國立臺灣大學電機資訊學院資訊工程學系

碩士論文

Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Taiwan University

Master's Thesis

優化手機影片摘要生成：運用生成式圖片轉文字模型與 AITW 資料集

Enhancing Mobile Video Captioning: Utilizing Generative Image-to-text Transformers with AITW Dataset

蔡博揚

Po-Yang Tsai

指導教授: 廖世偉 博士

Advisor: Shi-Wei Liao Ph.D.

中華民國 113 年 6 月

June, 2024

doi:10.6342/NTU202401151

國立臺灣大學碩士學位論文
口試委員會審定書
MASTER'S THESIS ACCEPTANCE CERTIFICATE
NATIONAL TAIWAN UNIVERSITY

優化手機影片摘要生成：運用生成式圖片轉文字模型與 AITW 資料集

Enhancing Mobile Video Captioning: Utilizing Generative Image-to-text Transformers with AITW Dataset

本論文係蔡博揚君（學號 R10922A15）在國立臺灣大學資訊工程學系人工智慧碩士班完成之碩士學位論文，於民國 113 年 6 月 8 日承下列考試委員審查通過及口試及格，特此證明。

The undersigned, appointed by the Master Program of Artificial Intelligence offered by the Department of Computer Science and Information Engineering on 8 June 2024 have examined a Master's thesis entitled above presented by TSAI, PO-YANG (student ID: R10922A15) candidate and hereby certify that it is worthy of acceptance.

口試委員 Oral examination committee:

庾世雋	傅楸善	盧瑞山
（指導教授 Advisor）		

系（所）主管 Director: 陳祝嵩

Acknowledgements

經歷了許多困難，終於走到了這一步，缺少了任何人的幫助都無法完成碩士論文。首先要謝謝廖世偉教授的協助，教授的專業領域知識給予我論文方向上的教導，實驗室提供了我做實驗所需要的機器，協助我能順利的完成論文。接著要感謝 UR Program 的 Vincent 和 Yu-Siang，幫助我決定論文的主題並給於適當的建議，非常感謝團隊的協助。不能忘記 Google CV 組可愛的團隊們，大家一起努力前行。最後要謝謝適時和我討論的同學們，還有過去的自己，完成了這篇論文。在此送上最真誠的謝意。

doi:10.6342/NTU202401151

摘要

我們提供一個有效的方法，使用生成式圖片和文字的轉換器模型來為手機影片生成摘要，並訓練在 Android in the Wild 資料集。目前手機錄影都是由人工檢視做摘要，我們使用機器學習直接將視覺的資訊轉成文字。本論文使用的方法包含資料的前處理及三種微調策略來改善模型，包含雙學習率、增加時間序詞嵌入，以及可變輸入圖片解析度。實驗結果顯示微調方法明顯的提高了生成摘要的準確度，並且凸顯視覺語言模型，在手機應用程式中自動化問題報告過程的潛力，大量的減少人力與時間的同時提供高準確度的摘要。

關鍵字：影片摘要生成、Android in the Wild、視覺語言模型、機器學習、微調

doi:10.6342/NTU202401151

Abstract

This paper introduces a novel approach for mobile video captioning using the Generative Image-to-text Transformer model, with the Android in the Wild dataset. The process of summarizing mobile records is traditionally reliant on manual review. We address this challenge by employing machine learning techniques to convert visual information directly into texts. The methodology includes data preprocessing and three fine-tuning strategies, such as dual learning rates, increased temporal embeddings, and variable input image resolutions, to enhance the model's performance. Comprehensive experimentation shows that these fine-tuning techniques significantly improve the accuracy of generated captions. The results highlight the potential of vision-language models to automate the problem-reporting process in mobile applications, significantly reducing time and labor while ensuring high accuracy.

Keywords: Video Captioning, Android in the Wild, Vision-Language Model, Machine Learning, Fine-Tuning

doi:10.6342/NTU202401151

Contents

doi:10.6342/NTU202401151

doi:10.6342/NTU202401151

List of Figures

doi:10.6342/NTU202401151

doi:10.6342/NTU202401151

List of Tables

doi:10.6342/NTU202401151

Chapter 1 Introduction

With the advancements in technology, smartphones have been pervasive, profoundly impacting how we communicate, work, and entertain ourselves. As users increasingly rely on these devices, the number of problems has also increased, highlighting a pressing need for efficient problem-reporting mechanisms. Users typically record screens to demonstrate the problems and submit videos to developers. Manually reviewing the video, on the other hand, is not only time-consuming but also requires a large amount of human resources. To tackle this problem, we adopt an established model to transform mobile recordings into natural language captions. This approach aims to reduce the time and labor involved in manual video reviews while providing a concise summary of reported issues. Machine learning has advanced significant breakthroughs across various areas, especially in vision and language domains [6, 7, 19, 24, 28, 31]. Based on these fundamental advances, vision-language models [1, 13, 24, 35] have emerged as prominent tools to cope with more complex challenges, such as image captioning [17] and visual question answering [8]. Nevertheless, as machine learning models' size increases, training large models from scratch requires enormous computational time and resources. Consequently, it is common practice for researchers to release pre-trained model weights alongside new models. Utilizing pre-trained model weights not only facilitates transfer learning but also enhances performance due to large and diverse datasets [11, 25]. On the other hand, nu-

doi:10.6342/NTU202401151

merous datasets [5, 14, 33] pertain to mobile application screenshots, but there remains a deficiency in datasets comprising mobile screen recordings. Recently, Google launched Android in the Wild (AITW)[27], addressing the demand for extensive datasets that include mobile recordings. AITW offers an extensive collection of mobile recordings for research and development purposes. For the video captioning task, we use the Generative Image-to-text Transformer (GIT), introduced by Google in 2022, as our pre-trained model, along with AITW as our dataset. The main reason we chose GIT is that its simple structure aligns perfectly with AITW's visual data. GIT only consists of an image encoder and a text decoder, and it has achieved state-of-the-art performance on various tasks without other input features like audio or optical character recognition. The model's adeptness at distilling visual data into textual descriptions without auxiliary inputs makes it well-suited for our objective. We have implemented various fine-tuning strategies, such as dual learning rates, adjusting the number of temporal embeddings, and variable resolution inputs.

Here is a summary of this paper's primary contribution:

- In the video captioning domain, we present an original application of high-level instructions and mobile screenshots.
- We conducted a comprehensive analysis of the AITW dataset and refined it.
- We explore three fine-tuning methods to enhance the model performance.

doi:10.6342/NTU202401151

Chapter 2 Related Work

In the 2010s, VGG [29] and ResNet [10] respectively introduced deeper networks and residual connections, leading to considerable improvements in image processing tasks. The success of VGG encouraged the development of larger models. After that, ResNet's use of residual blocks effectively fixed the vanishing gradient problem and made it possible to build deeper neural network architectures. These two models aided the groundwork for image processing techniques and influenced later vision-language model development. In 2015, the initial development of the vision-language deep learning models combined a convolutional neural network for image processing with recurrent neural network (RNN) or long short-term memory network (LSTM) [32] for text processing. This model structure first employed an image encoder and a text decoder. Recent vision-language models have adopted this architecture, resulting in significant improvements. In 2017, Google presented the attention mechanisms in Transformer [31] and also adopted the residual blocks from ResNet. Transformer replaced RNN and has since contributed to both the vision and language fields.

doi:10.6342/NTU202401151

2.1 Vision Language Model

A type of deep learning model known as the vision-language model extracts visual information from images or videos and produces texts that are readable texts. by humans. The application includes image captioning, visual question answering, and other tasks based on questions about visual content. With recent advances in both the vision and language domains, vision-language models have made significant progress.

