Human-Guided Generation of Sketches and Prototypes

Forrest Huang

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2022-175
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2022/EECS-2022-175.html

July 18, 2022

Human-Guided Generation of Sketches and Prototypes

by

Zifeng Forrest Huang

A dissertation submitted in partial satisfaction of the

requirements for the degree of

Doctor of Philosophy

in

Computer Science

in the

Graduate Division

of the

University of California, Berkeley

Committee in charge:

Professor John Canny, Chair
Professor Alexei Efros
Professor Björn Hartmann
Professor Kosa Goucher-Lambert

Summer 2022

Human-Guided Generation of Sketches and Prototypes

Abstract

Human-Guided Generation of Sketches and Prototypes

by

Zifeng Forrest Huang

Doctor of Philosophy in Computer Science

University of California, Berkeley

Professor John Canny, Chair

Sketching and prototyping are central to creative activities that improve and advance many aspects of human lives. They enable non-experts to express themselves through drawing, or help User Interface (UI) designers explore diverse alternatives through low-fidelity prototyping. Generating these sketches and prototypes, however, typically requires significant expertise that casual users might not possess, and may be effortful and time-consuming even for professional users.

In this dissertation, I will introduce multiple deep-learning methods and systems that can generate sketches and prototypes. The generation of these artifacts is designed to be guided by annotations in familiar modalities (e.g., generating user interfaces from text descriptions). The presented generation systems and methods include Sketchforme, a system that generates individual sketched scenes from text descriptions; Scones, a system that iteratively generates and refines sketched scenes based on users' multiple text instructions; and Words2ui, a collection of methods that can create UI prototypes from high-level text descriptions. This research creates unique affordances, advances the state-of-the-art of creativity support tools, contributes benchmark metrics, and explores novel interaction paradigms in diverse domains from non-expert sketching to professional UI design. These research contributions can serve as important building blocks towards future multi-modal systems that enable more effective and efficient sketching and prototyping for all.

To Mum, Dad, and Anthony.

Contents

List of Figures

List of Tables

Acknowledgments

The Ph.D. program has been a challenging yet extremely rewarding experience for me. Throughout this arduous but productive journey, I am forever grateful to have the support of many mentors, colleagues, friends, and family members. They have made these past few years in graduate school the most intellectually stimulating and fruitful learning experience that I have ever had. This dissertation and my entire research career would not have been possible without their guidance and assistance. They have helped me improve not only as a researcher and collaborator, but also as a human being. With the limited amount of text that follows, I hope to express my unlimited gratitude towards them.

First and foremost, I would like to express my deepest gratitude towards my research advisor Professor John Canny, for his endless support of my work, for connecting me with relevant academic and industry collaborators, and for putting his trust in me by taking me as his student. Over the past few years, he has been patient and encouraging in advising each of my projects, especially at times when I faced significant challenges. He offered insightful yet practical suggestions, especially with his expertise in Deep Learning which I was not proficient in at the beginning of my studies. He taught me to be scientific, systematic, and methodical about understanding and diagnosing cutting-edge technical methods, and to holistically consider long-term research vision and directions in each of our projects. I have benefited tremendously from these invaluable lessons from his advising throughout the Ph.D. program, and I believe they will considerably strengthen my future research career.

I would like to thank each of the members of my dissertation committee: Professor Kosa Goucher-Lambert, Professor Björn Hartmann, and Professor Alexei Efros. I have worked closely with Professor Kosa Goucher-Lambert on interdisciplinary research between Computer Science, Mechanical Engineering, and Design. He provided me with many fresh perspectives with his domain expertise in Mechanical Engineering and Human-Centered Design, and encouraged me to think of implications that my research could have beyond my immediate field of studies. These perspectives have inspired projects presented in this dissertation and my overall research vision. Professor Björn Hartmann kept me connected with the wonderful Human-Computer Interaction research community and offered high-level research and career advice at multiple critical instances in graduate school. He has also built a supportive community of students that had graciously provided me with feedback and ideas about research and teaching. While I have not interacted with Professor Alexei Efros beyond the context of the dissertation committee, various works presented in this dissertation were inspired by Professor Efros' and his students' research, which we will cover in greater detail in the dissertation.

