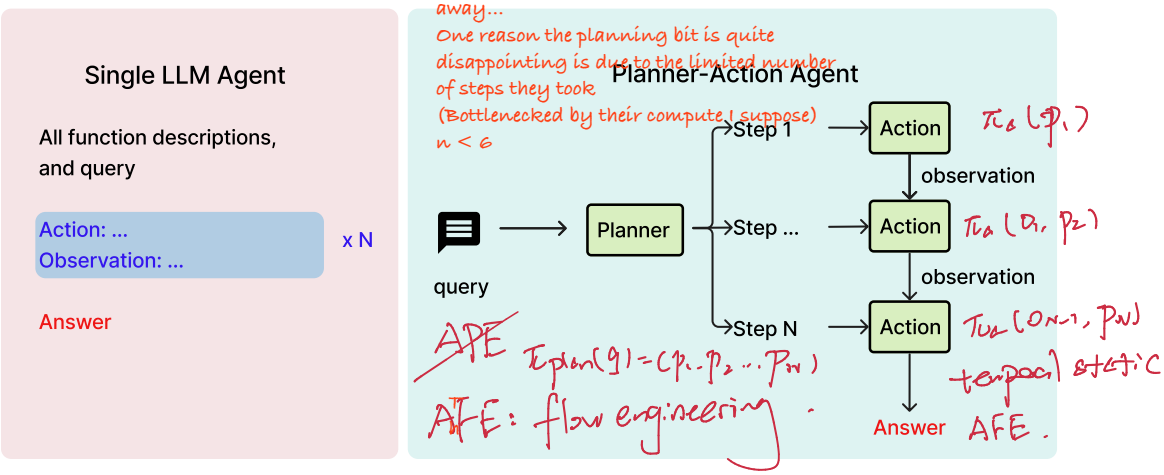

Figure 2: Comparison of Single LLM Agent and Planner-Action Agent frameworks. (left) Single LLM Agent: A unified model performs both task planning and action execution. (right) Planner-Action Agent: A specialized planner model decomposes the task into subtasks, while a separate action model executes each subtask sequentially.

The modular design of our framework offers several advantages:

- **Specialization**: Separating planning and action execution allows optimization of each model for its specific role, enhancing performance in complex tasks.
- **Scalability**: Independent scaling of planning and action capabilities efficiently accommodates varying task complexities.
- **Interpretability**: Explicit separation of phases improves transparency in the decision-making process.
- **Adaptability**: Easier integration of domain-specific knowledge or constraints into either phase without system-wide changes.

3.2 Planning dataset

Our framework uses the Octopus model as the action model, requiring training only for the planner agent. We fine-tune the planner agent with the following dataset format:

<|user|>{user's query}<|end|>

Plan = sequence of queries separated by <nexa_split> token

<|assistant|> {query$_1$}<nexa_split>{query$_2$}<nexa_split>...<nexa_split>{query$_n$}.<|end|>

Alright, so here we are effectively training a 'pathfinder' model, which aims at searching for a sequence of steps which correctly map input state towards output state here. It directly predicts out the entire policy unrolled path in here.

At another level, we are also training an automatic flow-engineer to write out a sequence of prompts for calling a LLM (octopus in this case) N times. Where each time we do pi(p_i), and (p_1, p_2, ..., p_N) is already provided by the planner model from the start.

In AlphaGo, random policy is used to initialize training, we set clear winning condition and randomize paths which consists of actions sampling from pre-defined 'constrained' action space. So, random action sampling is a good way to collect training data, as long as we have a heuristic for termination and filtering (picking the winning paths)

Special tokens like <|user|> and <|assistant|> are used for chat model pretraining but are optional otherwise. We set n as 1-5, based on our finding that most of tasks on mobile app consist of fewer than 5 steps. The dataset generation and curation process includes:

1. **Dataset collection**: Given the available functions F, we use a large language model (GPT-4) to generate diverse queries answerable by these functions. We increase the model's temperature setting to ensure query variety. Responses are then generated in the specified dataset format. Importantly, while function descriptions are used during generation, they are not included in the final dataset. Instead, the planner model internalizes this function information during training.

2. **Data validation**: We employ the same language model as a validation tool to assess the correctness of query-response pairs. Despite some errors in the initial generation process, we found that the model effectively classifies generated content as valid or invalid, allowing us to filter out incorrect outputs and maintain dataset quality. Big claims

Example data points with different number of sub-steps are shown below:

One-step example:

query: Tell me today's stock market.
response: Search for today's stock market news and latest updates<|end|>

Two-step example:

query: Find and email me Jensen Huang's news in English.
response: Find articles about Jensen Huang in English<nexa_split> Email the first article found to myself<|end|>

Three-step example: At this level, you would really hope the phone just 'gets you' — I'd never state myself this clearly anyways ...

query: Find and increase the screen brightness by 20% to better view the quarterly report document before connecting to the office printer via Bluetooth.
response: Find the quarterly report document in the system<nexa_split> Increase screen brightness by 20% to improve visibility<nexa_split> Connect to the office printer via Bluetooth to print the quarterly report<|end|>

Four-step example: Nexa AI, MultiOn. SimulAI.

query: Find and email the project proposal document to project.manager@company.com, then look up the submission deadline and schedule it in my calendar with a reminder.
response: Locate the project proposal document<nexa_split> Send the project proposal via email to the project manager at project.manager@company.com<nexa_split> Find the project proposal submission deadline<nexa_split> Schedule a calendar event for the project proposal submission deadline with a reminder<|end|>

Five-step example:

query: I have a meeting tomorrow morning, please find my presentation then connect to the conference room projector via Bluetooth. After that, increase the screen brightness then take a screenshot of the final summary slide and email it to all participants.
response: Find the presentation for the meeting tomorrow<nexa_split> Connect to the conference room projector via Bluetooth<nexa_split> Increase screen brightness by 20%<nexa_split> Take a screenshot of the final summary slide<nexa_split> Email the screenshot to all meeting participants<|end|>

These folks, once grounded on a real problem, marches very close to AGI already ... Think about this: we include training of the action model, and we have evaluation on action result.
Out of 10 queries, my action model take care of 6, fails on 4. I go back to planner model to 'decompose' these 4, and see how much action model solve, loop doesn't stop until I resolve all the case here, once I do, I add them into the training dataset. MultiOn uses 70B, Nexa sticks with 3B, latter wins, former learns.
Of course, the plan should be a graph in the first place. Combining these ideas, and we have a project proposal ...

For the visualization of the dataset collection, please see Figure 3. Example function descriptions are in Appendix 7.1.