In 2020, Vision Transformer (ViT) [7] applied the attention architecture, originally designed for natural language processing, in the vision domain. ViT introduced an algorithm that splits an image into multiple **patches**. Transformer blocks view these patches as words and process them for training. ViT eventually became the foundation of modern vision-language models [9, 19, 30]. In 2021, OpenAI presented Contrastive Language–Image Pretraining (CLIP) [24], which bridges the gap between computer vision and natural language understanding. CLIP uses contrastive learning [4] and leverages a large amount of image-text data pairs from the internet for pre-training. CLIP functions by creating a high-dimensional feature mapping that aligns image-text pairs with related concepts. Researchers can easily adapt the pre-trained image-to-text module for various tasks. In 2022, Microsoft proposed the Generative Image-to-text Transformer (GIT) [35]. It consists of an image encoder and a text decoder. The image component is a CLIP module while the text component is a Transformer module. Despite its relatively simple and straightforward structure, GIT achieved state-of-the-art performance across numerous datasets, such as COCO [17] and MSVD [3], as well as tasks, such as image captioning and video captioning. In this paper, we chose GIT as our base model because of its ability to transfer learning and effectiveness in translating images into languages.

doi:10.6342/NTU202401151

2.2 Mobile Video Captioning

Datasets comprising mobile screenshots, such as [5, 14, 33], have facilitated advancements in analyzing smartphone user interfaces (UI) and behaviors [1, 15]. Although these datasets provide invaluable insights into UI designs or descriptive text, there remains a gap in the field. Specifically, there is a deficiency in datasets that offer a large volume of data accompanied by general prompts. The gap underscores the pressing need for a dataset, specifically for video captioning tasks, that connects user interactions with commands. To address this critical gap, Google introduced Android in the Wild (AITW) [27] in 2023. This dataset is designed for training device-control models that can interpret natural language commands or screenshots and perform the corresponding actions. Before the advent of AITW, existing datasets (e.g. UIBert [2]) were limited to step-by-step commands specifying particular UI elements. AITW, in contrast, employs concise commands that describe high-level goals, marking a significant shift away from the limitations of previous datasets. You can see this when you ask a question (e.g. ”When is my next meeting?”) or issue a command (e.g.”Check Android version”). AITW’s extensive volume, consisting of up to 700,000 records with a total of 5,600,000 screenshots, is another notable contribution. The abundance of data enhances the model’s ability to understand user commands. This collection not only provides comprehensive user interactions but also enriches each interaction with detailed contextual data. Each screenshot contains several fields: **episode-id**, **episode-length**, **goal-info**, **action-type**, **annotations-positions** and other digital information. Because we want to build a model capable of analyzing users’ mobile screen records, our focus is particularly on **goal-info**, which serves as the high-level prompt describing the sequence of screenshots. We expect our model to understand

doi:10.6342/NTU202401151

the screen records and provide explanations for them. Therefore, we utilized the **goal-info** data as ground truth and screenshots as inputs to train our video captioning model.

doi:10.6342/NTU202401151

Chapter 3 Methodology

Our objective is to develop a model capable of converting video inputs into textual prompts. We choose the GIT model [35] as the base model and fine-tune the model on the AITW dataset [27]. This process involves an initial analysis and preprocessing of the AITW dataset, followed by strategies to enhance the fine-tuning process. We chose the GIT model primarily because of its straightforward architecture, which is ideal for the visual data in the AITW dataset. Without relying on additional input features such as audio or optical character recognition, the GIT model, which consists only of an image encoder and a text decoder, has shown state-of-the-art performance across a variety of tasks.

3.1 Model Structure

The GIT model structure is composed of an image encoder and a text decoder as shown in Figure 3.1. The image encoder takes raw images as input and generates encoded features. They are then combined with temporal embeddings and fed into a text decoder to produce the text description. The image encoder is a CLIP-based [24] encoder and has already been pre-trained on contrastive learning tasks. This is because recent studies show that contrastive learning models work very well [4, 36]. The text decoder is a transformer module [31], which consists of multiple transformer blocks. Each block

doi:10.6342/NTU202401151

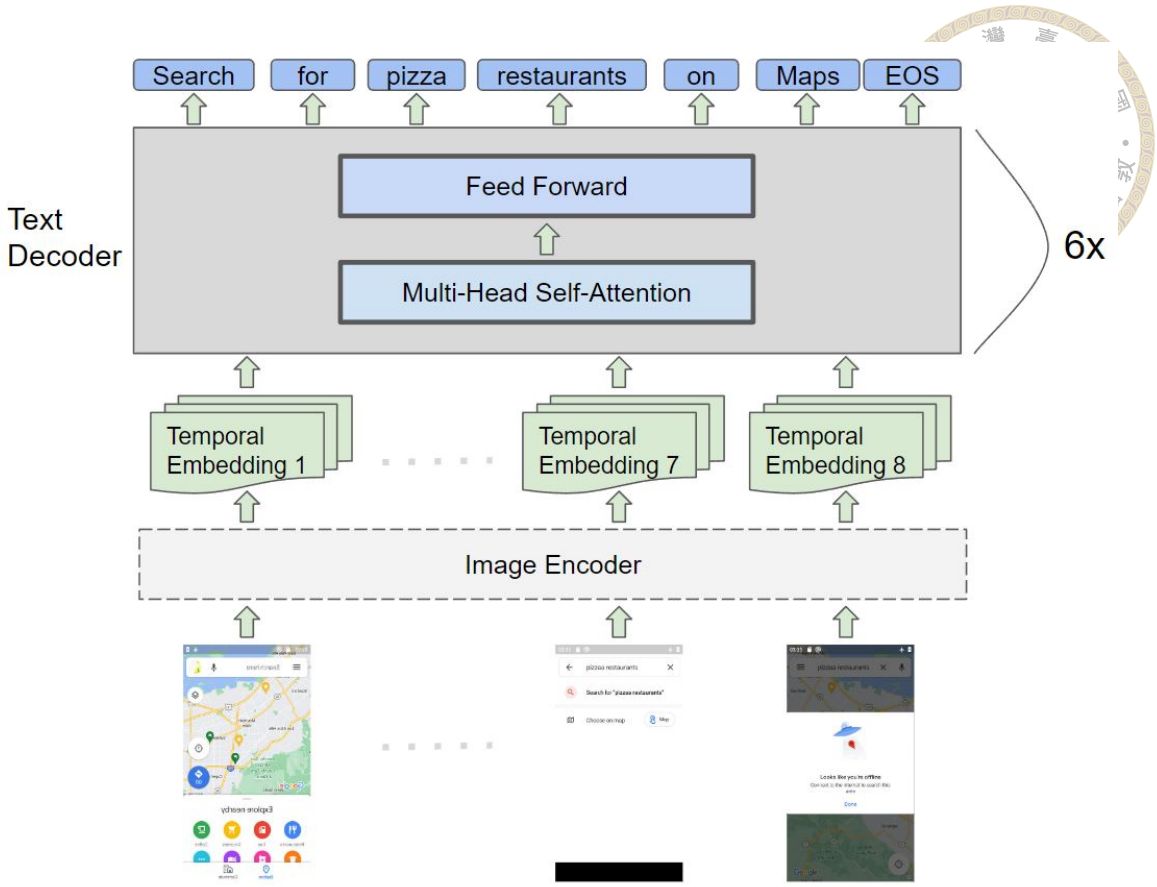


Figure 3.1: The GIT model architecture is comprised of an image encoder and a text decoder. Input screenshots are first encoded, added with temporal embeddings, and then concatenated before being sent into the text decoder.

is composed of one self-attention layer and one feed-forward layer and is randomly initialized, in accordance with the experiment findings [34]. Moreover, the functionality of temporal embeddings [26] is to capture time-dependent patterns. The original paper configures the GIT model with six temporal embeddings, meaning it can accepts a maximum of six input images.

3.2 Data Preprocessing

For our study, we used Android in the Wild (AITW) [27], which was introduced by Google. Before delving into the training details, we provide a brief overview of the data format. AITW encompasses up to 700,000 episodes with a total of 5,600,000 screenshots.

doi:10.6342/NTU202401151

The dataset is divided into five sections: **{GoogleApps, Install, WebShopping, General, Single}**. It is important to note that **Single**, which includes only single-step tasks, is excluded from our study because it contradicts our requirement for video data. Descriptions of the categories are shown in Table 3.1.

Table 3.1: Categories of AITW

Name	**Description**
GOOGLE APPS	Google apps tasks
INSTALL	App installation and login tasks
WEB SHOPPING	Web shopping tasks
GENERAL	Misc web/app tasks

Each episode in the dataset represents a sequence of screenshots illustrating an instruction's steps. While each screenshot contains fifteen fields, only some fields are relevant to our topic, as detailed below. The key points are **images** and **goal_info**, which serve as training inputs and ground truth.

episode_id: The episode's unique identifier from which the example is taken.

episode_length: The total count of steps in the episode.

goal_info: The natural language instruction demonstrated by the episode.

image/channels, image/height, image/width: The number of channels, height, and width of the screenshot.

step_id: The zero-indexed step number of the example within the episode (e.g., if step_id is 4, this is the fifth step of the episode).

During our experiments, we observed that some episodes within the dataset were missing screenshots. The number of missing ones varied, with some episodes lacking

doi:10.6342/NTU202401151

one or two screenshots, while others were missing up to half of their screenshots. According to our observations, the missing screenshots typically occur in the latter part of the episodes, which is a factor that could potentially impact the training performance of our model because the latter part shows the final steps of the tasks. To assess the impact of these missing episodes on model accuracy, we conducted a trial using a subset of the dataset. The findings confirmed our hypothesis that missing screenshots adversely affect performance. Section 4.2 provides additional information. We discovered that approximately 100,000 episodes were missing from the dataset, leaving about 500,000 episodes intact. These 500,000 episodes serve as the setting for the remaining experiments.

3.3 Scheme

In this section, we cover three fine-tuning methods.

