博士論文

Teaching AI Through Object Demonstrations and Language Instructions

(物体教示と言語指示による AI 学習)

Zhongyi Zhou

周　中一

Department of Electrical Engineering and Information Systems

The University of Tokyo

This dissertation is submitted for the degree of

Doctor of Philosophy

December 1, 2023

Abstract

AI has proven to be highly effective in addressing a multitude of research challenges. However, the development of AI solutions for specific applications still demands specialized expertise and substantial resources. This phenomenon limits the width of AI impact: only a small number of AI professions can leverage and customize AI as intelligent tools to solve their problem.

The main cause of this issue lies in the lack of interactive systems that enable humans to intuitively teach AI. To facilitate a natural interaction between non-expert users and an interactive system, humans should be able to perform teaching behaviors similar how they are engaged in normal social events.

In this dissertation, I focus on investigating two interaction techniques in teaching events: 1) demonstrations and 2) instructions. I built two interactive systems (*i.e.*, LookHere and InstructPipe) that allow users to teach AI by performing object demonstrations and language instructions, respectively. LookHere leverages users' gestural interactions people naturally perform in their object demonstration process to predict the target object that users want to specify. InstructPipe enables users to start prototyping an AI pipeline in visual programming by text-based instructions. Both studies in the two projects reveal a significant workload drop when a system leverages humans' natural teaching capability. Qualitative results further show that the reduced perceived workload inspire more creative uses of the systems, and that visualization of the system prediction enhances AI transparency.

Acknowledgements

I want to first thank my advisor, Koji Yatani. Thank you for introducing me to the exciting field of Human-Computer Interaction. I really appreciate your encouragement when things failed and when I tasted bitter paper rejection. I also feel encouraged by the times we celebrated our awards together during this journey. Research is hard. With you at my back, I feel no fear of exploring the unknown and the uncertainty.

I appreciate all the people I met at the university. I feel lucky that everyone I met is so friendly, and all of the events together with you constitute my wonderful Ph.D. journey in my life.

Pursuing a Ph.D. degree in a foreign country that speaks my second foreign language is never easy. I want to thank Takeo Igarashi and Yinqiang Zheng who provided great advice for my career. Thank you Takeo Igarashi for kindly introducing me to the Japanese academic career and for your advice on the possible career after the Ph.D. life. Thank you Yinqiang Zheng for many discussions about both research and life.

I want to thank all IIS-Lab fellows I have collaborated with, Shixian Geng, Kazuhiro Shinoda, Keitaro Shimizu, Anran Xu, Zefan Sramek, Minghui Chen, Shitao Fang, Hiroki Katori, Rei Sawano, Ginshi Shimojima, Shunpei Norihama, Takuma Masuda, Kakeru Miyazaki, Kosuke Yamamoto, Ryo Yoshikawa, Yuya Umeda, Yudai Shimada, Hiroki Nakano, Haruma Hirabayashi, Simo Hosio, Arissa Janejera Sato, Akari Doi, Michihiko

Ueno, Dimas Antony Chacon Salas, Shoko Sano, Ryo Takashima, Hirotaka Hayashi, Yuta Hirai, Takaharu Yamada, Nami Ogawa, Yuji Sugiyama, Asahi Takenouchi, Ikutoshi Katayama, Ryoto Suzuki, Naho Tomiki, Shinichiro Yoshida, Hao Xu, Carla F. Griggio, Tatsuhiko Sakaguchi, Hiroaki Masaki, Hidenori Matsui, Ayuka Ito, Siwook Choi, Yuki Tsubouchi, Kosuke Hatai, Kiyotaka Eguchi, Kengo Shibata, Peihan Tu, Daisuke Shibata, Benio Shimada, Toby Chong, Yuya Munekata. Life in IIS-Lab is great. Without any of you, it would be like a puzzle missing its most crucial pieces.

Thank you Misako Motooka, and all the staff at the IME, EEIS, OIS, and WINGS offices. Your professional support on my onboarding, accounting, administrative processes, and beyond can always smooth things that would otherwise overwhelm me if I did it myself. I also want to thank all the staff in the Japanese Language Class in the School of Engineering. Thank you for your great skills in teaching foreigners the Japanese language that support our life in this country.

Thank you Yuta Nakagawa for organizing the Friday Soccer event every week for years, and thank you to everyone who joins the game. Thank you CSSAUT, the soccer Association for Chinese students at UTokyo, especially the current and previous team captions, Ruiyang Xu, Wenpeng Xie, Yunxuan Hou, Yanyuan Fu, and Jiawei Chen, for organizing the events and the matches. I enjoyed every game with you.

I want to thank all my collaborators at Google. Thank you for all of your support that constituted my great experience during my internship.

Thank you Ruofei Du for hosting my internship. Thank you for sharing with me your enthusiasm for creating interactive systems and for demonstrating how to efficiently collaborate with many experts within a large organization.

I also value your guidance on my project proposals. It is great to have a mentor who sees what I see and believes in what I envision. Throughout the collaboration, I feel supported and recognized for all my efforts.

I want to thank my co-authors in the InstructPipe project. Thank you Jing Jin, Vrushank Phadnis, Xiuxiu Yuan, Jun Jiang, Xun Qian, Jingtao Zhou, Yiyi Huang, Zheng Xu, Yinda Zhang, Kristen Wright, Jason Mayes, Mark Sherwood, Johnny Lee, Alex Olwal, David Kim, Ram Iyengar, Na Li, and Ruofei Du. Without any of you, we cannot make such a great project.

Thank you Fengyuan Zhu, Xuan Gong, Karl Rosenberg, K Z, Danhang "Danny" Tang, Mingsong Dou, Zhengyang Shen, Anuva Kulkarni, Yingtao Tian, Yunfan Chen, and Daichi Hirono, for your great support of my work and beyond.

I am also lucky to receive support from external fellows.

Thank you Xiang 'Anthony' Chen for sharing your vision on the human-AI interaction research. Thank you Jiahao "Nick" Li. I enjoyed our academic collaboration and casual meals when you were interning in Tokyo.

Thank you Nadir Weibel for kindly inviting me to the UbiComp/ISWC 2023 organization committee. I appreciate all the support from other organizers, especially those from the general chairs (Monica Tentori, Nadir Weibel, and Kristof Van Laerhoven) and workshop chairs (Franceli L. Cibrian, Tengxiang Zhang, and Varun Mishra). Your speedy reply and action on my requests really helped us publish the proceedings successfully.

Thank you, Mingming Fan, Gonzalo Ramos, Yang Li, Ranjay Krishna, Quan Li, and Chun Yu for hosting my talk in your group. I appreciate your feedback on my talk as well as our discussion on our vision of future research in the community.

Finally, I want to thank my parents. Thank you for your remote support in my Ph.D. life. I enjoyed sharing with you interesting stories at school and listening to how things were changing in our hometown.

最后，我想感谢我的父母。感谢你们在我攻读博士期间给予的无形支持。我非常享受与你们分享学校里的有趣故事以及聆听故乡的变化。

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background and Motivation

AI technology has demonstrated its capability to solve various research challenges. From the accuracy surge in the ImageNet classification problem [88] to the latest breakthrough of foundational models [27, 29, 113] that stimulated people's dream of Artificial General Intelligence (AGI) [109], there exists no signal that progress made by AI research will slow down in the near future.

However, the major impact of AI breakthroughs still lies in the classical AI fields (*e.g.*, computer vision and natural language processing), and customizing AI for a broader range of practical usages is still challenging for the general population. The major reason is that developing new AI models requires professional AI knowledge, and there is a lack of interactive systems that enable users to leverage AI as a tool to solve their practical problems. On the other hand, even AI experts still struggle to develop useful AI applications completely by themselves. The development of an AI production typically requires more effort than training ML model(s): the production first needs to satisfy users' requirements in society, and a created model needs to be deployed in a sustainable and stable platform. The community typically solves this issue through a team effort with members with various expertise backgrounds:

1. the user experience (UX) researchers first investigate the users' requirements and the issues, and then write a report to product managers (PMs).
2. PMs then propose several AI features and consult with engineers and scientists to decide on the implementation plans.
3. Engineers and scientists implement the system and maintain the system.
4. UX researchers then collect user feedback that forms a development loop.

While a good team effort can effectively find representative user requirements and solve them with professional solutions, the whole development is still a highly costly process. I argue that the main cause of this issue is the lack of interactive systems that enable *end-users themselves* to intuitively customize their own AI models. Therefore, our current society has to invest limited resources in those focus areas, in which the community first distills humans' shared interests and conducts the development process on the focus areas.

In practice, to develop an AI model for a target use, developers typically refer to an existing AI solution and conduct customization upon this existing solution. One commonly approach is transfer learning [118], in which a model adapts the knowledge it learns in a related domain to a target domain. The major benefit is that the ML model can effectively reach a decent accuracy with a limited number of data. Moreover, various ML benchmanks [102, 156] make developers' choices on base ML architectures converge to several outstanding ones, and recent advance in ML packages [166, 145] further allow developers to make such ML model by as few as one line of code. Therefore, when developers create AI solutions, they typical perform AI customization, in which they fine-tune (or assemble) related AI solutions to serve for their need.

While this procedure typically requires advanced ML skills, in essence, the development task is equivalent to teaching an AI with a new concept. With the increasing capability of ML packages that enhance ML accessibility for developers, could it also be possible for non-experts to intuitively teach their own AI in the future?

1.2 Research Goal and Focus

In this dissertation, I aim to build interactive systems that empower non-expert users when they teach AI. To build such systems that enable natural human-computer interaction between non-experts and AI systems, I raise the following research question that inspires my research projects: *What is the role of AI and humans when humans teach AI?*

On the human end, people are talented at using various interactions to express their intentions at social events (*i.e.*, human-human interaction). When humans play the roles of teachers in such events, people naturally perform two kinds of interactive behaviors for effective and natural teaching: 1) demonstrations and 2) instructions. Demonstrations allow teachers to explicitly show an abstract concept, and instructions enable teachers to explain concepts using natural language. On the computer end, an interactive system requires fundamental input for adapting an ML model for customized uses. Example input includes a labeled dataset that enables a fine-tuning process or a prompt that guides the inference of a Large Language Model (LLM).

To this end, my Ph.D. dissertation focuses on building two interactive systems that empower users to teach AI through object demonstrations and language instructions, separately. In the first project, I explored humans' gestural interactions with objects and leveraged the cues implied in such interactions to inform the ML process. In the second project, I studied methods to facilitate users to prototype a visual programming pipeline by giving language instructions. Together, these projects demonstrated an early-step exploration of two major approaches to how humans can intuitively teach AI.

1.3 Teaching AI and AI Customization

Two important phrases will be frequently used in this dissertation: 1) "teaching AI" and 2) "AI customization". While both phrases share similar concepts, "teaching AI" emphasizes the human behaviors within the human-computer interaction process and the goal of such behaviors is AI customization. While these two phrases can be used interchangeably in most expressions, I want to highlight that the main focus of this dissertation is to leverage humans' teaching behaviors, including teaching by demonstration and teaching instruction, to better support the AI customization process.

AI is another important keyword in this dissertation. I use a broader definition of AI, encompassing not only a singular machine learning model, but also systems comprising various functional modules, whether each of they are machine learning-based or otherwise. The AI customization process refers to users' endeavors to tailor AI solutions to their needs, irrespective of whether these solutions derive from a singular model or a constellation of different models. This broader interpretation aligns more closely with the intended scenarios of interactive machine teaching aimed at non-expert users, whose primary objective is to develop functional solutions without a particular focus on the underlying machine components.

Chapter 2

Related Work

2.1 Interactive Machine Learning

Latest ML models usually require a large number of data to train the algorithm. This causes a massive burden for humans to label considerable data. For example, for the multi-person pose detection task in computer vision [17], human labelers need to pinpoint the pixel positions of all 18 landmarks of each person in the view (i.e., two eyes, two ears, a nose, a neck, two shoulders, two elbows, two wrists, two hips, two knees, and two ankles.). Such tasks treated humans as an oracle [5], which is only responsible for an endless labeling task. Not only in the field of computer science, other researchers also applied ML to their projects which proved to have promising results [139, 171]. To facilitate the fundamental ecosystem for ML development, researchers created the necessary dataset for various applications by labeling a large number of data [83, 143]. Despite the success of knowledge transfer, the increasing application scope implies that humans need to continuously serve for machines through infinite labeling tasks. It becomes clearer that further consolidation of such human-machine collaboration would cause disastrous results for both the technology community and society.

One prominent attribution of this issue is the lack of interaction between humans and machines during the learning process. Traditional ML development strategies only ask humans to label a whole dataset at the beginning and then initialize ML development by fixing that dataset. This separation between humans and ML strongly eliminated the efficiency of the work. On the one hand, human labelers are usually the experts of the tasks, but they do not have a chance to inspect the learning progress, nor to provide further valuable feedback for the learning. This agnostic phenomenon forces researchers to create a sufficiently large dataset, being as diverse as possible, despite the redundancy and the burden to the human labelers. On the other hand, ML cannot ask questions for the experts to solve its confusion, since the teaching samples is fixed while it is learning. Although the reason for the confusion may be the ambiguity of the label or the sufficient data, ML researchers have to build an algorithm that is intelligent enough to solve these issues only by itself.

Interactive Machine Learning (IML) breaks this barrier and introduces a bi-directional communication between humans and machines. Dudley and Kristensson's work [47] summarizes the workflow between humans and machines in an IML system. It is important to note that IML is different from Active Learning (AL). AL only concerns the active role of machine learning in which the machine learning algorithm would actively ask for further clarification when encountering confusion. It implies that humans still play a passive role: do nothing if ML does not ask for anything. Based on this, IML further allows the user to actively explore ML and provide feedback on learning if they consider it necessary.

2.1.1 Interactive Machine Teaching

When a modern machine learns a new concept, it usually asks for a large number of abstract data (e.g., ImageNet Dataset [39]) for its training process [82]. This learning approach differs from the principle of human-human instruction. A human teacher usually teaches the student in an intuitive way, like providing a demonstration and explanation of the idea by talking or using body language. Inspired by this, researchers have created novel interaction approaches for users to intuitively teach machines new concepts, which provides abundant information to machines at the same time. For example, users can instruct machines by sketching [49], speaking natural languages [101, 125, 111], and demonstrating the workflow of the task [97, 99, 98]. Crayons, a pioneering IML system, allowed users to create an image segmentation machine interactively through simple sketches [49]. Users can iteratively examine the prediction results from Crayons and draws simple curves on the interface to correct the prediction for fine-tuning its algorithm. Teachable Reality [108] applies the concept of IMT into AR authoring tools and shows that IMT can effectively enhance the customizability of the toolkits and support an expensive creation supported by accessible ML models. Feng et al. [50] further explore the application of IMT into social platforms, and highlight the effectiveness of IMT in enhancing user control and personalization in social media feeds.

Considerable AI research demonstrates that deep and large ML models can effectively scale up the machine intelligence [61, 105]. However, finetuning a large model typically requires many samples from humans so that an ML model can fully understand the concept. Previous studies revealed that users tended to become frustrated during such a tedious labeling process [5]. Researchers mainly solved this issue by creating tools to accelerate labelling [82, 90], or only ask for highly informative human input [28, 38]. For example, Laielli et al. built LabelAR [90], which automatically collected data on AR anchors placed by the user. The user can first anchor AR boxes over target objects. Then the interface computes 24 arrows and guides the user to move the camera under these directions which are evenly distributed in the 3D world. Their evaluation study showed that LabelAR supported a fast and accurate data collection experience, by comparing it with two baseline methods [57, 173]. Goodfellow et al. built Generative Adversarial Network (GAN) [59] that supported the following systems to generative verisimilar data like human-face images [140], dancing videos [93], and summary sentences [103]. However, the generative model requires the input of a high-dimension latent vector. This made it hard for a user to explore optimal data since it is impractical for the user to adjust the latent vector in, e.g., 128-d space. Torre-Ortiz et al. [38] created a brain-computer interface (BCI) that sensed Electroencephalography (EEG) as high-dimensional feedback for the exploration in the latent space. Their user study showed a high convergence speed of the generative model and potential for future applications. Chiu et al. [28] developed a framework that allowed the user to explore their ideal data by simply using a 1D slider interface. They applied Singular Value Decomposition (SVD) which compresses the optimization space into a dynamic 1D subspace while maintaining as much information as possible. This method supported a faster convergence as well as an "easy-to-use" user experience, according to their evaluation study.

The aforementioned work requires humans to spend time on a 1-on-1 teaching session for machines to learn new concepts. This process can be time-consuming. Is it possible for machines to learn concepts interactively, but only by themselves without disturbing users to spend time teaching machines some simple concepts? Recent research provided solutions to this question mainly by exploiting users' behaviors in history, inspired by the approaches in building recommendation systems for advertisement [128, 136, 159]. This method activates an automatic learning strategy, in which ML models look for personal preferences by exploring users' historical behaviors, like clicking events [10, 25] and eye-contact interests [54]. For example, Gebhardt et al. [54] exploited the user's gazing directions to predict the object that interests the user. This method can intellectualize Mix Reality (MR) interfaces, which only show the user labels of the objects that interest them. The authors demonstrated the utility of this function when there existed many virtual objects within the user's visual field.

2.1.2 Interactive Visualization for Human Perception

After humans provide instructions to the ML model, it operates a learning process that updates its understanding of the concept based on the new teaching data. ML researchers usually summarize this optimization process by the following equation:

$$\theta_{t+1} \leftarrow \theta_t + \Delta\theta_t \tag{2.1}$$

θ_t represents the model parameter at the tth update. $\Delta\theta_t$ is the updating value determined by its optimizer.

However, simply presenting the parameters to the IML user does not help users understand the machine. This would discourage users from providing further instruction for teaching the unknowns of the machine. Instead of showing these mathematically sufficient data (i.e., the model parameter), explainable AI researchers let ML models "explain" themselves by providing visual saliency maps [49, 185] or example data [15, 52]. Class Activation Mapping (CAM) solved the challenge of explaining the predictions of Convolutional Neural Networks (CNNs) by highlighting spatial grids that have high weight values to support the prediction [185]. Cai et al. [15] supported the explanation of image classification prediction by showing a group of similar images in its training dataset. The system also provides "comparative explanations", meaning the system would present top-k prediction labels and include one training image as an example for explanation. Their user study showed that the explanation design helped users better understand the system, though "comparative explanations" may sometimes cause further confusion.

The technology of interpreting machine learning above allows humans to understand the static working principle of ML models at each iterative step during the IML workflow. Since humans need to repeatedly review the performance of ML models, how to coherently visualize the dynamic improvement of ML models plays a critical role in building the closed-loop in IML workflow.

AnchorViz is an IML tool that can decompose the unlabelled dataset semantically and visualize a dynamic knowledge graph of the model based on several labeled example data (i.e., 'anchors") [149]. The system placed the anchors on a circle and all the other unlabelled data within this circle. AnchorViz distributed these unlabelled data according to their semantic similarity among all the anchors on the circle. This distributing strategy facilitates multiple clusters. The system would compute the accuracy of the clusters to support the user to decide whether they need further teach the data in this cluster. By such iteratively teaching through filling the knowledge graph of the model, the model can quickly learn the knowledge with as few labels and few human efforts as possible. Hohman et al. [66] found that many applied ML need to be evolving to follow the change in customers' preference. According to their interviews with experts, companies usually solve this issue by feeding new types of data instead of innovating a new algorithm. However, there are no existing interfaces and visualization tools to help engineers examine how the new data affects the result (or model prediction accuracy) throughout the trials. Therefore, they created CHAMELEON, which they designed for the purpose of visualizing the performance comparison among all the production versions. Participants in the case studies found several potential use cases of the system, including helping them find outliers, encouraging instance-level analysis, and capturing data processing changes.

2.2 Programming by Demonstration

Computers run according to predefined programs. This implies that the most fundamental approach to interacting with computer systems, including AI systems, is to program. However, programming is a specialized skill that demands formal training, making it challenging for the general population to master. To address this issue, HCI researchers invent an interaction scheme in which non-experts can "program" computer systems by performing simple demonstrations [84, 112, 33]. Programming by Demonstration is an interaction method that *"combine(s) the simple interface of macros with the expressiveness of a scripting language"* [92]. Such an interaction method can effectively reduce technical barriers when users, *e.g.*, command a computer by writing programs.

Inspired by these concepts, several generations of researchers built various applications that allow non-expert users to program by demonstration [42]. For example, Li et al. [99] created PUMICE, a multi-modal agent for non-programmer users that can learn new concepts and ask for confirmation for ambiguous users' commands. The system allows the user to teach it by demonstration. For example, the system would learn how to place a new order, like ordering a cup of cappuccino, and the user shows the procedure one time. Ten participants with different levels of programming experience attended their user study, in which they were required to use PUMICE for four tasks. For example, the user needs to teach the smartphone to order a pepperoni pizza if there is enough money left in the food budget. The result of their user study showed that participants completed the tasks within a similar period of time, regardless of the programming experience. One critical issue in most programming by demonstration systems is that the systems may have access to more personal information than necessary to learn users' teaching intentions. To address these shared issues of prior systems, Li et al. [98] built a system that automatically obfuscated privacy-threatening regions in a GUI when users are performing programming by demonstration.

2.3 Instruction-based Interactive Systems

Other than demonstrations, humans also perform instructions to facilitate efficient communication when humans want to teach concepts to other people. Despite the wide variety of concepts people want to deliver at social events, natural languages alone usually satisfy humans' desire to convey their concepts. Such facts inspire many researchers to simplify various human-computer interactions into natural languages and build applications with simple and effective agency. The AI Chatbot, such as ChatGPT [113] and Bard [154], is one of the most impactful applications in the society. Following such breakthroughs of the foundational models which can demonstrate their problem-solving capability in a wide range of tasks, developers have built various AI applications for incorporating humans' instructions. AutoGPT [1] is an application that can take multi-step actions given human instruction. For example, when the user instructs the system to "summarize the news today", the system may first gather news from the internet, identify key points in each article, and draft a summarization. Such widely interest in instruction-based interactive systems motivates researchers to further tailor the foundational models for better interpreting users' instructions and generating more feasible content. InstructGPT [114] and FLAN-PaLM [30] are exemplary projects that initialize such exploration, in which researchers incorporate the instruction tuning that simulates diverse uses of LLM based on instructions. Other than upgrading models for better perception of human instructions, researchers further explore the philosophy of the instruction-based human-computer interaction. Chain-of-thought prompting [164, 86] is one of the most representative prompting techniques. Instead of prompting the inputs and outputs of a target task as in-context examples, researchers suggest adding the reasoning processing between the I/Os (*i.e.*, the chain-of-thoughts), and experiments show that such reasoning annotations can effectively improve model performance.

Instruction-based interactive system design also extends from text-based tasks [75] to multimodal tasks [4, 181, 126]. Researchers have applied the prompt-based interaction design into vision-based generation [13, 180] and recognition [126, 95], and the generation of our modality like music [3] and motions [9, 155]. The prosperity in foundational models among multiple modalities stimulates people's dream of artificial general intelligence (AGI) [109], in which an AI system can perceive multimodal data in our society and reply in multimodal messages. Despite the wide enthusiasm in the community, researchers also point out various concerns of instruction-based systems [78, 174] For example, Zamfirescu-Pereira et al. [174] criticize the overuse of instruction-based principles to design interactive systems. Their studies reveal that non-expert users find it difficult to frame clear instructions.

Chapter 3

Gesture-aware Interactive Machine Teaching with In-situ Object Annotations

3.1 Introduction

Interactive Machine Teaching (IMT) [127, 141] aims to enhance users' teaching experience during the creation of Machine Learning (ML) models. IMT systems are primarily designed for non-ML-experts, and allow such users to provide training data through demonstrations. Vision-based IMT (V-IMT) systems utilize cameras to capture users' demonstrations. For example, in Teachable Machine [19] users can create a computer vision classification model by showing different views of each object (class) to a camera. Despite its low burden for providing training samples, existing work [189] revealed that an ML model trained through a V-IMT system might recognize an object by using visual features unrelated to it. For example, even if a user performs a demonstration of a book, the model may use visual features in the background. Failure to address this error properly could result in the degraded

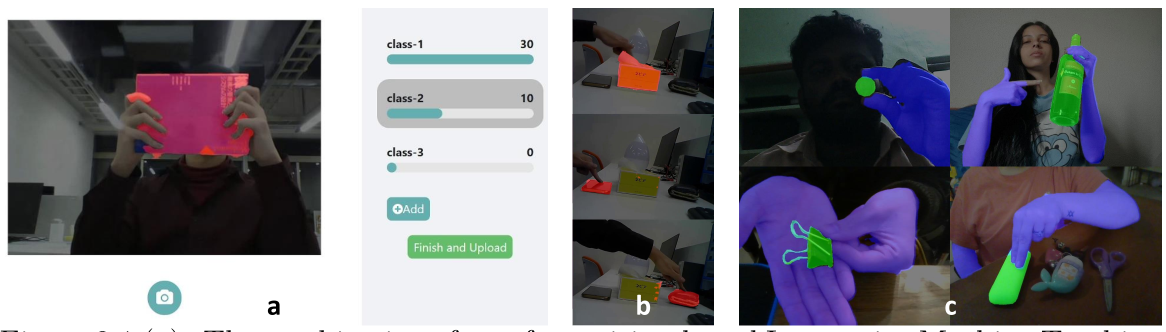


Figure 3.1 (a): The teaching interface of our vision-based Interactive Machine Teaching system, LookHere. LookHere provides a segmentation mask (an object highlight) on the object guided by users' deictic gestures in real time during teaching. This segmentation mask is used for model training as additional information for training classifiers. (b): Users' deictic gestures guides in-situ object annotations. (c): Example images in our *HuTics* dataset that enables the implementation of LookHere. *HuTics* includes 2040 labeled images that capture how 170 people use deictic gestures to present an object.

performance of the model when it is deployed to real applications. Therefore, users should have the capability to specify portions of an image that a model should emphasize in learning to achieve reliable classification.

One approach to address this issue is to perform annotations on the objects of interest [134] and feed them into the model as well. Advances in annotation tools reduce user workload by simplifying necessary interactions to clicks [110, 147] or sketches [130, 178]. Despite their lowered burden, existing annotation tools are not well tailored toward V-IMT systems, and users thus have to perform annotations in a post-hoc manner. This would degrade the overall experience of V-IMT systems [146]. Annotation approaches that are more deeply integrated into V-IMT thus need to be explored.

To this end, this work examines V-IMT systems that can integrate object annotations into the teaching process. We observe that when users are doing demonstrations for teaching, they may hold or point to the object of interest. These deictic gestures in demonstrations thus are indicative of what visual features a model should focus on. Therefore, this work focuses on the integration of annotations by leveraging deictic gestures that humans naturally perform during the teaching process.

Note that, in this work, we use the term deictic gestures to represent a wide range of gestures whose purpose is to indicate the object of interest [133, 31] while it typically represents pointing gestures in HCI research [80].

Our V-IMT system, called LookHere[1], embeds in-situ object annotations inferred from users' deictic gestures into the teaching process. LookHere provides real-time visualizations, named *object highlights*, on what portions of the given frame the system is considering as the region of the target object (the red mask in Figure 3.1a). Depending on the deictic gestures users are performing, our system infers different object regions (Figure 3.1b). To achieve this gesture-aware object segmentation, we created *HuTics*, a dataset consisting of 2040 images collected from 170 people that include various deictic gestures and objects with segmentation mask annotations (Figure 3.1c). Our technical evaluation shows that our object highlights can achieve the accuracy of 0.718 (mean Intersection over Union; *mIoU*) and can run at 28.3 fps. Our user evaluation confirms that participants were able to build accurate models while being liberated from post-hoc manual object annotations.

This work offers the following contributions:

- A vision-based IMT system, LookHere, which integrates in-situ object annotations guided by users' deictic gestures into the teaching process,
- The development of real-time object highlights, which offers users feedback on the object region inferred from their deictic gestures,
- The *HuTics* dataset[2], which contains 2040 labeled images from 170 participants interacting with various target objects using deictic gestures, and
- Our evaluations that confirm LookHere's benefits through quantitative and qualitative results.

[1]The source code is available at https://github.com/zhongyi-zhou/GestureIMT

[2]The dataset is available at https://zhongyi-zhou.github.io/GestureIMT/.

3.2 Related Work

3.2.1 Interactive Machine Teaching

Machine Teaching is a term that has been used by both HCI [141, 67, 132] and ML [191, 192] communities with different definitions. To avoid conflicts, we utilized the term of Interactive Machine Teaching defined by Ramos *et al.* [127]: IMT is *"an IML (interactive machine learning) process in which the human-in-the-loop takes the role of a teacher, and their goal is to create a machine-learned model"*. This definition emphasizes user experience rather than mathematical challenges, and is well aligned with the scope of this work.

IMT systems were typically designed for non-ML-experts to build their own ML models without requiring technical knowledge and skills [48]. Existing work conducted qualitative studies with ML novice users and presented user requirements and design guidelines for IMT systems [55, 131, 132]. For example, Fiebrink *et al.*'s work [51] suggested informing users of *"where and how the model was likely to make mistakes"* so that users can systematically assess the benefits and risks in their applications. Yin *et al.* [172] found that non-ML-experts users evaluated the model only based on its accuracy, and they were not often aware of the potential unreliability of the model when it was used in another application.

Zhou and Yatani [189] enhanced the model assessment process by visualizing the image regions that were highly weighed for predictions. They further found that simple teaching without further fine-grained annotations [19] could cause unexpected failures. Effective approaches for specifying the portions of the image in the teaching process are still under-explored.

This work introduces a V-IMT system that exploits users' deictic gestures to present objects for identifying the region which a model should focus on for learning. This interface design can solve the issue of accidental use of unrelated visual features by ML models by exploiting interactions people would normally perform during the teaching phase.

3.2.2 Interactive Annotations

One standard approach to addressing accidental use of unrelated visual features is to provide annotations (e.g., a segmentation mask over the target object) and inform a system of where a model should focus. While offering useful information, annotation is generally a tedious manual task. Drawing a polygon-based contour on an object [134] is a common approach to generating a segmentation mask, but this is generally very time-consuming. By incorporating computer vision methods, research has demonstrated different ways to reduce input from users [90], including clicks [110, 147], sketches [130, 178], and mouse drags [22].