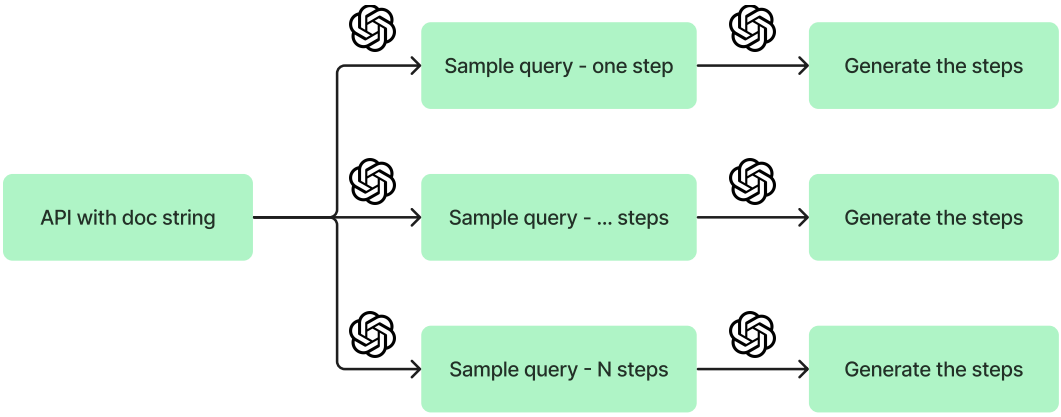

Figure 3: Dataset Collection Process for Planner Model Training. First, we identify the number of steps required, setting N from 1 to 5 in our current case. Next, we generate corresponding queries and steps for each query.

3.3 Benchmark design

Our evaluation relies on a carefully constructed test dataset. This dataset is designed to represent the complexities of real-world planning, employing a multi-stage approach that integrates automated generation, expert validation, and empirical testing.

The process begins with the automated generation of an initial dataset comprising 1,000 data points using GPT-4. These data points then undergo a rigorous quality assurance process to ensure their integrity and relevance. The quality assessment criteria are as follows:

- Each step must correspond to an existing function;
- The sequential order of steps must be correct.

To ensure the reliability of our evaluation, we incorporate an additional phase of manual verification. This phase involves selecting a subset of examples for end-to-end model execution, thereby validating the accuracy of our results and providing a comprehensive assessment of our model's performance.

For the evaluation of our proposed planning model, we employ GPT-4 as an oracle to determine the correctness of the generated plans. This choice is based on empirical observations indicating GPT-4's high proficiency in our specific use case.

4 Experimental Design

Our experimental design assesses the Octo-planner's performance for on-device AI agent planning. We aim to determine the optimal configuration for deploying efficient, accurate planning models on resource-constrained devices while maintaining adaptability to new domains and functions. Our experiments focus on four key areas:

1. Performance and efficiency trade-offs between full fine-tuning and LoRA.
2. Multi-LoRA accuracy in handling different function sets simultaneously.
3. Performance comparison across various base models and sizes.
4. The impact of dataset size on accuracy, ranging from 100 to 1000 training examples.

We conduct supervised fine-tuning on our curated dataset, using Phi-3 Mini and a few other alternatives as the base model. Training includes both full fine-tuning and LoRA techniques. For all experiments, we set the dataset size to be 800 times the number of available functions and perform fine-tuning on an NVIDIA A100 GPU. We use optimized hyperparameters across both techniques: a learning rate of 5×10^{-6}, batch size of 4, a warm-up ratio of 0.2 with 2 epochs. For LoRA, we set the target_modules to be `all-linear`.

Only the linear layers are fine-tuned here ... (why?)
The inferior performance of LoRA compared to full-fine tuning is likely due to this restriction, too ...

5 Results

5.1 Full fine-tuning vs LoRA

Table 1 presents a detailed comparison of full fine-tuning and LoRA approaches for our planning model. Our experiments reveal significant differences in performance across these methods. Full fine-tuning achieves the highest accuracy at 98.1%, demonstrating superior performance. In contrast, LoRA performance depends on rank size. With rank 64 and alpha 256, LoRA achieves 85.1% accuracy, while reducing to rank 16 and alpha 32 drops accuracy to 72.9%. These results highlight the trade-off between model performance and computational efficiency when using LoRA. While full fine-tuning provides better accuracy, LoRA offers a more resource-efficient alternative, with performance varying based on rank configuration.

Table 1: Full fine-tuning vs LoRA benchmark

Training Configuration	Benchmark Accuracy
Full model training	98.1%
LoRA, rank = 64, lora_alpha = 256, target_modules `"all-linear"`	85.1%
LoRA, rank = 16, lora_alpha = 32, target_modules `"all-linear"`	72.9%

This avoids weight collapse

5.2 Multi-LoRA Training and Merging

While LoRA-based training is effective for specific function sets, real-world applications often require handling new or expanded sets of functions. To address this challenge, we propose merging multiple LoRA weights, each trained on distinct function subsets, with the same base model. This approach creates a composite model that combines knowledge from various function sets, offering a scalable solution for complex, multi-domain queries in resource-constrained environments.

To evaluate this method, we constructed a benchmark dataset by randomly selecting functions from each LoRA's domain with equal probability, combining them into workflows. Queries and plans were generated using GPT-4. For example, when testing two merged LoRAs, queries may involve Android functions, E-commerce functions, or both, with equal likelihood.

The code block below shows sample queries from our benchmark dataset and the corresponding inference results from the multi-LoRA merged model:

Android functions only:

query: Find the quarterly report for Q3 and print it for the upcoming board meeting.
response: Locate the quarterly report document for Q3<|nexa_split|>Print the located quarterly report document<|end|>

E-commerce functions only:

query: Find ergonomic office chairs and then buy the selected one using my default shipping address and credit card
response: Search for ergonomic office chairs<nexa_split>Purchase the selected ergonomic office chair using the default shipping address and credit card<|end|>

Android and E-commerce functions:

query: Find ergonomic office chair list doc and print it for my shopping
response: Search for ergonomic office chair list doc<nexa_split>Print the ergonomic office chair list doc <|end|>

Table 2 presents the performance results of our multi-LoRA merging technique. Each individual LoRA was trained with consistent hyperparameters: rank 64, lora_alpha 256, and target_modules set to "all-linear". The single-domain Android function set LoRA achieves 85.1% accuracy. When

merging LoRAs from two domains (Android and E-Commerce), accuracy slightly decreases to 82.2%. Further merging yields lower accuracies: 78.9% for three domains (adding Video Streaming), and 69.7% for four domains (adding Travel). These results reveal a pattern of gradual accuracy decline as we integrate more function sets, with a steeper drop occurring after the third domain is added.