Beyond my immediate dissertation committee, there are many professors and instructors at UC Berkeley that have played vital roles in my academic and professional development. I would like to thank Professor Eric Paulos, who served as the examiner of my preliminary examination and a primary organizer of visit day events each year that have also reinforced my decision to pursue graduate studies at UC Berkeley. I attended classes taught by Professor Aydin Buluç, Professor Trevor Darrell, Professor James Demmel, Professor Armando Fox,

Professor Gerald Friedland, Professor Chris Hoofnagle, Professor Jitendra Malik, Professor James F. O'Brien, Professor Kimiko Ryokai, Professor Dawn Song, Professor Jennifer Urban, Professor Katherine Yelick, Professor Stella Yu, Dr. Allen Y. Yang, and Xavier Malina. I would like to thank all of them who have prepared highly applicable technical content and provided me with important research training in their classes. I would also like to thank postdoctoral scholars Vivek Rao and Qian Yu for providing insightful and actionable suggestions on my research projects with their experience and expertise.

I am fortunate to have the support and mentorship from many mentors in industry research. I would like to especially thank Jeffrey Nichols, who was my manager and mentor for my first first-author publication in graduate school. He provided me with extensive guidance not only on specific projects but also long-term advice in terms of research planning and career development. He also connected me to other researchers and domain experts whose opinions are constructive, important, and relevant to my research. I would like to thank David Ha, who has been an avid supporter of my work, for meeting me regularly to discuss novel ideas, for promoting and steering my projects to achieve maximal impact, and for endorsing my application to computing resources. I would like to thank my most recent manager Yang Li, who has supported me in pursuing ambitious projects, and provided in-depth and pivotal research ideas and recommendations that have led to the success of our research collaborations. Further, I would also like to thank the following industry researchers who have provided research advice and engineering support for my Ph.D. work: Peggy Chi, Tao Dong, Douglas Eck, Daniele Grandi, Gang Li, James Lin, David Salesin, Karl D.D. Willis, and Xin Zhou. Finally, I would like to thank Google Cloud for providing computational resources for experiments conducted in this dissertation, and Adobe for awarding me with an honorable mention for its research fellowship program in 2020.

My research career towards pursuing a Ph.D. degree would not have been realizable without my undergraduate research experiences at the University of Illinois at Urbana-Champaign. I would like to thank my undergraduate thesis advisor Professor Ranjitha Kumar, who introduced me to the Human-Computer Interaction research community, supported me to attend conferences as an undergraduate student, and trusted me to be extensively involved in her research projects. Many of these projects that we completed then have set up great foundations for the research work presented in this dissertation. I would also like to thank Professor Matthew Caesar for providing me with my first research experience in Computer Science, and Professor Cinda Heeren for guiding me through planning and managing a research mentorship program to serve the wider community of engineering students.

The magnificent and diverse research community at UC Berkeley has provided me with great colleagues to learn research and technical skills from and alongside with. First, I would like to thank the past and current members of CannyLab (Professor John Canny's Research Group), where I learned about video captioning and system administration from David Chan, protein modeling from Roshan Rao, news and natural-language processing techniques from Philippe Laban, robotics and reinforcement learning from Daniel Seita, contrastive learning and advanced attention mechanisms from Suhong Moon, autonomous driving from Jinkyu