As these annotation tools are not specifically designed for the integration into V-IMT systems, users would have to use them in a post-hoc manner. This does not thus fully exploit the user interaction that occurred in the teaching phase for inferring segmentation masks on target objects. Instead of proposing another annotation approach, our work utilizes users' deictic gestures toward target objects when they are performing demonstrations to a camera. In this manner, our system achieves in-situ object annotations while teaching in V-IMT systems.

3.2.3 Interactions Using Deictic Gestures

Prior research found that infants already have an ability to perform and interpret hand gestures [20, 107]. Inspired by this inherent human capability, HCI research has developed various interfaces using deictic gestures [165, 161]. One of the earliest work in this space is "Put-that-there" [11], in which users can manipulate virtual objects through a combination of deictic gestures and natural languages. Interface applications of deictic gestures also include drone manipulations [21], Human-Robot

Interaction (HRI) [133, 124] and commutations in Mixed Reality [16]. Sauppe *et al.* [133] demonstrated a human-like robot that can perform deictic gestures, and found that these gestures can contribute to improving communicative accuracy in interactions with users.

Besides deictic gestures, research has investigated different aspects of hand-object interactions. By assuming that the object under humans' manipulations would follow 1-DOF movements, Hartanto *et al.* [64] created a method to segment the object and classify the type of object motions (pure displacement motion by the prismatic joint or pure rotational motion by the revolute joints). Other work built datasets of hand-object interactions [18, 137, 94], and aimed to derive data-driven approaches for recognizing these interactions. Lee and Kacorri [94] created the TEgO dataset and a system for people with visual impairments to recognize a pre-defined set of daily-life objects and assist interaction with them.

This work extends the application of deictic gestures to V-IMT systems and allows users to perform in-situ annotations while teaching in real time. More importantly, LookHere advances the generalizability by removing those constraints in prior work (e.g., pre-defined object categories [94] or specific motions associated with holding [64]).

3.3 Research Challenges and Questions

3.3.1 Challenges in Existing V-IMT Systems

After reviewing the existing V-IMT systems, the authors summarized our perceived challenges in the following two aspects:

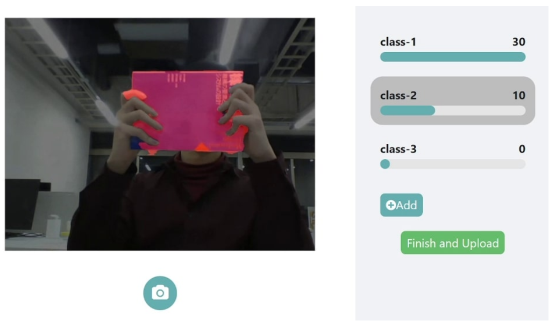
C1. ML models created through simplified processes supported by V-IMT can be unreliable because they may learn features unrelated to target objects. One major shortcoming of V-IMT is that ML models created through such systems may unpredictably attend to unrelated objects, which is aligned with the findings by Zhou and Yatani's work [189]. To simplify the creation of ML models for non-experts [48, 141], V-IMT systems typically only ask users to perform several demonstrations to the camera [19, 53]. During teaching, the computer not only captures the object to be classified, but also other unrelated backgrounds or objects. A model thus may consider those unrelated features as critical components of the target objects while users expect that it would only capture the features on the target objects. This discrepancy could result in unexpected inaccuracy when the model is brought to actual use. This can greatly degrade the usability of the created model and affect users' trust [172] toward it.

C2. Post-hoc annotations can diminish the overall usability of V-IMT systems. A naïve approach to solve the aforementioned issue is to ask users to specify what they want to be included in models (i.e., annotate the image regions of the objects). Existing work has successfully simplified data annotations [148, 110], but these interfaces are mostly designed for more professional use [81]. Furthermore, creating a reliable ML model usually requires the user to provide many samples per class. Performing annotations on many images repeatedly in addition to teaching through V-IMT systems can thus be overwhelming to non-expert users [146]. This also can discourage casual use of ML, which many V-IMT systems envision.

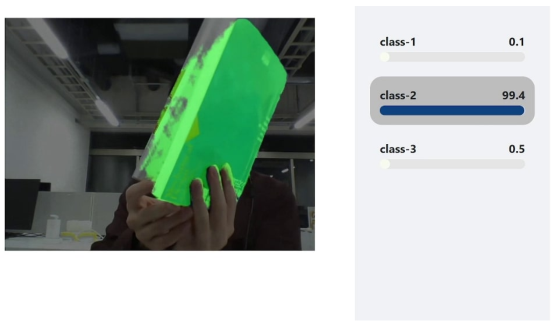
While formative user studies could further confirm these challenges, we decided not to execute them as they are already well explained in the existing literature. Our user evaluation results presented in Section 3.8 also confirm these challenges well.

3.3.2 Research Questions

This work explores approaches to solve both challenges by *integrating annotations into the teaching process.* To achieve this integration, we exploit how people interact



(a) Teaching Interface.



(b) Model Assessment Interface.

Figure 3.2 The screenshots of LookHere. (a) In this teaching interface, real-time object highlights are provided. The number of samples per class is presented on the right side of the view; (b) In this model assessment interface, the saliency map visualizations for the prediction of a specified class (i.e., class 2 in this example) are shown along with the prediction confidence score. This feedback informs users of what visual features in a given frame a model is weighed for predictions.

with objects of interest using *deictic gestures* when they perform demonstrations to a camera. For example, they may hold the object with both hands or may point to the object with their index fingers. Such human behaviors are known in prior HCI research [79] that led to diverse gesture-based applications [122, 94]. Therefore, we hypothesized that such deictic gestures would be an important cue for in-situ annotation. Accordingly, we derive the following two research questions to be answered through this research:

RQ1. *How can users' deictic gestures toward objects of interest be utilized for annotations during teaching?*

RQ2. *Can such in-situ annotations inferred from users' deictic gestures reduce the overall teaching workload while maintaining the model accuracy?*

RQ1 asks for technical approaches for leveraging humans' deictic gestures for the integration and the corresponding implementation. RQ2 investigates the efficacy of such gesture-aware annotation methods. Our design and implementation of LookHere in the following content explore RQ1, and our evaluation study answers RQ2 using multiple metrics (i.e., time consumption, model accuracies and subjective workload).

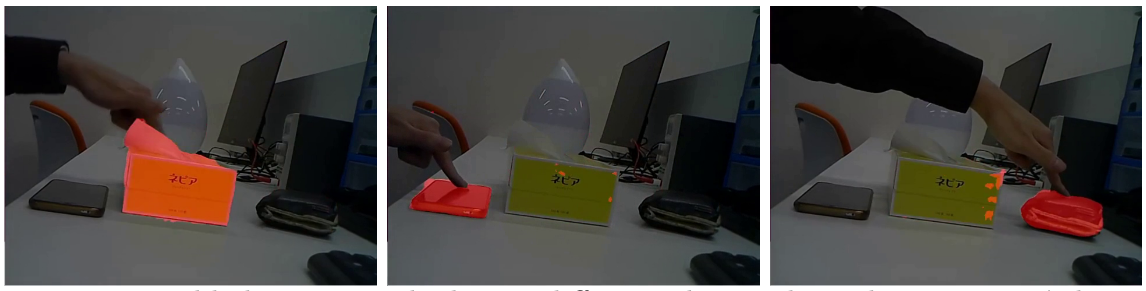


Figure 3.3 Highlights are overlaid onto different objects depending on users' deictic gestures.

3.4 LookHere

Our V-IMT system, LookHere, considers users' gestures to objects for building accurate ML models. Unlike existing workflows in V-IMT, LookHere directly integrates the annotation process into the teaching process. More specifically, LookHere includes a function called *object highlights* to inform which part of the camera view the system is considering as the region of the object to be learned. In the assessment phase, LookHere also supports a model assessment process by providing a similar visualization, allowing the user to assess whether the trained model attends to the correct features.

Besides these two features explained in this section, the architecture and interaction walkthrough are similar to existing V-IMT systems. In our current implementation, users can train a multi-class classifier (i.e., classifying different objects). To define a class, the user first selects the corresponding class (see the top-right corner of Figure 3.2a). Then, they can perform demonstrations of the object to the camera, and the system captures the frame when the user clicks a camera button. The number of frames collected for each class is presented as a bar graph. After finishing teaching for the three classes, users may either click the "Add" button to include more classes or the "Finish and Upload" button to finish the teaching session. Appendix A.1.1 shows our detailed configurations in the ML process after teaching.

3.4.1 Object Highlights and In-situ Object Annotation

During the teaching process, users can receive visual feedback about which portion of the camera view LookHere is currently considering as the region of the objects of interest. As is shown in Figure 3.2a, our system infers the object region based on deictic gestures users are performing (e.g., holding or pointing to an object for teaching). Users may simply change how to perform gestures to express different target objects, as shown in Figure 3.3. LookHere incorporates a gesture-aware algorithm (see details in the next section) to achieve this adaptive highlight on objects.

Another advantage of providing this highlight in real time during the teaching process is to help users avoid including erroneous demonstrations. Users can easily opt out of such frames by not clicking the camera button. In this manner, LookHere takes a mixed-initiative approach [68] for teaching.

When the user records the current frame by a button click, the system stores the RGB image as well as the inferred object segmentation mask. Both data are used for model training. In this manner, LookHere achieves in-situ object annotations during the teaching process.

3.4.2 Model Assessment with Saliency Map Visualizations

After the teaching phase, LookHere offers the model assessment mode like other V-IMT systems [53, 19]. However, unlike these systems, LookHere provides saliency map visualizations for users to confirm whether the created model is considering appropriate visual features. Figure 3.2b illustrates an example of the view in this assessment phase. The interface presents two visualizations for the users: bar graphs to present confidence score distributions (Figure 3.2b right) and real-time saliency map visualizations (Figure 3.2b left). The confidence score shows how confidently the model considers that the current frame belongs to the corresponding class. In the example of Figure 3.2b, the model is 99.4% confident that the object in the

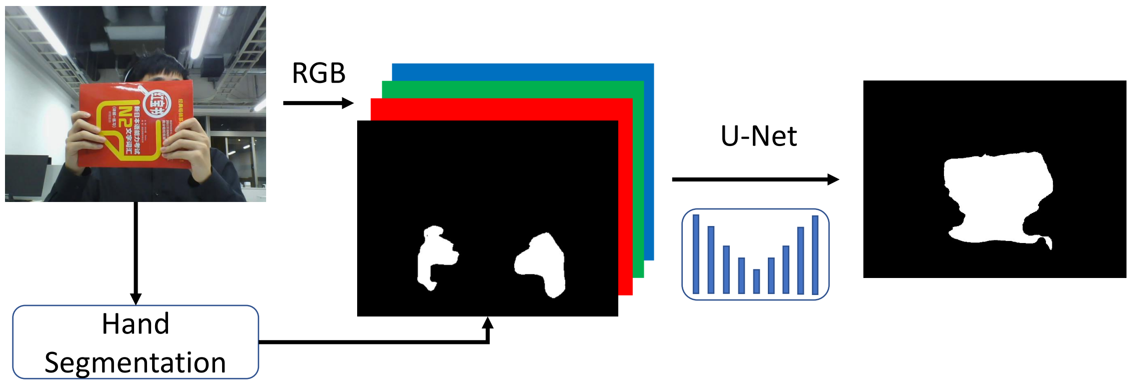


Figure 3.4 The generation process of object highlights. LookHere first performs a hand segmentation with the given RGB image. The system then feeds both the RGB image and segmentation mask into U-Net, which predicts a segmentation mask of the object guided by deictic gestures.

frame belongs to is class 2 (which is configured as a “book” class in Figure 3.2a). Real-time saliency map visualizations then help users understand which portion of the frame the model considers as the object of interest (a book in this example). Existing work [182, 189] leveraged CAM methods to present such visualization while we introduce a new method for more accurate visualizations by utilizing the object segmentation masks originally generated for object highlights (see more details for Section 3.5.2).

3.5 Implementation

The current prototype of LookHere is implemented as a web-based interface, and most of the computations are executed at the back end. We use WebRTC to synchronize the video between the interface and server for real-time image processing. The two key features presented in Section 3.4 are supported by two technical components: gesture-aware object highlights and joint training. We explain the details of the implementation of these components in this section.

3.5.1 Gesture-aware Object Highlights

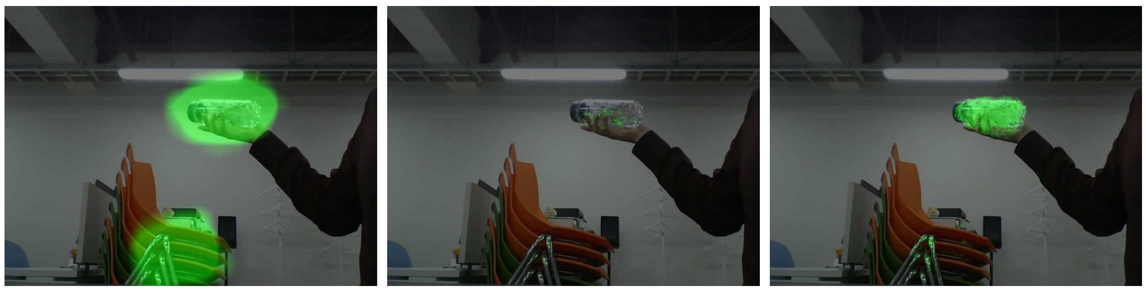
Figure 3.4 summarizes the workflow of our gesture-aware object highlight algorithm. The algorithm first applies a hand segmentor on the input image and predicts a hand segmentation mask. It then feeds both the original RGB image and the hand segmentation mask into U-Net [129], which outputs a segmentation mask of the object that is referred to by the users' deictic gestures.

Hand Segmentation

We utilize Li *et al.*'s algorithm [96] trained on the LIP dataset [58] to perform real-time hand segmentation. The LIP dataset parses a person into 20 body parts and garments (e.g., "left-leg", "gloves" and "pants"), and we regard the segmentation result of "left-arm" and "right-arm" as the portion of hands. We note that the definition of "arm" in the LIP dataset includes both arms and hands that are not covered by clothes or gloves. We notice that the publicly-available model provided by the authors of the LIP dataset is not suitable because it utilizes resnet-101 backbone [65], which is a very deep CNN architecture and is not executable in real time. Therefore, we re-design their methods based on resnet-18, a much lighter model with the same encoding approach. We then tested this light-weighted model on the LIP dataset. The result $mIoU$ accuracy of the light model is 0.621, and that of the original model using resnet-101 is 0.680. This demonstrates that our light model for real-time uses can still achieve comparable accuracy to the original deep model.

Object Highlights

As explained in Section 3.4.1, object highlights offer immediate feedback on what portions of the image frame the model to be trained should focus on. To avoid losing the generalizability of V-IMT, LookHere should be able to segment the object of interest in an object-agnostic manner. To tackle this challenge, we feed the RGB image



(a) CAM ($\Lambda = 0$). (b) $\Lambda = 1$. (c) $\Lambda = 0.718$

Figure 3.5 Visual comparison of saliency maps with different settings of Λ. The parameter Λ in Equ. 3.1 controls the weight balance between the results by CAM and our trained model.

concatenated with the hand segmentation mask inferred from the hand segmentor (Section 3.5.1) into a recognition model as shown in Figure 3.4. Intuitively, this hand segmentation mask carries the information of what objects in the frame users are specifically referring to in their demonstrations.

The current implementation uses U-Net [129] as the encoder-decoder architecture. It performs the best as well as the fastest among four commonly-used segmentation model architectures (see Appendix A.1.3 for detailed data). The network uses the EfficientNet [153] backbone, a design toward high computation efficiency. To train this U-Net, we use our own dataset which we will explain in Section 3.6.

3.5.2 Joint Classification and Segmentation for Saliency Map Visualizations

Saliency map visualizations are useful for users to understand what specific portions of a given image are weighed more in their ML models. Existing work [189, 182] created saliency maps of a classification model through CAM methods [135, 185]. CAM methods are primarily used for simple classification models trained by the dataset without segmentation masks. Unlike existing V-IMT systems, our training data accompany the object segmentation masks inferred during the teaching phase.

We thus devise a new model training approach for LookHere to exploit this unique information resource to achieve more accurate saliency maps.

LookHere identifies the areas to be highlighted by saliency maps through solving a classification and segmentation problem jointly. This means that our backend model predicts a class as well as infers the segmentation of the object of interest at the same time. More specifically, we train the model through a joint loss function (l_{joint}), which is a weighted sum of classification loss (l_{cls}) and segmentation loss (l_{seg}): $l_{joint} = l_{cls} + \lambda \cdot l_{seg}$. λ is a trade-off weight that determines the relative importance between the classification loss and segmentation loss. In our current prototype, we set λ to 1, making both of them equally important in the training process.

While the segmentation masks originally created for object highlights can be useful for training our backend model for saliency maps as we discussed above, they may also contain some errors because the generated mask is not always perfect. Such errors may lead to degradation in the accuracy of segmentation inference for saliency maps. To eliminate this effect, we introduce another parameter (Λ) to control the balance between the inference results by our backend model and CAM methods:

$$\Lambda \cdot Out + (1 - \Lambda) \cdot CAM \tag{3.1}$$

Out represents the segmentation output of the our backend model and CAM is the CAM inference result. A larger Λ value means that the system weighs more on our inference result for the output for saliency maps.

We found that taking such trade-off in consideration can greatly improve the accuracy of our saliency maps in some challenging cases. Figure 3.5 illustrates the effect of Λ in a case where a user is holding a plastic bottle. The saliency map visualization can be quite erroneous when we only use the results of CAM (Figure 3.5a). This approach would include regions that are not related to the object of interest. On the other hand, when we only use the prediction by our backend model, the result tends to be overly conservative (Figure 3.5b). One reason of this issue is over-fitting. In this example, we deliberately used different backgrounds for training and testing. As the bottle in this example was transparent, the model might have included (or overfit) some visual features of the background during training. Such features would not appear when the background was changed when being tested, and this could thus explain why our model can be very conservative.

By choosing an appropriate value for Λ, the saliency map can visualizes the object region more precisely (Figure 3.5c). We chose Λ value to be the accuracy of our object highlights in our current implementation and technical evaluation (i.e., 0.718 using EfficientNet-b0 backbone). It is out of our scope to investigate how to achieve optimization on this parameter.

3.6 Deictic Gesture Dataset

3.6.1 Motivation of Data Collection

As explained in the previous section, the backend model for object highlights needs training data of how people perform deictic gestures to objects to a camera. Among existing related human-object datasets [36, 35, 137, 94], TEgO [94] is the one that best fits our task. TEgO includes 5758 labeled egocentric images of hand-object interactions. For each image, there is a hand segmentation mask and a point-level annotation of the object location, which is not immediately sufficient for our purpose (object segmentation). We therefore attempted to infer the segmentation mask of the object by emulating a click-based interactive segmentation method [148]. We then manually inspected all the generated results and removed data samples where the inferred segmentation masks were completely inaccurate. This constitutes our customized dataset with automatically-synthesized object segmentation masks, called TEgO-Syn (n=5232).

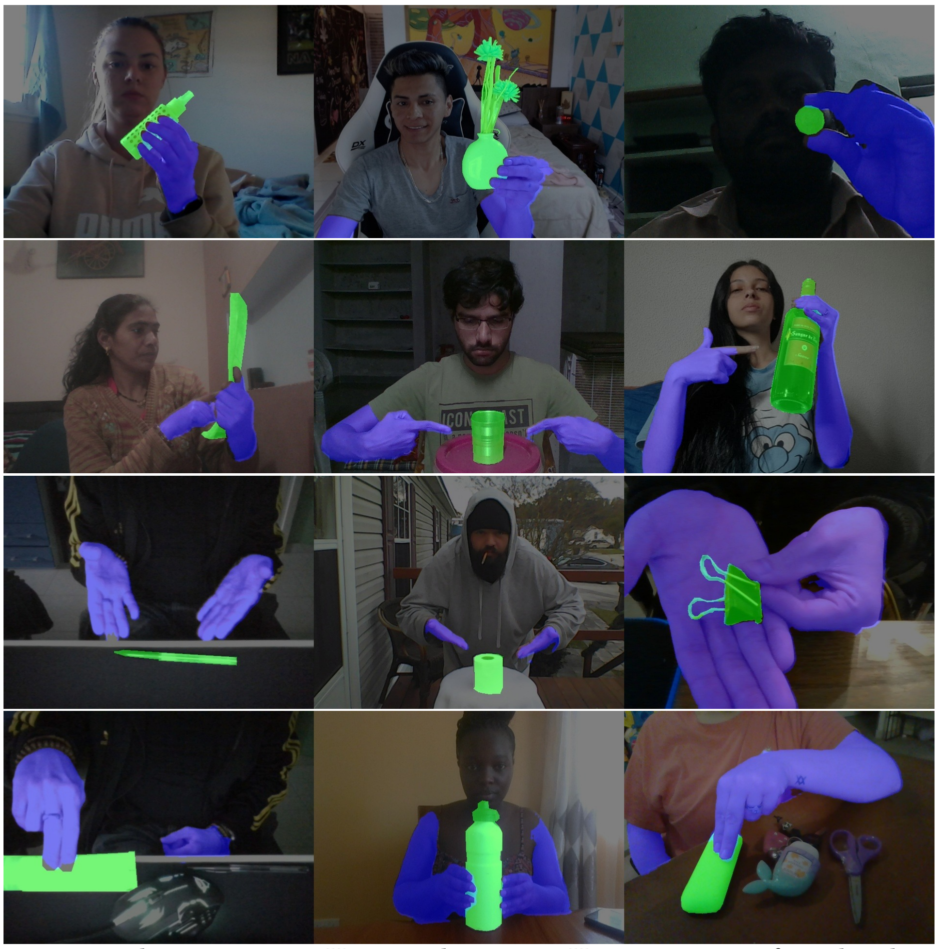


Figure 3.6 Example images in *HuTics* dataset. *HuTics* covers four kinds of deictic gestures to objects: exhibiting (top-left), pointing (top-right), presenting (bottom-left) and touching (bottom-right). The hands and objects of interest are highlighted in blue and green, respectively.

The trained network using TEgO-Syn achieved $mIoU$=0.895 on the testing set, showing a seemly-promising result. Appendix A.1.2 provides our detailed training configurations. To further evaluate the robustness of the network in real applications, we experimented with this model with images where various objects were presented through different deictic gestures. Our observations showed that the model was not robust enough which we will further confirm in Section 3.6.3. We then summarized three main reasons why TEgO still cannot fit our target task:

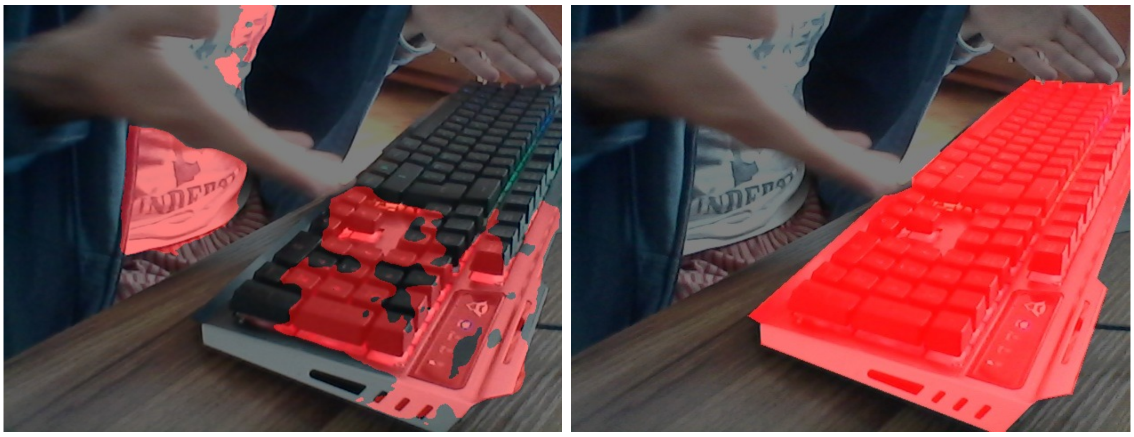
- **A limited set of gestures**. All data in TEgO were collected from two participants, which is insufficient to cover how different people interact with the object using gestures.
- **A limited set of objects**. TEgO-Syn includes 5232 images of 19 objects. Training on a small set of objects repeatedly enables the model to over-fit the features of these specific objects, which is harmful to our target task, i.e., object-agnostic segmentation.
- **Egocentric images**. The images in the TEgO dataset are taken from the egocentric view. Our system uses a front-facing camera, which is a common configuration in V-IMT [19].

3.6.2 ***HuTics*** Dataset

To address the three issues above, we created our own dataset. We recruited crowd-workers on Amazon Mechanical Turk, aiming to enhance the diversity of the dataset.[3] In each task, the worker needed to upload 12 images in total that clearly showed how they would use deictic gestures to express the references to objects. For collecting a diverse set of images from each worker, we first classified deictic gestures into four categories based on Sauppe *et al.*'s taxonomy [133]: pointing, presenting, touching and exhibiting. We then asked the workers to take three different photos for each gesture category. We also provided example pictures to clarify our expectations to the workers.

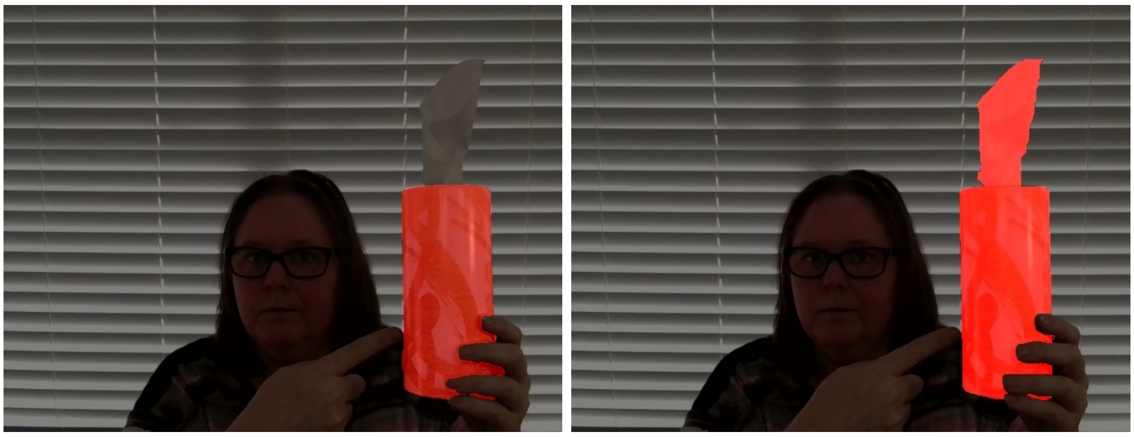
We collected 2040 qualified images from 170 crowd-workers (M: 99; F: 71) in total. The average age of the workers was 34 (*SD*: 9.2). Example unqualified submissions included images that were highly blurry or where no gesture was involved at all. The crowd-workers spent 15 minutes on average to complete the task, and

[3]We received IRB approval for this data collection at our university.



Prediction. Ground truth.

(a) An example with the model trained with TEgO-Syn (IoU=0.366).



Prediction. Ground truth.

(b) An example with the model trained with *HuTics* (IoU=0.719).

Figure 3.7 Visual comparison of predictions by the models trained with the two datasets (TEgO-Syn and *HuTics*).

we paid each participant 2 dollars. We then recruited another five people on our local crowdsourcing platform to annotate object segmentation masks on the collected images. On average, each annotation worker labeled 408 images, and we compensated them with approximately 78 dollars on average in our local currency. During the annotation, the workers used AnnoFab [73], an online polygon-based tool, to label the segmentation masks.

Figure 3.6 presents example images with the annotated object segmentation masks. Unlike TEgO, our dataset contains a wide range of objects, deictic gestures,

Table 3.1 Comparison of *HuTics* and TEgO-Syn.

	HuTics	TEgO-Syn
# Participants	170	2
Object Types	Uncontrolled	Controlled
View	Front-facing	Egocentric
# of Images	2040	5232
Annotation	Segmentation mask	Pointed-based
Target Task	Object-agnostic segmentation specified by gestures	Object recognition for people with visual impairments

backgrounds, and environmental conditions. Table 3.1 summarizes a comparison of *HuTics* v.s. TEgO-Syn.

3.6.3 Performance of Object Highlights on *HuTics*

We used the data from 80% of the participants in *HuTics* (*i.e.*, 1632 images from 136 people) for training and 20% for testing. We trained our algorithm using the same configuration above, and the network achieves *mIoU*=0.718 and 0.806 using EfficientNet-b0 and EfficientNet-b3 backbone on the testing set, respectively. Running on one GTX 2080Ti GPU, our implementation of the algorithm was able to reach 28.3 fps and 24.0 fps with the EfficientNet-b0 and EfficientNet-b3 backbone, respectively. For comparison, we trained another model with the same network architecture using TEgO-Syn and tested with images in *HuTics*. The accuracy of that model was *mIoU*=0.368, much lower than that of the same network using *HuTics* for training. This significant accuracy drop from 0.895 (tested on TEgO-Syn) further confirms our observations discussed in Section 3.6.1.

Figure 3.7 shows a visual comparison of the results between the networks trained on TEgO-Syn and *HuTics*. Each example in Figure 3.7 are the one in our testing set that has the closest IoU values (0.366 and 0.719) to the corresponding mean IoU values (0.368 and 0.718). We therefore use the model trained on *HuTics* in our current prototype implementation.

3.7 User Study

We conducted a comparative user study to evaluate how LookHere could improve the experience of V-IMT in terms of time cost for teaching, accuracy performance on models created, and subjective user workload.

3.7.1 Interface Conditions

Besides LookHere, we included the following three interface conditions to represent existing V-IMT and object annotation methods.

- *NaïveIMT*: This represents the most common design in current V-IMT systems [19, 53]. In this condition, participants would only perform object demonstrations during the teaching phase. Participants would not have an opportunity to specify which regions of the recorded images would represent the object for a given class. We implemented this naïve IMT system based on the source code of Zhou and Yatani [189] available online.

- *Contour*: In addition to the teaching process with the naïve IMT system, this condition would involve a manual annotation procedure in a post-hoc manner. In this condition, participants would be asked to perform contour-based annotations. This annotation style is widely used in IMT systems for medical purposes [12]. We used AnnoFab [73] for post-hoc contour-based annotations in this study.