Table 2: Multi-LoRA Benchmark

Training Configuration	Benchmark Accuracy (%)
LoRA for Android	85.1
Merged for Android, E-Commerce	82.2
Merged for Android, E-Commerce, video streaming	78.9
Merged for Android, E-Commerce, video streaming, travelling	69.7

5.3 Full fine-tunning with different base models

Table 3 presents the benchmark accuracy for different base models after full fine-tuning. Google Gemma 2b achieved 85.6% accuracy, while the larger Gemma 7b excelled with 99.7%. Microsoft Phi-3 also performed strongly at 98.1%. These results indicate that our framework adapts well to various on-device LLMs, with larger models generally achieving higher accuracy.

Table 3: Different base model benchmark

Base model	Benchmark Accuracy
Google Gemma 2b	85.6%
Google Gemma 7b	99.7%
Microsoft Phi-3 Mini	98.1%

5.4 Full fine-tuning with different dataset sizes

Our default training dataset contains 1000 data points, evenly distributed across 1-5 step sequences (200 each) to represent varying task complexities. We investigated the impact of dataset size on model performance to optimize function set integration efficiency and address synthetic data generation costs. Table 4 shows the benchmark accuracy for various training dataset sizes:

Table 4: Different training dataset size benchmark

Training Dataset Size	Benchmark Accuracy
1000	98.1%
500	92.5%
250	85.3 %
100	78.1%

The results show a clear correlation between dataset size and accuracy. The full 1000-point dataset achieves 98.1% accuracy, while reducing to 500 data points drops accuracy to 92.5%. Further reductions to 250 and 100 data points result in accuracies of 85.3% and 78.1%, respectively. These findings suggest that for optimal performance, a training dataset of more than 1000 data points is recommended.

6 Conclusion

This paper introduces the Octo-planner, an on-device planning agent designed to work alongside action agents like Octopus V2. By separating planning and action execution, we improve specialization and adaptability. Our approach fine-tunes Phi-3 Mini (a 3.8 billion parameter LLM) to serve as a planning agent capable of running locally on edge devices with 97% success in in-domain tests. We've reduced computational demands, improving latency and battery life, and implemented a multi-LoRA technique for expanding model capabilities without full retraining.

The Octo-planner contributes to addressing AI deployment concerns including data privacy, latency, and offline functionality. It represents an advancement towards practical, sophisticated AI agents for personal devices. By open-sourcing our model weights, we aim to drive innovation in on-device AI, promoting the development of efficient, privacy-respecting applications that enhance daily life without compromising performance or security.

Limitations and future work

Our current model, while effective for specific mobile phone use cases, has limitations in its broader applicability. Unlike frameworks such as ReAct, which alternate between planning steps and executing actions based on real-time feedback, our model conducts all its planning in advance. This upfront planning approach, while efficient for straightforward tasks, may be less adaptable to complex or unpredictable scenarios where conditions might change during execution.

Future work will focus on exploring an iterative planning methodology that refines plans based on real-time observations, improving adaptability to dynamic environments. We also plan to investigate the integration of our planning model with diverse action models, extending its capabilities beyond mobile applications to areas such as IoT, robotics, and smart home systems. These advancements will address current limitations and expand the versatility of our on-device planning model, bridging the gap between efficient, localized AI processing and the complex demands of real-world applications.

References

[1] Nicholas R Jennings and Michael Wooldridge. Applications of intelligent agents. *Agent technology: foundations, applications, and markets*, pages 3–28, 1998.

[2] David L Poole and Alan K Mackworth. *Artificial Intelligence: foundations of computational agents*. Cambridge University Press, 2010.

[3] Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks, 2023.

[4] Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Samuel Stevens, Boshi Wang, Huan Sun, and Yu Su. Mind2web: Towards a generalist agent for the web, 2023.

[5] An Yan, Zhengyuan Yang, Wanrong Zhu, Kevin Lin, Linjie Li, Jianfeng Wang, Jianwei Yang, Yiwu Zhong, Julian McAuley, Jianfeng Gao, Zicheng Liu, and Lijuan Wang. Gpt-4v in wonderland: Large multimodal models for zero-shot smartphone gui navigation, 2023.

[6] Boyuan Zheng, Boyu Gou, Jihyung Kil, Huan Sun, and Yu Su. Gpt-4v(ision) is a generalist web agent, if grounded, 2024.

[7] Jing Yu Koh, Robert Lo, Lawrence Jang, Vikram Duvvur, Ming Chong Lim, Po-Yu Huang, Graham Neubig, Shuyan Zhou, Ruslan Salakhutdinov, and Daniel Fried. Visualwebarena: Evaluating multimodal agents on realistic visual web tasks, 2024.

[8] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul R. Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, Jack Krawczyk, Cosmo Du, Ed Chi, Heng-Tze Cheng, Eric Ni, Purvi Shah, Patrick Kane, Betty Chan, Manaal Faruqui, Aliaksei Severyn, Hanzhao Lin, YaGuang Li, Yong Cheng, Abe Ittycheriah, Mahdis Mahdieh, Mia Chen, Pei Sun, Dustin Tran, Sumit Bagri, Balaji Lakshminarayanan, Jeremiah Liu, Andras Orban, Fabian Güra, Hao Zhou, Xinying Song, Aurelien Boffy, Harish Ganapathy, Steven Zheng, HyunJeong Choe, Ágoston Weisz, Tao Zhu, Yifeng Lu, Siddharth Gopal, Jarrod Kahn, Maciej Kula, Jeff Pitman, Rushin Shah, Emanuel Taropa, Majd Al Merey, Martin Baeuml, Zhifeng Chen, Laurent El Shafey, Yujing Zhang, Olcan Sercinoglu, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, Alexandre Frechette, Charlotte Smith, Laura Culp, Lev Proleev, Yi Luan, Xi Chen, James Lottes, Nathan Schucher, Federico