3.3.1 Dual Learning Rates

We have implemented a dual learnings rate in our model, inspired by referenced papers [18, 35]. Dual learning rates refers to training different modules at different learning rates. Transfer learning typically freezes the pre-trained model components while training the newly added modules. For our analysis, the learning rate of the text decoder and the temporal embeddings are five times that of the image encoder. The image encoder is not frozen but is trained at a smaller pace because it needs to adapt to features not present in its pre-training data, such as mobile screenshots. On the other hand, the new text decoder and temporal embeddings undergo training at a faster pace.

doi:10.6342/NTU202401151

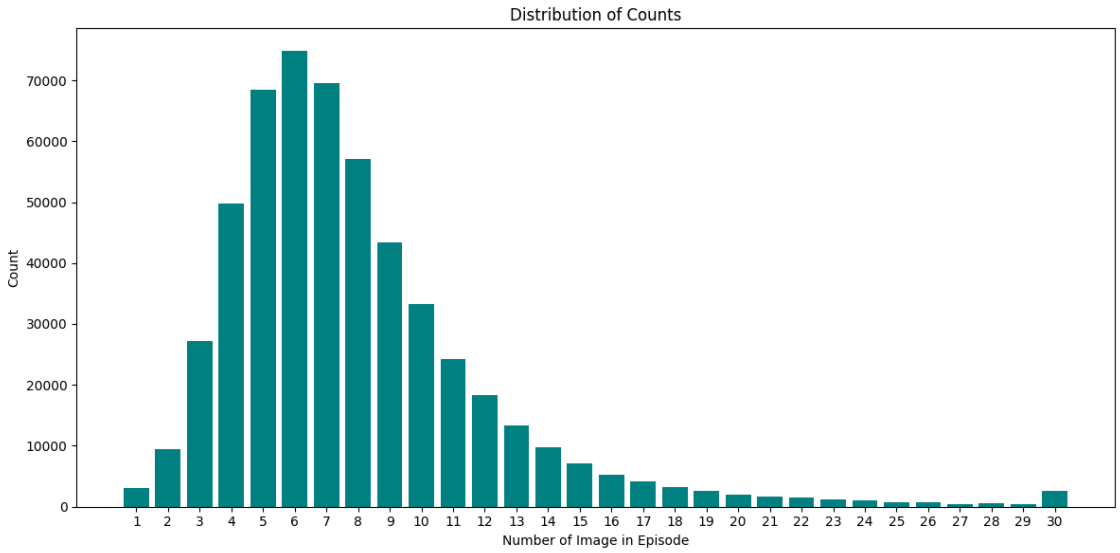


Figure 3.2: Distribution of numbers of images per episode. Note that any count exceeding 30 was reported as 30.

3.3.2 Number of Temporal Embeddings

We conducted tests using different numbers of temporal embeddings to determine an optimal configuration. Designed for image-related tasks such as image captioning and visual question answering, the original GIT model treats multiple images concatenated as a video when adapted to the video domain. Curious about the rationale behind using six images, we reached out to the authors. Their response indicated that the choice of six images was arbitrary and lacking in specific significance. Given this insight, we decided to select the number of images based on statistical analysis, aiming to find the most effective configuration for our video domain adaptation. According to statistics from AITW, both the median and average number of screenshots per episode are 8, but the most frequently occurring number of screenshots is 6, as shown in Figure 3.2. As a result, we decided to set the number of temporal embeddings at 6 and 8 for our analysis. Note that we reported any count above 30 was reported as 30 in our data presentation.

doi:10.6342/NTU202401151

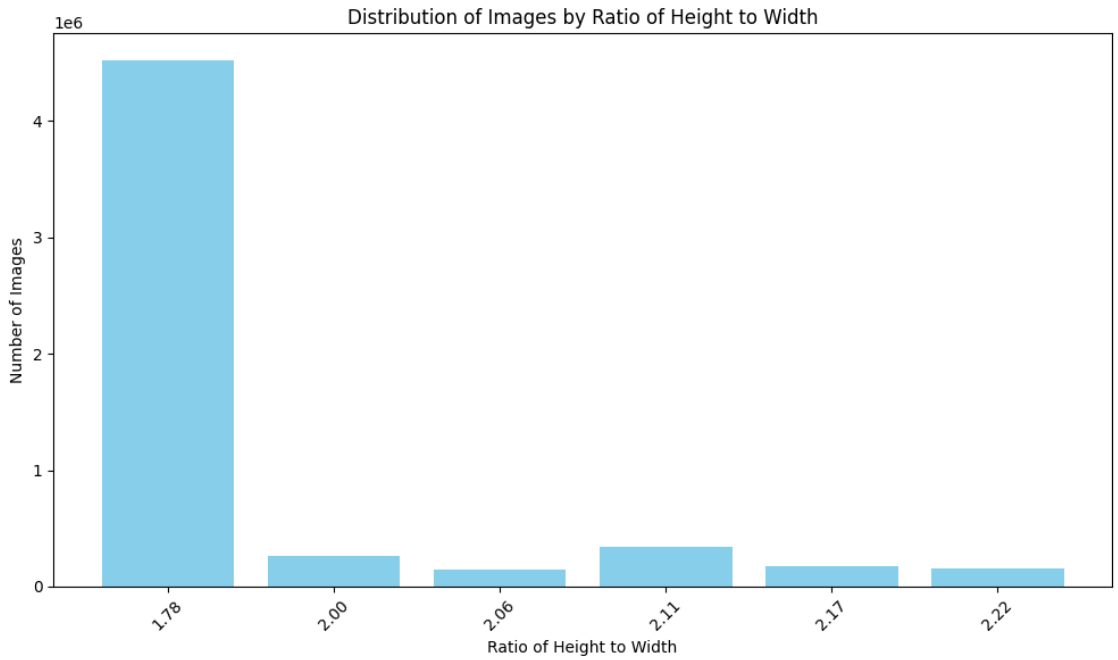


Figure 3.3: Distribution of images by ratio of height to width

3.3.3 Input Image Ratio

For the screenshots, we have implemented **variable resolution inputs**. The method of variable resolution inputs was introduced by Google in the paper: "Pix2Struct: Screenshot Parsing as Pretraining for Visual Language Understanding" [12]. The image encoder scales the input image to a predefined resolution, which is 224 by 224 in our case. This process would distort the image input resolution because mobile screens are usually rectangles instead of squares, thus affecting the recognition of documents, UIs, and icons on mobiles. To address this issue, we then show the steps to find the most appropriate resolution and the strategy to divide patches. The pre-trained model divides images into patches of 16x16 pixels. When applied to a standard image size of 224x224 pixels, this results in an arrangement of 14x14 patches, totaling 196 squares. To better accommodate the aspect ratios of our dataset's screenshots, we sought a configuration as close as possible to this 196-patch setup. We first counted the aspect ratio of all screenshots; the result is shown in

doi:10.6342/NTU202401151

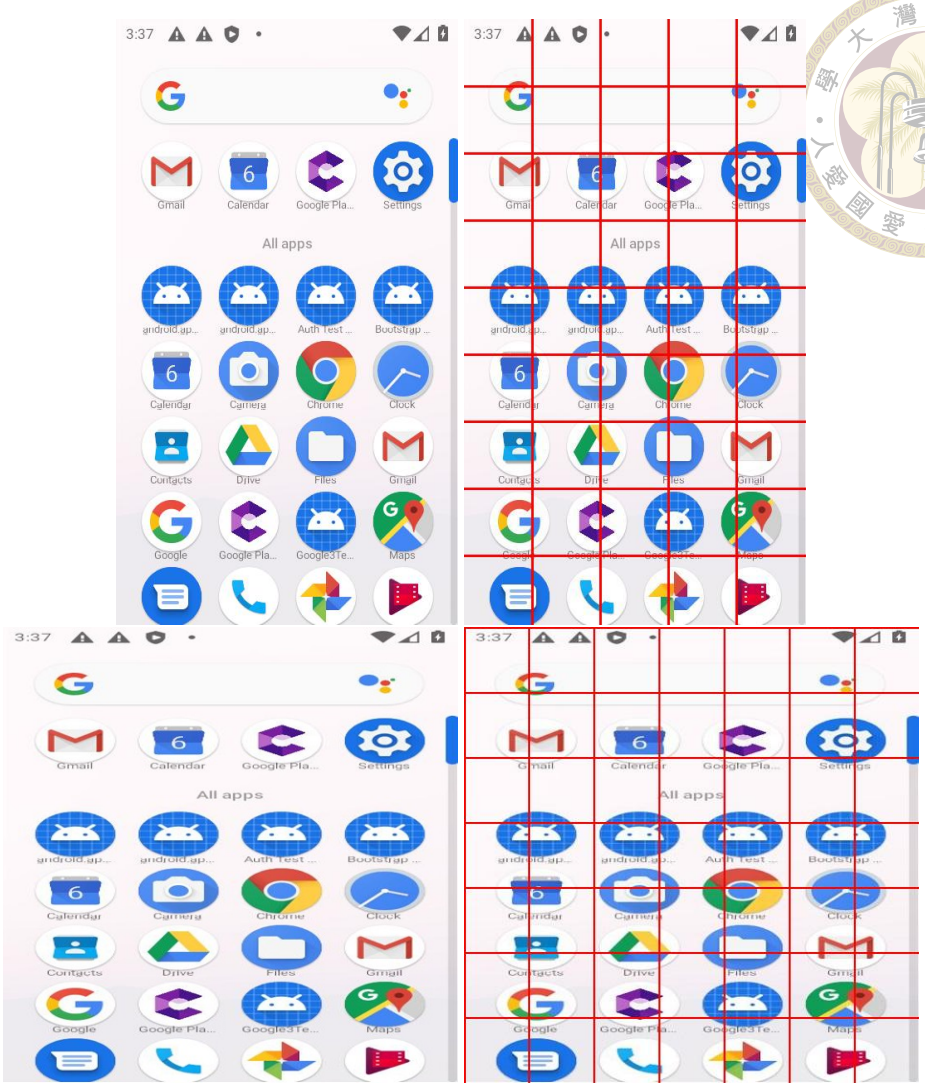


Figure 3.4: The examples demonstrate different approaches to image resizing and patch division. The upper figures are original figure and 9 by 5 grid figure. The lower figures are resized square figure and 7 by 7 grid figure.