- *Click*: The third reference condition included a click-based annotation method [148]. We decided to include this condition as the annotation process would be more lightweight than a contour-based approach. We used RITM [148] as the click-based annotation tool.

All these three reference conditions involve the teaching process using the naïve IMT system. To shorten the overall study time, we decided to ask participants to perform teaching under the two conditions of *NaïveIMT* and LookHere. After this teaching task, participants were then asked to perform annotations under the two conditions of *Contour* and *Click* using the data recorded under the *NaïveIMT* condition. In this manner, we liberated the participants from performing the same tasks repeatedly with *NaïveIMT* for the *Contour* and *Click* conditions.

We counter-balanced both the condition order of *NaïveIMT* and LookHere and that of *Contour* and *Click* across participants. The order of tasks of teaching and annotation was fixed (the teaching process was the first).

3.7.2 Evaluation Metrics

Teaching and Annotation Time

We measured how long it took for participants to finish the model creation process under each interface condition. Specifically, we recorded the teaching/annotation time from when the participants started uploading/annotating the first sample to when they finished the last (30th) sample.

Model Accuracy

We measured both classification accuracy and segmentation accuracy (i.e., mean Intersection over Union or *mIoU*) of the created models. We randomly used 80% of the data for training and the rest for testing. For classification accuracy, we utilized cross-condition validation. Specifically, we tested three models trained by data collected from *naïveIMT* on the data collected from LookHere, and vice versa. In terms of the object segmentation accuracy, we only performed cross validation for data collected from LookHere because there were no ground-truth segmentation annotations in LookHere to validate models created from *naïveIMT*. This ensured that LookHere gained no advantage over the three comparative conditions.

NASA-TLX

NASA Task Load Index (TLX) [63] is a standard metric for perceived workload. We included this to understand how different conditions could affect the experience of creating ML models with different configurations of IMT systems.

3.7.3 Procedure

At the beginning of the study, we told participants that their goal was to create four AI models to classify and detect objects by the given systems. For the teaching tasks, we allowed participants to use any object available in our experimental space. They were also welcome to bring their own belongings in the study. We did not limit the set of objects to be used in this experiment in order not to lose the validity of the study. After they had explanations about the two teaching methods (*NaïveIMT* and LookHere) and became comfortable with using both, they were asked to create three classes for classification, and generate 30 images for each class through the given teaching method. After completing teaching with the two methods, participants were given an opportunity to take a break.

We next moved to the annotation tasks with the two interfaces (*Contour* and *Click*). We provided explanations about these two tools, and participants were given practice time to become comfortable with using them. Participants were then asked to annotate all the images they captured under the *NaïveIMT* condition. They were instructed to perform each annotation task as fast and accurately as possible. Participants were allowed to take a break between the two sets of tasks (i.e., using the two annotation tools).

Participants were asked to fill in NASA-TLX questionnaires after finishing each of the two task sessions (teaching and annotation). In this manner, we ensured that participants remembered their experience of the conditions. When participants were rating NASA-TLX for *Contour* and *Click*, we explicitly asked them to consider their overall workload for the combination of the teaching method with *NaïveIMT* and the given annotation approach.

After completing both the teaching and annotation tasks, we conducted semi-structured interviews with participants. We first interviewed them about their overall experience and perceived benefits and shortcomings of the methods used in the study. This helped us collect their immediate use experience of each condition without being biased by the performance of the resulting models (i.e., the accuracy of the created models). We next offered our model assessment interface (Figure 3.2b) for all the four resulting models. Participants were allowed to freely use them and check whether their created model would function accurately. We then interviewed them about how they perceived their four models and would characterize them differently.

The whole study takes approximately 3.5 hours on average. We offered each participant compensation of approximately 40 USD in a local currency at the end of the experiment.

3.7.4 Apparatus

We set the video frame rate to be 24 fps across all the conditions. We used the EfficientNet-b0 backbone in our object highlights (i.e., the lightest model), aiming to understand the effectiveness of such feedback even under the least accurate setting.

3.7.5 Participants

We recruited 12 non-expert participants (P1 – P12) for this study. None of them had experience in studying or working in the fields related to AI or ML. Eight of them were female, and the rest were male. The age of participants ranged from 23 to 28.

Table 3.2 The mean values and standard deviations of the task completion time and accuracies (classification and segmentation) across the four interface conditions.

		LookHere		*NaïveIMT*	*Click*	*Contour*
		Λ=1	Λ=0.718			
Time [s]		104 (44)		67 (10)	1,197 (228)	1,483 (407)
Acc.	Cls.	0.824 (0.158)		0.847 (0.190)	0.880 (0.141)	0.833 (0.159)
	Seg.	0.578 (0.233)	0.605 (0.153)	0.139 (0.095)	0.716 (0.167)	0.732 (0.151)

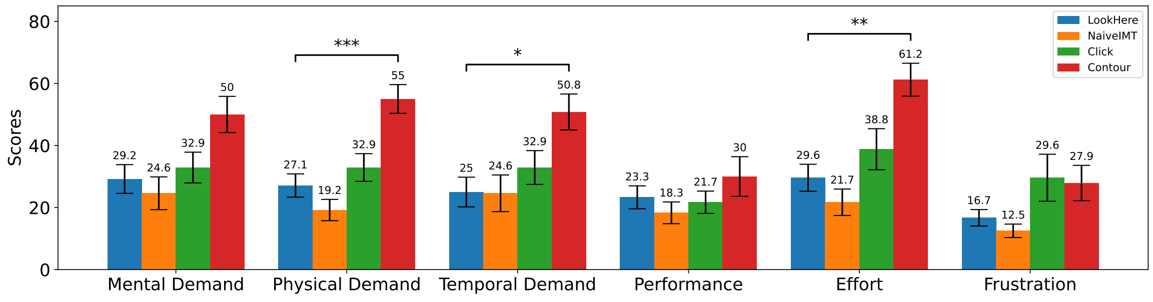


Figure 3.8 The results of the mean NASA-TLX scores across the six subscales. The error bars represent the standard errors. The observed significant differences are indicated with $*$, $**$ or $***$ (p<.05, p<.01 and p<.001, respectively).

3.8 Results

3.8.1 Quantitative Results

Table 3.2 presents the mean completion time and model accuracy in the study. With respect to the task completion, the *Contour* and *Click* conditions exhibited much longer time than the other two conditions (11.5 and 14.3 times than the LookHere condition, respectively). One-way repeated-measure ANOVA found that the factor of the conditions was significant ($F(3, 33)$=134.02, p<.001, η^2=.92). We then used Scheffe's multiple comparison procedure to compare the take completion time under the LookHere condition against the three reference conditions. We found that the completion time under the LookHere condition was significantly shorter than those under the *Contor* and *Click* conditions (both $p < .001$). This result clearly suggests that LookHere successfully removed the effort for object annotations.

We next looked into the accuracies of the models created under the four conditions. As shown in Table 3.2, the mean accuracies for classification (predicting the correct class for the given image from the three classes defined by each participant) did not show large differences. Our one-way repeated-measure ANOVA did not find a significant effect of the interfaces ($F(3,33)$=.460, p=.712, η^2=.04).

We further examined the segmentation accuracies. As shown in Table 3.2, the accuracy under the *NaïveIMT* condition was clearly lower than those with the other methods. One-way repeated-measure ANOVA found a significant effect of the interface conditions ($F(3,33)$=105.98, p<.001, η^2=.91). Scheffe's multiple comparison procedure revealed a significant difference between LookHere and *NaïveIMT* (p<.001). This result confirms that the models created with data collected under the *NaïveIMT* condition did not necessarily weigh the visual features in the objects of interest, implying potential unreliability in actual use.

We also compared the segmentation predictions on the same trained model with two different Λ values: 1 and 0.718 (the default configuration in our current implementation). While the accuracy was improved by 0.027 with the value of 0.718, this difference was not significant. Future research should investigate how Λ should be configured to achieve the best performance, but this result implies that the combination of CAM results and the inference by the backend object segmentation model could offer improvements.

We next examined the NASA-TLX results. Figure 3.8 shows the mean values of the raw NASA-TLX subscales. One-way repeated-measure ANOVA on each subscale found significant effects by the conditions in mental demand ($F(3,33)$=7.37, p<.001, η^2=.40); physical demand ($F(3,33)$=16.27, p<.001, η^2=.60); temporal demand ($F(3,33)$=7.86, p<.001, η^2=.42); effort ($F(3,33)$, p<.001, η^2=.57); and frustration ($F(3,33)$=3.23, p<.05, η^2=.23). No significant result was found in performance ($F(3,33)$=1.55, p=.22, η^2=.12). Post-hoc Scheffe's procedure revealed significant differences in physical demand, temporal demand and effort between LookHere and *Contour* (p<.001, p<.05, and p<.01, respectively). These results suggest that the contour-based annotation method significantly impacted the user experience negatively.

In summary, the quantitative results show that LookHere was able to achieve the best balance of task completion time and model accuracy. We further looked into how different user experience of the four interface conditions was through our qualitative results.

3.8.2 Qualitative Results

We transcribed the interviews and extracted quotes that were related to user experience and opinions about the four interfaces tested. We then performed the open coding approach to categorize the quotes and derive them in a bottom-up manner.

Burden for post-hoc annotations

Six participants (P1, P4, P5, P9, P10 and P12) explicitly mentioned that post-hoc annotations were tedious and reduced the perceived usability of an overall V-IMT system. While nine participants preferred the *Click* annotation method to *Contour*, all participants agreed that both approaches were *"time-consuming"*.

> *"[A good process] shouldn't contain the annotation process because it is the most time-consuming one and requires lots of effort. On the contrary, these two (NaïveIMT and LookHere) are very comfortable to use because there is only one step."* [P4]

All participants considered LookHere as *"efficient"* because it does not involve explicit post-hoc annotations. This was clear from the task completion time and NASA-TLX results, and our qualitative data were also indicative. In particular, P1 appreciated that LookHere combined the teaching and annotation process:

"It can greatly improve the user experience in terms of not only time consumption but also the sense of satisfaction." [P1]

Participants were also satisfied with the accuracy of their created models achieved through LookHere. Despite the simplified teaching experience, they could not notice the accuracy difference between LookHere and *Contour*.

"I prefer to use [LookHere]. First, its accuracy is good, and it's easy to use ... It is a user-friendly design, not requiring much effort and time." [P10]

"In terms of effort and performance, [LookHere] is definitely a cost-effective choice ... Speaking of [Contour], it requires much effort, but its result is not that good, probably similar to [LookHere]. It makes me feel that it is not worthwhile." [P6]

Uncertainty in teaching with *NaïveIMT*

Participants expressed their concerns about whether the model created with the *NaïveIMT* approach correctly interpreted their teaching.

"Because you can't find its focus, as a user, you can't confirm whether it (the computer) understands my idea." [P8]

"[NaïveIMT] is very convenient to use but I am afraid that the performance would be bad." [P3]

On the other hand, object highlights shown in LookHere offered our participants more confidence that the regions of the objects would be considered more.

"[Different from NaïveIMT, LookHere] is simple, and it also has visualizations. It can let users keep well informed whether the object is recognized [by the computer]." [P5]

In case object highlights were out of place, participants adjusted their deictic gestures until they were well overlaid onto the objects. This offered a sense of control as P12 commented:

"On one hand, the procedure is simple, and on the other hand, [LookHere] itself has already drawn that pattern (object highlights). [Even though it sometimes has errors,] I can change some positions [of the object] and it can [successfully] capture this [object] ... It provides a sense of control. Unlike [NaïveIMT], I do not know what it captures [within each image]." [P12]

Limitations

Participants pointed out limitations of LookHere, and some further provided suggestions on how we can improve the current prototype. For example, P11 raised an issue that users could not interact with LookHere using bimanual interaction since one hand is required to manipulate the mouse, clicking on the camera icon in Figure 3.2a.

Additionally, P4 pointed out that there was a lack of further teaching/clarification support when object highlights fail. P4 further suggested that V-IMT should integrate more functions so that users can better correct object highlights in erroneous cases, rather than passively avoid teaching these samples.

"In terms of [LookHere], is it possible to utilize the Click function there? For example, when I hold something, [if object highlights are erroneous at this moment,] can I tell the computer which region I want it to recognize [by clicking on the object]? ... In the current design, I can only change the position (of the object) to adjust it (the highlight), and this makes me feel quite inactive (i.e., not in good control of object highlights)." [P4]

3.9 Discussion

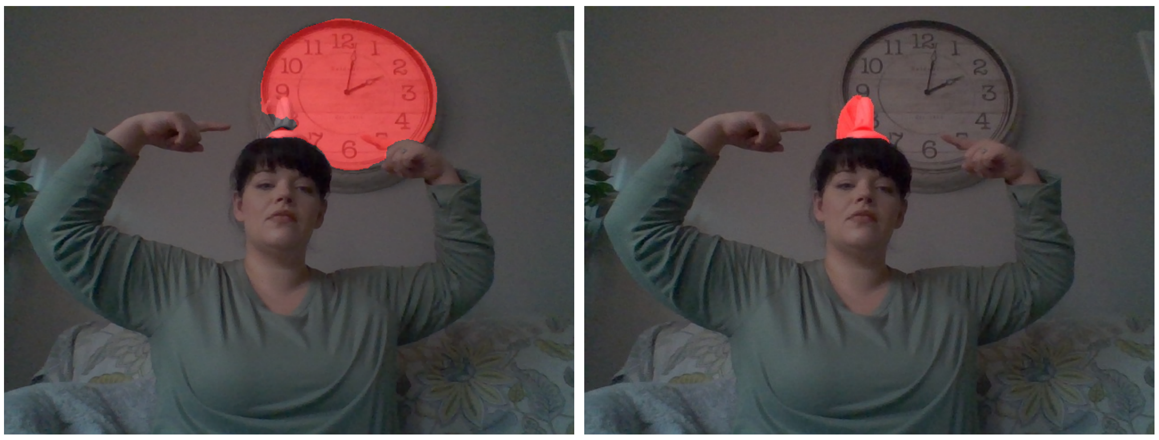
As mentioned in Section 3.2, this work aims to (1) explore technical solutions for the integration of object annotations into the teaching process (i.e., RQ1) and (2) study the effectiveness of the solution (i.e., RQ2). We answer RQ1 through our implementation of LookHere as explained in Section 4 and 5. Our evaluation results further show that LookHere can effectively reduce users' workload during the model creation process while maintaining similar model accuracies, answering RQ2.

Despite its effectiveness, we also found several drawbacks of our systems that limited user experience in practice. In the following content, we share our insights about how future work can improve our system and how to extend our research questions to exploit more human interactions to achieve in-situ annotations in IMT.

Additionally, our dataset, HuTics, which is designed for the gesture-aware object-agnostic segmentation task, is one important contribution of this work. We further show that our approach and dataset can also be used to support other HCI projects by demonstrating several example applications.

3.9.1 Depth-aware Object Highlights

Despite the effectiveness of object highlights to support efficient teaching, we still observed typical erroneous cases that remain to be addressed. Figure 3.9 shows an example where IoU was low (more examples can be found in Appendix A.2). The person in this figure is pointing at an object on her head, but our algorithm incorrectly highlights the clock in the background. Our object highlights also tend to fail in cases where a person is pointing at an object at a distance (e.g., buildings or furniture that are not close to hands). These failure cases are mainly caused by the current implementation that uses 2D hand segmentation features without a 3D understanding of the scene. A future system may consider obtaining a richer set of information through 3D scene reconstruction [34], 3D hand pose estimation [18, 71] from RGB



(a) Prediction. (b) Ground truth.

Figure 3.9 A failure case of object highlights.

images, or directly use depth cameras [74]. Future research should further study how to simplify the aforementioned feature extractors to be used in real time for V-IMT systems or how to use depth sensors to support V-IMT systems [177, 176].

3.9.2 Voice Input and In-situ Correction

As mentioned in Section 3.8.2, we observed several limitations of our interface design. To enable bimanual interactions with the object, future research can investigate how to use technologies like voice input or facial expression recognition to replace a button click. For example, when users want the system to sample the current frame, they can simply say “collect” or smile to the system while performing bimanual deictic gestures.

In addition, as mentioned in Section 3.8.2, future systems should study how to enable users to *actively* correct object highlights when they observe prediction errors. Although the segmentation annotation in LookHere allows users to choose appropriate frames for teaching, the role of users in this Human-AI collaboration is relatively passive. When users observe the failure case of object highlights, they should be given an opportunity to *actively* correct the error [146], instead of *passively* avoiding those data. Allowing such in-situ correction initiated by users can further empower the ability of IMT systems to *"leverage human capabilities and human knowledge beyond labels"* [127], achieving better human-AI collaboration.

3.9.3 Other Modalities and Privacy Issues

While this work focuses on studying how to use deictic gestures to enable in-situ annotations, they are not the only human interaction that future IMT research can exploit. As we discussed in Section 3.9.1, our gesture-aware annotation approach may not function with some deictic gestures users may perform. More importantly, humans also innately perform other interactions as a cue of objects of interest. For example, future research can study how to use gaze tracking technologies to capture the object of interest that is difficult to hold by hand (e.g., buildings or scenery). While examining other modalities is out of scope of this work, future work on this aspect is encouraged.

Despite the benefits from collecting fine-grained annotation by sensing additional human interactions, such systems without proper designs may cause severe privacy issues. We therefore encourage future research to study how to balance privacy protection and the benefits from in-situ annotations in IMT.

3.9.4 Applications of the Object-agnostic Segmentation Model Trained on *HuTics*

One important contribution of this work is our object-agnostic segmentation model and its dataset, *HuTics*. Although our original objective was to enable LookHere, we envision that our model can be used for a broader range of applications, not limited to IMT research.

Intelligent Virtual Background

Using our model, developers can create an intelligent virtual background used in online meeting systems which is aware of the object users are trying to present to others. Segmentation algorithms for virtual backgrounds do not typically consider the behavior of object presentation. Therefore, virtual backgrounds often hide the objects held by users, diminishing user experience in certain scenarios. Our model can address this issue and show the object held by the user while preserving virtual backgrounds to support a better communication experience.

Gesture-guided Portrait Mode

Portrait mode in recent smartphones allows users to have a focus effect (e.g., blurring the background to highlight a person in the foreground). Using our model, such a portrait mode may create a focus effect on the objects held by users intelligently. Our object-agnostic segmentation model thus has a potential to enrich user experience of photo shooting with smart devices.

Supports for People with Visual Impairments

Prior work demonstrated an assistive technology for people with visual impairments by recognizing objects held by users. While it only recognized 19 objects constrained by the dataset used in that project, future assistive systems may develop a more generalizable approach by using our model trained on *HuTics*. They can first locate an object held by users using our object-agnostic segmentation model, and then apply a classification model trained on large-scale datasets that cover thousands of objects (e.g., 1000 classes in ImageNet [40]), achieving the goal of recognizing various objects for supporting activities of people with visual impairments.

3.10 Summary

This work demonstrates LookHere, a V-IMT system that allows users to annotate objects in real time during the teaching phase by exploiting users' deictic gestures.

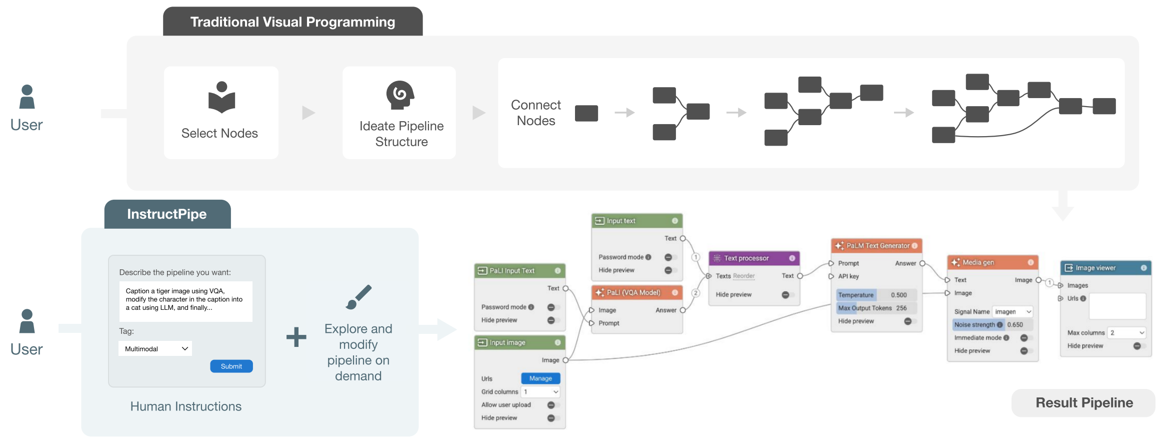
We build our own dataset (*HuTics*), consisting of 2040 front-facing images of deictic gestures and objects to achieve our implementation. Our user study results show that LookHere successfully removed substantial user effort on post-hoc manual annotations. However, the models created through LookHere did not show significant differences in their accuracies compared to those using the data with manual annotations.

Chapter 4

InstructPipe: Building Visual Programming Pipelines with Human Instructions

4.1 Introduction

A *visual programming* interface provides users with a node-graph editor to program. As opposed to writing code in a code editor, the node graph allows users to build a pipeline in a visual workspace with nodes and edges. This approach effectively reduces technical barriers for users to prototype creative applications. Advances in machine learning (ML) further stimulate growing interest in visual programming. Open-sourced ML libraries (*e.g.*, TensorFlow [2], PyTorch [121], and Hugging Face [166]) provide users with various encapsulated modules to accelerate AI project development and experimentation. Meanwhile, this also provides valuable protocols for visual programming developers to create systems for ML applications [45], where ML practitioners can interactively test "off-the-shelf" models on the node-graph editor. Recent foundational models like large language models (LLMs) [158, 14, 7]



Describe the pipeline you want:

Caption a tiger image using VQA, modify the character in the caption into a cat using LLM, and finally...

Tag: Multimodal

Temperature 0.500
Max Output Tokens 256
Signal Name imagen
Noise strength 0.650
Grid columns 1
Max columns 2

Figure 4.1 Traditional visual programming requires users to select nodes, conceive pipeline structure, and then create the pipeline with a node-graph editor. In contrast, InstructPipe streamlines this process by instantiating a multi-modal machine learning pipeline directly from human instructions, enabling users to further iterate and interact with the pipeline across diverse applications such as news summarization, image style transfer, and real-time AR effects.

and findings on Chain-of-Thought [164] further stimulate a community-wise interest in visual programming [167, 169, 46], which provides users interactive experiences to explore AI chains.

Despite the development of visual programming platforms in various domains, we observed that existing systems share one similar characteristic: users usually start a creative process in the workspace *"from scratch"*. This implies that users need to 1) select nodes, 2) ideate the pipeline structure, and finally, 3) connect nodes from *a completely empty workspace*. For users who are unfamiliar with a particular visual programming platform, such processes can easily overwhelm them, degrading their overall programming experience.

Similar issues also exist when users write programs using text-based editors (there exist many built-in functions in a particular programming language and multiple variables in a program), but advances in LLM assistants show that such challenges can be effectively reduced. For example, GitHub Copilot [56] makes it possible for users to generate code by simply describing users' requirements in natural language. Even though the generated code is not absolutely correct, the AI assistance usually finishes a large portion of the task, and programmers may only need to make a few edits to achieve a correct result. To this end, we raise the following questions that motivate our work: *How can we build such an assistant to benefit visual programming users?*

In this work, we built InstructPipe, an AI assistant for visual programming users to generate a pipeline through natural language instructions (Figure 4.1). We implemented InstructPipe on Visual Blocks [45], a visual programming system for prototyping ML pipelines. One major technical challenge in implementing InstructPipe lies in the lack of visual programming data. Different from the standard approach to building a copilot for the text-based editor (*i.e.*, training LLMs using large-scale text-based programs online), it is impractical to collect sufficient data for a particular visual programming platform. We addressed this issue by decomposing the generation process into three steps. Using two separate LLM modules, the system first scopes the potentially useful nodes and then generates pseudocode for a target pipeline. InstructPipe then compiles the pseudocode and renders the pipeline on the node-graph editor to facilitate further user interaction. Our technical evaluation suggests that InstructPipe reduces user interactions by 81.1% compared to building pipelines from scratch. Our system evaluation with 16 participants demonstrated that InstructPipe significantly reduced users' workload in their creative process. Qualitative results further reveal that InstructPipe effectively supports novices' *on-boarding* experience of visual programming systems and allows them to easily prototype a concept for various purposes. As one pioneering work on visual programming copilot, we also observed new challenges caused by humans' cognitive characteristics and proposed future technical directions toward a next-level, open-ended AI prototyping assistant.

In summary, this work offers the following contributions:

1. InstructPipe, an AI assistant that enables users to *build ML pipelines from human instructions*,

2. System design and technical development of InstructPipe, which includes two LLM modules and a code interpreter that generate codes for a visual programming pipeline, compile the code, and render the pipeline in an interactive node-graph editor,

3. A technical and a user evaluations that demonstrate effectiveness of InstructPipe, and the corresponding findings that reveal new challenges for the HCI community.

4.2 Related Work

4.2.1 Visual Programming

The operation of computer systems is defined by a computer program. However, *"the program given to a computer for solving a problem need not be in a written format"* [152]. This future-looking statement, dating back to 1960s, inspired several generations of researchers to design and build visual programming systems.

Today, visual programming systems (*e.g.*, LabView [85], Unity Graph Editor [157], PromptChainer [167], and Visual Blocks [45]) typically feature a node graph editor, providing users with a visual workspace to "write" their program. Recent work further explored the application of visual programming in education [76, 87], authoring support [183, 179], and robotics [37, 70, 69]. Zhang et al. [183] connected the visual programming tool to the concept of *teaching by demonstration* [190], allowing users to rapidly customize AR affects in video creation. FlowMatic [179] extended traditional visual programming interfaces into 3D virtual environments, providing users with immersive authoring experiences.

More recently, findings on LLM Chains [169] and Chain-of-Thought [164] further stimulated researchers to build visual programming tools that chain LLM modules. Developers want to explore various ways to chain an LLM module for various application, and in such scenario, visual programming provides a great platform for users to focus on the high-level exploration. Example research work and industrial products include PromptChainer [167], LangFlow [91], and ComfyUI [32].

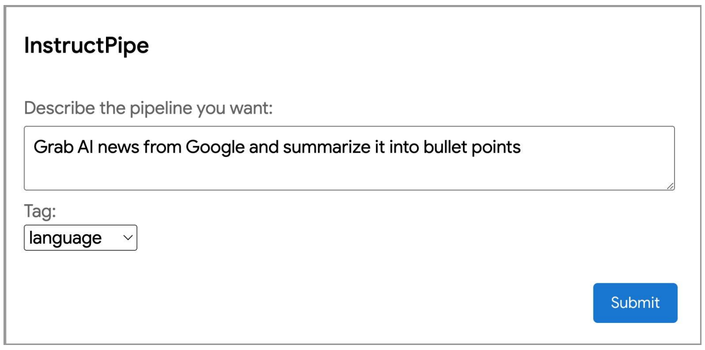
InstructPipe offers technical contributions that allow users to generate a pipeline using text-based instruction, providing users with new experience beyond building a pipeline from scratch.

4.2.2 LLMs and Their Applications in Interactive Systems

Early multimodal ML work uses language models to solve Visual Question Answering (VQA) [6, 8], but these solutions are limited to simple questions and cannot perform effective reasoning and problem solving [106]. LLMs revolutionized AI's reasoning capability [164, 186] in language, which motivated researchers to build LLM applications in various domains beyond NLP [151, 142, 119]. For instance, LLMs augmented the perceptual and planning intelligence in robotics [142], supported autonomous driving [100] and assisted clinical processes in medical science [89, 144].

The advances in LLMs empower recent interactive systems [123, 120] with enhanced machine intelligence. Recent research leverages LLMs to edit visualization [163, 138], receive AI explanation [168], facilitate communications [104], understand user interface [162], and study simulated social behaviors [119]. The revolution also motivated HCI researchers to design new interfaces for LLMs [77, 150]. Graphologue [77] augmented LLM response by displaying interactive diagrams on the side of the response text, which visualizes the semantic logics in a paragraph. Sensecape [150] provides users with a workspace to explore long LLM response in a hierarchical structure.

InstructPipe focuses on utilizing LLMs for generating pipelines in visual programming with human instructions. This work shares similar vision with Prompt2Model [160] and VisProg [60]. Prompt2Model [160] finetunes a BERT model [41] using data generated by instructions. VisProg [60] produces Python code

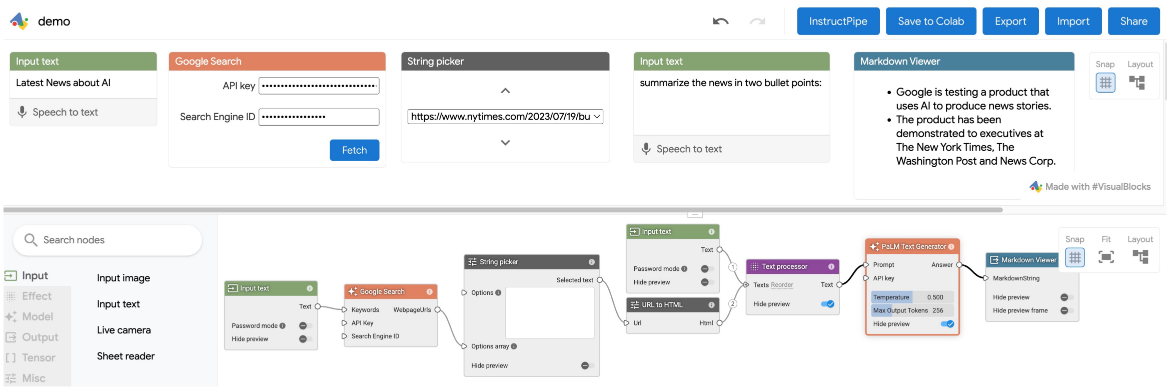

InstructPipe

Describe the pipeline you want:

Grab AI news from Google and summarize it into bullet points

Tag: language

(a) InstructPipe's instruction dialog.



Latest News about AI

API key ••••••••••••••••••••••••••••••

Search Engine ID ••••••••••••••••

https://www.nytimes.com/2023/07/19/bu

summarize the news in two bullet points:

- Google is testing a product that uses AI to produce news stories.
- The product has been demonstrated to executives at The New York Times, The Washington Post and News Corp.

Temperature 0.500

Max Output Tokens 256

(b) InstructPipe's visual programming interface.