Lebron, Alban Rrustemi, Natalie Clay, Phil Crone, Tomas Kocisky, Jeffrey Zhao, Bartek Perz, Dian Yu, Heidi Howard, Adam Bloniarz, Jack W. Rae, Han Lu, Laurent Sifre, Marcello Maggioni, Fred Alcober, Dan Garrette, Megan Barnes, Shantanu Thakoor, Jacob Austin, Gabriel Barth-Maron, William Wong, Rishabh Joshi, Rahma Chaabouni, Deeni Fatiha, Arun Ahuja, Gaurav Singh Tomar, Evan Senter, Martin Chadwick, Ilya Kornakov, Nithya Attaluri, Iñaki Iturrate, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Xavier Garcia, Thanumalayan Sankaranarayana Pillai, Jacob Devlin, Michael Laskin, Diego de Las Casas, Dasha Valter, Connie Tao, Lorenzo Blanco, Adrià Puigdomènech Badia, David Reitter, Mianna Chen, Jenny Brennan, Clara Rivera, Sergey Brin, Shariq Iqbal, Gabriela Surita, Jane Labanowski, Abhi Rao, Stephanie Winkler, Emilio Parisotto, Yiming Gu, Kate Olszewska, Ravi Addanki, Antoine Miech, Annie Louis, Denis Teplyashin, Geoff Brown, Elliot Catt, Jan Balaguer, Jackie Xiang, Pidong Wang, Zoe Ashwood, Anton Briukhov, Albert Webson, Sanjay Ganapathy, Smit Sanghavi, Ajay Kannan, Ming-Wei Chang, Axel Stjerngren, Josip Djolonga, Yuting Sun, Ankur Bapna, Matthew Aitchison, Pedram Pejman, Henryk Michalewski, Tianhe Yu, Cindy Wang, Juliette Love, Junwhan Ahn, Dawn Bloxwich, Kehang Han, Peter Humphreys, Thibault Sellam, James Bradbury, Varun Godbole, Sina Samangooei, Bogdan Damoc, Alex Kaskasoli, Sébastien M. R. Arnold, Vijay Vasudevan, Shubham Agrawal, Jason Riesa, Dmitry Lepikhin, Richard Tanburn, Srivatsan Srinivasan, Hyeontaek Lim, Sarah Hodkinson, Pranav Shyam, Johan Ferret, Steven Hand, Ankush Garg, Tom Le Paine, Jian Li, Yujia Li, Minh Giang, Alexander Neitz, Zaheer Abbas, Sarah York, Machel Reid, Elizabeth Cole, Aakanksha Chowdhery, Dipanjan Das, Dominika Rogozińska, Vitaliy Nikolaev, Pablo Sprechmann, Zachary Nado, Lukas Zilka, Flavien Prost, Luheng He, Marianne Monteiro, Gaurav Mishra, Chris Welty, Josh Newlan, Dawei Jia, Miltiadis Allamanis, Clara Huiyi Hu, Raoul de Liedekerke, Justin Gilmer, Carl Saroufim, Shruti Rijhwani, Shaobo Hou, Disha Shrivastava, Anirudh Baddepudi, Alex Goldin, Adnan Ozturel, Albin Cassirer, Yunhan Xu, Daniel Sohn, Devendra Sachan, Reinald Kim Amplayo, Craig Swanson, Dessie Petrova, Shashi Narayan, Arthur Guez, Siddhartha Brahma, Jessica Landon, Miteyan Patel, Ruizhe Zhao, Kevin Villela, Luyu Wang, Wenhao Jia, Matthew Rahtz, Mai Giménez, Legg Yeung, James Keeling, Petko Georgiev, Diana Mincu, Boxi Wu, Salem Haykal, Rachel Saputro, Kiran Vodrahalli, James Qin, Zeynep Cankara, Abhanshu Sharma, Nick Fernando, Will Hawkins, Behnam Neyshabur, Solomon Kim, Adrian Hutter, Priyanka Agrawal, Alex Castro-Ros, George van den Driessche, Tao Wang, Fan Yang, Shuo yiin Chang, Paul Komarek, Ross McIlroy, Mario Lučić, Guodong Zhang, Wael Farhan, Michael Sharman, Paul Natsev, Paul Michel, Yamini Bansal, Siyuan Qiao, Kris Cao, Siamak Shakeri, Christina Butterfield, Justin Chung, Paul Kishan Rubenstein, Shivani Agrawal, Arthur Mensch, Kedar Soparkar, Karel Lenc, Timothy Chung, Aedan Pope, Loren Maggiore, Jackie Kay, Priya Jhakra, Shibo Wang, Joshua Maynez, Mary Phuong, Taylor Tobin, Andrea Tacchetti, Maja Trebacz, Kevin Robinson, Yash Katariya, Sebastian Riedel, Paige Bailey, Kefan Xiao, Nimesh Ghelani, Lora Aroyo, Ambrose Slone, Neil Houlsby, Xuehan Xiong, Zhen Yang, Elena Gribovskaya, Jonas Adler, Mateo Wirth, Lisa Lee, Music Li, Thais Kagohara, Jay Pavagadhi, Sophie Bridgers, Anna Bortsova, Sanjay Ghemawat, Zafarali Ahmed, Tianqi Liu, Richard Powell, Vijay Bolina, Mariko Iinuma, Polina Zablotskaia, James Besley, Da-Woon Chung, Timothy Dozat, Ramona Comanescu, Xiance Si, Jeremy Greer, Guolong Su, Martin Polacek, Raphaël Lopez Kaufman, Simon Tokumine, Hexiang Hu, Elena Buchatskaya, Yingjie Miao, Mohamed Elhawaty, Aditya Siddhant, Nenad Tomasev, Jinwei Xing, Christina Greer, Helen Miller, Shereen Ashraf, Aurko Roy, Zizhao Zhang, Ada Ma, Angelos Filos, Milos Besta, Rory Blevins, Ted Klimenko, Chih-Kuan Yeh, Soravit Changpinyo, Jiaqi Mu, Oscar Chang, Mantas Pajarskas, Carrie Muir, Vered Cohen, Charline Le Lan, Krishna Haridasan, Amit Marathe, Steven Hansen, Sholto Douglas, Rajkumar Samuel, Mingqiu Wang, Sophia Austin, Chang Lan, Jiepu Jiang, Justin Chiu, Jaime Alonso Lorenzo, Lars Lowe Sjösund, Sébastien Cevey, Zach Gleicher, Thi Avrahami, Anudhyan Boral, Hansa Srinivasan, Vittorio Selo, Rhys May, Konstantinos Aisopos, Léonard Hussenot, Livio Baldini Soares, Kate Baumli, Michael B. Chang, Adrià Recasens, Ben Caine, Alexander Pritzel, Filip Pavetic, Fabio Pardo, Anita Gergely, Justin Frye, Vinay Ramasesh, Dan Horgan, Kartikeya Badola, Nora Kassner, Subhrajit Roy, Ethan Dyer, Víctor Campos Campos, Alex Tomala, Yunhao Tang, Dalia El Badawy, Elspeth White, Basil Mustafa, Oran Lang, Abhishek Jindal, Sharad Vikram, Zhitao Gong, Sergi Caelles, Ross Hemsley, Gregory Thornton, Fangxiaoyu Feng, Wojciech Stokowiec, Ce Zheng, Phoebe Thacker, Çağlar Ünlü, Zhishuai Zhang, Mohammad Saleh, James Svensson, Max Bileschi, Piyush Patil, Ankesh Anand, Roman Ring, Katerina Tsihlas, Arpi Vezer, Marco Selvi, Toby