Figure 3.3 We found that most of the images' ratios are 1.78, which is close to 1.8 or 9:5. Other images have ratios equal to or greater than 2. Note that 9 multiplied by 5 equals 45, which is the closest number to 49. By doubling the 9x5 ratio, we get 18 by 10 patches, resulting in 180 patches, a close approximation to 196. We adjusted the input resolution to 288x160 pixels by multiplying the number of patches (18 and 10) by the patch size of 16. This resolution accommodates the aspect ratios prevalent in our dataset while closely aligning with the encoder's capabilities. In Figure 3.4, we present the example figures.

doi:10.6342/NTU202401151

Chapter 4 Evaluation

4.1 Experiment Setting

We conducted all experiments on an Nvidia GeForce RTX 4090 GPU. We employed the Android in the Wild [27] dataset, allocating 80% for training and 20% for validation. Given the GPU's memory constraints, we chose the smaller GIT-Base model from the two versions proposed in the GIT model [35]. The image encoder was pre-trained [36] while the text decoder was randomly initialized with six transformer blocks. The hidden dimension was 768, similar to BERT [6], and the model had 0.7 billion model parameters. We utilized the Pytorch [23] framework. The batch size is 32. Under the standard settings, including eight temporal embeddings and variable resolution inputs, the learning rate was 5e-5. In the dual learning rate scenario, the learning rate of the image encoder was 1e-5 while the text decoder and the temporal embeddings were 5e-5. The training utilized the AdamW optimizer [20] with a weight decay of 1e-5 and a cosine decay to zero, including a warm-up phase. Each configuration ran for three epochs. To assess the model's performance, we applied the BLEU-4, CIDEr, and ROUGE-L metrics[16, 21, 22], which are commonly used in language generation tasks.

One critical issue was how to select six input screenshots for an episode. At first, we randomly chose six screenshots, but soon realized that earlier screenshots were often irrel-

doi:10.6342/NTU202401151

evant to the task, as shown in Figure A.1. After reviewing the original paper, we noticed they reset the environment to a random starting screen for each episode. Consequently, we selected the last six screenshots of an episode as our training inputs.

Another issue was that we found that the AITW dataset missed some of the screenshots in some episodes when we compared the generated captions to the screenshots. The detailed examples can be found in Figure A.2 and A.3. Therefore, we deleted these episodes, and Table 4.1 displays the number of remaining episodes after the deletion of missing episodes. To verify the impact of this deletion, we have trained the standard setting on a subset of AITW. The subset represented one-eightieth of the data in AITW, reducing the training time. We compared performance between the original and modified subsets using different learning rates. The best results were obtained at 5e-5, as shown in Table 4.2. At optimal learning rates like 5e-5 and 2.5e-5, the model performed slightly better on the modified subset. Notably, for other learning rates, the performance differed significantly. For subsequent experiments, we utilized the modified AITW dataset.

Table 4.1: The numbers of deleted and remaining episodes.

Name	Deleted	Remaining (Episodes)
GOOGLE APPS	88,406	537,136
INSTALL	5,109	20,651
WEB SHOPPING	1,176	26,885
GENERAL	3,636	5,840

4.2 Result

We evaluated the proposed methods, dual learning rate, eight temporal embeddings, and variable resolution inputs on the AITW dataset. Table 4.3 shows the results. Since we could not find any relevant reference papers, we established the standard setting as the

doi:10.6342/NTU202401151

Table 4.2: The scores of normal and modified subset of AITW.

Learning Rate	BLEU-4	CIDEr	ROUGE-L
Original Subset			
7.5e-5	49.5	41.0	60.3
5.0e-5	**60.5**	**50.8**	**67.9**
2.5e-5	59.5	50.0	67.4
1.0e-5	53.1	43.7	62.7
Modified Subset			
7.5e-5	55.5	45.2	64.0
5.0e-5	**61.1**	**50.1**	**68.7**
2.5e-5	60.2	49.9	68.6
1.0e-5	58.5	47.7	67.1

baseline.

Table 4.3: The scores from applying a single method to the AITW dataset. For brevity, we denote dual learning rate, variable resolution inputs, and eight temporal embeddings as ” 2 LR”, ”Pix” and ”8 Img” respectively.

Method	BLEU-4	CIDEr	ROUGE-L
Standard	64.6	57.1	72.7
2 LR	66.7	58.5	73.8
Pix	64.3	56.6	72.4
8 Img	**69.5**	**58.4**	**74.3**

The *dual learning rate* setup, with 1e-5 for the image encoder and 5e-5 for the temporal embeddings and the text decoder. The results suggest that different modules require specific learning rates to achieve better performance. A lower learning rate is suitable for the pre-trained module. The text decoder with a larger learning rate processes relatively simple sentence templates from the AITW dataset effectively. In contrast, the image encoder deals with the more complex task of recognizing patterns in mobile screenshots, which requires a finer adjustment.

The *eight temporal embeddings* significantly improved the BLEU-4 score and slightly increased the CIDEr and ROUGE-L scores compared to the baseline, achieving the best scores. We set the number of temporal embeddings from six to eight depending on the

doi:10.6342/NTU202401151

average number of screenshots per episode. The possible reason why the BLEU-4 score getting better is that the BLEU-4 score emphasizes the correct order of texts, while CIDEr and ROUGE-L focus on the occurrence of texts in sentences. The increase in embeddings likely enhanced the model's ability to accurately construct the sentences by extracting more visual information from the episodes. The choice of frames was another possible factor for the higher performance on eight temporal embeddings. The current choice of the latter frames is heuristic and requires further improvement. We leave this as future work.

The *variable resolution inputs* lowered all scores relative to the baseline, indicating that this method was ineffective in our experiments. A possible reason is that the resolution changes did not improve the image encoder's performance or optimize the model weight utilization. The original input size was 224x224 pixels, totaling 50,176 pixels, while our method used 288x160, totaling 46,080 pixels. This trade-off did not yield any improvements. In the following section, we explore all combinations of our methods to evaluate their effectiveness.

4.3 Ablation Study

Table 4.4 analyzes the importance of each method on the AITW dataset. When applying the dual learning rate, the image encoder's learning rate was 1e-5 while the text decoders and the temporal embeddings' learning rates were 5e-5.

Eight temporal embeddings. This method's score strongly improves compared to the baseline and further enhances the outcomes when combined with variable resolution inputs and dual learning rates. The result shows that eight temporal embeddings is the most

doi:10.6342/NTU202401151

effective solution. **Dual learning rate.** Employing only the dual learning rate achieves modest results. It improves the scores for the other two methods as well. **Variable resolution inputs.** While this method alone is ineffective for this task, when combined with all methods, it achieves the highest scores among all metrics.

Table 4.4: The ablation study on the AITW dataset. We refer to dual learning rate, variable resolution inputs, and eight temporal embeddings as '2 LR', 'Pix', and '8 Img' respectively for simplicity.

Method	BLEU-4	CIDEr	ROUGE-L
Standard	64.6	57.1	72.7
2 LR	66.7	58.5	73.8
Pix	64.3	56.6	72.4
8 Img	69.5	58.4	74.3
2 LR + 8 Img	70.5	58.9	74.7
Pix + 8 Img	69.2	58.1	73.9
2 LR + Pix	66.1	58.1	73.8
2 LR + Pix + 8 Img	**70.7**	**59.5**	**74.9**

doi:10.6342/NTU202401151

Chapter 5 Conclusion

In this paper, we employ the GIT model to train on the AITW dataset, marking the first application of high-level instructions and mobile screenshots in the video captioning domain. We conduct a comprehensive analysis of the AITW dataset and refine it accordingly. Additionally, we explore three fine-tuning methods aimed at enhancing model performance. The adjustment of temporal embeddings yields the most significant improvement, whereas the dual learning rate method achieves a modest enhancement as well. The variable resolution input, however, has minimal impact. These findings provide a foundation for further research in mobile video captioning. We still have room for improvements, such as frame selection and additional module adjustments, which we leave for future work.

doi:10.6342/NTU202401151

References

[1] G. Baechler, S. Sunkara, M. Wang, F. Zubach, H. Mansoor, V. Etter, V. Cărbune, J. Lin, J. Chen, and A. Sharma. Screenai: A vision-language model for ui and infographics understanding. arXiv preprint arXiv:2402.04615, 2024.