Figure 4.2 The user interface of InstructPipe. The user can first click on the "InstructPipe" button on the top-right corner of the interface in (b). A dialog will appear, and the user can input the instruction and select a category tag. InstructPipe then renders a pipeline on (b), in which the user can interactively explore and revise.

from instructions with task-dependent few-shot prompting. However, both prior arts lack an interactive workspace that facilities novices to use. In contrast, InstructPipe generates and compiles a pipeline (without fine-tuning), while rendering the pipeline in a visual programming interface, facilitating an interactive, collaborative, and explainable workflow.

4.3 InstructPipe

InstructPipe is an AI assistant that enables users to generate a visual programming pipeline by simply providing text-based instructions [187]. We implemented InstructPipe on Visual Blocks [45], a visual programming system for prototyping

ML pipelines that handles texts and images. Technically, InstructPipe takes user's instruction (texts) as *inputs* and returns a directed graph consisting of nodes and directed links (directed edges), and the nodes are selected from our node library. Note that InstructPipe assigns the node parameter with default values, which implies that InstructPipe leaves the parameter tuning task on the visual programming interface to the user.

4.3.1 User Workflow

To generate a pipeline, users can first click the "InstructPipe" button on the top-right corner of the interface (Figure 4.2b). The system then activates a simple dialog (Figure 4.2a) in which users can 1) provide a description of their target pipeline and 2) tag the pipeline. The tag can be "language", "visual", or "multimodal". After users click the "Submit" button in the dialog, InstructPipe renders a visual programming pipeline on the node-graph editor. Based on the result, users can further refine the pipeline in the visual programming interface.

4.3.2 Primitive Nodes

InstructPipe supports 27 primitive nodes in Visual Blocks, including seven I/O nodes (*e.g.*, live camera and markdown viewer) and 20 processor nodes. section B.1 provides a description for each of the 27 nodes. Compared to related work [60, 151] that automates ad hoc ML inferences in specific use scenarios, we aim at an open-ended use case with diverse primitive nodes. Figure 4.3 visualizes 17 (out of 20 in total) processor nodes according to the I/Os of the nodes. The remaining three nodes are "Google Sheet" (which takes a Google Sheet URL as input and outputs the sheet data), "image_mixer" (which combines multiple images), and "virtual_sticker" (which casts a sticker on a person's face on a live camera). "Metadata" in Figure 4.3 indicates intermediate data used in ML pipelines, which can be a segmentation mask that

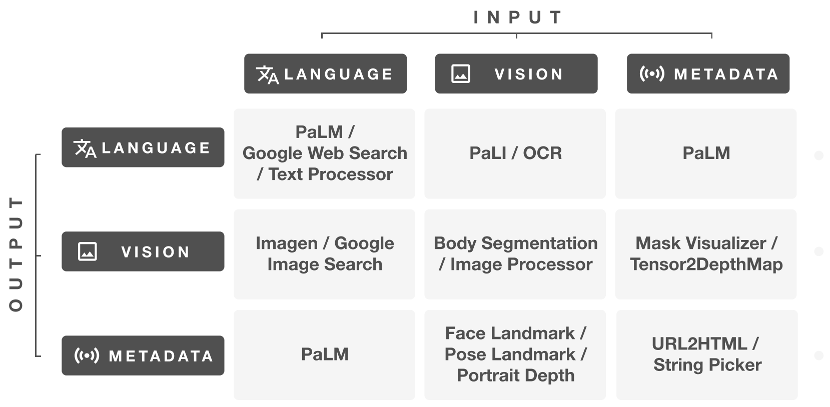

INPUT
LANGUAGE
VISION
METADATA
OUTPUT
LANGUAGE
VISION
METADATA
PaLM / Google Web Search / Text Processor
PaLI / OCR
PaLM
Imagen / Google Image Search
Body Segmentation / Image Processor
Mask Visualizer / Tensor2DepthMap
PaLM
Face Landmark / Pose Landmark / Portrait Depth
URL2HTML / String Picker

Figure 4.3 An overview of major primitive processor nodes supported by InstructPipe. 17 of the 20 processor nodes are categorized by the type of accepted I/O into a 3×3 matrix. Note that "PaLM" represents two nodes in InstructPipe, *i.e.*, a text generation model and a chat model of PaLM [7].

describes an input image, or a URL that describes a target news that users want to read. As shown in the matrix, InstructPipe contains a wide range of nodes that supports the creation of complex ML pipelines.

4.4 Pipeline Generation from Instructions

InstructPipe leverages LLMs to generate visual programming pipelines from text instructions. There exist two prevailing approaches for customizing LLMs: 1) fine-tuning LLMs [13, 104], and 2) few-shot prompting [119, 44]. For our task, fine-tuning LLMs requires a substantial volume of annotated data comprising pairs of pipelines and prompts. Additionally, a growing list of nodes poses new challenges to scaling this approach with new data annotations. In comparison, few-shot prompting can seem more practical [60, 164, 170], but it is challenging to design an efficient prompt that fits within a reasonable number of tokens. The configuration file of the 27 nodes alone includes 8.2k tokens. Moreover, because of the combinatorial explosion of the 27 nodes in the system, it is not clear how many prompt examples are needed and how we can construct in-context pipeline examples.

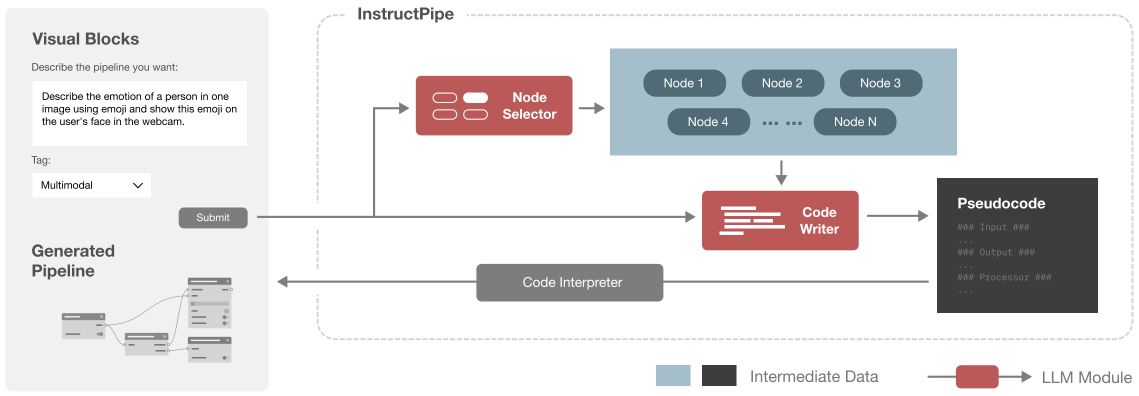
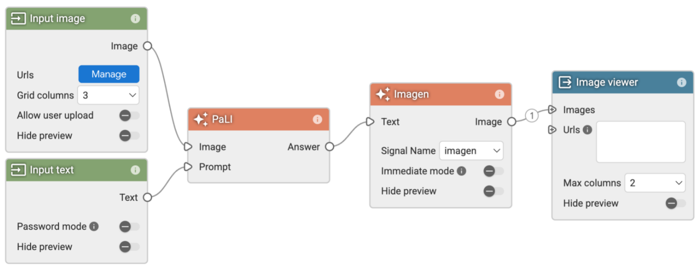


Figure 4.4 Workflow of InstructPipe. First, users describe their desired pipeline in natural language and designate it with a language, image, or multi-modal tag. InstructPipe then feeds user instructions into a node selector to identify a relevant set of nodes. Subsequently, both the instructions and the relevant nodes with their description are input into a code writer to produce pseudocode. Finally, a code interpreter parses the pseudocode, rectifies errors, and compiles a JSON-formatted pipeline, allowing users to refine and interact with it further within Visual Blocks's node-graph editor.

To this end, we implement InstructPipe with a two-stage LLM refinement prompting strategy, followed by a pseudocode interpretation step to render a pipeline. Figure 4.4 illustrates the high-level workflow of the InstructPipe implementation. InstructPipe includes two LLM modules (highlighted in red): 1) a Node Selector (§4.4.2) and 2) a Code Writer (§4.4.3). Given a user instruction and a pipeline tag, we first devise the Node Selector to identify a list of potential nodes that would be used according to the instruction. In the Node Selector, we prompt the LLM with a very brief description of each node, aiming to filter out unrelated nodes for a target pipeline. The selected nodes and the original user input (the prompt and the tag) are then fed into the Code Writer, which generates pseudocode for the desired pipeline. In Code Writer, we provide the LLM with detailed description and examples of each selected nodes to ensure LLMs have a thorough understanding of each candidate node. Finally, we employ a Code Interpreter to parse the pseudocode and render a visual programming pipeline for the user to interact with.



(a) Pipeline.

```
###Input###
input_image_1: input_image();
input_text_1: input_text(text="caption this image in detail");

###Output###
image_viewer_1: image_viewer(images=imagen_1_out);

###Processor###
pali_1_out = pali_1: pali(image=input_image_1,
prompt=input_text_1);
imagen_1_out = imagen_1: imagen(text=pali_1_out);
```

(b) pseudocode.

Figure 4.5 A pair example of pipeline and pseudocode. In the first line of code under "processor", pali_1_out, pali_1, pali and image=input_image_1, prompt=input_text_1 represents output variable id, node id, node type, and node arguments, respectively.

4.4.1 Pipeline Representation

The Visual Blocks system represents a pipeline as a Directed Acyclic Graph (DAG) in JSON format[1]. To compress the verbose JSON file, InstructPipe represents pipelines as pseudocode [60, 151], which can be further compiled into a JSON-formatted pipeline. The pseudocode representation is highly token-efficient. Figure 4.5 shows an example in which the pseudo compresses the JSON-based pipeline representation (2.8k tokens) into a 123-token pseudocode representation. The efficiency does come with a cost: it loses some fine-grained annotations of each node like property values (*e.g.*, layout of the nodes, parameters of a segmentation model, and the degree of blurring parameters in an "image processor" node). InstructPipe leaves the task of finetuning

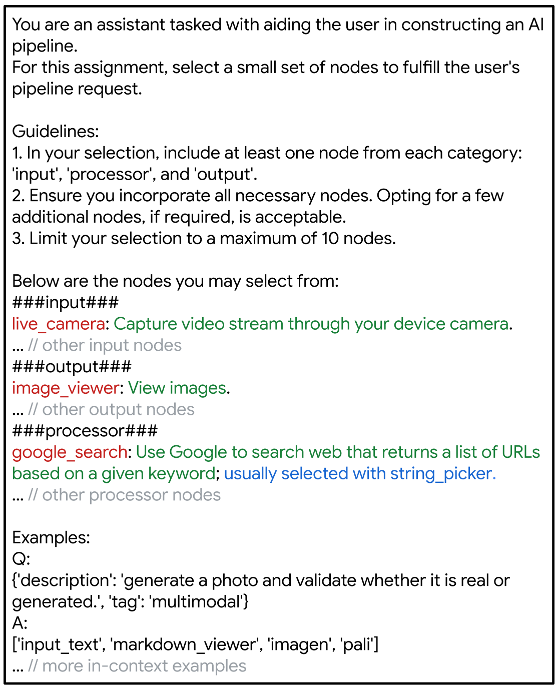
[1]Pipeline in JSON files: see supplementary materials for examples.

these parameters to the user and focuses on generating the graphic structure of a visual programming pipeline.

Figure 4.5 provides an example of a pipeline (Figure 4.5a) and its corresponding pseudocode (Figure 4.5b). The syntax design is inspired by TypeScript. The structure is inspired by how academic papers present pseudocode [184]: an algorithm block typically starts with specifying the input/output and then explains the intermediate processor. We highlight the first line under the processor module (*i.e.*, the operation of the PaLI node) in Figure 4.5b in four different colors, representing four different components in the programming language. "pali_1_out" represents the output variable name of the node. "pali_1" is the unique ID of the node. The green symbol after the colon, *i.e.*, "pali", specifies the type of the node. In this example, the node with the ID of "*pali*_1" is a "pali" node. The rest part in the bracket, *i.e.*, "image=input_image_1, prompt=input_text_1", defines the arguments of the nodes. In the input pseudocode, we do not annotate the output variable (*i.e.*, there are no red colors in the highlighted line under the input module) because all the input nodes only export one value, and the output variable name is automatically annotated as the same symbol as the node id (*i.e.*, "input_text_1"). Note that InstructPipe generates the text input (*i.e.*, the property value) in the "input text" node. Therefore, the argument in "text="caption this image in detail"" does not indicate that the "input_text" node accepts input edges, but accepts the node property input.

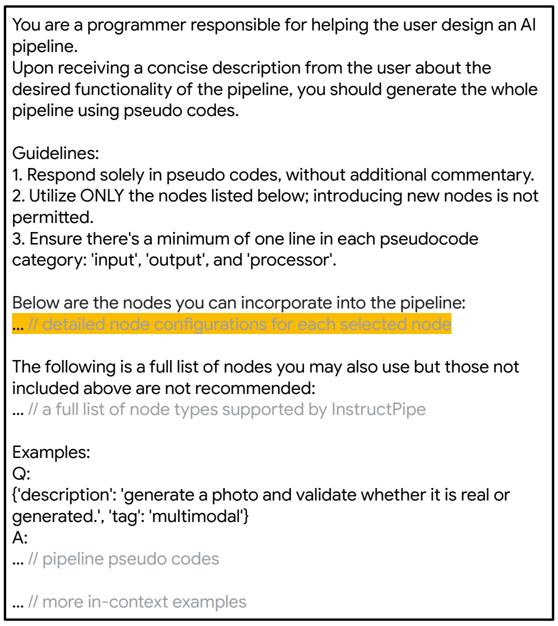
4.4.2 Node Selector

Node Selector aims to filter unrelated nodes by providing an LLM with a short description of each node. Figure 4.6 shows the prompt we use in Node Selector. Followed by a general task description and guidelines, we list all the node types with a short description that explains the function of the node. Several nodes come with recommendation(s) when the users interact with Visual Blocks, and we also include



```
You are an assistant tasked with aiding the user in constructing an AI
pipeline.
For this assignment, select a small set of nodes to fulfill the user's
pipeline request.

Guidelines:
1. In your selection, include at least one node from each category:
'input', 'processor', and 'output'.
2. Ensure you incorporate all necessary nodes. Opting for a few
additional nodes, if required, is acceptable.
3. Limit your selection to a maximum of 10 nodes.

Below are the nodes you may select from:
###input###
live_camera: Capture video stream through your device camera.
... // other input nodes
###output###
image_viewer: View images.
... // other output nodes
###processor###
google_search: Use Google to search web that returns a list of URLs
based on a given keyword; usually selected with string_picker.
... // other processor nodes

Examples:
Q:
{'description': 'generate a photo and validate whether it is real or
generated.', 'tag': 'multimodal'}
A:
['input_text', 'markdown_viewer', 'imagen', 'pali']
... // more in-context examples
```

Figure 4.6 The prompt structure for the Node Selection module. Each node description is formated as "{node types}: {short descriptions of the nodes}; {recommended node(s)}". The node recommendation is optional.



```
You are a programmer responsible for helping the user design an AI
pipeline.
Upon receiving a concise description from the user about the
desired functionality of the pipeline, you should generate the whole
pipeline using pseudo codes.

Guidelines:
1. Respond solely in pseudo codes, without additional commentary.
2. Utilize ONLY the nodes listed below; introducing new nodes is not
permitted.
3. Ensure there's a minimum of one line in each pseudocode
category: 'input', 'output', and 'processor'.

Below are the nodes you can incorporate into the pipeline:
... // detailed node configurations for each selected node

The following is a full list of nodes you may also use but those not
included above are not recommended:
... // a full list of node types supported by InstructPipe

Examples:
Q:
{'description': 'generate a photo and validate whether it is real or
generated.', 'tag': 'multimodal'}
A:
... // pipeline pseudo codes

... // more in-context examples
```

Figure 4.7 The prompt structure for the Code Writer module. Detailed node configurations, see supplementary materials for examples, are listed in the highlighted region.

such node recommendations information in the prompt. The main intuition of this prompt design is based on how the existing open-source libraries (*e.g.*, Numpy [62]) present a high-level overview of all functions[2]. The documentation typically presents a list of supported functions (in each category), followed by a short description so that developers can quickly find their desired functions. At the end of the prompt, we provide a list of Q&A as few-shot examples to support the LLM to learn and adapt to the context of the task.

4.4.3 Code Writer

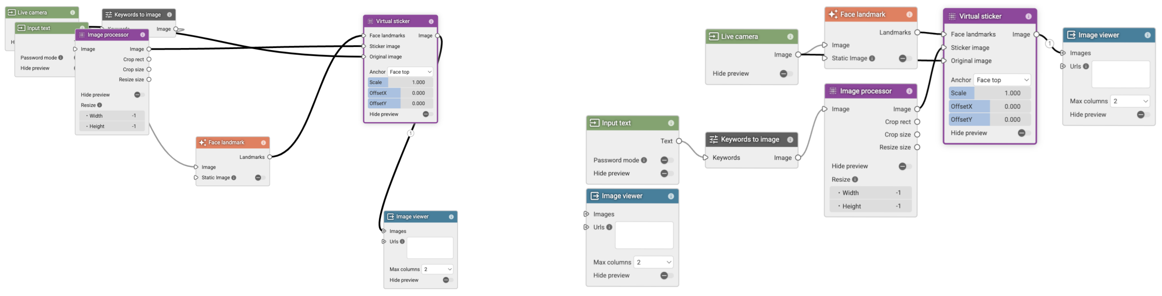
With a pool of selected nodes, the Code Writer module can write pseudocode for rendering a target pipeline. Figure 4.7 shows the structure of the prompt utilized in

[2]See an example in the following link: https://numpy.org/doc/1.25/reference/routines.array-manipulation.html

this LLM module. Similar to §4.4.2, the prompt starts with a general introduction and several guidelines. The major difference in the prompt design in this stage lies in the granularity of each node introduction. We provide a detailed configuration for each selected node with additional information, including 1) input data types, 2) output data types, and 3) an example, represented in pseudocodes, showing how this node connects to other nodes. We put a detailed explanation of the full node configuration in the supplementary materials. Similar to the previous LLM module (§4.4.2), the prompt design here is also inspired by the documentation of several popular code libraries. Specifically, we gain inspiration from low-level function-specific documentation[3], which typically includes a detailed description and data types in the input/output, followed by one or more examples of how developers can use this function with a few lines of codes.

The prompt also includes a list of Q&A as few-shot examples. However, providing few-shot examples in this stage is non-trivial. The reason lies in the dynamics of the node selection pool: a combination of all the nodes causes many possible selections, and it is impossible to design a list of few-shot examples for each possibility. Therefore, we only created an example list for each pipeline tag (*i.e.*, "language", "visual", and "multimodal") and intended to utilize these few-shot examples to cover most of the use cases. The intuition behind this design is: the in-context examples serve to list possible creative ideas of the pipelines in each tag, which can be condensed into few-shot examples rather than traversing all the possible combinations. This implies that the in-context examples may include nodes that are not selected in the prompts. According to our preliminary tests, such out-of-scope nodes cause negative effects on the generation results: LLMs tend to also *"invent"* new nodes that do not exist in our system, causing failure in node rendering. Note that we also observed this issue when combining the prompts used in [60]. We eliminate this issue by using the

[3]See an example in numpy.shape: https://numpy.org/doc/1.25/reference/generated/numpy.shape.html#numpy-shape



(a) Before layout optimization. (b) After layout optimization.

Figure 4.8 A comparison of the same generated pipeline before and after layout optimization. (a): Each node is assigned with a default property value when convert pseudocode into a JSON file for rendering the pipeline. Such default values will cause sub-optimal visual effects. (b): Our layout optimization process re-arranges the layout for better presentation of the pipeline.

prompt contents between the node configurations and the in-context examples (*i.e.*, the contents start with "the following is a full list of ..." in Figure 4.7). This aims to inform LLMs of the exceptions for invention.

4.4.4 Code Interpreter

Finally, InstructPipe employs a code interpreter to parse the generated pseudocode and compile a JSON-formatted pipeline with an automatic layout. We delineate the graph compilation and rendering procedure below:

1. **Lexical Analysis**: InstructPipe first tokenizes each line of the pseudocode is into output variable id, node id, node type, and node arguments (§4.4.1).

2. **Graph Generation**: The tokenized results allow us to build a graph structure in a node-graph editor that connects each node as specified by the pseudocode. We then generate JSON-formatted code, which Visual Blocks uses to render the pipeline in the node-graph editor. Note that the JSON code includes far more parameters than those used in InstructPipe for defining the graphic structure. InstructPipe fills texts in the "Input Text" node based on LLM-generated pseudocode and uses default values for the rest nodes. For example, by default, the temperature and the max output tokens for the PaLM node are set to 0.5 and 256, respectively. If users are

not satisfied with the default values, they can interactively adjust the parameters in the node-graph editor.

3. **Graph Rendering and Optimization**: A problem with employing the default parameters is that it results in a chaotic distribution of nodes in the node-graph editor: each node is located in a predefined position, and the edges go across the workspace without a reasonable arrangement. Therefore, InstructPipe traverses the graph with a breadth-first search (BFS) and arranges the nodes based on their depth values in the DAG. As shown in Figure 4.8, such post-processing can effectively enhance the visualization of a generated pipeline.

Our exploration shows that LLMs may not always generate accurate code as expected. One typical issue is that Code Writer tends to invent illegal nodes out of our library of 27 nodes. To address this, InstructPipe disposes such lines that leverage illegal nodes. If legal nodes use the output of such an illegal node, InstructPipe discards the edge connection between them to prevent bugs when running the pipeline. Another common issue is that the generated code may not have a correct order. For example, the generated code may be "Line 1; Line 3; Line 4; Line 2;", in which "Line 1 – 4" represents four lines of code, and the correct order should be from "Line 1" to "Line 4". Such ordering issues would cause an input value of the current line to become undefined, because its definition is mistakenly placed afterward. We address this issue by stacking the line that is not ready for execution, similar to graph construction by traversing an adjacency list. The Code Interpreter runs in a loop to interpret the code until all legal lines in the stack become ready.

4.5 Technical Evaluation

Evaluating our system is challenging. Ideally, the evaluation set should cover 1) diverse visual programming pipelines (*i.e.*, the pipeline factor) and 2) a variety of instructions created by different individuals for each pipeline (*i.e.*, the human factor). However, such a combination requires a significant number of user evaluation sessions, which is impractical to deploy. For example, if an evaluation set includes 50 pipelines and 20 instructions for each pipeline from 20 different participants, it requires 1,000 user study sessions. Asking participants to attend multiple pipeline creation sessions seems efficient, but it introduces new learning transfer effects to the study. The learn effects usually require additional user study design with counterbalancing, which eventually increases the total number of required sessions.

To address this issue, we decouple the two factors that cause variations by decomposing the whole evaluation into two evaluation sessions. The major benefit of this design is that it allows us to deploy the evaluation tasks with a reasonable workload while maintaining the rigorousness of the whole evaluation. In our first evaluation session (*i.e.*, technical evaluation), we focus on understanding the technical performance of InstructPipe among a variety of pipelines. In the second evaluation session (*i.e.*, user evaluation), we control the pipeline factor and examine the human factor by recruiting different participants for prototyping controlled pipelines. The first technical evaluation provides crucial data on the “accuracy” of InstructPipe, which guides us to select such representative pipelines with as little bias as possible. More importantly, the user evaluation further allows us to understand how user experiences are affected under two conditions: with and without InstructPipe.

In this section, we focus on explaining the first step of our evaluation, the technical evaluation, and the following user evaluation is covered in the next section.

4.5.1 Data Collection

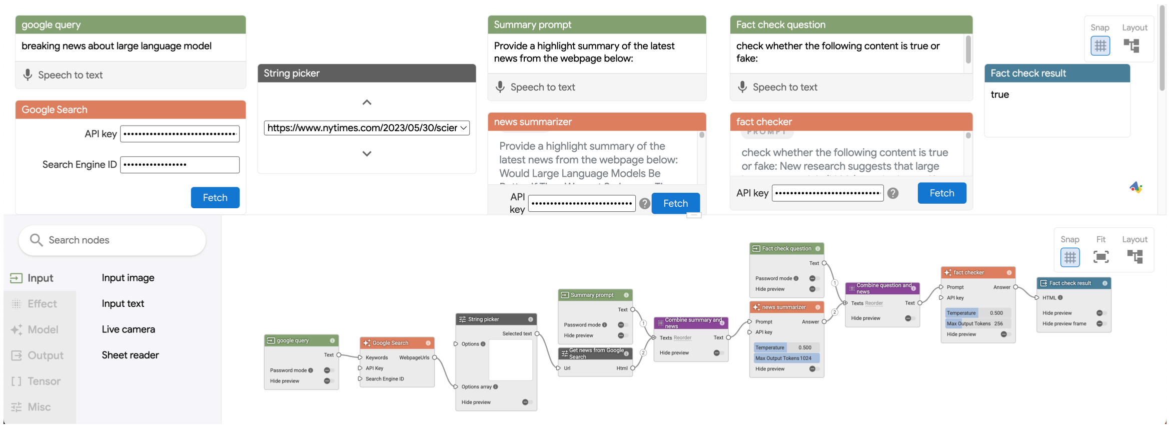
To conduct the technical evaluation, we first organized a two-day hybrid workshop with 23 participants, aiming to collect pipelines that Visual Blocks users build for their creative usage. Six attendees claimed that they had prior experience in using Visual Blocks. At the beginning of the workshop, we gave a 15-minute tutorial walking the participants through the nodes and the pipeline-building process using Visual Blocks. After the tutorial, attendees created pipelines independently. We required the participants to export the JSON file from Visual Blocks and upload it to our Google Drive every time they finished a pipeline. We also emphasized that they should caption their pipelines when they export the files. Such data pairs (pipeline/caption) by the participants constitute important raw data we use for the technical evaluation.

Note that Visual Blocks includes more primitive nodes than the 27 nodes covered by InstructPipe. The workshop is an open-ended creation process, in which participants are free to any node available in Visual Blocks. The InstructPipe feature was not available in the workshop.

4.5.2 Data Post-Processing

After the workshops, one author carefully examined each collected pipeline and found several critical issues in the raw data:

- **Incomplete pipelines**. There exist pipelines uploaded by the participants that were incomplete.
- **Isolated graphs**. There exist pipelines that include at least one isolated subgraph. The isolated subgraph, as opposed to the main graph, is defined as a graph (or a node) that has no connection to the main graph in the pipeline that provides the main functionality of the pipeline. The "Image viewer" node on the bottom-left corner of Figure 4.8b We observed that some participants typically would like to



google query: breaking news about large language model

https://www.nytimes.com/2023/05/30/scier

Summary prompt: Provide a highlight summary of the latest news from the webpage below:

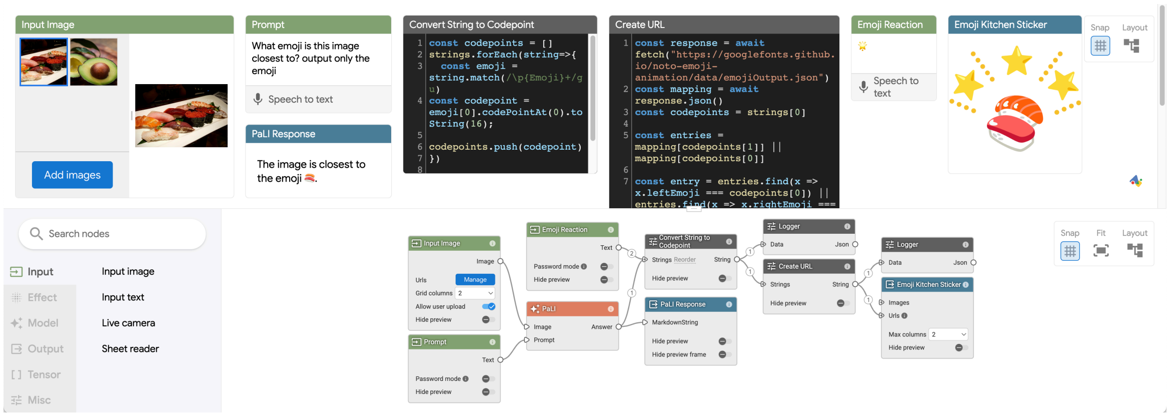
news summarizer: Provide a highlight summary of the latest news from the webpage below: Would Large Language Models Be

Fact check question: check whether the following content is true or fake:

fact checker: check whether the following content is true or fake: New research suggests that large

Fact check result: true

(a) Search news from Google, summarize it and then conduct fact check. Input: a keyword for Google Search; Output: a summarization of the news and a fact-check result.



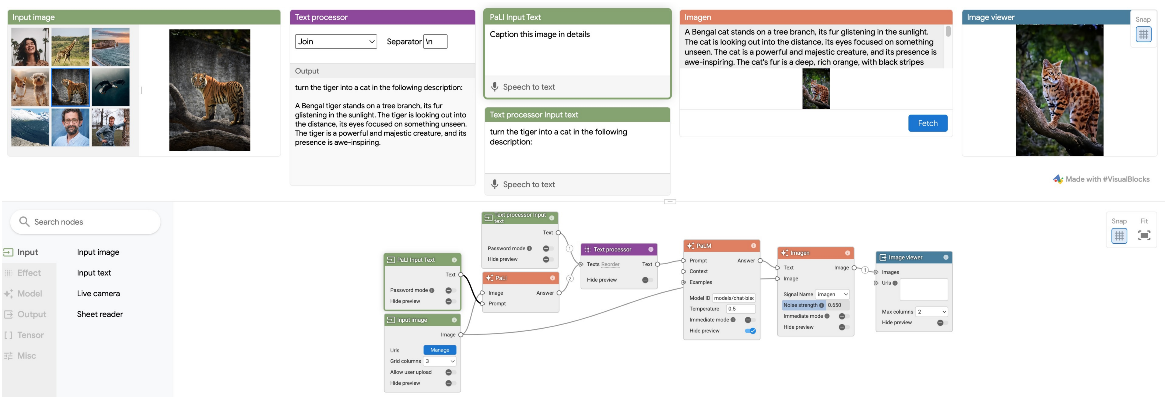
Prompt: What emoji is this image closest to? output only the emoji

PaLI Response: The image is closest to the emoji 🍣.

```
const codepoints = []
strings.forEach(string=>{
  const emoji =
string.match(/\p{Emoji}+/g
u)
const codepoint =
emoji[0].codePointAt(0).to
String(16);

codepoints.push(codepoint)
})
```

```
const response = await
fetch("https://googlefonts.github.
io/noto-emoji-
animation/data/emojiOutput.json")
const mapping = await
response.json()
const codepoints = strings[0]

const entries =
mapping[codepoints[1]] ||
mapping[codepoints[0]]

const entry = entries.find(x =>
x.leftEmoji === codepoints[0]) ||
entries.find(x => x.rightEmoji ===
```

(b) Generating an emoji from a photo. Input: a photo uploaded by the user; Output: an emoji generated from the photo.