Shevlane, Mikel Rodriguez, Tom Kwiatkowski, Samira Daruki, Keran Rong, Allan Dafoe, Nicholas FitzGerald, Keren Gu-Lemberg, Mina Khan, Lisa Anne Hendricks, Marie Pellat, Vladimir Feinberg, James Cobon-Kerr, Tara Sainath, Maribeth Rauh, Sayed Hadi Hashemi, Richard Ives, Yana Hasson, Eric Noland, Yuan Cao, Nathan Byrd, Le Hou, Qingze Wang, Thibault Sottiaux, Michela Paganini, Jean-Baptiste Lespiau, Alexandre Moufarek, Samer Hassan, Kaushik Shivakumar, Joost van Amersfoort, Amol Mandhane, Pratik Joshi, Anirudh Goyal, Matthew Tung, Andrew Brock, Hannah Sheahan, Vedant Misra, Cheng Li, Nemanja Rakićević, Mostafa Dehghani, Fangyu Liu, Sid Mittal, Junhyuk Oh, Seb Noury, Eren Sezener, Fantine Huot, Matthew Lamm, Nicola De Cao, Charlie Chen, Sidharth Mudgal, Romina Stella, Kevin Brooks, Gautam Vasudevan, Chenxi Liu, Mainak Chain, Nivedita Melinkeri, Aaron Cohen, Venus Wang, Kristie Seymore, Sergey Zubkov, Rahul Goel, Summer Yue, Sai Krishnakumaran, Brian Albert, Nate Hurley, Motoki Sano, Anhad Mohananey, Jonah Joughin, Egor Filonov, Tomasz Kępa, Yomna Eldawy, Jiawern Lim, Rahul Rishi, Shirin Badiezadegan, Taylor Bos, Jerry Chang, Sanil Jain, Sri Gayatri Sundara Padmanabhan, Subha Puttagunta, Kalpesh Krishna, Leslie Baker, Norbert Kalb, Vamsi Bedapudi, Adam Kurzrok, Shuntong Lei, Anthony Yu, Oren Litvin, Xiang Zhou, Zhichun Wu, Sam Sobell, Andrea Siciliano, Alan Papir, Robby Neale, Jonas Bragagnolo, Tej Toor, Tina Chen, Valentin Anklin, Feiran Wang, Richie Feng, Milad Gholami, Kevin Ling, Lijuan Liu, Jules Walter, Hamid Moghaddam, Arun Kishore, Jakub Adamek, Tyler Mercado, Jonathan Mallinson, Siddhinita Wandekar, Stephen Cagle, Eran Ofek, Guillermo Garrido, Clemens Lombriser, Maksim Mukha, Botu Sun, Hafeezul Rahman Mohammad, Josip Matak, Yadi Qian, Vikas Peswani, Pawel Janus, Quan Yuan, Leif Schelin, Oana David, Ankur Garg, Yifan He, Oleksii Duzhyi, Anton Älgmyr, Timothée Lottaz, Qi Li, Vikas Yadav, Luyao Xu, Alex Chinien, Rakesh Shivanna, Aleksandr Chuklin, Josie Li, Carrie Spadine, Travis Wolfe, Kareem Mohamed, Subhabrata Das, Zihang Dai, Kyle He, Daniel von Dincklage, Shyam Upadhyay, Akanksha Maurya, Luyan Chi, Sebastian Krause, Khalid Salama, Pam G Rabinovitch, Pavan Kumar Reddy M, Aarush Selvan, Mikhail Dektiarev, Golnaz Ghiasi, Erdem Guven, Himanshu Gupta, Boyi Liu, Deepak Sharma, Idan Heimlich Shtacher, Shachi Paul, Oscar Akerlund, François-Xavier Aubet, Terry Huang, Chen Zhu, Eric Zhu, Elico Teixeira, Matthew Fritze, Francesco Bertolini, Liana-Eleonora Marinescu, Martin Bölle, Dominik Paulus, Khyatti Gupta, Tejasi Latkar, Max Chang, Jason Sanders, Roopa Wilson, Xuewei Wu, Yi-Xuan Tan, Lam Nguyen Thiet, Tulsee Doshi, Sid Lall, Swaroop Mishra, Wanming Chen, Thang Luong, Seth Benjamin, Jasmine Lee, Ewa Andrejczuk, Dominik Rabiej, Vipul Ranjan, Krzysztof Styrc, Pengcheng Yin, Jon Simon, Malcolm Rose Harriott, Mudit Bansal, Alexei Robsky, Geoff Bacon, David Greene, Daniil Mirylenka, Chen Zhou, Obaid Sarvana, Abhimanyu Goyal, Samuel Andermatt, Patrick Siegler, Ben Horn, Assaf Israel, Francesco Pongetti, Chih-Wei "Louis" Chen, Marco Selvatici, Pedro Silva, Kathie Wang, Jackson Tolins, Kelvin Guu, Roey Yogev, Xiaochen Cai, Alessandro Agostini, Maulik Shah, Hung Nguyen, Noah Ó Donnaile, Sébastien Pereira, Linda Friso, Adam Stambler, Adam Kurzrok, Chenkai Kuang, Yan Romanikhin, Mark Geller, ZJ Yan, Kane Jang, Cheng-Chun Lee, Wojciech Fica, Eric Malmi, Qijun Tan, Dan Banica, Daniel Balle, Ryan Pham, Yanping Huang, Diana Avram, Hongzhi Shi, Jasjot Singh, Chris Hidey, Niharika Ahuja, Pranab Saxena, Dan Dooley, Srividya Pranavi Potharaju, Eileen O'Neill, Anand Gokulchandran, Ryan Foley, Kai Zhao, Mike Dusenberry, Yuan Liu, Pulkit Mehta, Ragha Kotikalapudi, Chalence Safranek-Shrader, Andrew Goodman, Joshua Kessinger, Eran Globen, Prateek Kolhar, Chris Gorgolewski, Ali Ibrahim, Yang Song, Ali Eichenbaum, Thomas Brovelli, Sahitya Potluri, Preethi Lahoti, Cip Baetu, Ali Ghorbani, Charles Chen, Andy Crawford, Shalini Pal, Mukund Sridhar, Petru Gurita, Asier Mujika, Igor Petrovski, Pierre-Louis Cedoz, Chenmei Li, Shiyuan Chen, Niccolò Dal Santo, Siddharth Goyal, Jitesh Punjabi, Karthik Kappaganthu, Chester Kwak, Pallavi LV, Sarmishta Velury, Himadri Choudhury, Jamie Hall, Premal Shah, Ricardo Figueira, Matt Thomas, Minjie Lu, Ting Zhou, Chintu Kumar, Thomas Jurdi, Sharat Chikkerur, Yenai Ma, Adams Yu, Soo Kwak, Victor Ähdel, Sujeevan Rajayogam, Travis Choma, Fei Liu, Aditya Barua, Colin Ji, Ji Ho Park, Vincent Hellendoorn, Alex Bailey, Taylan Bilal, Huanjie Zhou, Mehrdad Khatir, Charles Sutton, Wojciech Rzadkowski, Fiona Macintosh, Konstantin Shagin, Paul Medina, Chen Liang, Jinjing Zhou, Pararth Shah, Yingying Bi, Attila Dankovics, Shipra Banga, Sabine Lehmann, Marissa Bredesen, Zifan Lin, John Eric Hoffmann, Jonathan Lai, Raynald Chung, Kai Yang, Nihal Balani, Arthur Bražinskas, Andrei Sozanschi, Matthew Hayes, Héctor Fernández Alcalde, Peter Makarov, Will Chen, Antonio Stella, Liselotte Snijders, Michael Mandl, Ante Kärrman, Paweł Nowak, Xinyi Wu, Alex Dyck, Krishnan Vaidyanathan, Raghavender R, Jessica Mallet, Mitch Rudominer, Eric Johnston, Sushil Mittal, Akhil Udathu, Janara Christensen, Vishal Verma, Zach Irving, Andreas Santucci, Gamaleldin