[2] C. Bai, X. Zang, Y. Xu, S. Sunkara, A. Rastogi, J. Chen, and B. A. y Arcas. Uibert: Learning generic multimodal representations for ui understanding. In International Joint Conference on Artificial Intelligence, 2021.

[3] D. Chen and W. Dolan. Collecting highly parallel data for paraphrase evaluation. In D. Lin, Y. Matsumoto, and R. Mihalcea, editors, Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 190–200, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

[4] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In H. D. III and A. Singh, editors, Proceedings of the 37th International Conference on Machine Learning, volume 119 of Proceedings of Machine Learning Research, pages 1597–1607. PMLR, 13–18 Jul 2020.

[5] B. Deka, Z. Huang, C. Franzen, J. Hibschman, D. Afergan, Y. Li, J. Nichols, and

doi:10.6342/NTU202401151

R. Kumar. Rico: A mobile app dataset for building data-driven design applications. In Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology, UIST '17, page 845–854, New York, NY, USA, 2017. Association for Computing Machinery.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In J. Burstein, C. Doran, and T. Solorio, editors, Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[7] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. ICLR, 2021.

[8] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the v in vqa matter: Elevating the role of image understanding in visual question answering. In 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 6325–6334, Los Alamitos, CA, USA, jul 2017. IEEE Computer Society.

[9] K. He, X. Chen, S. Xie, Y. Li, P. Doll'ar, and R. B. Girshick. Masked autoencoders are scalable vision learners. 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 15979–15988, 2021.

[10] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CVPR, 2016.

doi:10.6342/NTU202401151

[11] D. Hendrycks, K. Lee, and M. Mazeika. Using pre-training can improve model robustness and uncertainty. In K. Chaudhuri and R. Salakhutdinov, editors, Proceedings of the 36th International Conference on Machine Learning, volume 97 of Proceedings of Machine Learning Research, pages 2712–2721. PMLR, 09–15 Jun 2019.

[12] K. Lee, M. Joshi, I. Turc, H. Hu, F. Liu, J. Eisenschlos, U. Khandelwal, P. Shaw, M.-W. Chang, and K. Toutanova. Pix2struct: Screenshot parsing as pretraining for visual language understanding. arXiv preprint arXiv:2210.03347, 2022.

[13] J. Li, D. Li, C. Xiong, and S. C. H. Hoi. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. In International Conference on Machine Learning, 2022.

[14] Y. Li, J. He, X. Zhou, Y. Zhang, and J. Baldridge. Mapping natural language instructions to mobile ui action sequences. arXiv preprint arXiv:2005.03776, 2020.

[15] Y.-P. Y. Lik-Hang Lee and P. Hui. Perceived user reachability in mobile uis using data analytics and machine learning. International Journal of Human–Computer Interaction, 0(0):1–24, 2024.

[16] C.-Y. Lin. ROUGE: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics.

[17] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In European Conference on Computer Vision, 2014.

doi:10.6342/NTU202401151

[18] E. Liner and R. Miikkulainen. Improving neural network learning through dual variable learning rates. In 2021 International Joint Conference on Neural Networks (IJCNN), pages 1–7, 2021.

[19] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pages 9992–10002, 2021.

[20] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In International Conference on Learning Representations, 2019.

[21] G. Oliveira dos Santos, E. L. Colombini, and S. Avila. CIDEr-R: Robust consensus-based image description evaluation. In W. Xu, A. Ritter, T. Baldwin, and A. Rahimi, editors, Proceedings of the Seventh Workshop on Noisy User-generated Text (W-NUT 2021), pages 351–360, Online, Nov. 2021. Association for Computational Linguistics.

[22] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. Bleu: a method for automatic evaluation of machine translation. In P. Isabelle, E. Charniak, and D. Lin, editors, Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics.

[23] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In Advances in

doi:10.6342/NTU202401151

Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc., 2019.

[24] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. arXiv preprint arXiv:2103.00020, 2021.

[25] A. Radford and K. Narasimhan. Improving language understanding by generative pre-training. In arxiv, 2018.

[26] V. Ramanathan, K. D. Tang, G. Mori, and L. Fei-Fei. Learning temporal embeddings for complex video analysis. 2015 IEEE International Conference on Computer Vision (ICCV), pages 4471–4479, 2015.

[27] C. Rawles, A. Li, D. Rodriguez, O. Riva, and T. P. Lillicrap. Androidinthewild: A large-scale dataset for android device control. In Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track, 2023.

[28] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 28. Curran Associates, Inc., 2015.

[29] K. Simonyan and A. Zisserman:. Very deep convolutional networks for large-scale image recognition. ICLR, 2015.

[30] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, and H. Jegou. Training data-efficient image transformers and distillation through attention. In M. Meila

doi:10.6342/NTU202401151

and T. Zhang, editors, Proceedings of the 38th International Conference on Machine Learning, volume 139 of Proceedings of Machine Learning Research, pages 10347–10357. PMLR, 18–24 Jul 2021.

[31] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems, volume 30. Curran Associates, Inc., 2017.

[32] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan. Show and tell: A neural image caption generator. In CVPR, pages 3156–3164. IEEE Computer Society, 2015.

[33] B. Wang, G. Li, X. Zhou, Z. Chen, T. Grossman, and Y. Li. Screen2Words: Automatic Mobile UI Summarization With Multimodal Learning. In The 34th Annual ACM Symposium on User Interface Software and Technology. ACM, Oct. 2021.

[34] J. Wang, X. Hu, P. Zhang, X. Li, L. Wang, L. Zhang, J. Gao, and Z. Liu. Minivlm: A smaller and faster vision-language model. ArXiv, abs/2012.06946, 2020.

[35] J. Wang, Z. Yang, X. Hu, L. Li, K. Lin, Z. Gan, Z. Liu, C. Liu, and L. Wang. Git: A generative image-to-text transformer for vision and language. arXiv preprint arXiv:2205.14100, 2022.

[36] L. Yuan, D. Chen, Y.-L. Chen, N. C. F. Codella, X. Dai, J. Gao, H. Hu, X. Huang, B. Li, C. Li, C. Liu, M. Liu, Z. Liu, Y. Lu, Y. Shi, L. Wang, J. Wang, B. Xiao, Z. Xiao, J. Yang, M. Zeng, L. Zhou, and P. Zhang. Florence: A new foundation model for computer vision. ArXiv, abs/2111.11432, 2021.