Text processor: Join, Separator \n

Output: turn the tiger into a cat in the following description:

A Bengal tiger stands on a tree branch, its fur glistening in the sunlight. The tiger is looking out into the distance, its eyes focused on something unseen. The tiger is a powerful and majestic creature, and its presence is awe-inspiring.

PaLI Input Text: Caption this image in details

Text processor Input Text: turn the tiger into a cat in the following description:

Imagen: A Bengal cat stands on a tree branch, its fur glistening in the sunlight. The cat is looking out into the distance, its eyes focused on something unseen. The cat is a powerful and majestic creature, and its presence is awe-inspiring. The cat's fur is a deep, rich orange, with black stripes

(c) Turning a tiger into a cat. Input: an image of a tiger; Output: an image of a cat.

Figure 4.9 Example pipelines participants built in the workshops. The input and out of the pipelines are shown in the sub-captions.

explore the system by working on a separate sub-space. While we acknowledge its usefulness, leaving such “redundant” graphs in the raw data for the evaluation would cause issues when we calculate the number of user interactions (*i.e.*, the metric used in the evaluation that will be defined in the next subsection).

- **Low-quality captions**. While we explicitly required the participants to write descriptive captions, we found some captions written by the participants were either empty or low-quality (*e.g.*, *“newsletter”*, *“image editing”* and *“[participant name]-demo”*).

The observation motivated us to post-process the raw data to present more rigorous evaluation results. We first removed incomplete pipelines and the isolated graphs in each pipeline (if there are any). Although the low-quality captions may reflect real-world scenarios on how the users may use our system, such a hypothesis cannot be officially validated. In our data collection process, participants are situated in the scenarios of writing captions instead of instructing an AI assistant. When users generate instruct pipelines, they may be aware of the importance of writing good instructions, and thus, such low-quality captions would not be as usual. More importantly, the technical evaluation focuses on considering the variation of the pipeline, instead of text instructions. In the next user evaluation (§4.6), we will explicitly situate a new group of participants in instructing the assistant and observe the user behaviors and the variation of human instructions on the same pipelines. Therefore, we were motivated to enhance the quality of the captions for fair technical evaluations. Two authors individually annotated the caption of each pipeline separately by referring to the original captions and pipelines authored by the participants. It is important to note that we finished the workshop and the data annotation task *before* we completed the system implementation. The two authors had no experience using InstructPipe before completing the annotation. We believed this process could effectively enhance the quality of the captions while maintain fairness of the technical evaluation.

As we clarified in §4.5.1, the workshop is designed to be an open-ended creation process. This indicates that the dataset inevitably includes out-of-scope nodes like "custom scripts" (in which the participants write code to process the input data and return custom outputs; see Figure 4.9b for an example) and "TFLite model runner" (which call a custom TensorFlow model with a URL input of the model in the TF-Hub). As we explained in §4.4.1, InstructPipe focuses on generating the graphic structure of a pipeline without considering the property value within each node. Different from the majority of nodes in Visual Blocks, nodes like "custom scripts" and "TFLite model runner" derive their intrinsic functionality from the property values. Without this information, the entire pipeline's rationale is obscured. Therefore, we removed pipelines that contained "out-of-scope" nodes in our technical evaluation.

The final 48 pipelines (out of 64 pipelines) are comprised of 23 language pipelines, seven visual pipelines, and 18 multi-modal pipelines. Figure 4.9 shows three pipelines created by the participants. Figure 4.9b is an example of the pipelines that include out-of-scope nodes, and therefore is not included in the final 48 pipelines. In the technical evaluation, we ran our generation algorithm on the pipeline captions six times (three times for each caption $\times$ two captions from two authors for each pipelines) and evaluated the generation results using the metric that will be introduced below.

4.5.3 Metric: The Number of User Interactions

Evaluating a generated pipeline is more complex than other tasks that have a universally recognized definition of accuracy. In this project, intuitively, an accurate generation implies that users only need to do very few edits (or even no edits) based on the generated result. This intuition inspires our definition of "the number of user interactions" as the core metric used in our evaluations:

*The Number of User Interactions is defined as the **minimal** number of user interactions needed to **complete** the pipeline from a generated pipeline.*

Note that there are countless ways to modify a generated pipeline toward a complete pipeline in practice. Nevertheless, the **minimal** number of user interactions, representing the "smartest" way(s) to make a modification, is deterministic, and this is an objective metric that can fairly reveal how many minimal efforts users need to spend to achieve their goal. A pipeline is considered **complete** when it satisfies the given instruction. We calculate the number of interactions from the sum of two events: 1) adding or deleting a node and 2) adding or deleting an edge between nodes. Note that when a node with edges connected is deleted, our system will auto-remove these edges. In such instances, we only register one interaction for the node deletion.

Our definition of the number of user interactions has two important implications. First, a complete pipeline after user interaction does not need to be the same as the corresponding pipeline in the dataset. As long as it fulfills the task described in the caption, we consider the pipeline complete. Second, our definition does not consider interactions of modifying property values of a node, *e.g.*, typing in a text box or selecting a value in a drop-down box. We argue that such interactions are highly node-dependent and are hard to quantify objectively. More importantly, as we explain in §4.4.1, the generation of property values is out of the scope of this work.

In the technical evaluation with various pipelines, it is unfair to report an averaged *absolute* value of user interactions because the complexity of the pipelines varies dramatically. For instance, the user may need three edits based on a generated result to complete a large pipeline that requires 20 edits from scratch. In another pipeline, the user also needs to do three edits starting from the generated result, but the whole pipeline only takes three edits to finish. Averaging these *absolute* values does not provide reasonable insights into how accurate the generation is. Therefore, we reported an averaged *ratio* of user interactions required to complete a pipeline "from

Table 4.1 The ratio of human interactions in the technical evaluation. Results are reported as mean ± standard deviation.

Overall	Language	Visual	Multimodal
$\mathbf{18.9 \pm 20.3\%}$	$17.4 \pm 20.6\%$	$17.6 \pm 23.7\%$	$20.8 \pm 16.0\%$

our generated pipeline" to that "from scratch" as our target metric in the technical evaluation.

4.5.4 Results

Table 4.1 summarizes the results of the technical evaluation. Compared to building a pipeline from scratch, InstructPipe allows the user to finish a pipeline with **18.9%** of the user interactions (as defined in §4.5.3), demonstrating the effectiveness of the InstructPipe support. **Seven** generated pipelines directly satisfied with instructions without user interactions in all six trials, and **38** generated pipelines completed at least once in any of the six trials.

4.6 User Evaluation

While the technical evaluation demonstrates the accuracy of InstructPipe among various real pipelines created by participants, it is still unclear what is the actual user experience when real users go through the entire system workflow. Additionally, there is a lack of subject variation on a controlled group of pipelines in the technical evaluation. Therefore, we conducted an in-person user study of InstructPipe with another group of participants, aiming to provide more insights into our system performance. The study recruitment was in accordance with the ethics board of our institution. We obtained participant consent before the study began.

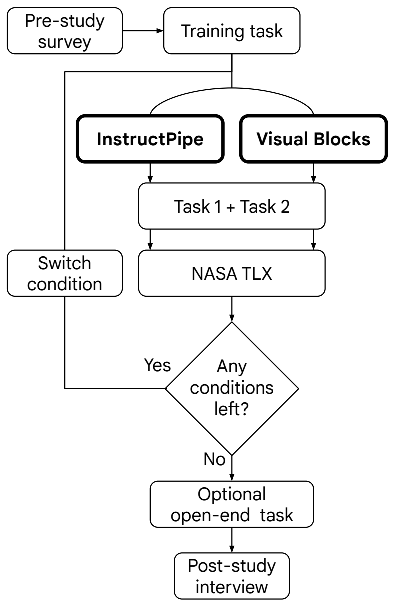


Figure 4.10 A flow diagram of the user study. After a training session, participants completed the two tasks in each condition in the sequence determined by the counterbalancing protocol.

4.6.1 Study Design

In the user evaluation, we aimed to investigate how the interface condition (with InstructPipe and without InstructPipe; the independent variable) affects the user experience and behaviors (the dependent variable). We will refer to these two interface conditions as "InstructPipe" and "Visual Blocks" in the following content. Figure 4.10 visualizes the complete study flow. In each condition, participants completed the two pipelines with counterbalance (referred to as Task 1 and Task 2 in Figure 4.10).

We carefully designed this study to ensure a fair study that can be completed with reasonable efforts. The following elaborates how we made two important decisions that affected the rigorousness of our study:

- **Two controlled pipelines**. Our user evaluation focuses on two controlled pipelines. While we acknowledge more pipelines (*e.g.*, four, six, or even more, instead of two in our design) can enhance the solidness of the study, such design also dramatically increases the required groups of the user study session. For example, counterbalancing four controlled pipelines with two interface conditions results in

$4! \times 2 = 48$ study orders. If there are four participants for each unique order, the whole study requires $48 \times 4 = 192$ participants, which is far more than a standard number in HCI system papers (*i.e.*, recruiting 10 - 20 participants for evaluation). We believe two pipelines fit the best in our case since they require $2! \times 2 \times 4 = 16$ participants.

- **Pipeline selection**. With the limited number of pipelines we can choose in the study, it becomes highly important how we select the two pipelines. There are two critical factors we considered in the pipeline selection process: representativeness and diversity. The representativeness implies that the selected pipelines should represent the averaged performance of InstructPipe. The diversity further suggests that the selected pipelines should provide various experiences to simulate the actual use scenarios in which InstructPipe may work well in some pipelines but not always. By following this guideline, we first selected four candidates, and the final decision was made after a pilot study with one participant to test the level of pipeline difficulty. The two resultant pipelines are composed of eight nodes with seven edges and six nodes with six edges, respectively. Using the instructions from two authors, the averaged ratio of human interactions in these two pipelines are 27.8% and 5.2%, respectively. See section B.3 for more details of the pipelines.

4.6.2 Procedure

The study starts with a 10-15 minute hands-on training of both conditions. The training included 1) all the Visual Blocks interactions needed to complete the subsequent steps of the experiment and 2) all the nodes that participants need to use for creating the pipelines they are assigned in the main sessions. Participants were also encouraged to try building a pipeline independently and to ask questions.

After the training, participants progressed to an unmoderated session where they were asked to build pipelines under the given conditions. We verbally described the pipelines to participants as below, and participants are not allowed to read our scripts:

- **Text-based pipeline**: get the latest news about New York using Google Search and compile a high-level summary of one of the results.
- **Real-time multimodal pipeline**: create a virtual sunglasses try-on experience using your web camera.

During the task, participants were allowed to consult with us for technical help. If participants were unable to make progress, we provided hints. We provided such support to ensure that each participant spent a reasonable amount of time on each task and had sufficient time for the following tasks. As shown in the results (§4.6.5), we provided far more support in the "Visual Blocks" condition. Therefore, we argued that this is a fair moderation in the study because it favors our comparative conditions (*i.e.*, it makes it easier for users to finish the pipeline under the "Visual Blocks" condition). As an optional extension to the study, eight participants were offered an open-ended pipeline creation, where participants prototype their own ideas with InstructPipe. This optional section was offered based on the progress of the participant in the previous sections and time constraints so that the study duration was controlled within the time we guaranteed in our recruitment process.

After trying all pipeline-condition combinations, participants answered open-ended questions in a semi-structured interview. Participants provided their general outlook of each condition, listed pros and cons, identified potential future use cases, and critiqued the user interface for future improvements.

In total, participants spent 55-65 minutes in the study.

4.6.3 Participants

We recruited 16 participants from an internal participant pool at Google. A diverse variety of job profiles were represented, but no software engineers were included. See Table (Table 4.2) for a full breakdown. Note that we intentionally recruited novice users, as we envision them as intended users of InstructPipe. The criteria for selecting the novice users was based on self-evaluation ratings of the below prompts:

- Please provide a self-evaluation of your programming experience
- Please provide a self-evaluation of your machine learning (ML) skillset

Table 4.2 Participant demographics for the user study, showing various demographic characteristics and skills relevant to InstructPipe.

ID	Job Title	Self-identified Gender	Age Group	Programming Experience	Machine Learninig Skill	LLM Usage
P1	Product Manager	Woman	25 - 34	Beginner	Beginner	At least once a month
P2	Image Tuning Engineer	Man	35 - 44	Intermediate	Beginner	At least once a week
P3	Program Manager	Woman	45 - 54	No experience	No experience	At least once a week
P4	Hardware Engineer	Man	35 - 44	Intermediate	No experience	At least once a month
P5	Technical Program Manager	Man	35 - 44	Beginner	No experience	At least once a day
P6	Senior Hardware Engineer	Man	35 - 44	Beginner	No experience	At least once a month
P7	Technical Program Manager	Woman	18 - 24	Beginner	Beginner	Never used it
P8	Technical program manager	Man	25 - 34	No experience	No experience	Multiple hours every day
P9	Solutions Engineer	Man	25 - 34	Beginner	No experience	At least once a month
P10	Program Manager	Man	55 - 64	Beginner	Beginner	At least once a month
P11	Program Manager	Woman	35 - 44	No experience	No experience	Never used it
P12	Lab Manager	Man	35 - 44	Intermediate	Beginner	At least once a week
P13	Partner Development Manager	Man	25 - 34	Beginner	Beginner	At least once a week
P14	Hardware Engineer	Man	25 - 34	Beginner	Beginner	At least once a week
P15	Global Supply Manager	Man	25 - 34	Beginner	No experience	At least once a month
P16	Global Supply Manager	Woman	55 - 64	No experience	No experience	At least once a week

4.6.4 Metrics

In addition to the qualitative data from the interview, we measured the following quantitative data.

Task Completion Time

Back-end logs were used to collect timestamps for starting and ending events. Then, completion times for each condition were calculated per task for each participant.

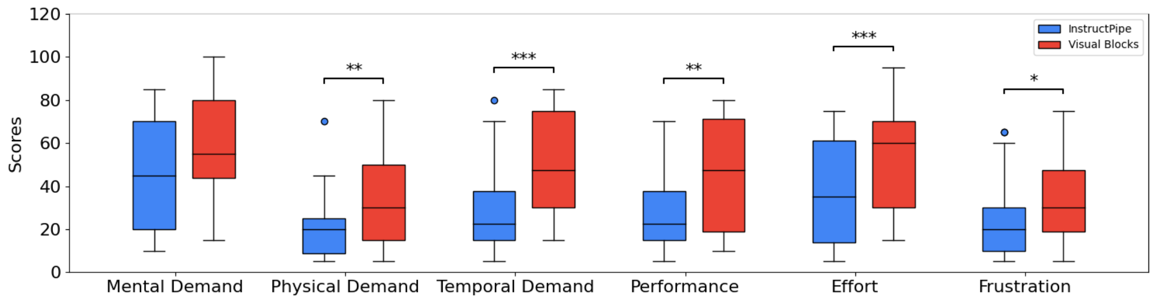


Figure 4.11 Raw-TLX results. The statistic significance is annotated with *, **, or *** (representing p<.05, p<.01, and p<.001, respectively).

Table 4.3 Task completion time and the number of human interactions in the user study (N=16).

System	**Time** (secs)			**# Interactions**		
	Median	IQR	p	Median	IQR	p
InstructPipe	**203.5**	**156.25**	p <.001	**5.0**	**4.25**	p <.001
Visual Blocks	304.5	124.25		16.0	6.0	

The Number of User Interactions

We used the number of user interactions (introduced in §4.5.3) to measure the user's objective workload. Unlike the results in §4.5.4, we report an absolute value here because all the pipelines are controlled in the system evaluation.

Perceived Workload

The raw task load index (Raw-TLX) questionnaire was used to measure participant's perceived workload [63]. This questionnaire was a subset of the NASA-TLX (part I). Participants filled out the questionnaire after each condition (InstructPipe or Visual Blocks).

4.6.5 Results

InstructPipe Reduces Users' Workload

Table 4.3 shows the results of two objective metrics measured in the study. The Wilcoxon signed ranks test found significant differences on both scales ($p < .001$).

Figure 4.11 further visualizes the results of users' perceived workload in six subscales. The Wilcoxon signed ranks test revealed significant differences on five subscales, all but "Mental Demand" (see §4.7.2 for more explanations and discussion). Furthermore, the test indicates that all participants unanimously vote InstructPipe provides lower or equal workload on the subscales of "Physical Demand", "Temporal Demand", "Performance" and "Effort" ($W = 0$). These quantitative results, with both objective and subjective metrics, demonstrate that InstructPipe can effectively reduce users' workload during the pipeline creation process.

Users' qualitative feedback is also aligned with our quantitative results. Participants complimented that InstructPipe is *"helpful"*[P16] and *"obviously easier (to use) than [Visual Blocks]"*[P1]. P11 and P6 further elaborated how InstructPipe enhances the user experience when the user builds a visual programming pipeline:

> *"I feel like I can talk in natural language, and it (InstructPipe) can write the code for me."* [P11]

On-boarding Support of Visual Programming

P1, P5, and P9 explicitly mentioned that there is a *"learning curve"* in the visual programming system for beginners, which validates our statements in §4.1 that motivates this project.

> *"There is a learning curve to it (using the visual programming system) for sure, because you have to, like, read each node carefully"* [P1]

P1's comment matches with our observation of participants' behaviors during the study. In the Visual Blocks condition, we observed that people were more easily stuck in their creative purposes, which required our support[4]. Typical supports include 1)

[4]In the whole study, we encouraged the users to talk to us while keeping their focus on their task. While we offer explicit hints only when we consider a participant is stuck in an issue, it is still difficult to strictly define what conversation is " providing hints" and count them quantitatively. Therefore, we cannot fairly report quantitative data for "the number of hints" in both conditions. Here, we faithfully report this finding based on our observation and experience.

guiding participants if we notice they go too far away from the correct pipeline and 2) reminding them of an important node for the pipeline, although we introduced all the necessary nodes in our training session.

To this end, participants commented that InstructPipe is a good onboarding tool in visual programming systems, especially for non-experts, to get familiarized with the system by having a ready solution.

> *"[InstructPipe] lets you know these nodes exist [when the pipeline appears after the instruction]. It' s like a super speedy tutorial."* [P7]

> *"If you don' t have experience in visual programming, you will appreciate [InstructPipe] much more ... With [InstructPipe], the structure is there, and I feel less worried about making mistakes. It' s, like, giving you examples. It' s easier than starting from scratch."* [P5]

Anecdotally, three participants asked for InstructPipe during the Visual Blocks condition.

Integration into The Existing Workflow

InstructPipe is a feature available in Visual Blocks. In the interviews, participants particularly expressed their strong appreciation of this design as an AI assistant that enhances, instead of completely replacing, the existing user workflow (*i.e.*, the creative process purely using the node-graph editor):

> *"[The pipeline generated by [InstructPipe] could be pretty close to what I want ... Or maybe sometimes not, but that's okay. I got most of the blocks there, and then it's up to me to figure out how to connect them."* [P6]

While most participants shared similar opinions with P6, P15 raised one concern about such integration. The concern lies in the mental experience of the state that transits from the prompting task to the node-graph editing process:

> *"Rather than fixing it (the generated pipeline) on my own [on the node-graph editor], I would have gone back and changed my prompt ... I'm pretty sure I could have gotten it closer ... because I just spent so much time figuring out what the prompt should be. That's kind of like* ***already where my brain was*** *and I knew that something was wrong there (the prompt), but I would* ***have to switch over to the other mode*** *(visual programming) of figuring out what was wrong on the pipeline."* [P15]

Use Scenarios: Accessible ML Prototyping and Education

In the open-ended session, we observed participants could efficiently utilize InstructPipe to prototype a pipeline for various daily life or business purposes. For example, P14 tried InstructPipe with *"summarize real estate price increase in San Diego California over 2023"*. Compared to using LLM chatbots, InstructPipe helps the user quickly build a more explainable pipeline in which the user can track (or modify) the information resources. P4 prototyped an interactive VQA app by "*Describe the product in the camera*". P13 further shared his thoughts on how this rapid and accessible prototyping experience can support future business:

> *"It (InstructPipe)'s going to facilitate the prototype building for PMs (Product Managers) ... I was a PM ... Back then ... My biggest fear is to code ... I have lots of ideas, but my challenge is how to translate an idea into the technical world and see a prototype. I think that this app expedites me in that process a lot."* [P13]

Another emerging theme was regarding educating kids on programming as explained in the following:

> *"With [InstructPipe], I don't need to teach them (kids) to code for them to build something ... Some kids like to code, some kids like to create stuff but don't*

want to be bored with learning the syntax of coding ... Using [InstructPipe], I can see kids can build, like, customized chat-bots or interactive Wikipedia." [P13]

Limitations and Future Directions

Across the study sessions, we consistently observed a specific user behavior pattern: participants typically paused their pace when a generated pipeline appeared in the workspace. At these times, some participants used soliloquy, as in saying "Let me see", while others kept a focused stare on the workspace. These human behaviors strongly indicate that the participants were engaging in deep, contemplative thought.

This observation suggests that participants typically need to perceive a pipeline that appears in the workspace. Such a sense-making process brings new challenges to their creative process:

"[Using InstructPipe] is a little mentally demanding ... I have to debug ... If it doesn't help (generating an almost 100% correct pipeline), I have to go through all the nodes ... I don't like debugging." [P13]

Additionally, we observed that several participants spent more time crafting their prompts than the others. P15 is the participant who spent the most time writing the prompt. The following comments explain the reason, and how the prompting process causes extra mental workload to him/her/them.

"I'm a relatively visual thinker ... Getting the prompt right requires me to think in a way that is a lot more like precise and like getting it figured out without working it out live ... [When writing prompts,] you' re just putting them (every detail in a whole pipeline) all out [in one short prompt]" [P15]

In addition to the lack of the original visual thinking experience in visual programming, P13 also warned that such simplification of the creative process into prompting experience may sacrifice users' hands-on experiences:

"I'm very hands-on with techs. I would like to understand what's going on [rather than prompting LLMs to generate everything for me]. I want to like think for myself and then compile all the information myself." [P13]

4.7 Discussion and Future Directions

4.7.1 Human-AI Collaboration in Prototyping Open-ended ML Pipelines

Our technical evaluation (§4.5.4) shows that InstructPipe reduces the number of user interactions to 18.9 % (±20.3%). There are two implications from the results.

- InstructPipe automates *most* pipeline components with a single prompt.
- InstructPipe is *not* able to automate *all* the pipeline creation processes.

Such takeaways *differ from existing findings* that show LLMs can achieve full automation of ML inference [151, 60]. The main reason is that existing work built their ad hoc solutions for target use scenarios, respectively. In contrast, InstructPipe covers a wide range of mainstream ML models (§4.3.2) and aims for an open-ended use case. Our results show that LLMs (*i.e.*, GPT-3.5-turbo) still fail to write robust code with prompt engineering techniques. As we mentioned in §4.4.4, the main attribute of such failure is that LLMs tend to invent nodes (*i.e.*, the node type in the pseudocode) that do not exist in our node library, which causes execution error. Note that we explicitly instructed the LLM that *"introducing new nodes is not permitted"* in the system prompt for the Code Writer module (Figure 4.7). We also observed similar issues when we combined all the official prompts of four scenarios in VisProg [60].

While the execution error in existing work indicates the failure of the full solution, InstructPipe renders the executable components of the generation pipeline in a node-graph editor. This facilitates a collaboration protocol between humans and AI in which humans do not need to start a project from scratch. Results in §4.6.5 revealed that a wide range of participants appreciated this design of human-AI collaboration. We believe the main reason behind this observation is that most non-expert users prefer working on a task from a template (although sometimes it is not perfect) rather than starting the whole project from scratch. The visual programming system provides non-experts with a novice-friendly interface to work with AI in a visual platform. In this manner, *"users can do what matters most – building software by letting AI do the redundant work"* [175].

4.7.2 Three Attributes to Mental Workload

While most participants are positive about our integration of InstructPipe into the existing user workflow of Visual Blocks, results from the user evaluation also reveal several challenges in such human-AI collaboration. Results in §4.6.5 show that InstructPipe failed to significantly reduce novice users' mental demand. We summarized its major causes into three aspects.

- **Instruction**. P15's comment in §4.6.5 summarizes the first aspect that causes mental burden. Although the "instruction-to-pipeline" process is fast and seems effortless, the process of framing a prompt is one factor that may overwhelm users, especially those who are more accustomed to visual thinking. In essence, an instructor of an LLM system (the user of InstructPipe) should be clear about the problem they want to solve and preferably what pipeline they want, which causes a mental burden to the user. Zamfirescu-Pereira et al.'s study shows that non-experts may not prompt LLMs well [174], which further explains why non-expert participants might find the instruction process mentally demanding.
- **Perception**. The integration of LLM supports into the visual programming interface enables a "multimodal programming" experience [43], in which, users

can program through both verbal and visual approaches. Despite the enhanced flexibility of the system, it also causes additional perceptual burden because users need to *switch their "brain mode" between "visual thinking" and "text-based thinking"*. This finding is aligned with Dual Coding Theory (DCT) [116], in which Paivio hypothesized that human brains process information using two different channels: verbal and visual. Interestingly, our results seemingly contradict with multiple psychological findings based on DCT that show a combination of verbal and visual information actually helps humans' memory process [115, 117]. For example, people feel it easier to remember a new word if they learn the word using a vocabulary card with a figure with texts. Based on these existing theories on humans' mind processes, we make the following explanation on the finding in our project: the cause of mental workload does not imply that people dislike a multimodal workspace, but that *users require a transparent interface that connects humans' mental model to the AI reasoning process, both verbally and visually.* When users frame the instruction in the InstructPipe dialogue, there is a lack of feedback system from the AI assistant that visualizes how AI interprets the instructions in real time. As a result, humans need to either first estimate the AI prediction or make no expectations when they perform instruction. Even for experts, it is hard to accurately predict AI generation, and thus, the generated pipeline is usually unexpected. When users frame the instructions, the current interface design is purely based on texts, and the visual channel in the brain is off. When the generation is complete, the user suddenly needs to switch on a sleeping (visual) channel, which causes a mental workload. One possible approach to addressing this issue is to expose users to the multimodal information when they perform instructions. For example, future work can investigate methods to visualize a generated pipeline on-the-fly when users are framing and typing their instructions. In this manner, users' mental models are continuously synchronized with the AI assistant before the final generation, which should eliminate users' cognitive burdens when the final generation result appears.

- **Debugging**. When a rendered pipeline does not match with users' expectations, users then are required to debug the generated results (see P13's comment §4.6.5). In essence, debugging is one professional programming skill, which explains why it mentally overwhelms beginner-level users. InstructPipe made the first step to facilitate users to 1) perceive the results and 2) take actions on the results, *i.e.*, two important elements in a debugging process, on the node-graph editor. Results in our study reveal that this first step is insufficient to support users so that users still feel like they are debugging the codes as professional programmers. Future work should investigate how to further support users' debugging process with visual programming interfaces. For example, when execution fails, future work can consider visualizing messages that provide cues on the possible issues and provide actionable guidance on the interface for users to fix the issues easily.

On the other hand, it is important to note that we distill the attributes mainly from a subset of participants who explicitly mentioned that they realized an increase in mental workload when using InstructPipe. Those participants typically spent more time writing prompts and working on the node-graph editor than the average performance (*e.g.*, P15 spent the most time ideating and writing the instructions). Therefore, we argue that the aforementioned issues do not represent the average user experience, and the issues are distilled from explicit comments on their negative experience. In fact, we observed that most participants were relatively decisive and could quickly write their instructions and modify a generated result in the node-graph editor. This observation aligns with our qualitative results in Figure 4.11: the median score of "Mental Demand" in the InstructPipe condition is lower than the baseline condition.

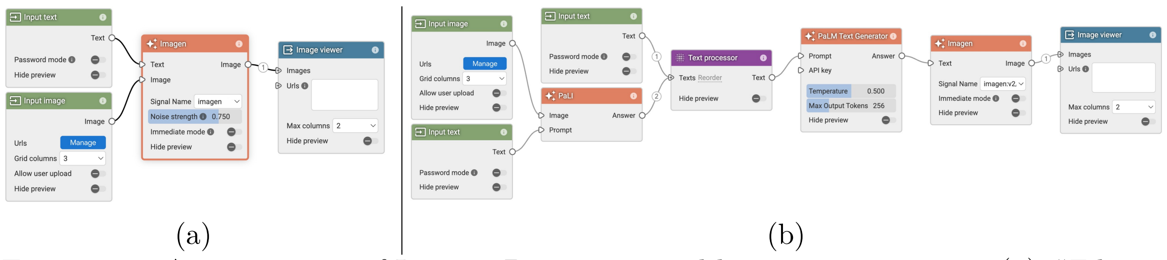


Figure 4.12 A comparison of InstructPipe generated by two instructions: (a) *"Edit an image by updating the image caption"*; (b) *"Caption a tiger image using VQA, modify the character in the caption into a cat using LLM, and finally generate a cat image based on the updated caption"*. See Figure 4.9c for the complete pipeline.

4.7.3 Instructing LLMs Poses Challenges for Both Novices and, Potentially, Experts

As we explained above, one reason that caused mental workload is that non-experts found it challenging to instruct LLM. More interestingly, we found that *even we, the inventors of InstructPipe, failed to write optimal instructions.* As mentioned in §4.5.2, two authors annotated captions for the pipelines offline, and we observed multiple imperfections, especially for the complex ones. For instance, the two captions of Figure 4.9c are *"Describe the image and turn it into a cat image"* and *"Edit an image by updating the image caption"*. Neither caption explicitly describes the detailed pipeline flow clearly, and therefore, all the six evaluation trials (§4.5) were incomplete (see Figure 4.12a for one example). The average ratio of user interactions is 45.8%, more than twice the average value for our multimodal pipelines (20.8%). To further understand the cause of the failure, another author improved the instruction into *"Caption a tiger image using VQA, modify the character in the caption into a cat using LLM, and finally generate a cat image based on the updated caption"*. The resulting pipeline is significantly improved but still not perfect (Figure 4.12b). The user only needs to turn "Imagen" into another mode so that it also accepts the input "image" node. Revisiting the improved instruction, we instructed InstructPipe with *"generate a cat image based on THE updated caption"*, which actually missed the input image.