Kim, and deep learning visualization from Biye Jiang. I would like to thank members of the Berkeley Institute of Design, consisting primarily of members of b-crew (Professor Björn Hartmann's research group) and Co-Design Lab (Professor Kosa Goucher-Lambert's research group). This includes Eldon Schoop, who is a close collaborator on our projects and taught me about machine-learning debugging tools; Elisa Kwon, who is another close collaborator on our projects and taught me about inspirational stimuli in design; Jeremy Warner, who taught me about vector style transfer; Andrew Head, who taught me about interactive tools for programming notebooks; Richard Lin, who taught me about hardware debugging; James Smith, who taught me about optimizing the data-collection processes with Augmented Reality; Bala Thoravi Kumaravel, who taught me about building interactive and usable tools in Virtual Reality; J.D. Zamfirescu-Pereira, who taught me about applications of large language models; Shm Almeda, who taught me about interactive tools for digital design; Nate Weinman, who taught me about programming education; Yakira Mirabito, who taught me about design decision-making; Ananya Nandy, who taught me about functional similarities in engineering design; and Doris Jung-Lin Lee, who taught me about accelerating data science workflows. I would also like to thank a few members of the Hybrid Ecology Lab (Professor Eric Paulos' research group), including Cesar Torres, Molly Nicholas, and Sarah Sterman, who have widened my horizons with some of the most novel and innovative interaction methods, and helped me prepare for my preliminary examination. Furthermore, I would like to thank Chandan Singh and Colorado Reed from Berkeley AI Research for the research discussions and social events. Apart from the research community, I would also like to thank the support staff I interacted the most with at the EECS department at UC Berkeley: Jean Nguyen, Shirley Salanio, and Patrick Hernan. They have been professional and helpful in handling documentations and inquiries about both regular procedures and extraordinary circumstances throughout the Ph.D. program.

I am grateful to have my closest friends in the bay area and beyond. I would like to thank Calvin Leung, Kelvin Kwok, Darren So, Ricky Yeung, Markus Au Yeung, Kyle Wong, Godfrey Chan, Tim Chiu, Victor Cheung, Annie Jin, and Evan Huang. They have made various phases of graduate school significantly easier through enthusiastically celebrating my tiniest achievements, and continuously providing mental and emotional support through challenges and obstacles.

Finally, I would like to thank my family for tirelessly supporting me not only in this Ph.D. program, but through all stages of my life. My parents' nurturing has granted me the passion for technology and innovation from a very young age, and growing up with my brother has taught me the qualities and characteristics of a good collaborator. Most importantly, they have been my biggest champion through the ups and downs of this journey of life. Their unconditional love has given me tremendous strength to weather through storms in the past, present, and future, and they can expect the same commitment and contribution from me in my lifetime.

Chapter 1

Introduction

Communicating novel visual ideas and materializing them with concrete prototypes are some of humans' most unique capabilities that drive innovation and improvements in many aspects of our lives. As the descendent of one of the earliest forms of communication [129], sketches are widely used to convey visual ideas in diverse fields. The abstract yet expressive nature of sketches enables humans to quickly and succinctly communicate conceptual and high-level ideas. Sketchers can transform their intents into concrete illustrations and artifacts, and communicate these concepts tangibly while leaving out unnecessary details. Therefore, sketches are popular among artists for expressing creative thoughts [49, 90, 108, 129], among engineers for communicating hard-to-verbalize ideas [41, 66], and among educators for teaching complex and unintuitive concepts [2, 43, 94]. Most notably, the advantages of sketch-based problem-solving and communication are manifested in design processes where designers frequently sketch in iterative design, critique, and review sessions [12, 19, 29, 31, 41, 109, 146, 164].

Following through on the envisionment of creative ideas through sketching, prototypes at various abstract levels, or *fidelities*, are often created to help us further realize, refine, and evaluate the novel solution we developed for a particular problem. For instance, user interface (UI) designers might prototype using low-fidelity mock-ups with abstract graphics or high-fidelity screenshots [189]. These prototypes allow designers to better gauge the merits and issues of their design solutions, while being able to quickly iterate without devoting significant effort to building out a polished solution or product.

As both sketching and prototyping have facilitated creative activities in many areas, each of these two processes and their sub-steps has been established and well-researched in design and creativity research fields (see a review of the background of sketching and prototyping in creative processes in Chapter 2). However, producing aesthetic and functional sketches and prototypes still requires extensive expertise, considerable experience, and non-trivial effort. Attempting to lower this barrier of execution, Human-Computer Interaction (HCI) researchers have since developed a number of computational *Creativity Support Tools* to reduce users' workload during these processes [175]. Many of these tools were designed to support exploratory search [78, 105, 110, 157, 166], assist prototype creation [95, 106, 120],

and encourage collaboration [107], all to further elevate and improve users' creative processes.