doi:10.6342/NTU202401151

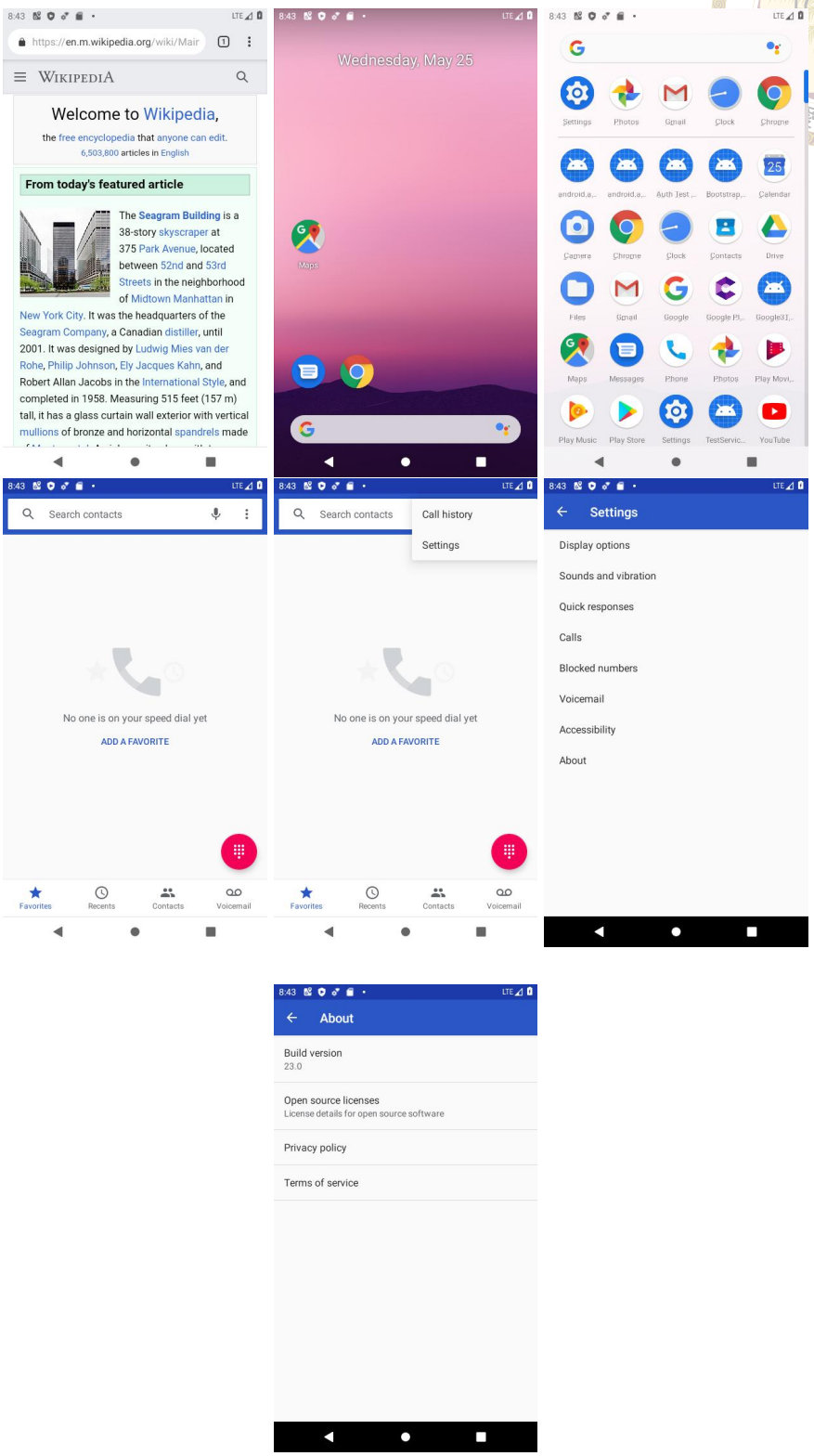
Appendix A — Examples

We provide several illustrative examples in this chapter. As described in the AITW paper [27], each episode starts from a randomly selected screen. Upon observing many contiguous episodes, we discovered that an episode might begin where another has ended. Therefore, the starting screenshot is irrelevant to the task. Examples of random start screenshots are depicted in Figure A.1.

One critical issue may affect the training is the missing episodes. When examining the generated captions alongside the screenshots, we found that our model sometimes generated correct captions but the screenshots contradicted the goal_info. This difference may be attributed to some episodes lacking the latter screenshots, which contains the steps displaying operations. Instances of episodes missing screenshots are shown in Figure A.2 and A.3.

We give one example for each of four categories in Figure A.4, A.5, A.6 and A.7. The generated captions may be partially the same to the goal_info in these examples. We would discuss the possible reasons and further solutions.

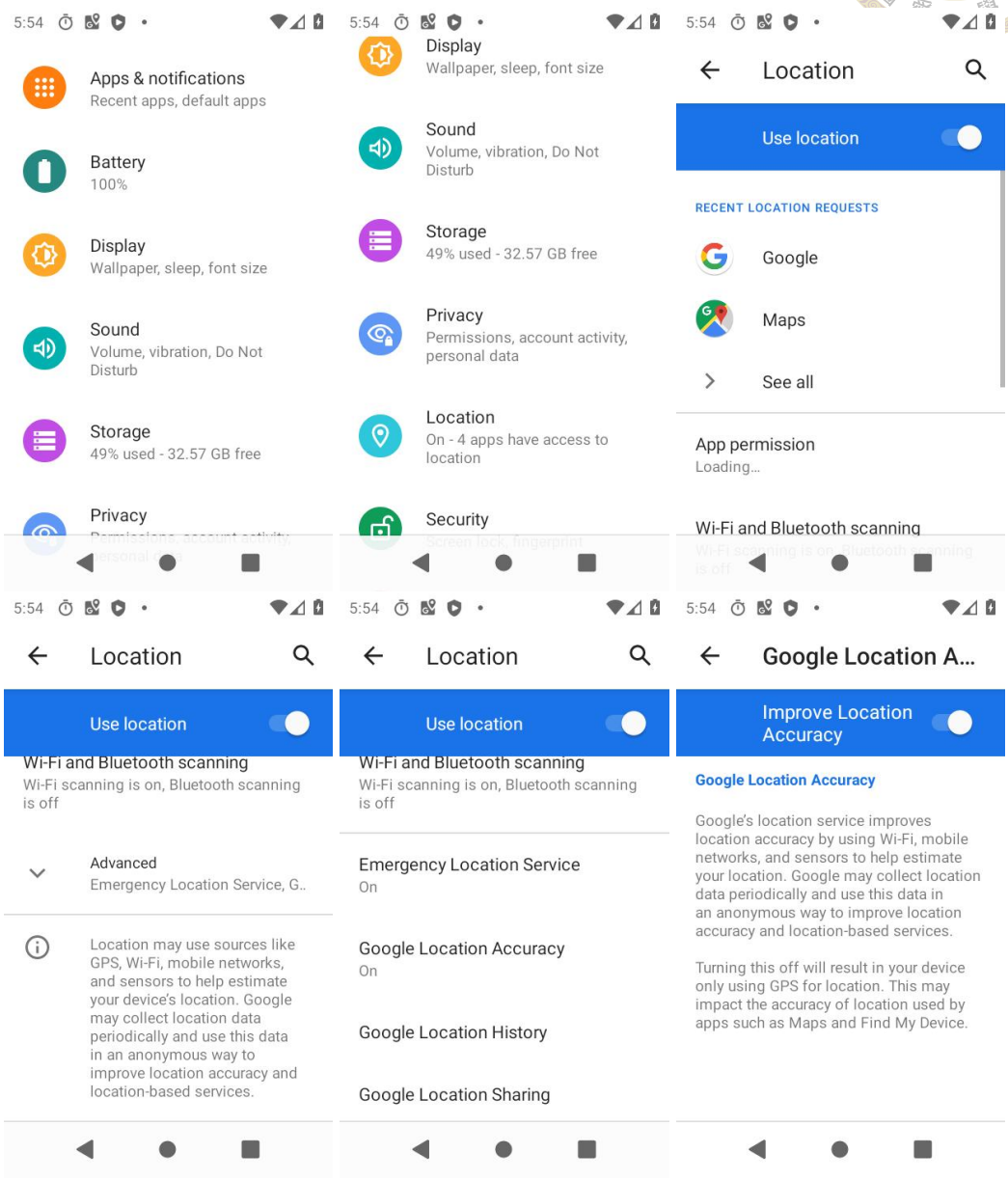
doi:10.6342/NTU202401151



Goal_info: check out phone information.

Figure A.1: Example 1 of random start episodes shows the first figure starting at the last scene of the previous episode.