The important takeaway is while natural languages are proven to be one promising communication media that connects humans and AI systems [26, 162], *instructions may not be the best format to facilitate such connection.* We believe the reason is that instructions are still not intuitive to humans: AI typically requires flawless and unambiguous instructions while humans tend to express their intention using ambiguous natural languages in conversations. We encourage future work to investigate alternative interaction mediums beyond instructions to further enhance user experience in human-AI collaboration.

4.7.4 Online InstructPipe

In this project, we made the following assumption: *visual programming and its LLM support are both offline.* Note that InstructPipe does have connection to the Internet, and the "offline" here implies that the system does not dynamically update its node library with online resources: every node is pre-defined by the system developers. This statement shows one major limitation of InstructPipe we observed in the open-ended session of our user evaluation: the 27 nodes covered by InstructPipe are insufficient to facilitate all the creative instructions from participants. For example, P9 prompted InstructPipe using *"giving me [an] image person walking [a] dog with webcam of myself"*. After some clarification, we found that P9 wanted to generate a video in which P9 is walking a dog by taking the webcam as the input of P10's visual appearance. Such a pipeline is infeasible in our system because it requires a text-driven video generation model. While an extension of InstructPipe with such a text-to-video model as a primitive node can solve this specific issue, we argue that, in practice, users may request many more nodes for their customized uses. Therefore, it is rather important for researchers to investigate a generalizable solution for issues of this category.

In this project, InstructPipe enables users to prototype an ML pipeline with human instruction, and we focus our exploration on a fixed node library. An extension of

InstructPipe to a system that covers a wide range of possible *online* functions is still under-explored. With a next generation of InstructPipe that can find ML models, define nodes and implement the node dynamically with human instruction, we believe the aforementioned issues (*e.g.*, the one raised by P10) can be effectively addressed. In the past several years, the community has already established ML libraries and API services for various models (*e.g.*, by Hugging Face [72]). Such ecosystems provide crucial resources for AI-oriented visual programming systems to dynamically define their primitive nodes, and thus provide more powerful support for users. We encourage future work to investigate how to build a model selector (similar to the Node Selector in Figure 4.4) that can intelligently select the correct online API to be called in a node. We believe such "online InstructPipe" would provide an unprecedented user experience in which researchers can brainstorm with the system, and the system will automatically return a pipeline with state-of-the-art ML models to accelerate researchers' creative workflow.

4.8 Limitations

As we discussed in the main content, we notice multiple limitations of InstructPipe. In our current implementation, InstructPipe focuses on generating the graph structure in the pipeline (§4.4.1), and therefore the system is not able to generate property value of the nodes. In some nodes like "tf-model-runner" that exists in Visual Blocks (but not covered by InstructPipe), such property value plays a critical role that directly defines the functionality of the node. Additionally, when InstructPipe detects an undefined generated node, it directly disposes the line without leveraging such information for other purposes (§4.4.4). In the evaluation, we were not able to conduct a large-scale user study with thousands of user study sessions to understand the performance of InstructPipe with both variations from the pipeline factors and the human factors (§4.5). Our quantitative results show that InstructPipe fail to significantly reduce participants' mental workload (fig. 4.4). Qualitative results further validate this finding, and reveal several issues that causes the mental burden when users use InstructPipe (§4.7.2).

We also observe two major limitations of the project that we did not mention in the main content of the chapter, and encourage future research to conduct further investigation. First, our user study participants only engaged InstructPipe in one hour, and therefore, it is unclear whether users will still frequently use and appreciate InstructPipe supports in a long-term manner. Additionally, all participants are all non-experts, hence we are not able to verify the effectiveness of InstructPipe for users with other levels of expertise. Second, InstructPipe currently cannot detect harmful data or misuse of AI. We believe such feature is crucial especially when future work builds "online InstructPipe"(§4.7.4) which greatly enhances the generalizability of ML pipeline prototyping capability. Future work should study effective methods to eliminate potential harmful uses.

4.9 Summary

In this chapter, we introduced InstructPipe, an AI assistant that facilitates users to build ML visual programming pipelines with instruction. We implemented our InstructPipe by decomposing the task into three modules: a node selection module, a code writer and a code compiler. Results in our technical and system evaluations demonstrate that InstructPipe provided users' satisfactory "on-boarding" experience of visual programming systems and allow them to rapidly prototype an idea. We further discussed the issues we observed concerning LLMs in visual programming, related to both human factors and technical implementations. We hope that InstructPipe will engage a diverse community to easily develop creative machine learning pipelines.

Chapter 5

Discussion

5.1 The Roles of Humans, AI Model and Interactive Systems

This dissertation explores two interaction techniques humans naturally perform during teaching. The evaluations in both projects reveal that a system that leverages humans' natural teaching behaviors can effectively reduce humans' teaching efforts. More importantly, qualitative results from the studies show that the reduced workload can lead to an expansion of new use cases of similar systems, which would not be reasonable if non-experts had to spend much effort.

The intuition of the system design of both LookHere and InstructPipe comes from the exploration of the following question: *what are the roles of humans, AI models, and interactive systems, respectively, in their collaboration?*

5.1.1 Humans

Humans should be provided opportunities to express their teaching intention as naturally as how they teach humans. In this dissertation, LookHere and InstructPipe allow users to teach AI by object demonstrations and natural instructions. I believe the two projects are only the early-stage exploration of these two critical human behaviors. As we discussed in both chapters, we observed multiple limitations in our systems, and future work should conduct an in-depth investigation. In addition to object demonstration and natural instruction, I believe that humans can inherently perform many more interactions that remain underexploited. With more researchers engaging in this research fields, I envision a future when there will be a general interactive system that can interpret most of people's intentions in their daily-life behaviors.

5.1.2 AI Models

The mission of AI models is to adapt to user requirements that humans express in their interactions. The approach for such adaptation strongly depends on the ML community's research findings. The most classical method of training an ML model for a customized purpose requires a large-scale dataset, but, in practice, it is unreasonable to request users for data at scale. To achieve quick and cheap adaptation, there exist two approaches in the community: 1) fine-tuning and 2) prompting. LookHere fine-tunes a model pre-trained on the ImageNet [40] based on the limited number of data collected from the interface. While results show that it can achieve high accuracy, the fine-tuning process still takes time to finish. Additionally, the time would scale up if the backend model becomes larger. InstructPipe, on the other hand, uses prompting protocol in the LLM to achieve simple AI adaptation. Extensive studies have demonstrated the effectiveness of prompting LLMs through text-based instructions, but its performance on vision-based models is still underexplored.

The design of how I adapt the backend models toward users' intention is limited to the state-of-the-arts optimization findings in ML. In the future, I believe there will be more approaches that allows efficient adaption, and future work should investigate how such adaptation methods can effectively incorporate take human factors into consider for system creation.

5.1.3 Interactive Systems

Interactive systems bridge humans' interactive behaviors with the AI adaptation process. One intuition that motivates the system implementation of my projects is that *the system should predict the hidden information implied in the user behaviors.* For example, LookHere contains a real-time vision model that predicts the segmentation mask of a target object that is specified by the user. The system then leverages such implications as pseudo-labels when fine-tuning its backend model. Note that the system may utilize an ML model for achieving such prediction of human implication, but this model differs from the backend AI model of the system as we discussed in §5.1.2.

5.2 Applications

With an increasing number of contributions that lower the technical barriers for a general population to customize AI, I envision a future in which *humans will easily utilize AI as simple tools to support daily-life events, similar to how modern citizens freely utilize the internet.* Figure 5.1 shows the application spaces of AI customization technology in different fields. The space divides the applications into two dimensions: 1) the level of user engagement and 2) the purpose of using AI. The three columns shows three levels of user engagements (direct interaction, indirect interaction and no interaction), and the three rows show three purposes (functional, entertainment and prototyping).

Teachable Reality [108] is an early-step system that incorporates an IMT toolkit into a downstream application. Their finding reveals the importance of IMT toolkits that supports users to *"lower the barrier to creating functional AR prototypes"*. In the future, I believe there will be more systems that explore such benefits of IMT toolkits and build systems that engage more users to support their creative process.

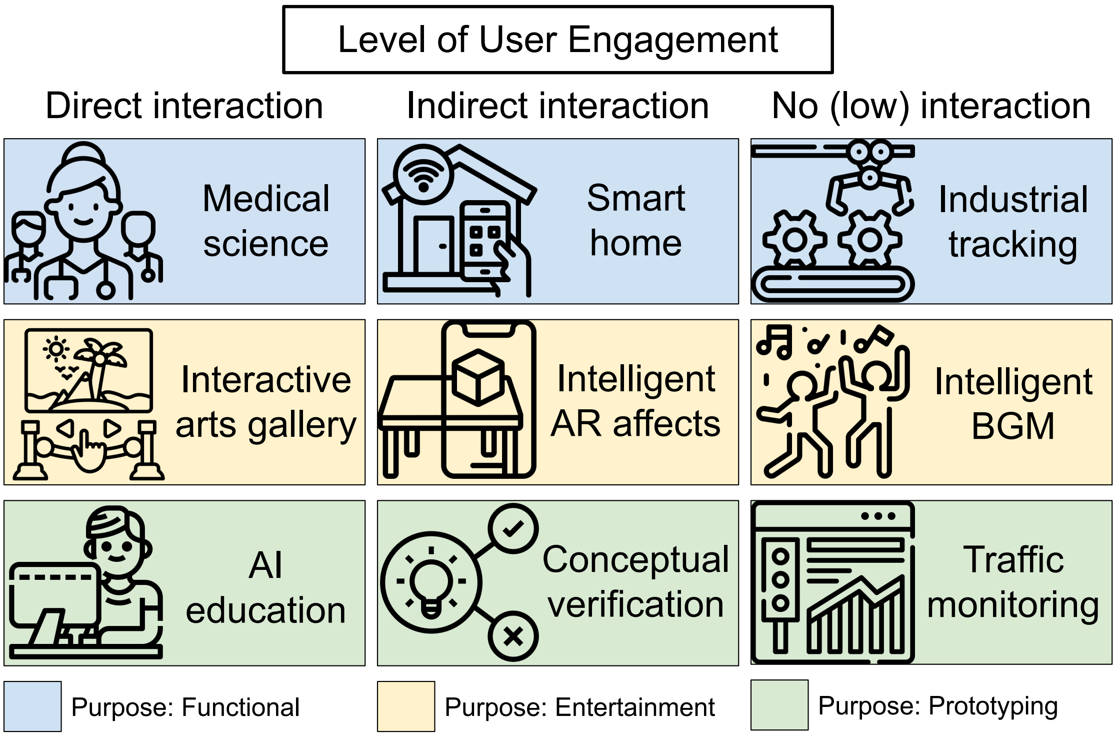


Figure 5.1 The application space of AI customization tools. Three rows (blocks in blue, yellow and green) represent three different *purposes* of using customized AI ("Functional", "Entertainment" and "Prototyping"). Three columns show three *levels of user engagement* of the customization process.

5.3 Future Directions

5.3.1 Joint Instructions and Demonstrations

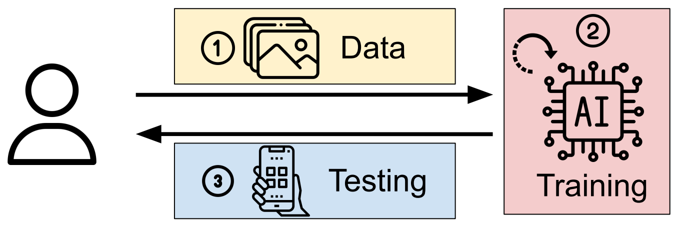
While this dissertation includes two projects that supports users to teach AI by object demonstrations and language instructions separately, it is still underexplored how a system can effectively provide a combined support. Intuitively, people utilize these two interactive techniques in different scenarios, and a system that supports a combined input should provide users more systematical experience for express their intentions. Demonstrations are useful for expressing concepts that are hard to explicitly describe verbally. At the same time, simple demonstrations usually suffer from ambiguity issues if there is no explicit specification on the details of the concepts. Language-based instructions can be one useful communication media to address such issues, in which human teachers can make comments which provide specific guidance on their concepts.

5.3.2 Module-based Customization

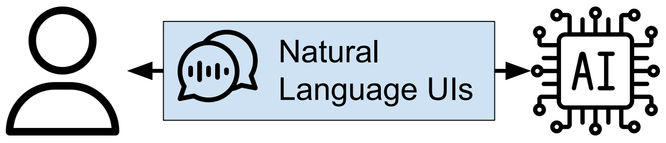
One important difference between InstructPipe and LookHere is that InstructPipe considers the AI model to be customized as a pipeline composed of multiple ML modules. With the increasing intelligence of modern AI applications, such configuration will be prevalent in future system design.

Similar to how recent deep learning architecture slowly converges to several dominant networks (*e.g.*, a stack of multiple Transformers blocks), I envision such phenomenon will also exists in many major ML benchmarks. The breakthrough in classical ML problem benchmarks (*e.g.*, classification, segmentation, VQA) will be not be as significant, and for each problem, there will be a solution that represents the most accurate prediction. The community has established multiple platforms (*e.g.*, Hugging Face) that encapsulate a complex network into one module so that programmers can call each network with one line of code. Such platforms make it possible for future systems to quickly deploy off-the-shelf models into the back-end model. As a result, the remaining task for AI customization becomes how users choose and connect a large number of off-the-shelf models to constitute their preferred personal AI ecosystem.

More importantly, recent advance in foundational models further facilitate this future for two reasons. First, foundational models typical provide prompting protocol for users to customize their usages, and the customization from prompt engineering usually can achieve a decent accuracy. Second, fine-tuning foundational models are much more difficult than other small models because they may not open-sourced and fine-tuning the models require a huge computation resources which may not be reasonable for non-experts.



(a) Conventional IML.



(b) IML with LLMs.

Figure 5.2 A comparison of the conventional interactive machine learning (IML) workflow and the IML with LLMs.

5.3.3 Human-in-the-loop ML with LLMs

For years, researchers have been exploring what are the roles of human beings in the future world with AI. Figure 5.2a shows an example workflow of interactive machine learning (i.e., human-in-the-loop ML):

1. An interactive system collects data from the users.
2. The AI model then is trained on data.
3. The user can test the model after the training finishes.

The major limitation of this workflow lies in two aspects: 1) users need to repetitively provide multiple samples as training data; 2) training a robust model is time-consuming. Both aspects lower the user experience when they interact with AI.

With recent advance in LLMs, I argue that the community is ready to upgrade this workflow towards a more natural human-AI collaboration process. Figure 5.2b visualizes the interaction flow between AI and humans bridged by LLMs. Since LLMs provide support for natural language conversation, the user can simple communicate with the AI on their requirements (similar to the projects proposed in Section 2.1 and

Section 2.2). This implies that both limited in the conventional IML workflows will be effectively eliminated. Firstly, users will no longer be required to provide repetitive training samples. Instead, they can easily prompt the model, and even brainstorm with the LLM support, which constitutes a more natural interaction similar to how humans interact with humans in a creative process. Second, since prompting LLM does not involve a training process, the result will be ready for users to test the model immediately after sending the prompt. In other words, users will no longer need to wait for the training process to receive feedback. Will LLMs revolutionize Interactive Machine Learning (IML) research by *re-defining how and why humans interact with AI*? Future work should conduct in-depth investigation on this research question, and provide a new research philosophical guidance if they give a positive answer.

Chapter 6

Conclusion

While AI demonstrates its capability to solves multiple research challenges, there is still a lack of interactive systems that allows users to intuitively teach AI for creating downstream applications. In this dissertation, I built two interactive systems that empower humans' teaching capability in their AI customization process by leverage their object demonstrations and language instructions, respectively. In the first project, I built LookHere, which leverages humans' deictic gestures that people naturally perform when they demonstrate an object. Lookhere predicts the segmentation mask of the target object by using the hand-object interactions in real time. In the second project, I created InstructPipe, a system that allows users to prototype an AI pipeline with text-based instructions. Compared to the conventional method in which users start building a visual programming pipeline from scratch, our study shows that InstructPipe effectively enhance non-experts' on-boarding experience of the system, and open up various new use cases. Together, this dissertation provides answers on how interactive systems can leverage interactions people naturally perform in the teaching process to support their teaching process. This work is a step toward creating AI that learns from intuitive human interactions, paving the way for more personalized and user-friendly AI applications.

Publications

Peer-reviewed Publication related to this thesis

1. **Zhongyi Zhou** and Koji Yatani. Gesture-aware Interactive Machine Teaching with In-situ Object Annotations. In The 35th Annual ACM Symposium on User Interface Software and Technology (UIST '22). http://dx.doi.org/10.1145/3526113.3545648

2. **Zhongyi Zhou** and Koji Yatani. Enhancing Model Assessment in Vision-based Interactive Machine Teaching through Real-time Saliency Map Visualization. In The Adjunct Publication of the 34th Annual ACM Symposium on User Interface Software and Technology (UIST ' 21 Demo). https://doi.org/10.1145/3474349.3480194

3. **Zhongyi Zhou**. Exploiting and Guiding User Interaction in Interactive Machine Teaching. In The Adjunct Publication of the 35th Annual ACM Symposium on User Interface Software and Technology (UIST ' 22 Doctoral Symposium). https://doi.org/10.1145/3526114.3558529

Peer-reviewed Publications not related to this thesis

Full Papers

- Anran Xu, **Zhongyi Zhou**, Kakeru Miyazaki, Ryo Yoshikawa, Simo Hosio, and Koji Yatani. DIPA2: An Image Dataset with Cross-cultural Privacy Perception Annotations. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. 7, 4, Article 192 (December 2023).

- Zefan Sramek, Arissa J. Sato, **Zhongyi Zhou**, Simo Hosio, and Koji Yatani. SoundTraveller: Exploring Abstraction and Entanglement in Timbre Creation Interfaces for Synthesizers. In Proceedings of the 2023 ACM Designing Interactive Systems Conference (DIS '23).

- Zhihang Zhong, Mingdeng Cao, Xiao Sun, Zhirong Wu, **Zhongyi Zhou**, Yinqiang Zheng, Stephen Lin, and Imari Sato. Bringing Rolling Shutter Images Alive with Dual Reversed Distortion. In Computer Vision - ECCV 2022 - 17th European Conference.

- **Zhongyi Zhou**, Anran Xu and Koji Yatani. SyncUp: Vision-based Practice Support for Synchronized Dancing. Proc. ACM Interact. Mob. Wearable Ubiquitous Technol. (IMWUT) 5, 3, Article 143 (September 2021), 25 pages.

Short Papers:

- Ruofei Du, Na Li, Jing Jin, Michelle Carney, Xiuxiu Yuan, Kristen Wright, Mark Sherwood, Jason Mayes, Lin Chen, Jun Jiang, Jingtao Zhou, **Zhongyi Zhou**, Ping Yu, Adarsh Kowdle, Ram Iyengar, and Alex Olwal. Experiencing Visual Blocks for ML: Visual Prototyping of AI Pipelines. In The Adjunct Publication of the 36th Annual ACM Symposium on User Interface Software and Technology (UIST '23 Demo).

- Anran Xu, **Zhongyi Zhou**, Kakure Miyazaki, Ryo Yoshikawa, Simo Hosio, Koji Yatani. DIPA: An Image Dataset with Cross-cultural Privacy Concern Annotations. In IUI 2023 Open Science track.

- Hirotaka Hayashi, Anran Xu, **Zhongyi Zhou** and Koji Yatani. Vision-based Scene Analysis toward Dangerous Cycling Behavior Detection Using Smartphones. In Adjunct Proceedings of the 2021 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2021 ACM International Symposium on Wearable Computers (UbiComp-ISWC ’21).

Bibliography

[1] [n. d.]. GitHub - Significant-Gravitas/AutoGPT: AutoGPT is the vision of accessible AI for everyone, to use and to build on. Our mission is to provide the tools, so that you can focus on what matters. — github.com. https://github.com/Significant-Gravitas/AutoGPT. [Accessed 05-02-2024].

[2] 2015. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems. https://www.tensorflow.org/ Software available from tensorflow.org.

[3] Andrea Agostinelli, Timo I Denk, Zalán Borsos, Jesse Engel, Mauro Verzetti, Antoine Caillon, Qingqing Huang, Aren Jansen, Adam Roberts, Marco Tagliasacchi, et al. 2023. Musiclm: Generating music from text. *arXiv preprint arXiv:2301.11325* (2023).

[4] 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. 2022. Flamingo: a Visual Language Model for Few-Shot Learning. arXiv:2204.14198 [cs.CV]

[5] Saleema Amershi, Maya Cakmak, William Bradley Knox, and Todd Kulesza. 2014. Power to the people: The role of humans in interactive machine learning. *Ai Magazine* 35, 4 (2014), 105–120.

[6] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. 2017. Neural Module Networks. arXiv:1511.02799 [cs.CV]

[7] 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. 2023. PaLM 2 Technical Report. arXiv:2305.10403 [cs.CL]

[8] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C. Lawrence Zitnick, and Devi Parikh. 2015. VQA: Visual Question Answering. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV).*

[9] Tenglong Ao, Zeyi Zhang, and Libin Liu. 2023. GestureDiffuCLIP: Gesture Diffusion Model with CLIP Latents. *ACM Trans. Graph.* 42, 4, Article 42 (jul 2023), 18 pages. https://doi.org/10.1145/3592097

[10] Bin Bi, Hao Ma, Bo-June (Paul) Hsu, Wei Chu, Kuansan Wang, and Junghoo Cho. 2015. Learning to Recommend Related Entities to Search Users. In *Proceedings of the Eighth ACM International Conference on Web Search and Data Mining* (Shanghai, China) *(WSDM '15).* Association for Computing Machinery, New York, NY, USA, 139–148. https://doi.org/10.1145/2684822.2685304

[11] Richard A. Bolt. 1980. "Put-That-There" : Voice and Gesture at the Graphics Interface. *SIGGRAPH Comput. Graph.* 14, 3 (July 1980), 262–270. https://doi.org/10.1145/965105.807503

[12] Dimitrios Bounias, Ashish Singh, Spyridon Bakas, Sarthak Pati, Saima Rathore, Hamed Akbari, Michel Bilello, Benjamin A Greenberger, Joseph Lombardo, Rhea D Chitalia, et al. 2021. Interactive Machine Learning-Based Multi-Label Segmentation of Solid Tumors and Organs. *Applied Sciences* 11, 16 (2021), 7488.

[13] Tim Brooks, Aleksander Holynski, and Alexei A. Efros. 2023. InstructPix2Pix: Learning to Follow Image Editing Instructions. arXiv:2211.09800 [cs.CV]

[14] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language Models are Few-shot Learners. *Advances in Neural Information Processing Systems* 33 (2020), 1877–1901. https://doi.org/10.48550/arXiv.2005.14165

[15] Carrie J. Cai, Jonas Jongejan, and Jess Holbrook. 2019. The Effects of Example-Based Explanations in a Machine Learning Interface. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 258–262. https://doi.org/10.1145/3301275.3302289

[16] Minghao Cai, Soh Masuko, and Jiro Tanaka. 2018. Gesture-Based Mobile Communication System Providing Side-by-Side Shopping Feeling. In *Proceedings of the 23rd International Conference on Intelligent User Interfaces Companion* (Tokyo, Japan) *(IUI '18 Companion)*. Association for Computing Machinery, New York, NY, USA, Article 2, 2 pages. https://doi.org/10.1145/3180308.3180310

[17] Z. Cao, G. Hidalgo Martinez, T. Simon, S. Wei, and Y. A. Sheikh. 2019. OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2019).

[18] Zhe Cao, Ilija Radosavovic, Angjoo Kanazawa, and Jitendra Malik. 2021. Reconstructing Hand-Object Interactions in the Wild. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 12417–12426.

[19] Michelle Carney, Barron Webster, Irene Alvarado, Kyle Phillips, Noura Howell, Jordan Griffith, Jonas Jongejan, Amit Pitaru, and Alexander Chen. 2020. Teachable Machine: Approachable Web-Based Tool for Exploring Machine Learning Classification. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3334480.3382839

[20] Maria Cristina Caselli. 1990. Communicative gestures and first words. In *From gesture to language in hearing and deaf children*. Springer, 56–67.

[21] Jessica R. Cauchard, Jane L. E, Kevin Y. Zhai, and James A. Landay. 2015. Drone & Me: An Exploration into Natural Human-Drone Interaction. In *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing* (Osaka, Japan) *(UbiComp '15)*. Association for Computing Machinery, New York, NY, USA, 361–365. https://doi.org/10.1145/2750858.2805823

[22] Chia-Ming Chang, Chia-Hsien Lee, and Takeo Igarashi. 2021. Spatial Labeling: Leveraging Spatial Layout for Improving Label Quality in Non-Expert Image Annotation. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems* (Yokohama, Japan) *(CHI '21)*. Association for Computing Machinery, New York, NY, USA, Article 306, 12 pages. https://doi.org/10.1145/3411764.3445165

[23] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. 2017. Rethinking Atrous Convolution for Semantic Image Segmentation. https://doi.org/10.48550/ARXIV.1706.05587

[24] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. 2018. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV).*

[25] Minmin Chen, Alex Beutel, Paul Covington, Sagar Jain, Francois Belletti, and Ed H. Chi. 2019. Top-K Off-Policy Correction for a REINFORCE Recommender System. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining* (Melbourne VIC, Australia) *(WSDM '19).* Association for Computing Machinery, New York, NY, USA, 456–464. https://doi.org/10.1145/3289600.3290999

[26] Weihao Chen, Chun Yu, Huadong Wang, Zheng Wang, Lichen Yang, Yukun Wang, Weinan Shi, and Yuanchun Shi. 2023. From Gap to Synergy: Enhancing Contextual Understanding through Human-Machine Collaboration in Personalized Systems. In *Proceedings of the 36th Annual ACM Symposium on User Interface Software and Technology* (San Francisco, CA, USA) *(UIST '23).* Association for Computing Machinery, New York, NY, USA, Article 110, 15 pages. https://doi.org/10.1145/3586183.3606741

[27] 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. 2023. PaLI: A Jointly-Scaled Multilingual Language-Image Model. arXiv:2209.06794 [cs.CV]

[28] Chia-Hsing Chiu, Yuki Koyama, Yu-Chi Lai, Takeo Igarashi, and Yonghao Yue. 2020. Human-in-the-Loop Differential Subspace Search in High-Dimensional Latent Space. *ACM Trans. Graph.* 39, 4, Article 85 (July 2020), 15 pages. https://doi.org/10.1145/3386569.3392409

[29] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311* (2022).

[30] Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling Instruction-Finetuned Language Models. arXiv:2210.11416 [cs.LG]

[31] Herbert H Clark. 2005. Coordinating with each other in a material world. *Discourse studies* 7, 4-5 (2005), 507–525.

[32] ComfyUI. 2023. ComfyUI. https://github.com/comfyanonymous/ComfyUI

[33] Allen Cypher and Daniel Conrad Halbert. 1993. *Watch what I do: programming by demonstration.* MIT press.

[34] Manuel Dahnert, Ji Hou, Matthias Niessner, and Angela Dai. 2021. Panoptic 3D Scene Reconstruction From a Single RGB Image. In *Advances in Neural Information Processing Systems*, M. Ranzato, A. Beygelzimer, K. Nguyen, P. S. Liang, J. W. Vaughan, and Y. Dauphin (Eds.), Vol. 34. Curran Associates, Inc., 8282–8293. https://proceedings.neurips.cc/paper/2021/file/46031b3d04dc90994ca317a7c55c4289-Paper.pdf

[35] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, , Antonino Furnari, Jian Ma, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. 2021. Rescaling Egocentric Vision: Collection, Pipeline and Challenges for EPIC-KITCHENS-100. *International Journal of Computer Vision (IJCV)* (2021). https://doi.org/10.1007/s11263-021-01531-2

[36] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. 2018. Scaling Egocentric Vision: The EPIC-KITCHENS Dataset. In *European Conference on Computer Vision (ECCV).*

[37] Chandan Datta, Chandimal Jayawardena, I Han Kuo, and Bruce A MacDonald. 2012. RoboStudio: A visual programming environment for rapid authoring and customization of complex services on a personal service robot. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems.* 2352–2357. https://doi.org/10.1109/IROS.2012.6386105

[38] Carlos de la Torre-Ortiz, Michiel M. Spapé, Lauri Kangassalo, and Tuukka Ruotsalo. 2020. Brain Relevance Feedback for Interactive Image Generation. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20).* Association for Computing Machinery, New York, NY, USA, 1060–1070. https://doi.org/10.1145/3379337.3415821

[39] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[40] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. 2009. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition.* 248–255. https://doi.org/10.1109/CVPR.2009.5206848

[41] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs.CL]

[42] Anind K. Dey, Raffay Hamid, Chris Beckmann, Ian Li, and Daniel Hsu. 2004. A CAPpella: Programming by Demonstration of Context-Aware Applications. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vienna, Austria) *(CHI '04)*. Association for Computing Machinery, New York, NY, USA, 33–40. https://doi.org/10.1145/985692.985697

[43] Griffin Dietz, Nadin Tamer, Carina Ly, Jimmy K Le, and James A. Landay. 2023. Visual StoryCoder: A Multimodal Programming Environment for Children' s Creation of Stories. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 96, 16 pages. https://doi.org/10.1145/3544548.3580981

[44] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A Survey on In-context Learning. arXiv:2301.00234 [cs.CL]

[45] Ruofei Du, Na Li, Jing Jin, Michelle Carney, Scott Miles, Maria Kleiner, Xiuxiu Yuan, Yinda Zhang, Anuva Kulkarni, Xingyu Liu, Ahmed Sabie, Sergio Orts-Escolano, Abhishek Kar, Ping Yu, Ram Iyengar, Adarsh Kowdle, and Alex Olwal. 2023. Rapsai: Accelerating Machine Learning Prototyping of Multimedia Applications Through Visual Programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 125, 23 pages. https://doi.org/10.1145/3544548.3581338

[46] Ruofei Du, Na Li, Jing Jin, Michelle Carney, Xiuxiu Yuan, Kristen Wright, Mark Sherwood, Jason Mayes, Lin Chen, Jun Jiang, Jingtao Zhou, Zhongyi Zhou, Ping Yu, Adarsh Kowdle, Ram Iyengar, and Alex Olwal. 2023. Experiencing Visual Blocks for ML: Visual Prototyping of AI Pipelines. In *Adjunct Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology (UIST)*. ACM. https://doi.org/10.1145/3586182.3615817

[47] John J. Dudley and Per Ola Kristensson. 2018. A Review of User Interface Design for Interactive Machine Learning. *ACM Trans. Interact. Intell. Syst.* 8, 2, Article 8 (June 2018), 37 pages. https://doi.org/10.1145/3185517

[48] John J. Dudley and Per Ola Kristensson. 2018. A Review of User Interface Design for Interactive Machine Learning. *ACM Trans. Interact. Intell. Syst.* 8, 2, Article 8 (jun 2018), 37 pages. https://doi.org/10.1145/3185517

[49] Jerry Alan Fails and Dan R. Olsen. 2003. Interactive Machine Learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces* (Miami, Florida, USA) *(IUI '03)*. Association for Computing Machinery, New York, NY, USA, 39–45. https://doi.org/10.1145/604045.604056

[50] KJ Feng, Xander Koo, Lawrence Tan, Amy Bruckman, David W McDonald, and Amy X Zhang. 2024. Mapping the Design Space of Teachable Social Media Feed Experiences. *arXiv preprint arXiv:2401.14000* (2024).