Elsayed, Elnaz Davoodi, Marin Georgiev, Ian Tenney, Nan Hua, Geoffrey Cideron, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Dylan Scandinaro, Heinrich Jiang, Jasper Snoek, Mukund Sundararajan, Xuezhi Wang, Zack Ontiveros, Itay Karo, Jeremy Cole, Vinu Rajashekhar, Lara Tumeh, Eyal Ben-David, Rishub Jain, Jonathan Uesato, Romina Datta, Oskar Bunyan, Shimu Wu, John Zhang, Piotr Stanczyk, Ye Zhang, David Steiner, Subhajit Naskar, Michael Azzam, Matthew Johnson, Adam Paszke, Chung-Cheng Chiu, Jaume Sanchez Elias, Afroz Mohiuddin, Faizan Muhammad, Jin Miao, Andrew Lee, Nino Vieillard, Jane Park, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Luowei Zhou, Jonathan Evens, William Isaac, Geoffrey Irving, Edward Loper, Michael Fink, Isha Arkatkar, Nanxin Chen, Izhak Shafran, Ivan Petrychenko, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Peter Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Evan Palmer, Paul Suganthan, Alfonso Castaño, Irene Giannoumis, Wooyeol Kim, Mikołaj Rybiński, Ashwin Sreevatsa, Jennifer Prendki, David Soergel, Adrian Goedeckemeyer, Willi Gierke, Mohsen Jafari, Meenu Gaba, Jeremy Wiesner, Diana Gage Wright, Yawen Wei, Harsha Vashisht, Yana Kulizhskaya, Jay Hoover, Maigo Le, Lu Li, Chimezie Iwuanyanwu, Lu Liu, Kevin Ramirez, Andrey Khorlin, Albert Cui, Tian LIN, Marcus Wu, Ricardo Aguilar, Keith Pallo, Abhishek Chakladar, Ginger Perng, Elena Allica Abellan, Mingyang Zhang, Ishita Dasgupta, Nate Kushman, Ivo Penchev, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnapalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Daniel Andor, Pedro Valenzuela, Minnie Lui, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Cassidy Hardin, Lucas Dixon, Lili Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Dayou Du, Dan McKinnon, Natasha Antropova, Tolga Bolukbasi, Orgad Keller, David Reid, Daniel Finchelstein, Maria Abi Raad, Remi Crocker, Peter Hawkins, Robert Dadashi, Colin Gaffney, Ken Franko, Anna Bulanova, Rémi Leblond, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Mark Omernick, Colton Bishop, Rachel Sterneck, Rohan Jain, Jiawei Xia, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Daniel J. Mankowitz, Alex Polozov, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Matthieu Geist, Ser tan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp, Christopher Yew, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Kathy Wu, David Miller, Nicolas Sonnerat, Denis Vnukov, Rory Greig, Jennifer Beattie, Emily Caveness, Libin Bai, Julian Eisenschlos, Alex Korchemniy, Tomy Tsai, Mimi Jasarevic, Weize Kong, Phuong Dao, Zeyu Zheng, Frederick Liu, Fan Yang, Rui Zhu, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Daniel Toyama, Evan Rosen, Sasan Tavakkol, Linting Xue, Chen Elkind, Oliver Woodman, John Carpenter, George Papamakarios, Rupert Kemp, Sushant Kafle, Tanya Grunina, Rishika Sinha, Alice Talbert, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Saaber Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Yeqing Li, Nir Levine, Ariel Stolovich, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Charlie Deck, Hyo Lee, Zonglin Li, Kyle Levin, Raphael Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Sho Arora, Christy Koh, Soheil Hassas Yeganeh, Siim Põder, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Zhiyu Liu, Anmol Gulati, Jasmine Liu, Xinyu Ye, Bart Chrzaszcz, Lily Wang, Nikhil Sethi, Tianrun Li, Ben Brown, Shreya Singh, Wei Fan, Aaron Parisi, Joe Stanton, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Guillaume Desjardins, Marco Cornero, Brona Robenek, Bhavishya Mittal, Ben Albrecht, Ashish Shenoy, Fedor Moiseev, Henrik Jacobsson, Alireza Ghaffarkhah, Morgane Rivière, Alanna Walton, Clément Crepy, Alicia Parrish, Zongwei Zhou, Clement Farabet, Carey Radebaugh, Praveen Srinivasan, Claudia van der Salm, Andreas Fidjeland, Salvatore Scellato, Eri Latorre-Chimoto, Hanna Klimczak-Plucińska, David Bridson, Dario de Cesare, Tom Hudson, Piermaria Mendolicchio, Lexi Walker, Alex Morris, Matthew Mauger, Alexey Guseynov, Alison Reid, Seth Odoom, Lucia Loher, Victor Cotruta, Madhavi Yenugula, Dominik Grewe, Anastasia Petrushkina, Tom Duerig, Antonio Sanchez, Steve Yadlowsky, Amy Shen, Amir Globerson, Lynette Webb, Sahil Dua, Dong Li, Surya Bhupatiraju, Dan Hurt, Haroon Qureshi, Ananth Agarwal, Tomer Shani, Matan Eyal, Anuj Khare, Shreyas Rammohan Belle, Lei Wang, Chetan