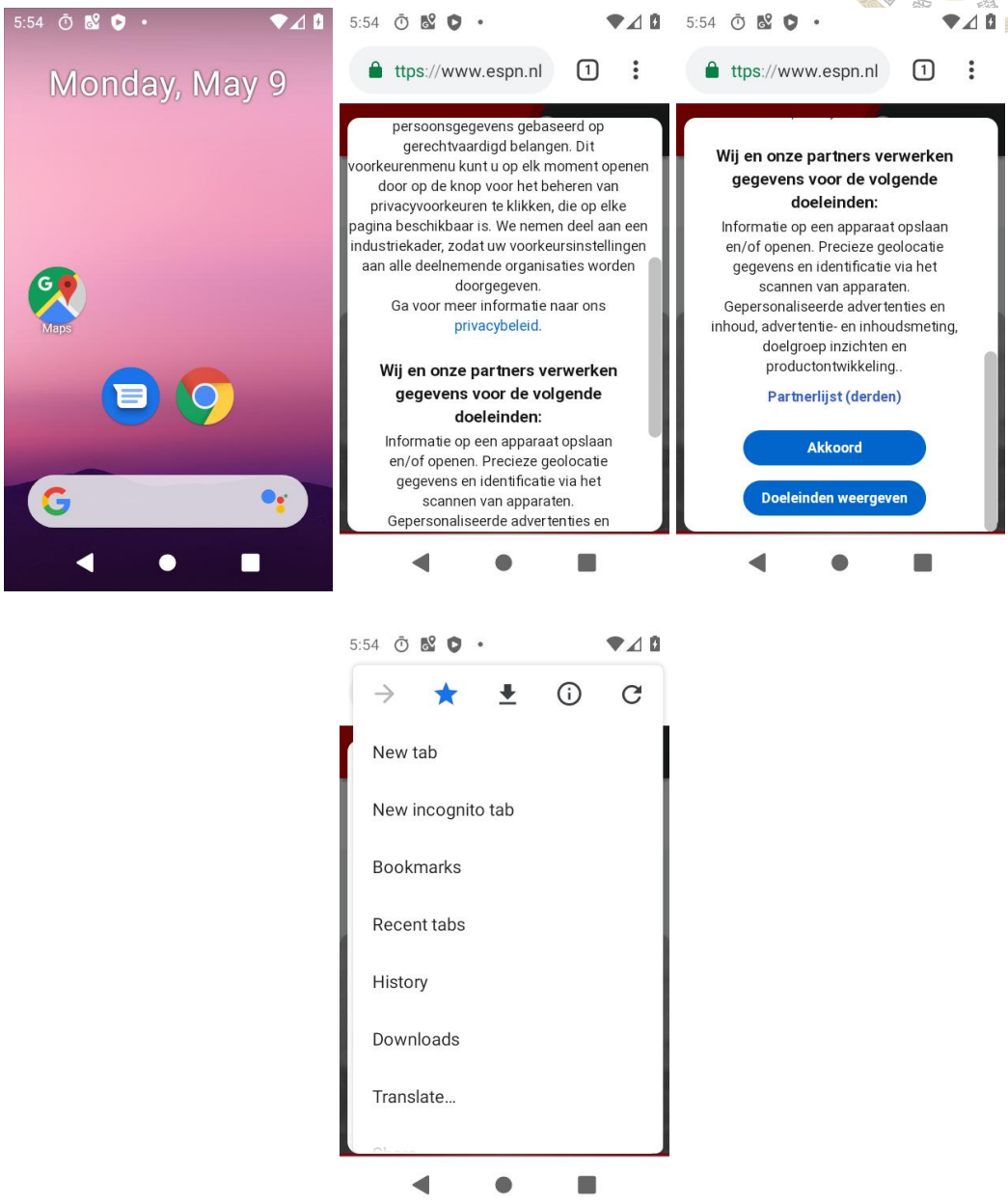
doi:10.6342/NTU202401151



Goal_info: turn off improve location accuracy.
Prediction : turn on improve location accuracy.

Figure A.2: Example 1 of missing episodes features our model accurately generating captions for scenes where a figure appears to be entering the 'improve location accuracy' page. The episodes seems to miss a figure toggling off the button.

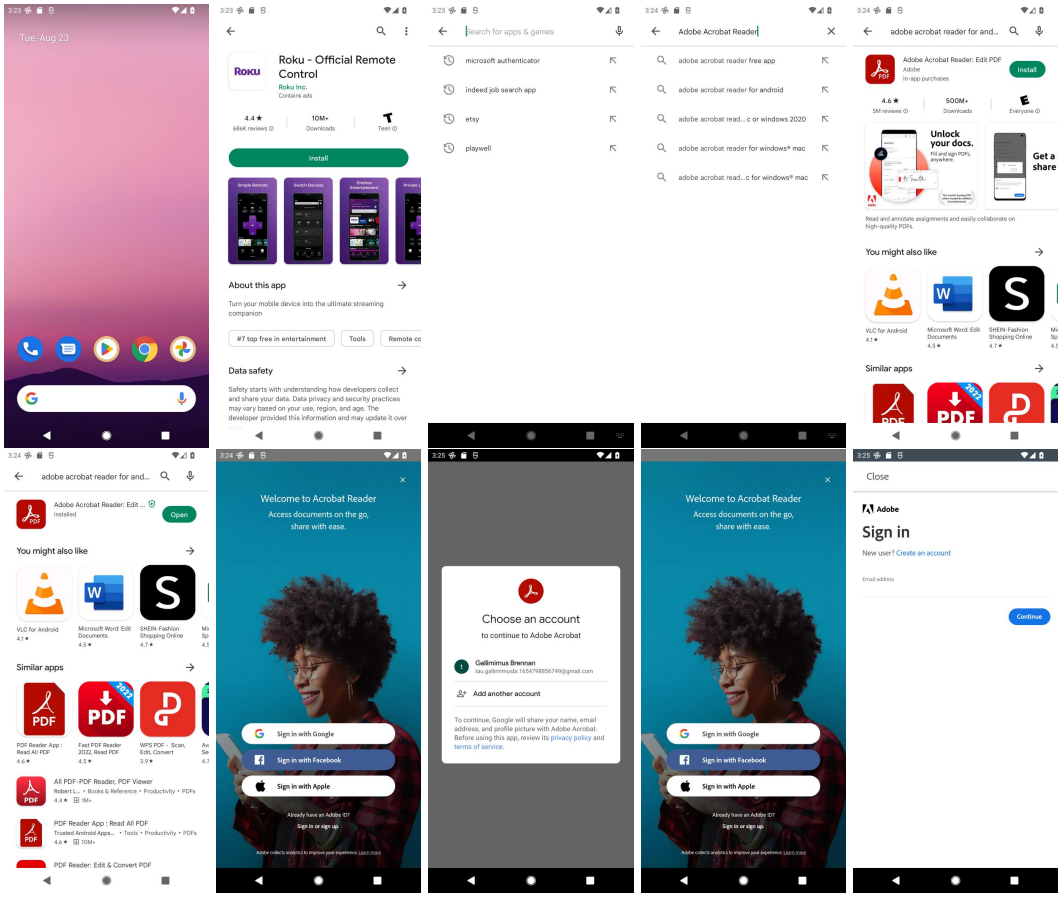
doi:10.6342/NTU202401151



Goal_info: open chrome and create a bookmark for the current page.
Prediction : read, delete, or share a saved page in the chrome app.

Figure A.3: Example 2 of missing episodes shows our model generating captions for scenes where a figure appears to be opening the toolbar. The episodes seems to miss the following steps.

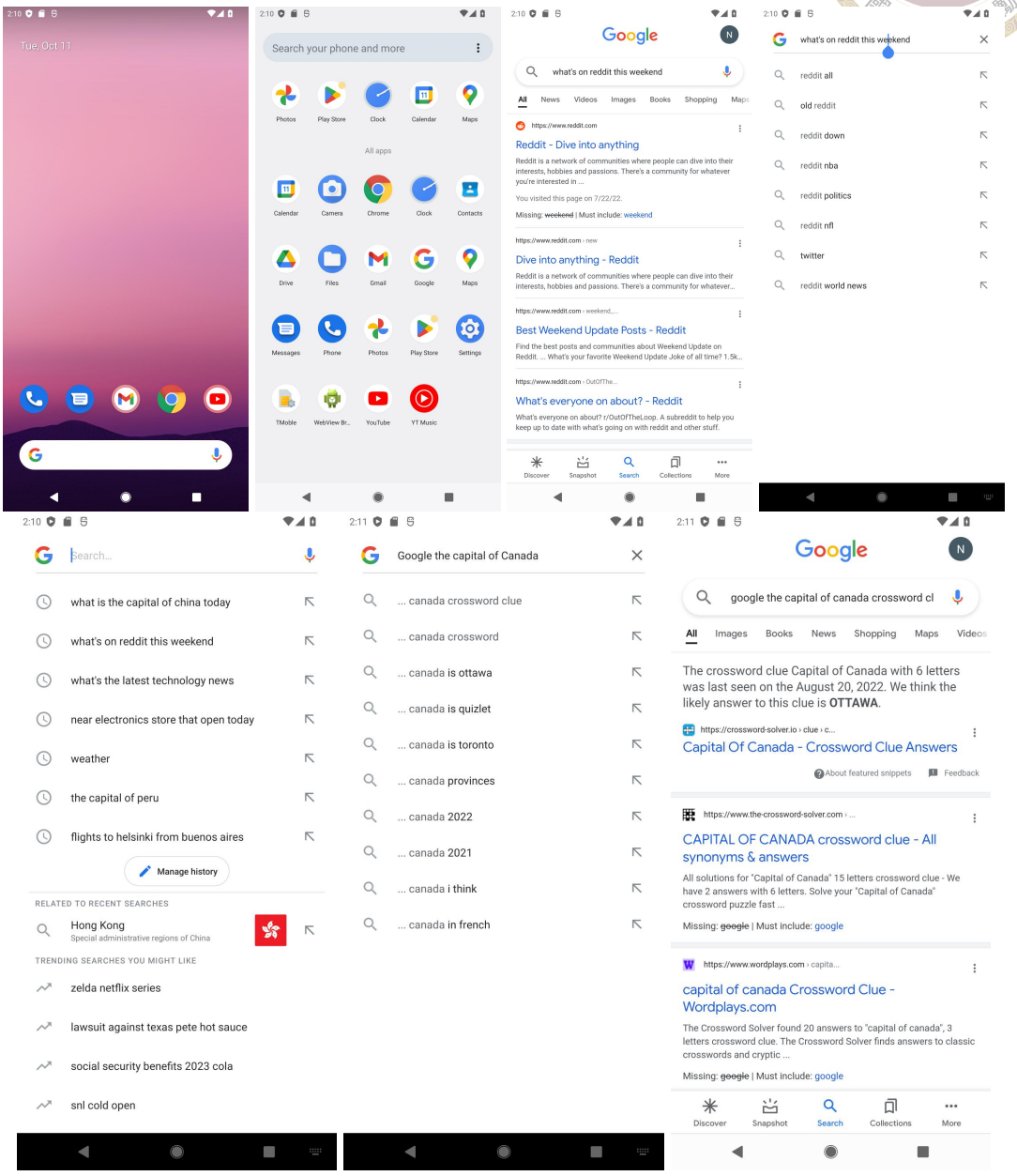
doi:10.6342/NTU202401151



Goal_info: open app ” adobe acrobat reader ” (install if not already installed), go to login, and select forgot password.
Prediction : open app ” adobe acrobat reader ” (install if not already installed) and go to login screen.