[51] Rebecca Fiebrink, Perry R. Cook, and Dan Trueman. 2011. Human Model Evaluation in Interactive Supervised Learning. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Vancouver, BC, Canada) *(CHI '11)*. Association for Computing Machinery, New York, NY, USA, 147–156. https://doi.org/10.1145/1978942.1978965

[52] James Fogarty, Desney Tan, Ashish Kapoor, and Simon Winder. 2008. CueFlik: Interactive Concept Learning in Image Search. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Florence, Italy) *(CHI '08)*. Association for Computing Machinery, New York, NY, USA, 29–38. https://doi.org/10.1145/1357054.1357061

[53] Jules Françoise, Baptiste Caramiaux, and Téo Sanchez. 2021. Marcelle: Composing Interactive Machine Learning Workflows and Interfaces. In *The 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 39–53. https://doi.org/10.1145/3472749.3474734

[54] Christoph Gebhardt, Brian Hecox, Bas van Opheusden, Daniel Wigdor, James Hillis, Otmar Hilliges, and Hrvoje Benko. 2019. Learning Cooperative Personalized Policies from Gaze Data. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 197–208. https://doi.org/10.1145/3332165.3347933

[55] Yolanda Gil, James Honaker, Shikhar Gupta, Yibo Ma, Vito D'Orazio, Daniel Garijo, Shruti Gadewar, Qifan Yang, and Neda Jahanshad. 2019. Towards Human-Guided Machine Learning. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 614–624. https://doi.org/10.1145/3301275.3302324

[56] GitHub. 2023. GitHub Copilot · Your AI pair programmer. https://github.com/features/copilot

[57] D. Goehring, J. Hoffman, E. Rodner, K. Saenko, and T. Darrell. 2014. Interactive adaptation of real-time object detectors. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*. 1282–1289. https://doi.org/10.1109/ICRA.2014.6907018

[58] Ke Gong, Xiaodan Liang, Dongyu Zhang, Xiaohui Shen, and Liang Lin. 2017. Look Into Person: Self-Supervised Structure-Sensitive Learning and a New Benchmark for Human Parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[59] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative Adversarial Nets. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2672–2680. http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[60] Tanmay Gupta and Aniruddha Kembhavi. 2023. Visual Programming: Compositional Visual Reasoning Without Training. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition.* arXiv. https://doi.org/10.48550/arXiv.2211.11559

[61] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. 2022. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence* 45, 1 (2022), 87–110.

[62] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. 2020. Array programming with NumPy. *Nature* 585, 7825 (Sept. 2020), 357–362. https://doi.org/10.1038/s41586-020-2649-2

[63] Sandra G Hart and Lowell E Staveland. 1988. Development of NASA-TLX (Task Load Index): Results of empirical and theoretical research. In *Advances in psychology.* Vol. 52. Elsevier, 139–183.

[64] Richard Sahala Hartanto, Ryoichi Ishikawa, Menandro Roxas, and Takeshi Oishi. 2020. Hand-Motion-guided Articulation and Segmentation Estimation. In *2020 29th IEEE International Conference on Robot and Human Interactive Communication (RO-MAN).* 807–813. https://doi.org/10.1109/RO-MAN47096.2020.9223433

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR).*

[66] Fred Hohman, Kanit Wongsuphasawat, Mary Beth Kery, and Kayur Patel. 2020. Understanding and Visualizing Data Iteration in Machine Learning. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20).* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376177

[67] Jonggi Hong, Kyungjun Lee, June Xu, and Hernisa Kacorri. 2020. Crowdsourcing the Perception of Machine Teaching. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI '20).* Association for Computing Machinery, New York, NY, USA, 1–14. https://doi.org/10.1145/3313831.3376428

[68] Eric Horvitz. 1999. Principles of Mixed-Initiative User Interfaces. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (Pittsburgh, Pennsylvania, USA) *(CHI '99).* Association for Computing Machinery, New York, NY, USA, 159–166. https://doi.org/10.1145/302979.303030

[69] Justin Huang and Maya Cakmak. 2017. Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts. In *Proceedings of the 2017 ACM/IEEE International Conference on Human-Robot Interaction*. 453–462.

[70] Justin Huang, Tessa Lau, and Maya Cakmak. 2016. Design and evaluation of a rapid programming system for service robots. In *2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*. IEEE, 295–302.

[71] Lin Huang, Jianchao Tan, Ji Liu, and Junsong Yuan. 2020. Hand-Transformer: Non-Autoregressive Structured Modeling for 3D Hand Pose Estimation. In *Computer Vision – ECCV 2020*, Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm (Eds.). Springer International Publishing, Cham, 17–33.

[72] HuggingFace. 2022. Spaces. https://huggingface.co/docs/transformers/preprocessing

[73] Kurusugawa Computer Inc. 2022. *AnnoFab*. Retrieved Apr 2, 2022 from https://annofab.com/

[74] Shahram Izadi, David Kim, Otmar Hilliges, David Molyneaux, Richard Newcombe, Pushmeet Kohli, Jamie Shotton, Steve Hodges, Dustin Freeman, Andrew Davison, and Andrew Fitzgibbon. 2011. KinectFusion: Real-Time 3D Reconstruction and Interaction Using a Moving Depth Camera. In *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology* (Santa Barbara, California, USA) *(UIST '11)*. Association for Computing Machinery, New York, NY, USA, 559–568. https://doi.org/10.1145/2047196.2047270

[75] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of Experts. arXiv:2401.04088 [cs.LG]

[76] Peiling Jiang. 2023. Positional Control in Node-Based Programming. In *Extended Abstracts of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI EA '23)*. Association for Computing Machinery, New York, NY, USA, Article 231, 7 pages. https://doi.org/10.1145/3544549.3585878

[77] Peiling Jiang, Jude Rayan, Steven P Dow, and Haijun Xia. 2023. Graphologue: Exploring Large Language Model Responses with Interactive Diagrams. *arXiv preprint arXiv:2305.11473* (2023).

[78] Eunkyung Jo, Daniel A. Epstein, Hyunhoon Jung, and Young-Ho Kim. 2023. Understanding the Benefits and Challenges of Deploying Conversational AI Leveraging Large Language Models for Public Health Intervention. In

Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (<conf-loc>, <city>Hamburg</city>, <country>Germany</country>, </conf-loc>) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 18, 16 pages. https://doi.org/10.1145/3544548.3581503

[79] Hernisa Kacorri, Kris M. Kitani, Jeffrey P. Bigham, and Chieko Asakawa. 2017. People with Visual Impairment Training Personal Object Recognizers: Feasibility and Challenges. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver, Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 5839–5849. https://doi.org/10.1145/3025453.3025899

[80] Maria Karam and m. c. schraefel. 2005. *A Taxonomy of Gestures in Human Computer Interactions.* Project Report. https://eprints.soton.ac.uk/261149/

[81] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. 2021. Layered Neural Atlases for Consistent Video Editing. *ACM Trans. Graph.* 40, 6, Article 210 (dec 2021), 12 pages. https://doi.org/10.1145/3478513.3480546

[82] Bongjun Kim and Bryan Pardo. 2018. A Human-in-the-Loop System for Sound Event Detection and Annotation. *ACM Trans. Interact. Intell. Syst.* 8, 2, Article 13 (June 2018), 23 pages. https://doi.org/10.1145/3214366

[83] Joohwan Kim, Michael Stengel, Alexander Majercik, Shalini De Mello, David Dunn, Samuli Laine, Morgan McGuire, and David Luebke. 2019. NVGaze: An Anatomically-Informed Dataset for Low-Latency, Near-Eye Gaze Estimation. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300780

[84] Russell B. Kline, Gary D. Hamor, Kenneth L. Krause, and Larry E. Druffel. 1978. Visual Demonstration of Program Execution. In *Papers of the SIGCSE/CSA Technical Symposium on Computer Science Education* (Detroit, Michigan) *(SIGCSE '78)*. Association for Computing Machinery, New York, NY, USA, 16–18. https://doi.org/10.1145/990555.990559

[85] Jeffrey Kodosky. 2020. LabVIEW. *Proc. ACM Program. Lang.* 4, HOPL, Article 78 (jun 2020), 54 pages. https://doi.org/10.1145/3386328

[86] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems* 35 (2022), 22199–22213.

[87] Anastasia Kovalkov, Avi Segal, and Kobi Gal. 2020. Inferring Creativity in Visual Programming Environments. In *Proceedings of the Seventh ACM Conference on Learning @ Scale* (Virtual Event, USA) *(L@S '20)*. Association for Computing Machinery, New York, NY, USA, 269–272. https://doi.org/10.1145/3386527.3406725

[88] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems* 25 (2012).

[89] Tiffany H Kung, Morgan Cheatham, Arielle Medenilla, Czarina Sillos, Lorie De Leon, Camille Elepaño, Maria Madriaga, Rimel Aggabao, Giezel Diaz-Candido, James Maningo, et al. 2023. Performance of ChatGPT on USMLE: Potential for AI-assisted medical education using large language models. *PLoS digital health* 2, 2 (2023), e0000198.

[90] Michael Laielli, James Smith, Giscard Biamby, Trevor Darrell, and Bjoern Hartmann. 2019. LabelAR: A Spatial Guidance Interface for Fast Computer Vision Image Collection. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 987–998. https://doi.org/10.1145/3332165.3347927

[91] LangFlow. 2023. LangFlow. https://github.com/logspace-ai/langflow

[92] Tessa A. Lau and Daniel S. Weld. 1998. Programming by Demonstration: An Inductive Learning Formulation. In *Proceedings of the 4th International Conference on Intelligent User Interfaces* (Los Angeles, California, USA) *(IUI '99)*. Association for Computing Machinery, New York, NY, USA, 145–152. https://doi.org/10.1145/291080.291104

[93] Hsin-Ying Lee, Xiaodong Yang, Ming-Yu Liu, Ting-Chun Wang, Yu-Ding Lu, Ming-Hsuan Yang, and Jan Kautz. 2019. Dancing to Music. In *Advances in Neural Information Processing Systems 32*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). Curran Associates, Inc., 3581–3591. http://papers.nips.cc/paper/8617-dancing-to-music.pdf

[94] Kyungjun Lee and Hernisa Kacorri. 2019. Hands Holding Clues for Object Recognition in Teachable Machines. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300566

[95] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. 2022. Grounded Language-Image Pre-Training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10965–10975.

[96] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. 2020. Self-Correction for Human Parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1. https://doi.org/10.1109/TPAMI.2020.3048039

[97] Toby Jia-Jun Li, Amos Azaria, and Brad A. Myers. 2017. SUGILITE: Creating Multimodal Smartphone Automation by Demonstration. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems* (Denver,

Colorado, USA) *(CHI '17)*. Association for Computing Machinery, New York, NY, USA, 6038–6049. https://doi.org/10.1145/3025453.3025483

[98] Toby Jia-Jun Li, Jingya Chen, Brandon Canfield, and Brad A. Myers. 2020. Privacy-Preserving Script Sharing in GUI-Based Programming-by-Demonstration Systems. *Proc. ACM Hum.-Comput. Interact.* 4, CSCW1, Article 060 (May 2020), 23 pages. https://doi.org/10.1145/3392869

[99] Toby Jia-Jun Li, Marissa Radensky, Justin Jia, Kirielle Singarajah, Tom M. Mitchell, and Brad A. Myers. 2019. PUMICE: A Multi-Modal Agent That Learns Concepts and Conditionals from Natural Language and Demonstrations. In *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology* (New Orleans, LA, USA) *(UIST '19)*. Association for Computing Machinery, New York, NY, USA, 577–589. https://doi.org/10.1145/3332165.3347899

[100] Xiwen Liang, Yangxin Wu, Jianhua Han, Hang Xu, Chunjing XU, and Xiaodan Liang. 2022. Effective Adaptation in Multi-Task Co-Training for Unified Autonomous Driving. In *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 19645–19658. https://proceedings.neurips.cc/paper_files/paper/2022/file/7c319b62e2257b34cb0e1040ced2e007-Paper-Conference.pdf

[101] Shih-Chieh Lin, Chang-Hong Hsu, Walter Talamonti, Yunqi Zhang, Steve Oney, Jason Mars, and Lingjia Tang. 2018. Adasa: A Conversational In-Vehicle Digital Assistant for Advanced Driver Assistance Features. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology* (Berlin, Germany) *(UIST '18)*. Association for Computing Machinery, New York, NY, USA, 531–542. https://doi.org/10.1145/3242587.3242593

[102] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*. Springer, 740–755.

[103] Linqing Liu, Yao Lu, Min Yang, Qiang Qu, Jia Zhu, and Hongyan Li. 2017. Generative Adversarial Network for Abstractive Text Summarization. arXiv:1711.09357 [cs.CL]

[104] Xingyu Liu, Vladimir Kirilyuk, Xiuxiu Yuan, Alex Olwal, Peggy Chi, Xiang Chen, and Ruofei Du. 2023. Visual Captions: Augmenting Verbal Communication With On-the-fly Visuals. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems (CHI)*. ACM, 20 pages. https://doi.org/10.1145/3544548.3581566

[105] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, et al. 2022. Swin transformer v2: Scaling up capacity and resolution. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 12009–12019.

[106] Zixian Ma, Jerry Hong, Mustafa Omer Gul, Mona Gandhi, Irena Gao, and Ranjay Krishna. 2023. CREPE: Can Vision-Language Foundation Models Reason Compositionally?. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 10910–10921.

[107] Andrew N Meltzoff. 1995. Understanding the intentions of others: re-enactment of intended acts by 18-month-old children. *Developmental psychology* 31, 5 (1995), 838.

[108] Kyzyl Monteiro, Ritik Vatsal, Neil Chulpongsatorn, Aman Parnami, and Ryo Suzuki. 2023. Teachable Reality: Prototyping Tangible Augmented Reality with Everyday Objects by Leveraging Interactive Machine Teaching. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (<conf-loc>, <city>Hamburg</city>, <country>Germany</country>, </conf-loc>) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 459, 15 pages. https://doi.org/10.1145/3544548.3581449

[109] Meredith Ringel Morris, Jascha Sohl-dickstein, Noah Fiedel, Tris Warkentin, Allan Dafoe, Aleksandra Faust, Clement Farabet, and Shane Legg. 2023. Levels of AGI: Operationalizing Progress on the Path to AGI. arXiv:2311.02462 [cs.AI]

[110] Eric N Mortensen and William A Barrett. 1998. Interactive segmentation with intelligent scissors. *Graphical models and image processing* 60, 5 (1998), 349–384.

[111] MA Viraj J Muthugala and AG Buddhika P Jayasekara. 2016. MIRob: An intelligent service robot that learns from interactive discussions while handling uncertain information in user instructions. In *2016 Moratuwa Engineering Research Conference (MERCon)*. IEEE, 397–402.

[112] B. A. Myers, B. V. Zanden, and R. B. Dannenberg. 1989. Creating Graphical Interactive Application Objects by Demonstration. In *Proceedings of the 2nd Annual ACM SIGGRAPH Symposium on User Interface Software and Technology* (Williamsburg, Virginia, USA) *(UIST '89)*. Association for Computing Machinery, New York, NY, USA, 95–104. https://doi.org/10.1145/73660.73672

[113] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[114] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs.CL]

[115] Allan Paivio. 1969. Mental imagery in associative learning and memory. *Psychological review* 76, 3 (1969), 241.

[116] Allan Paivio. 1991. Dual coding theory: Retrospect and current status. *Canadian Journal of Psychology/Revue canadienne de psychologie* 45, 3 (1991), 255.

[117] Allan Paivio, James M Clark, et al. 2006. Dual coding theory and education. *Pathways to literacy achievement for high poverty children* (2006), 1–20.

[118] Sinno Jialin Pan and Qiang Yang. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering* 22, 10 (2009), 1345–1359.

[119] Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2023. Generative Agents: Interactive Simulacra of Human Behavior. arXiv:2304.03442 [cs.HC]

[120] Joon Sung Park, Lindsay Popowski, Carrie Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2022. Social Simulacra: Creating Populated Prototypes for Social Computing Systems. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 74, 18 pages. https://doi.org/10.1145/3526113.3545616

[121] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems* 32 (2019).

[122] Siyou Pei, Alexander Chen, Jaewook Lee, and Yang Zhang. 2022. Hand Interfaces: Using Hands to Imitate Objects in AR/VR for Expressive Interactions. In *CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 429, 16 pages. https://doi.org/10.1145/3491102.3501898

[123] Zhenhui Peng, Xingbo Wang, Qiushi Han, Junkai Zhu, Xiaojuan Ma, and Huamin Qu. 2023. Storyfier: Exploring Vocabulary Learning Support with Text Generation Models. arXiv:2308.03864 [cs.HC]

[124] Gabriella Pizzuto and Angelo Cangelosi. 2019. Exploring Deep Models for Comprehension of Deictic Gesture-Word Combinations in Cognitive Robotics. In *2019 International Joint Conference on Neural Networks (IJCNN)*. 1–7. https://doi.org/10.1109/IJCNN.2019.8852425

[125] E. V. Polyakov, M. S. Mazhanov, A. Y. Rolich, L. S. Voskov, M. V. Kachalova, and S. V. Polyakov. 2018. Investigation and development of the intelligent voice assistant for the Internet of Things using machine learning. In *2018 Moscow Workshop on Electronic and Networking Technologies (MWENT)*. 1–5. https://doi.org/10.1109/MWENT.2018.8337236

[126] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. arXiv:2103.00020 [cs.CV]

[127] Gonzalo A. Ramos, Christopher Meek, Patrice Y. Simard, Jina Suh, and Soroush Ghorashi. 2020. Interactive machine teaching: a human-centered approach to building machine-learned models. *Hum. Comput. Interact.* 35, 5-6 (2020), 413–451. https://doi.org/10.1080/07370024.2020.1734931

[128] Arpit Rana and Derek Bridge. 2020. Navigation-by-Preference: A New Conversational Recommender with Preference-Based Feedback. In *Proceedings of the 25th International Conference on Intelligent User Interfaces* (Cagliari, Italy) *(IUI '20)*. Association for Computing Machinery, New York, NY, USA, 155–165. https://doi.org/10.1145/3377325.3377496

[129] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

[130] Carsten Rother, Vladimir Kolmogorov, and Andrew Blake. 2004. "GrabCut": Interactive Foreground Extraction Using Iterated Graph Cuts. *ACM Trans. Graph.* 23, 3 (aug 2004), 309–314. https://doi.org/10.1145/1015706.1015720

[131] Téo Sanchez, Baptiste Caramiaux, Jules Françoise, Frédéric Bevilacqua, and Wendy E. Mackay. 2021. How Do People Train a Machine? Strategies and (Mis)Understandings. *Proc. ACM Hum.-Comput. Interact.* 5, CSCW1, Article 162 (apr 2021), 26 pages. https://doi.org/10.1145/3449236

[132] Téo Sanchez, Baptiste Caramiaux, Pierre Thiel, and Wendy E. Mackay. 2022. Deep Learning Uncertainty in Machine Teaching. In *27th International Conference on Intelligent User Interfaces* (Helsinki, Finland) *(IUI '22)*. Association for Computing Machinery, New York, NY, USA, 173–190. https://doi.org/10.1145/3490099.3511117

[133] Allison Sauppé and Bilge Mutlu. 2014. Robot Deictics: How Gesture and Context Shape Referential Communication. In *Proceedings of the 2014 ACM/IEEE International Conference on Human-Robot Interaction* (Bielefeld, Germany) *(HRI '14)*. Association for Computing Machinery, New York, NY, USA, 342–349. https://doi.org/10.1145/2559636.2559657

[134] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOsmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. 2020. *opencv/cvat: v1.1.0*. https://doi.org/10.5281/zenodo.4009388

[135] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision*. 618–626.

[136] SM Seyednezhad, Kailey Nobuko Cozart, John Anthony Bowllan, and Anthony O Smith. 2018. A review on recommendation systems: Context-aware to social-based. *arXiv preprint arXiv:1811.11866* (2018).

[137] Dandan Shan, Jiaqi Geng, Michelle Shu, and David F. Fouhey. 2020. Understanding Human Hands in Contact at Internet Scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).*

[138] Leixian Shen, Enya Shen, Yuyu Luo, Xiaocong Yang, Xuming Hu, Xiongshuai Zhang, Zhiwei Tai, and Jianmin Wang. 2023. Towards Natural Language Interfaces for Data Visualization: A Survey. *IEEE Transactions on Visualization and Computer Graphics* 29, 6 (2023), 3121–3144. https://doi.org/10.1109/TVCG.2022.3148007

[139] Yichen Shen, Nicholas C Harris, Scott Skirlo, Mihika Prabhu, Tom Baehr-Jones, Michael Hochberg, Xin Sun, Shijie Zhao, Hugo Larochelle, Dirk Englund, et al. 2017. Deep learning with coherent nanophotonic circuits. *Nature Photonics* 11, 7 (2017), 441.

[140] Yujun Shen, Ceyuan Yang, Xiaoou Tang, and Bolei Zhou. 2020. InterFace-GAN: Interpreting the Disentangled Face Representation Learned by GANs. arXiv:2005.09635 [cs.CV]

[141] Patrice Y. Simard, Saleema Amershi, David M. Chickering, Alicia Edelman Pelton, Soroush Ghorashi, Christopher Meek, Gonzalo Ramos, Jina Suh, Johan Verwey, Mo Wang, and John Wernsing. 2017. Machine Teaching: A New Paradigm for Building Machine Learning Systems. https://doi.org/10.48550/ARXIV.1707.06742

[142] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA).* IEEE, 11523–11530.

[143] Nikhita Singh, Jin Joo Lee, Ishaan Grover, and Cynthia Breazeal. 2018. P2PSTORY: Dataset of Children as Storytellers and Listeners in Peer-to-Peer Interactions. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems* (Montreal QC, Canada) *(CHI '18).* Association for Computing Machinery, New York, NY, USA, 1–11. https://doi.org/10.1145/3173574.3174008

[144] Karan Singhal, Shekoofeh Azizi, Tao Tu, S. Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, Perry Payne, Martin Seneviratne, Paul Gamble, Chris Kelly, Nathaneal Scharli, Aakanksha Chowdhery, Philip Mansfield, Blaise Aguera y Arcas, Dale Webster, Greg S. Corrado, Yossi Matias, Katherine Chou, Juraj Gottweis, Nenad Tomasev, Yun Liu, Alvin Rajkomar, Joelle Barral, Christopher Semturs, Alan Karthikesalingam, and Vivek Natarajan. 2022. Large Language Models Encode Clinical Knowledge. arXiv:2212.13138 [cs.CL]

[145] Daniel Smilkov, Nikhil Thorat, Yannick Assogba, Ann Yuan, Nick Kreeger, Ping Yu, Kangyi Zhang, Shanqing Cai, Eric Nielsen, David Soergel, Stan Bileschi, Michael Terry, Charles Nicholson, Sandeep N. Gupta, Sarah Sirajuddin,

D. Sculley, Rajat Monga, Greg Corrado, Fernanda B. Viégas, and Martin Wattenberg. 2019. TensorFlow.js: Machine Learning for the Web and Beyond. https://doi.org/10.48550/arXiv.1901.05350

[146] Alison Smith-Renner, Ron Fan, Melissa Birchfield, Tongshuang Wu, Jordan Boyd-Graber, Daniel S. Weld, and Leah Findlater. 2020. *No Explainability without Accountability: An Empirical Study of Explanations and Feedback in Interactive ML*. Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376624

[147] Konstantin Sofiiuk, Ilia Petrov, Olga Barinova, and Anton Konushin. 2020. f-brs: Rethinking backpropagating refinement for interactive segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8623–8632.

[148] Konstantin Sofiiuk, Ilia A. Petrov, and Anton Konushin. 2021. Reviving Iterative Training with Mask Guidance for Interactive Segmentation. arXiv:2102.06583 [cs.CV]

[149] Jina Suh, Soroush Ghorashi, Gonzalo Ramos, Nan-Chen Chen, Steven Drucker, Johan Verwey, and Patrice Simard. 2019. AnchorViz: Facilitating Semantic Data Exploration and Concept Discovery for Interactive Machine Learning. *ACM Trans. Interact. Intell. Syst.* 10, 1, Article 7 (Aug. 2019), 38 pages. https://doi.org/10.1145/3241379

[150] Sangho Suh, Bryan Min, Srishti Palani, and Haijun Xia. 2023. Sensecape: Enabling Multilevel Exploration and Sensemaking with Large Language Models. *arXiv preprint arXiv:2305.11483* (2023).

[151] Dídac Surís, Sachit Menon, and Carl Vondrick. 2023. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128* (2023).

[152] William Robert Sutherland. 1966. *The on-line graphical specification of computer procedures*. Ph. D. Dissertation. Massachusetts Institute of Technology.

[153] Mingxing Tan and Quoc Le. 2019. EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 6105–6114. https://proceedings.mlr.press/v97/tan19a.html

[154] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, 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, 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, Ruibo Liu, Yunxuan Li, Sarah Cogan, Jeremy Chen, Chao Jia, Chenjie Gu, Qiao Zhang, Jordan Grimstad, Ale Jakse Hartman, Martin Chadwick, Gaurav Singh Tomar, Xavier Garcia, Evan Senter, Emanuel Taropa, 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, Yujing Zhang, Ravi Addanki, Antoine Miech, Annie Louis, Laurent El Shafey, Denis Teplyashin, Geoff Brown, Elliot Catt, Nithya Attaluri, 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, Vitaly 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, Hanzhao Lin, 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, Yong Cheng, 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, 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, YaGuang Li, 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, Gamaleldin Elsayed, Ed Chi, Mahdis Mahdieh, Ian Tenney, Nan Hua, Ivan Petrychenko, Patrick Kane, Dylan Scandinaro, Rishub Jain, Jonathan Uesato, Romina Datta, Adam Sadovsky, Oskar Bunyan, Dominik Rabiej, Shimu Wu, John Zhang, Gautam Vasudevan, Edouard Leurent, Mahmoud Alnahlawi, Ionut Georgescu, Nan Wei, Ivy Zheng, Betty Chan, Pam G Rabinovitch, 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, Sahitya Potluri, Jane Park, Elnaz Davoodi, Jiageng Zhang, Jeff Stanway, Drew Garmon, Abhijit Karmarkar, Zhe Dong, Jong Lee, Aviral Kumar, Luowei Zhou, Jonathan Evens, William Isaac, Zhe Chen, Johnson Jia, Anselm Levskaya, Zhenkai Zhu, Chris Gorgolewski, Peter

Grabowski, Yu Mao, Alberto Magni, Kaisheng Yao, Javier Snaider, Norman Casagrande, Paul Suganthan, Evan Palmer, Geoffrey Irving, Edward Loper, Manaal Faruqui, Isha Arkatkar, Nanxin Chen, Izhak Shafran, Michael Fink, 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, Marin Georgiev, Marcus Wu, Ricardo Aguilar, Keith Pallo, Abhishek Chakladar, Alena Repina, Xihui Wu, Tom van der Weide, Priya Ponnapalli, Caroline Kaplan, Jiri Simsa, Shuangfeng Li, Olivier Dousse, Fan Yang, Jeff Piper, Nathan Ie, Minnie Lui, Rama Pasumarthi, Nathan Lintz, Anitha Vijayakumar, Lam Nguyen Thiet, Daniel Andor, Pedro Valenzuela, Cosmin Paduraru, Daiyi Peng, Katherine Lee, Shuyuan Zhang, Somer Greene, Duc Dung Nguyen, Paula Kurylowicz, Sarmishta Velury, Sebastian Krause, Cassidy Hardin, Lucas Dixon, Lili Janzer, Kiam Choo, Ziqiang Feng, Biao Zhang, Achintya Singhal, Tejasi Latkar, Mingyang Zhang, Quoc Le, Elena Allica Abellan, 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, Sid Lall, Ken Franko, Egor Filonov, Anna Bulanova, Rémi Leblond, Vikas Yadav, Shirley Chung, Harry Askham, Luis C. Cobo, Kelvin Xu, Felix Fischer, Jun Xu, Christina Sorokin, Chris Alberti, Chu-Cheng Lin, Colin Evans, Hao Zhou, Alek Dimitriev, Hannah Forbes, Dylan Banarse, Zora Tung, Jeremiah Liu, Mark Omernick, Colton Bishop, Chintu Kumar, Rachel Sterneck, Ryan Foley, Rohan Jain, Swaroop Mishra, Jiawei Xia, Taylor Bos, Geoffrey Cideron, Ehsan Amid, Francesco Piccinno, Xingyu Wang, Praseem Banzal, Petru Gurita, Hila Noga, Premal Shah, Daniel J. Mankowitz, Alex Polozov, Nate Kushman, Victoria Krakovna, Sasha Brown, MohammadHossein Bateni, Dennis Duan, Vlad Firoiu, Meghana Thotakuri, Tom Natan, Anhad Mohananey, Matthieu Geist, Sidharth Mudgal, Sertan Girgin, Hui Li, Jiayu Ye, Ofir Roval, Reiko Tojo, Michael Kwong, James Lee-Thorp, Christopher Yew, Quan Yuan, Sumit Bagri, Danila Sinopalnikov, Sabela Ramos, John Mellor, Abhishek Sharma, Aliaksei Severyn, Jonathan Lai, Kathy Wu, Heng-Tze Cheng, 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, Mark Geller, Tian Huey Teh, Jason Sanmiya, Evgeny Gladchenko, Nejc Trdin, Andrei Sozanschi, 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, Abhimanyu Goyal, Diane Wu, Denese Owusu-Afriyie, Cosmo Du, Chloe Thornton, Jordi Pont-Tuset, Pradyumna Narayana, Jing Li, Sabaer Fatehi, John Wieting, Omar Ajmeri, Benigno Uria, Tao Zhu, Yeongil Ko, Laura Knight, Amélie Héliou, Ning Niu, Shane Gu, Chenxi Pang, Dustin Tran, Yeqing Li, Nir Levine, Ariel Stolovich, Norbert Kalb, Rebeca Santamaria-Fernandez, Sonam Goenka, Wenny Yustalim, Robin Strudel, Ali Elqursh, Balaji Lakshminarayanan, Charlie Deck, Shyam Upadhyay, Hyo Lee, Mike Dusenberry, Zonglin Li, Xuezhi Wang, Kyle Levin, Raphael

Hoffmann, Dan Holtmann-Rice, Olivier Bachem, Summer Yue, Sho Arora, Eric Malmi, Daniil Mirylenka, Qijun Tan, Christy Koh, Soheil Hassas Yeganeh, Siim Põder, Steven Zheng, Francesco Pongetti, Mukarram Tariq, Yanhua Sun, Lucian Ionita, Mojtaba Seyedhosseini, Pouya Tafti, Ragha Kotikalapudi, 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, Chenkai Kuang, Vinod Koverkathu, Christopher A. Choquette-Choo, Yunjie Li, TJ Lu, Abe Ittycheriah, Prakash Shroff, Pei Sun, Mani Varadarajan, Sanaz Bahargam, Rob Willoughby, David Gaddy, Ishita Dasgupta, 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, Yuan Liu, 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, Ivo Penchev, 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, Adam Kurzrok, Lynette Webb, Sahil Dua, Dong Li, Preethi Lahoti, 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, Taylan Bilal, Evgenii Eltyshev, Daniel Balle, 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, Adams Yu, Christof Angermueller, Xiaowei Li, 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, Kevin Brooks, Ken Durden, Harsh Mehta, Nikola Momchev, Elahe Rahimtoroghi, Maria Georgaki, Amit Raul, Sebastian Ruder, Morgan Redshaw, Jinhyuk Lee, Komal Jalan, Dinghua Li, Ginger Perng, Blake Hechtman, Parker Schuh, Milad Nasr, Mia Chen, Kieran Milan, Vladimir Mikulik, Trevor Strohman, Juliana Franco, Tim Green, Demis Hassabis, Koray Kavukcuoglu, Jeffrey Dean, and Oriol Vinyals. 2023. Gemini: A Family of Highly Capable Multimodal Models. arXiv:2312.11805 [cs.CL]

[155] Guy Tevet, Brian Gordon, Amir Hertz, Amit H Bermano, and Daniel Cohen-Or. 2022. Motionclip: Exposing human motion generation to clip space. In *European Conference on Computer Vision*. Springer, 358–374.