Tekur, Mihir Sanjay Kale, Jinliang Wei, Ruoxin Sang, Brennan Saeta, Tyler Liechty, Yi Sun, Yao Zhao, Stephan Lee, Pandu Nayak, Doug Fritz, Manish Reddy Vuyyuru, John Aslanides, Nidhi Vyas, Martin Wicke, Xiao Ma, Evgenii Eltyshev, Nina Martin, Hardie Cate, James Manyika, Keyvan Amiri, Yelin Kim, Xi Xiong, Kai Kang, Florian Luisier, Nilesh Tripuraneni, David Madras, Mandy Guo, Austin Waters, Oliver Wang, Joshua Ainslie, Jason Baldridge, Han Zhang, Garima Pruthi, Jakob Bauer, Feng Yang, Riham Mansour, Jason Gelman, Yang Xu, George Polovets, Ji Liu, Honglong Cai, Warren Chen, XiangHai Sheng, Emily Xue, Sherjil Ozair, Christof Angermueller, Xiaowei Li, Anoop Sinha, Weiren Wang, Julia Wiesinger, Emmanouil Koukoumidis, Yuan Tian, Anand Iyer, Madhu Gurumurthy, Mark Goldenson, Parashar Shah, MK Blake, Hongkun Yu, Anthony Urbanowicz, Jennimaria Palomaki, Chrisantha Fernando, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Denny Zhou, Komal Jalan, Dinghua Li, Blake Hechtman, Parker Schuh, Milad Nasr, Kieran Milan, Vladimir Mikulik, Juliana Franco, Tim Green, Nam Nguyen, Joe Kelley, Aroma Mahendru, Andrea Hu, Joshua Howland, Ben Vargas, Jeffrey Hui, Kshitij Bansal, Vikram Rao, Rakesh Ghiya, Emma Wang, Ke Ye, Jean Michel Sarr, Melanie Moranski Preston, Madeleine Elish, Steve Li, Aakash Kaku, Jigar Gupta, Ice Pasupat, Da-Cheng Juan, Milan Someswar, Tejvi M., Xinyun Chen, Aida Amini, Alex Fabrikant, Eric Chu, Xuanyi Dong, Amruta Muthal, Senaka Buthpitiya, Sarthak Jauhari, Nan Hua, Urvashi Khandelwal, Ayal Hitron, Jie Ren, Larissa Rinaldi, Shahar Drath, Avigail Dabush, Nan-Jiang Jiang, Harshal Godhia, Uli Sachs, Anthony Chen, Yicheng Fan, Hagai Taitelbaum, Hila Noga, Zhuyun Dai, James Wang, Chen Liang, Jenny Hamer, Chun-Sung Ferng, Chenel Elkind, Aviel Atias, Paulina Lee, Vít Listík, Mathias Carlen, Jan van de Kerkhof, Marcin Pikus, Krunoslav Zaher, Paul Müller, Sasha Zykova, Richard Stefanec, Vitaly Gatsko, Christoph Hirnschall, Ashwin Sethi, Xingyu Federico Xu, Chetan Ahuja, Beth Tsai, Anca Stefanoiu, Bo Feng, Keshav Dhandhania, Manish Katyal, Akshay Gupta, Atharva Parulekar, Divya Pitta, Jing Zhao, Vivaan Bhatia, Yashodha Bhavnani, Omar Alhadlaq, Xiaolin Li, Peter Danenberg, Dennis Tu, Alex Pine, Vera Filippova, Abhipso Ghosh, Ben Limonchik, Bhargava Urala, Chaitanya Krishna Lanka, Derik Clive, Yi Sun, Edward Li, Hao Wu, Kevin Hongtongsak, Ianna Li, Kalind Thakkar, Kuanysh Omarov, Kushal Majmundar, Michael Alverson, Michael Kucharski, Mohak Patel, Mudit Jain, Maksim Zabelin, Paolo Pelagatti, Rohan Kohli, Saurabh Kumar, Joseph Kim, Swetha Sankar, Vineet Shah, Lakshmi Ramachandruni, Xiangkai Zeng, Ben Bariach, Laura Weidinger, Tu Vu, Amar Subramanya, Sissie Hsiao, Demis Hassabis, Koray Kavukcuoglu, Adam Sadovsky, Quoc Le, Trevor Strohman, Yonghui Wu, Slav Petrov, Jeffrey Dean, and Oriol Vinyals. Gemini: A family of highly capable multimodal models, 2024.

[9] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob

McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.

[10] Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. Travelplanner: A benchmark for real-world planning with language agents, 2024.

[11] Huaixiu Steven Zheng, Swaroop Mishra, Hugh Zhang, Xinyun Chen, Minmin Chen, Azade Nova, Le Hou, Heng-Tze Cheng, Quoc V. Le, Ed H. Chi, and Denny Zhou. Natural plan: Benchmarking llms on natural language planning, 2024.

[12] MultiOn AI. Multion ai, 2024.

[13] Simular AI. Simular ai, 2024.

[14] Adept AI. Adept ai: Useful general intelligence, 2024.

[15] Rabbit Inc. Rabbit: Your pocket companion, 2024.

[16] Humane Inc. Humane: Wearable ai, 2024.

[17] Limitless AI. Limitless ai, 2024.

[18] Tianbao Xie, Danyang Zhang, Jixuan Chen, Xiaochuan Li, Siheng Zhao, Ruisheng Cao, Toh Jing Hua, Zhoujun Cheng, Dongchan Shin, Fangyu Lei, Yitao Liu, Yiheng Xu, Shuyan Zhou, Silvio Savarese, Caiming Xiong, Victor Zhong, and Tao Yu. Osworld: Benchmarking multimodal agents for open-ended tasks in real computer environments, 2024.

[19] William E Bishop, Alice Li, Christopher Rawles, and Oriana Riva. Latent state estimation helps ui agents to reason, 2024.

[20] Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. Webgpt: Browser-assisted question-answering with human feedback, 2022.

[21] Izzeddin Gur, Hiroki Furuta, Austin Huang, Mustafa Safdari, Yutaka Matsuo, Douglas Eck, and Aleksandra Faust. A real-world webagent with planning, long context understanding, and program synthesis, 2024.

[22] Wei Yu, Fan Liang, Xiaofei He, William Grant Hatcher, Chao Lu, Jie Lin, and Xinyu Yang. A survey on the edge computing for the internet of things. *IEEE access*, 6:6900–6919, 2017.

[23] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2024.

[24] Abdulmalik Alwarafy, Khaled A Al-Thelaya, Mohamed Abdallah, Jens Schneider, and Mounir Hamdi. A survey on security and privacy issues in edge-computing-assisted internet of things. *IEEE Internet of Things Journal*, 8(6):4004–4022, 2020.

[25] Pasika Ranaweera, Anca Delia Jurcut, and Madhusanka Liyanage. Survey on multi-access edge computing security and privacy. *IEEE Communications Surveys & Tutorials*, 23(2):1078–1124, 2021.

[26] Wei Chen and Zhiyuan Li. Octopus v2: On-device language model for super agent, 2024.

[27] OpenAI. Overview of assistants. https://platform.openai.com/docs/assistants/overview, 2024. Accessed: 2024-06-23.

[28] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[29] Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small llms are weak tool learners: A multi-llm agent, 2024.