Figure A.4: An example from the install category demonstrates the model’s ability to predict visual information. However, the screenshots lack of actions related to ”forgot password”.

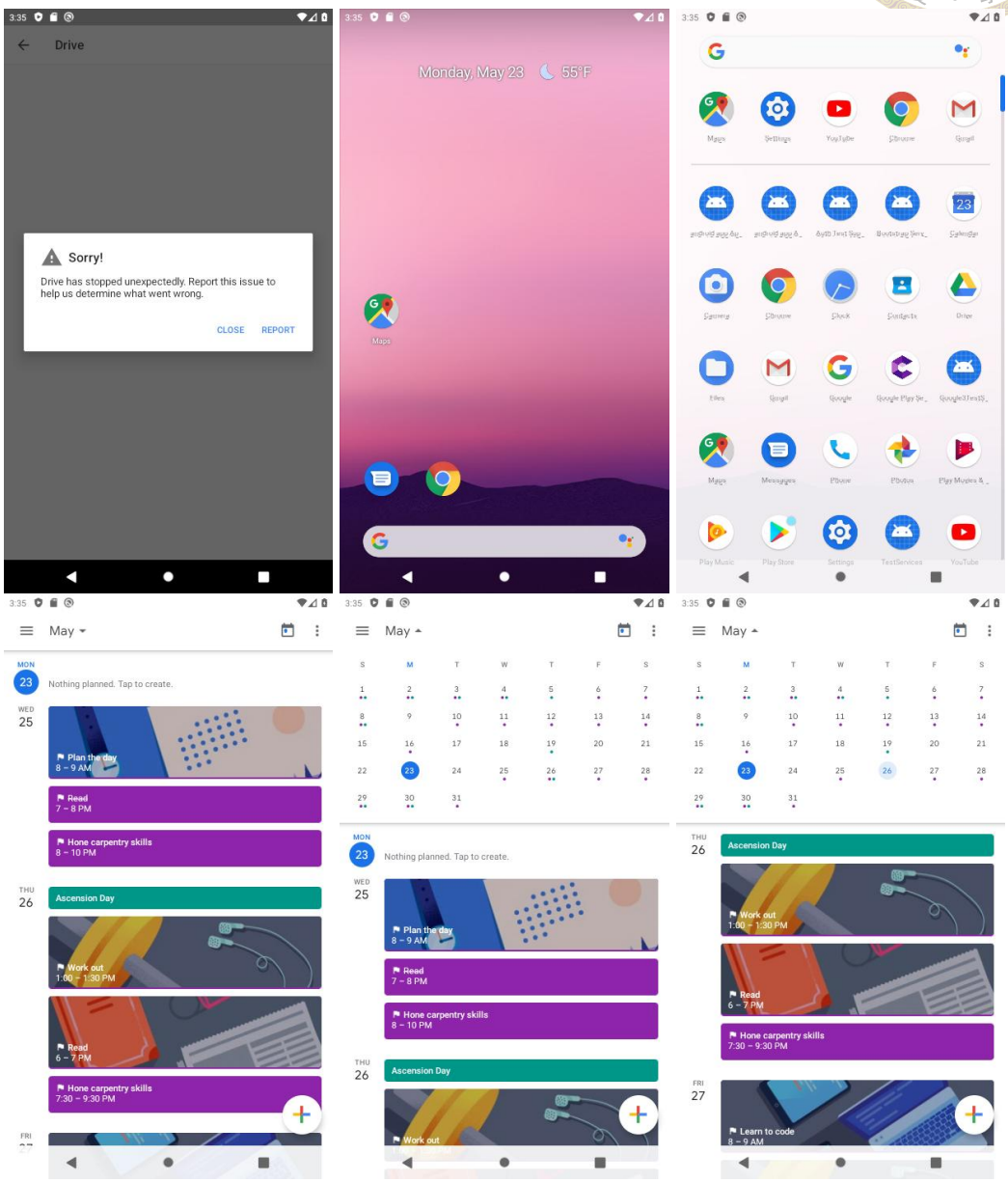
doi:10.6342/NTU202401151



Goal_info: google the capital of canada.
Prediction : google the capital of panama.

Figure A.5: An example from the general category shows that the model recognizes the query 'google the capital' but fails to identify the specific location. Using an optical character recognition module might address this deficiency.

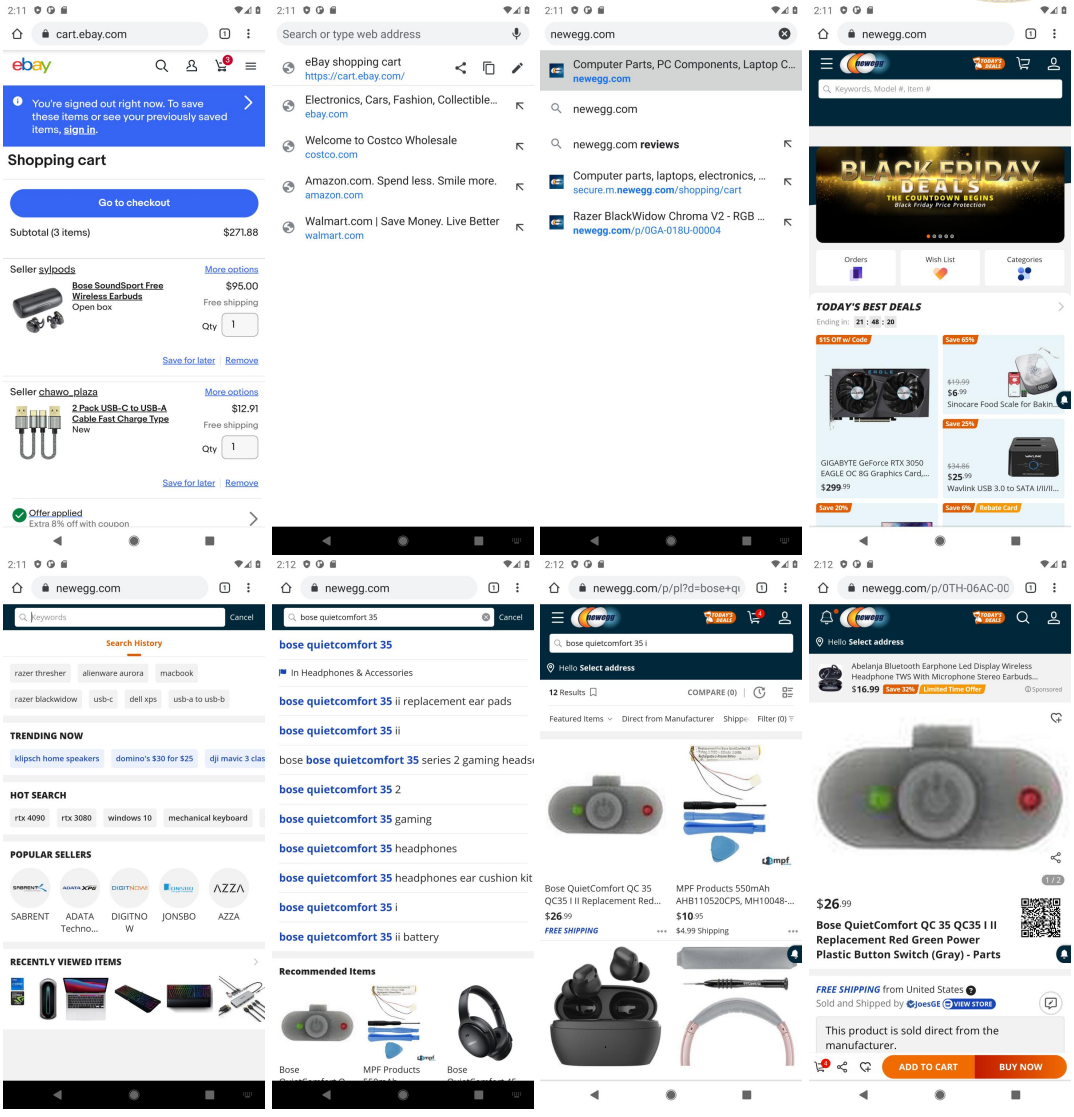
doi:10.6342/NTU202401151



Goal_info: when is my next appointment?
Prediction : do i have any events this weekend?

Figure A.6: An example from the google_apps category. The model predicts seeking for the events, although there remains a slight semantic gap.

doi:10.6342/NTU202401151



Goal_info: add " bose quietcomfort 35 " to the cart on newegg. com, then select checkout.

Prediction : search for bose quietcomfort 35 on newegg. com, select the first entry, and add it to the cart.

Figure A.7: An example from the web shopping category shows that the model understands the shopping steps and recognizes the specific product.

doi:10.6342/NTU202401151