[156] Jeyan Thiyagalingam, Mallikarjun Shankar, Geoffrey Fox, and Tony Hey. 2022. Scientific machine learning benchmarks. *Nature Reviews Physics* 4, 6 (2022), 413–420.

[157] Unity. 2023. Unity's Graph Editor. https://docs.unity.cn/Packages/com.unity.visualscripting@1.7/manual/vs-interface-overview.html#the-graph-editor

[158] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *Advances in Neural Information Processing Systems* 30 (2017). https://doi.org/10.5555/3295222.3295349

[159] Lucas Vinh Tran, Yi Tay, Shuai Zhang, Gao Cong, and Xiaoli Li. 2020. HyperML: A Boosting Metric Learning Approach in Hyperbolic Space for Recommender Systems. In *Proceedings of the 13th International Conference on Web Search and Data Mining* (Houston, TX, USA) *(WSDM '20)*. Association for Computing Machinery, New York, NY, USA, 609–617. https://doi.org/10.1145/3336191.3371850

[160] Vijay Viswanathan, Chenyang Zhao, Amanda Bertsch, Tongshuang Wu, and Graham Neubig. 2023. Prompt2Model: Generating Deployable Models from Natural Language Instructions. arXiv:2308.12261 [cs.CL]

[161] Tijana Vuletic, Alex Duffy, Laura Hay, Chris McTeague, Gerard Campbell, and Madeleine Grealy. 2019. Systematic literature review of hand gestures used in human computer interaction interfaces. *International Journal of Human-Computer Studies* 129 (2019), 74–94. https://doi.org/10.1016/j.ijhcs.2019.03.011

[162] Bryan Wang, Gang Li, and Yang Li. 2023. Enabling Conversational Interaction With Mobile UI Using Large Language Models. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 432, 17 pages. https://doi.org/10.1145/3544548.3580895

[163] Yun Wang, Zhitao Hou, Leixian Shen, Tongshuang Wu, Jiaqi Wang, He Huang, Haidong Zhang, and Dongmei Zhang. 2023. Towards Natural Language-Based Visualization Authoring. *IEEE Transactions on Visualization and Computer Graphics* 29, 1 (2023), 1222–1232. https://doi.org/10.1109/TVCG.2022.3209357

[164] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs.CL]

[165] Jacob O. Wobbrock, Meredith Ringel Morris, and Andrew D. Wilson. 2009. User-Defined Gestures for Surface Computing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '09)*. Association for Computing Machinery, New York, NY, USA, 1083–1092. https://doi.org/10.1145/1518701.1518866

[166] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's Transformers: State-of-the-Art Natural

Language Processing. *ArXiv Preprint ArXiv:1910.03771* (2019). https://arxiv.org/pdf/1910.03771

[167] Tongshuang Wu, Ellen Jiang, Aaron Donsbach, Jeff Gray, Alejandra Molina, Michael Terry, and Carrie J Cai. 2022. PromptChainer: Chaining Large Language Model Prompts through Visual Programming. In *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI EA '22)*. Association for Computing Machinery, New York, NY, USA, Article 359, 10 pages. https://doi.org/10.1145/3491101.3519729

[168] Tongshuang Wu, Marco Tulio Ribeiro, Jeffrey Heer, and Daniel S. Weld. 2021. Polyjuice: Generating Counterfactuals for Explaining, Evaluating, and Improving Models. arXiv:2101.00288 [cs.CL]

[169] Tongshuang Wu, Michael Terry, and Carrie Cai. 2022. AI Chains: Transparent and Controllable Human-AI Interaction by Chaining Large Language Model Prompts. In *CHI Conference on Human Factors in Computing Systems*. ACM. https://doi.org/10.1145/3491102.3517582

[170] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. arXiv:2210.03629 [cs.CL]

[171] Serena Yeung, Francesca Rinaldo, Jeffrey Jopling, Bingbin Liu, Rishab Mehra, N Lance Downing, Michelle Guo, Gabriel M Bianconi, Alexandre Alahi, Julia Lee, et al. 2019. A computer vision system for deep learning-based detection of patient mobilization activities in the ICU. *NPJ digital medicine* 2, 1 (2019), 1–5.

[172] Ming Yin, Jennifer Wortman Vaughan, and Hanna Wallach. 2019. Understanding the Effect of Accuracy on Trust in Machine Learning Models. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems* (Glasgow, Scotland Uk) *(CHI '19)*. Association for Computing Machinery, New York, NY, USA, 1–12. https://doi.org/10.1145/3290605.3300509

[173] Fisher Yu, Wenqi Xian, Yingying Chen, Fangchen Liu, Mike Liao, Vashisht Madhavan, and Trevor Darrell. 2018. Bdd100k: A diverse driving video database with scalable annotation tooling. *arXiv preprint arXiv:1805.04687* 2, 5 (2018), 6.

[174] J.D. Zamfirescu-Pereira, Richmond Y. Wong, Bjoern Hartmann, and Qian Yang. 2023. Why Johnny Can' t Prompt: How Non-AI Experts Try (and Fail) to Design LLM Prompts. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems* (Hamburg, Germany) *(CHI '23)*. Association for Computing Machinery, New York, NY, USA, Article 437, 21 pages. https://doi.org/10.1145/3544548.3581388

[175] Beiqi Zhang, Peng Liang, Xiyu Zhou, Aakash Ahmad, and Muhammad Waseem. 2023. Practices and Challenges of Using GitHub Copilot: An Empirical Study. In *Proceedings of the 35th International Conference on Software Engineering*

and Knowledge Engineering (SEKE2023). KSI Research Inc. https://doi.org/10.18293/seke2023-077

[176] Hao Zhang, Zi-Hao Bo, Jun-Hai Yong, and Feng Xu. 2019. InteractionFusion: Real-Time Reconstruction of Hand Poses and Deformable Objects in Hand-Object Interactions. *ACM Trans. Graph.* 38, 4, Article 48 (jul 2019), 11 pages. https://doi.org/10.1145/3306346.3322998

[177] Hao Zhang, Yuxiao Zhou, Yifei Tian, Jun-Hai Yong, and Feng Xu. 2021. Single Depth View Based Real-Time Reconstruction of Hand-Object Interactions. *ACM Trans. Graph.* 40, 3, Article 29 (jul 2021), 12 pages. https://doi.org/10.1145/3451341

[178] Jing Zhang, Xin Yu, Aixuan Li, Peipei Song, Bowen Liu, and Yuchao Dai. 2020. Weakly-Supervised Salient Object Detection via Scribble Annotations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

[179] Lei Zhang and Steve Oney. 2020. FlowMatic: An Immersive Authoring Tool for Creating Interactive Scenes in Virtual Reality. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 342–353. https://doi.org/10.1145/3379337.3415824

[180] Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. arXiv:2302.05543 [cs.CV]

[181] Pengchuan Zhang, Xiujun Li, Xiaowei Hu, Jianwei Yang, Lei Zhang, Lijuan Wang, Yejin Choi, and Jianfeng Gao. 2021. Vinvl: Revisiting visual representations in vision-language models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 5579–5588.

[182] Wencan Zhang, Mariella Dimiccoli, and Brian Y Lim. 2022. Debiased-CAM to Mitigate Image Perturbations with Faithful Visual Explanations of Machine Learning. In *CHI Conference on Human Factors in Computing Systems* (New Orleans, LA, USA) *(CHI '22)*. Association for Computing Machinery, New York, NY, USA, Article 182, 32 pages. https://doi.org/10.1145/3491102.3517522

[183] Yongqi Zhang, Cuong Nguyen, Rubaiat Habib Kazi, and Lap-Fai Yu. 2023. PoseVEC: Authoring Adaptive Pose-aware Effects Using Visual Programming and Demonstrations. In *ACM Symposium on User Interface Software and Technology*.

[184] Zhenyu Zhang, Xuxi Chen, Tianlong Chen, and Zhangyang Wang. 2021. Efficient lottery ticket finding: Less data is more. In *International Conference on Machine Learning*. PMLR, 12380–12390.

[185] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. 2016. Learning deep features for discriminative localization. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2921–2929.

[186] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, et al. 2022. Least-to-most prompting enables complex reasoning in large language models. *arXiv preprint arXiv:2205.10625* (2022).

[187] Zhongyi Zhou, Jing Jin, Vrushank Phadnis, Xiuxiu Yuan, Jun Jiang, Xun Qian, Jingtao Zhou, Yiyi Huang, Zheng Xu, Yinda Zhang, Kristen Wright, Jason Mayes, Mark Sherwood, Johnny Lee, Alex Olwal, David Kim, Ram Iyengar, Na Li, and Ruofei Du. 2023. InstructPipe: Building Visual Programming Pipelines with Human Instructions. arXiv:2312.09672 [cs.HC]

[188] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. 2018. Unet++: A nested u-net architecture for medical image segmentation. In *Deep learning in medical image analysis and multimodal learning for clinical decision support*. Springer, 3–11.

[189] Zhongyi Zhou and Koji Yatani. 2021. Enhancing Model Assessment in Vision-Based Interactive Machine Teaching through Real-Time Saliency Map Visualization. In *The Adjunct Publication of the 34th Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '21)*. Association for Computing Machinery, New York, NY, USA, 112–114. https://doi.org/10.1145/3474349.3480194

[190] Zhongyi Zhou and Koji Yatani. 2022. Gesture-Aware Interactive Machine Teaching with In-Situ Object Annotations. In *Proceedings of the 35th Annual ACM Symposium on User Interface Software and Technology* (Bend, OR, USA) *(UIST '22)*. Association for Computing Machinery, New York, NY, USA, Article 27, 14 pages. https://doi.org/10.1145/3526113.3545648

[191] Xiaojin Zhu. 2015. Machine teaching: An inverse problem to machine learning and an approach toward optimal education. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 29.

[192] Xiaojin Zhu, Adish Singla, Sandra Zilles, and Anna N. Rafferty. 2018. An Overview of Machine Teaching. https://doi.org/10.48550/ARXIV.1801.05927

Appendix A

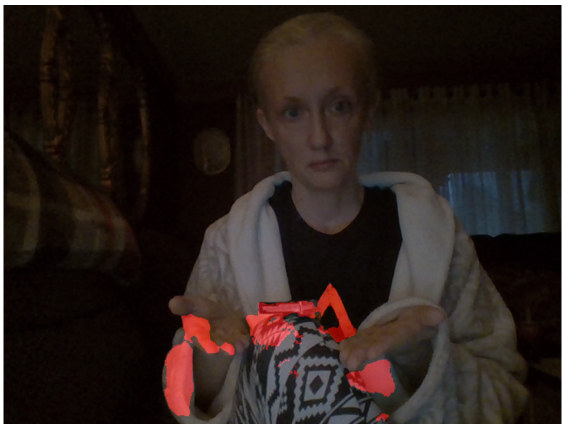
LookHere

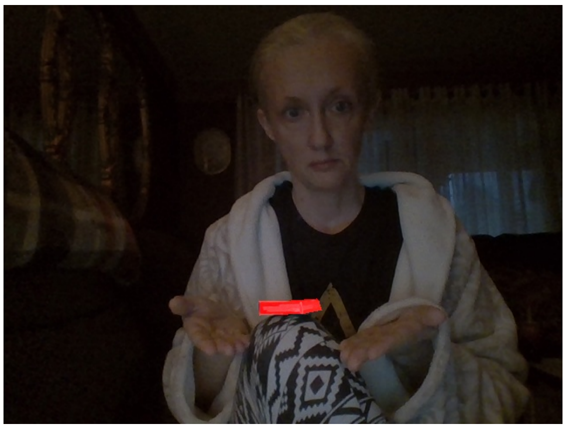
A.1 Implementation Details and Extra Technical Results

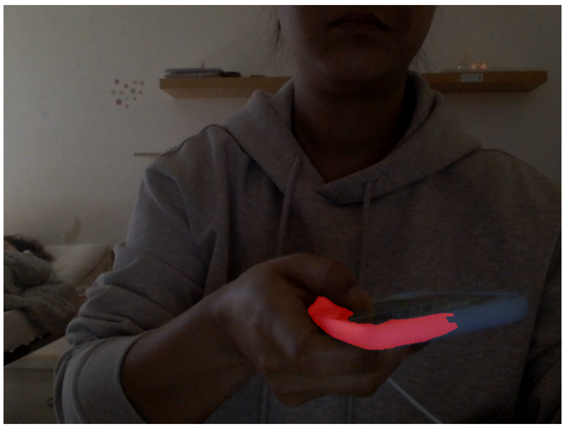
A.1.1 Configurations of Machine Learning Process in InstructPipe

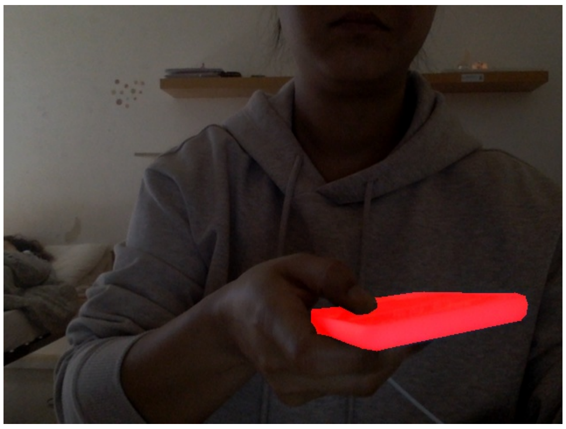
Each image captured in the front end is fixed at the size of 480×640 (height $\times$ width). We chose U-Net [129] with EfficientNet-b0 backbone [153] to be the machine learning model at the back end taught by users. During the training stage, InstructPipe uses Adam optimizer and performs fine-tuning on the model pretrained on ImageNet [40] for 50 epochs. The batch size is four, and the learning rate is 1e-4.

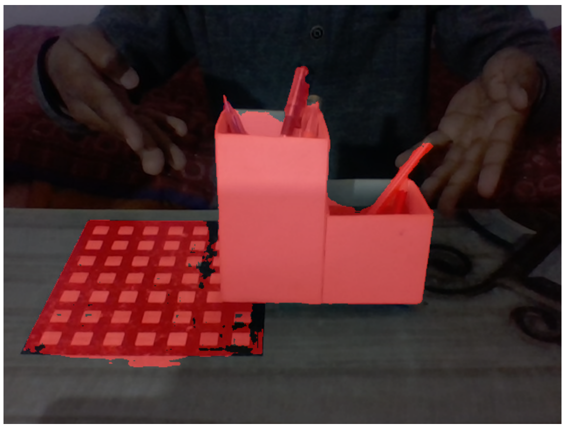
Note that we only use the encoder of the model for the *NaiїveIMT* condition (see details in Section 3.7.1) because there is no ground truth data of segmentation masks in this condition, which is necessary for training the decoder. We used CAM [185] to predict saliency maps using this classification model.


(a) Prediction.


(b) Ground truth.


(c) Prediction.


(d) Ground truth.


(e) Prediction.

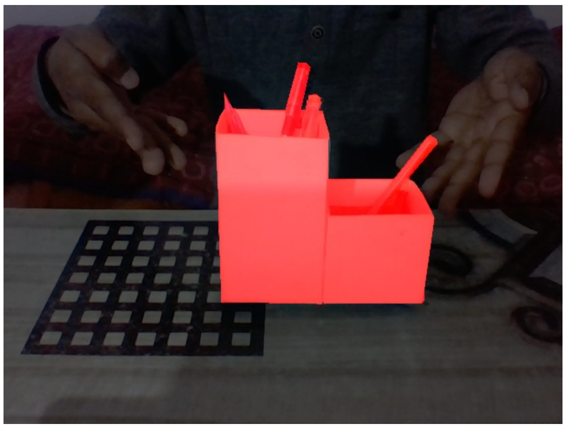

(f) Ground truth.

Figure A.1 Additional examples of object highlight prediction failure.

A.1.2 Configurations of Training Object Highlights

We fine-tuned the the network of object highlights pretrained on ImageNet using the Adam optimizer for 100 epochs. The learning rate maintains 1e-4 in the first 25 epochs,

Table A.1 Performance comparison of the four models for our object highlight.

	U-Net	UNet++	DeepLabV3	DeepLabV3+
mIoU	**0.718**	0.704	0.698	0.704
fps	**28.3**	24.7	22.5	27.8

and drops exponentially for the subsequent 50 epochs until it reach 1e-5 at epoch 75. It then maintains the learning rate of 1e-5 in the last 25 epochs. Because there is no validation set, we simply reported the accuracy based the model achieved at the end of the training without the early stop operation. We set the batch size to be 4.

We note that this training process is for object highlights in InstructPipe. The previous section explains how InstructPipe trains the model created by users (through demonstrations of objects).

A.1.3 Architecture Selections

We chose U-Net [129], UNet++ [188], DeepLabV3 [23] and DeepLabV3+ [24] for comparison because all of them are widely used different segmentation tasks and showed good performance. Table A.1 shows the results in which we compared *mIoU* and FPS of the trained models. The results show that U-Net was the most accurate as well as the fastest. Note that all of the architecture used EfficientNet-b0 as the backbone.

A.2 Object Highlight Prediction Failure

Figure A.1 shows additional examples in which our predictions of object highlights have large discrepancy with their ground truth.

Appendix B

InstructPipe

B.1 A Full List of 27 Nodes in InstructPipe

The following content shows 27 nodes InstructPipe covers in the generation process and their corresponding short description used in the Node Selector (§4.4.2):

B.1.1 Input Nodes

1. **live_camera**: Capture video stream through your device camera.
2. **input_image**: Select images to use as input to your pipeline. You can also upload your own images.
3. **input_text**: Add text to use as input to your pipeline.

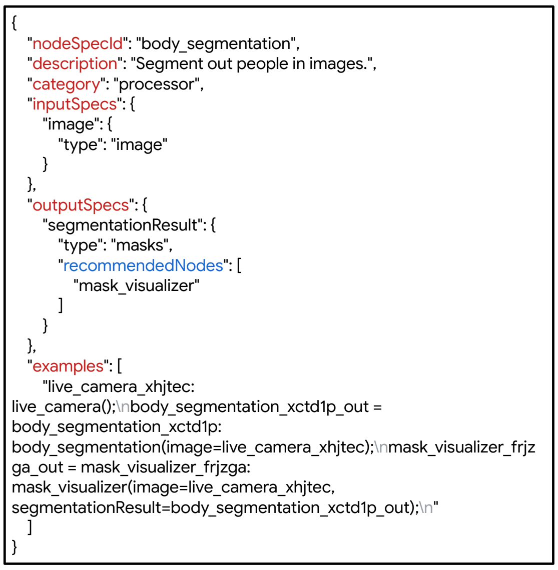
B.1.2 Output Nodes

1. **image_viewer**: View images.
2. **threed_photo**: Create a 3D photo effect from depthmap tensors.
3. **markdown_viewer**: Render Markdown strings into stylized HTML.

4. **html_viewer**: Show HTML content with styles

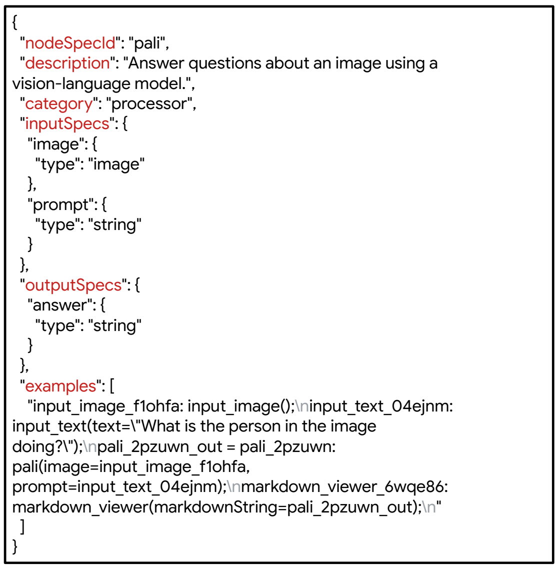
B.1.3 Processor Nodes

1. **google_search**: Use Google to search the web that returns a list of URLs based on a given keyword; usually selected with string_picker.
2. **body_segmentation**: Segment out people in images; usually selected with mask_visualizer.
3. **tensor_to_depthmap**: Display tensor data as a depth map.
4. **portrait_depth**: Generate a 3D depth map for an image; usually selected with tensor_to_depthmap, threed_photo.
5. **face_landmark**: Detect faces in images. Each face contains 468 keypoints; usually selected with landmark_visualizer, virtual_sticker.
6. **pose_landmark**: Generate body positional mappings for people detected in images; usually selected with landmark_visualizer.
7. **image_processor**: Process an image (crop, resize, shear, rotate, change brightness or contrast, add blur or noise).
8. **text_processor**: Reformat and combine multiple text inputs.
9. **mask_visualizer**: Visualize masks.
10. **string_picker**: Select one string from a list of strings; usually used with google_search.
11. **image_mixer**: Combine images and text into one output image. Requires two image inputs.
12. **virtual_sticker**: Use face landmarks data to overlay virtual stickers on images.



```
{
  "nodeSpecId": "body_segmentation",
  "description": "Segment out people in images.",
  "category": "processor",
  "inputSpecs": {
    "image": {
      "type": "image"
    }
  },
  "outputSpecs": {
    "segmentationResult": {
      "type": "masks",
      "recommendedNodes": [
        "mask_visualizer"
      ]
    }
  },
  "examples": [
    "live_camera_xhjtec:
live_camera();\nbody_segmentation_xctd1p_out =
body_segmentation_xctd1p:
body_segmentation(image=live_camera_xhjtec);\nmask_visualizer_frjz
ga_out = mask_visualizer_frjzga:
mask_visualizer(image=live_camera_xhjtec,
segmentationResult=body_segmentation_xctd1p_out);\n"
  ]
}
```

(a) Body segmentation



```
{
 "nodeSpecId": "pali",
 "description": "Answer questions about an image using a
vision-language model.",
 "category": "processor",
 "inputSpecs": {
  "image": {
   "type": "image"
  },
  "prompt": {
   "type": "string"
  }
 },
 "outputSpecs": {
  "answer": {
   "type": "string"
  }
 },
 "examples": [
  "input_image_f1ohfa: input_image();\ninput_text_04ejnm:
input_text(text=\"What is the person in the image
doing?\");\npali_2pzuwn_out = pali_2pzuwn:
pali(image=input_image_f1ohfa,
prompt=input_text_04ejnm);\nmarkdown_viewer_6wqe86:
markdown_viewer(markdownString=pali_2pzuwn_out);\n"
 ]
}
```

(b) PaLI

Figure B.1 Examples of node configuration used in Code Writer. The configuration is structured in a JSON format.

13. **palm_textgen**: Generate Text using a large language model.

14. **keywords_to_image**: Search for images by keywords.

15. **url_to_html**: Crawl the website by a given URL.

16. **image_to_text**: Extract text from an image using OCR service.

17. **pali**: Answer questions about an image using a vision-language model.

18. **palm_model**: Generate text using a large language model based on prompt and context.

19. **imagen**: Generate an image based on a text prompt.

20. **input_sheet**: Read string data from Google Sheets.

B.2 System Prompts Used in LLM Modules

Here we provide more details about the prompts we utilized in InstructPipe. The original txt files are also attached in the supplementary zip file.

B.2.1 Code Writer

The prompt in Code Writer is dynamic, which is dependent on the nodes chosen in Node Selector. Therefore, we cannot provide all the possible prompts in the supplementary materials. Here, we will focus on providing examples of two detailed node configurations utilized in InstructPipe. Figure 4.7 shows the structure of the prompt utilized in this LLM stage. Figure B.1 provides two examples of node configurations (*i.e.*, “Body segmentation” and “PaLI”) that InstructPipe may chose into the highlighted line(s). Each configuration includes keys of “nodeSpecId” (*i.e.*, node types), “description”, “category” and “examples”. For those nodes that support input and output edges, “inputSpecs” and “outputSpecs” specify the sockets and their corresponding valid data types. For example, the output socket name of “Body segmentation” is “segmentationResult”, and its data type is “masks”. Some nodes (*e.g.*, “Body segmentation”) include recommended node(s) (*e.g.*, “Mask visualizer” for “Body segmentation”), and our configuration also contains such information in the dictionary.

B.3 User Study Pipelines

Figure B.2 and Figure B.3 visualize two pipelines we required the participants to complete in our user study. Figure B.3 is a multimodal pipeline that allows participants to interact with AR effects in real time. Our technical evaluation shows that InstructPipe can generate this pipeline accurately: the averaged ratio of human interactions = 5.2%. Figure B.3 is a text-based pipeline that provides participants with

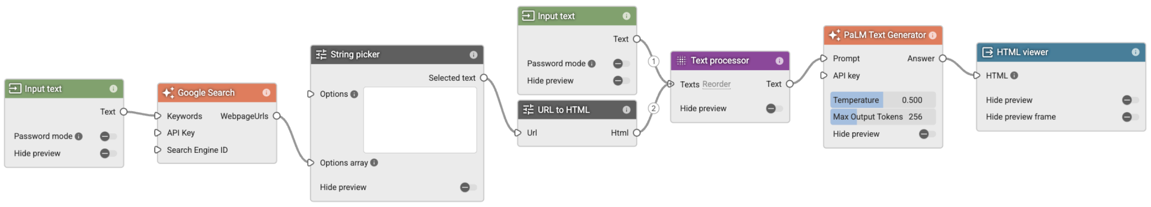


Figure B.2 Text-based pipeline. The "String picker" node provides users a drop-down menus to select one URL from a list of URLs returned by "Google Search". "PaLM Text Generator" is an LLM used to summarize the full HTML page.

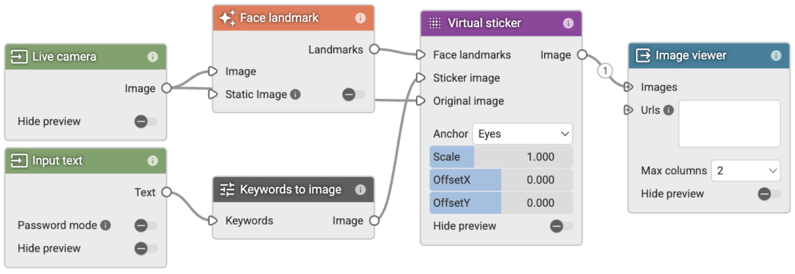


Figure B.3 Real-time multimodal pipeline. The "Keyword to image" node is used to search sunglasses image, and the "Virtual sticker" node anchors the sunglasses onto users' face.

a summary of the news searched from Google. The technical evaluation reveals that InstructPipe cannot generate this pipeline accurately: the averaged ratio of human interactions = 27.8%.

Note that even though InstructPipe may be able to complete the pipeline structure in Figure B.3 from users' instruction, we observed that participants still need to fine-tune their keywords to get an ideal pair of sunglasses. Additionally, the default anchor value is "Face top", so participants need to use the drop-down menu on the "Virtual sticker" node to change the value to "Eyes". This further motivates us to use the metric of "Time" in addition to the number of user interactions in our study.