[30] Yafei Hu, Quanting Xie, Vidhi Jain, Jonathan Francis, Jay Patrikar, Nikhil Keetha, Seungchan Kim, Yaqi Xie, Tianyi Zhang, Shibo Zhao, Yu Quan Chong, Chen Wang, Katia Sycara, Matthew Johnson-Roberson, Dhruv Batra, Xiaolong Wang, Sebastian Scherer, Zsolt Kira, Fei Xia, and Yonatan Bisk. Toward general-purpose robots via foundation models: A survey and meta-analysis, 2023.

[31] Roya Firoozi, Johnathan Tucker, Stephen Tian, Anirudha Majumdar, Jiankai Sun, Weiyu Liu, Yuke Zhu, Shuran Song, Ashish Kapoor, Karol Hausman, Brian Ichter, Danny Driess, Jiajun Wu, Cewu Lu, and Mac Schwager. Foundation models in robotics: Applications, challenges, and the future, 2023.

[32] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022.

[33] Yilun Du, Mengjiao Yang, Pete Florence, Fei Xia, Ayzaan Wahid, Brian Ichter, Pierre Sermanet, Tianhe Yu, Pieter Abbeel, Joshua B. Tenenbaum, Leslie Kaelbling, Andy Zeng, and Jonathan Tompson. Video language planning, 2023.

[34] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

[35] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

[36] Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training, 2023.

[37] Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. Instruction mining: When data mining meets large language model finetuning, 2023.

[38] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning, 2024.

[39] Peiqi Wang, Yikang Shen, Zhen Guo, Matthew Stallone, Yoon Kim, Polina Golland, and Rameswar Panda. Diversity measurement and subset selection for instruction tuning datasets, 2024.

[40] 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.

[41] Yiming Wang, Yu Lin, Xiaodong Zeng, and Guannan Zhang. Multilora: Democratizing lora for better multi-task learning, 2023.

[42] Soufiane Hayou, Nikhil Ghosh, and Bin Yu. Lora+: Efficient low rank adaptation of large models, 2024.

[43] Dawid J. Kopiczko, Tijmen Blankevoort, and Yuki M. Asano. Vera: Vector-based random matrix adaptation, 2024.

[44] Qingru Zhang, Minshuo Chen, Alexander Bukharin, Nikos Karampatziakis, Pengcheng He, Yu Cheng, Weizhu Chen, and Tuo Zhao. Adalora: Adaptive budget allocation for parameter-efficient fine-tuning, 2023.

[45] Shih-Yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation, 2024.

[46] Bojia Zi, Xianbiao Qi, Lingzhi Wang, Jianan Wang, Kam-Fai Wong, and Lei Zhang. Delta-lora: Fine-tuning high-rank parameters with the delta of low-rank matrices, 2023.

7 Appendix

7.1 Function/API description examples

```
def get_trending_news(query, language):
    """
    Retrieves a collection of trending news articles relevant to a specified query
    and language.

    Parameters:
    - query (str): Topic for news articles.
    - language (str): ISO 639-1 language code. The default language is English ('en
    '), but it can be set to any valid ISO 639-1 code to accommodate different
    language preferences (e.g., 'es' for Spanish, 'fr' for French).

    Returns:
    - list[str]: A list of strings, where each string represents a single news
    article. Each article representation includes the article's title and its URL,
    allowing users to easily access the full article for detailed information.
    """


def get_weather_forecast(location):
    """
    Provides a weather forecast for a specified location over a given number of days
    . Each day's forecast includes a brief description of the expected weather
    conditions.

    Parameters:
    - location (str): The location for which the weather forecast is desired. Can be
     a city name, ZIP code, or other location identifiers.

    Returns:
    - list[str]: A list of strings, each representing the weather forecast for one
    day. Each string includes the date and a brief description of the weather
    conditions. Formatted in 'YYYY-MM-DD: Description' format.
    """


def send_email(recipient, title, content):
    """
    Sends an email to a specified recipient with a given title and content.

    Parameters:
    - recipient (str): The email address of the recipient.
    - title (str): The subject line of the email. This is a brief summary or title
    of the email's purpose or content.
    - content (str): The main body text of the email. It contains the primary
    message, information, or content that is intended to be communicated to the
    recipient.

    Returns:
    """
```

```
37
38
39 def search_youtube_videos(query):
40     """
41     Searches YouTube for videos matching a query.
42
43     Parameters:
44     - query (str): Search query.
45
46     Returns:
47     - list[str]: A list of strings, each string includes video names and URLs.
48     """
49
50
51 def find_route_google_maps(origin, destination, mode):
52     """
53     Computes a route using Google Maps from an origin to a destination.
54
55     Parameters:
56     - origin (str): Starting location.
57     - destination (str): Target location.
58     - mode (enum): Mode of transportation, options include 'driving', 'walking', '
       bicycling', and 'transit'. The default mode is 'driving'.
59
60     Returns:
61     - List[str]:  The string provides the route details.
62     """
63
64
65 def send_text_message(contact_name, message):
66     """
67     Sends a text message to the specified contact.
68
69     Parameters:
70     - contact_name (str): The name of the recipient contact.
71     - message (str): The content of the message to be sent. This is what the
       recipient will receive.
72
73     Returns:
74     """
75
76
77 def create_contact(name, phone_number):
78     """
79     Creates a new contact entry in the device's address book.
80
81     Parameters:
82     - name (str): Full name of the contact. This should include first and last name.
83     - phone_number (str): phone number of the contact. The phone number should be
       provided in a standard format, preferably in E.164 format (e.g., +12345678900
       for an international format).
84
85     Returns:
86     """
87
88
89 def set_timer_alarm(time, label):
90     """
91     Sets a timer or alarm for a specified time.
92
93     Parameters:
94     - time (str): Alarm time in "HH:MM" 24-hour format. For example, "07:12" for
       7:12 AM.
95     - label (str): Custom label for the alarm, default is "alarm".
96
```

```
    Returns:
    """


def create_calendar_event(title, start_time, end_time):
    """
    Schedules a new event in the calendar.

    Parameters:
    - title (str): Event title.
    - start_time (str): Event start time as a string in ISO 8601 format "YYYY-MM-DD-HH-MM". For example, "2022-12-31-23-59" for 11:59 PM on December 31, 2022.
    - end_time (str): Event end time as a string in ISO 8601 format "YYYY-MM-DD-HH-MM". Must be after start_time. For example, "2023-01-01-00-00" for 12:00 AM on January 1, 2023.

    Returns:
    """


def set_volume(level, volume_type):
    """
    Sets the volume level for a specified type : "ring" , "media" , "alarm".

    Parameters:
    - level (int): Target volume level, from 0 (mute) to 10 (maximum).
    - volume_type (enum): The category of volume to adjust, select from "ring" , "media" , "alarm".

    Returns:
    """
```