Recent advances in the field of Machine Learning (ML) and Deep Learning (DL) have drastically improved computational systems' ability to comprehend and generate visual content [150, 151, 165, 195]. The development of neural-network architectures (e.g., convolutional neural networks [104] and attention-based Transformers [184]) have provided machines the ability to generate visual content conditioned on user-specified natural language and/or other accompanying semantic information. These model architectures have also advanced the state-of-the-art for computer vision tasks of sketching [62, 156] and prototyping [7, 60]. They provide great opportunities for sketching and prototyping applications that this dissertation aims to explore.

Inspired by work in the HCI community on creativity support tools and recent advances in DL, this dissertation investigates and presents computational systems that can **automatically generate sketches and prototypes** from scratch while **guided by users**. These automatic processes of creating creative artifacts drastically reduce the demands for skills and effort of users to engage with sketches and prototypes. We specifically designed these systems to also take users' guidance in various forms that they are familiar with (e.g., natural language) to grant them control over the generation outputs. These systems effectively transform users' creative expressions from modalities that they are proficient in, to modalities that were previously difficult for them to be expressive in. Through contributing these new systems towards guided generation of sketches and prototypes, this dissertation also explores novel affordances and interaction paradigms in diverse domains from non-expert sketching to professional UI/UX design. We believe these research contributions can serve as important building blocks towards future multi-modal systems that enable more effective and efficient creative expression for all.

1.1 Contributions

The core contributions of this dissertation are several computational systems that can create and generate sketches and prototypes to support various creative activities. These systems adapt state-of-the-art DL models to generate sketches and prototypes from scratch and support appropriate levels of user-control that the targeted audience can easily use: the artifacts generated by these systems are guided by naturalistic user-inputs, such as natural language authored by non-experts and professionals. Through developing these systems, we also contribute models that accomplish novel tasks and/or establish new state-of-the-art performance on established tasks. These systems and models which target various creative domains include:

- Sketchforme (Chapter 4), the first DL system that can **generate sketched scenes that contain multiple objects corresponding to individual text descriptions**. Sketchforme uniquely factors the complex sketch-generation task into layout composition and stroke rendering sub-tasks. We evaluated the quality of the generated sketches

of Sketchforme with a custom quantitative metric, a qualitative exploration, and a study of user-rating of the sketches. Our evaluation and study participants have found the generated sketches to be generally expressive and realistic. Using these generated sketches, we develop prototype applications to demonstrate that Sketchforme can potentially improve language-learning applications by adding visual hints to foreign language phrases. We also demonstrate that Sketchforme could support sketching assistants by auto-completing sketched scenes based on users' instructions and preferences.

- Scones (Chapter 5), which enables **iterative authoring and critique of sketched scenes using natural language**. It is the first DL system that can generate and modify sketched scenes given users' text commands across multiple turns and cycles of sketching and critiquing. Scones improves upon Sketchforme's novel workflow to continuously consider new text commands while composing scene layouts with object specifications, before rendering sketch strokes for each scene object. Scones exceeds state-of-the-art performance for an established text-based scene-layout modification task, and introduces a novel affordance of controlling the appearance and poses of generated object sketches with masks and aspect ratios. In an exploratory user evaluation of Scones, participants generally enjoyed the novel conversational sketching interaction paradigm that it is able to support, and were satisfied with the final sketches that they co-created with Scones.

- Words2ui (Chapter 6), which is a collection of multiple DL models that are all first of their classes to be able to **generate and retrieve user interface prototypes** in the form of low-fidelity mock-ups[1] guided by users' high-level specifications of the desired UIs. We also developed a set of novel benchmark metrics grounded on prior works to measure three aspects of success of the text-to-UI generation task. We evaluated our models using these metrics and found that our text-conditional models are able to generate UIs of similar quality and diversity as previous state-of-the-art unconditional models. We further built a filtering and rendering pipeline to present model outputs in the plausible form of low-fidelity UI mock-ups. Using these generated mock-ups, we performed a qualitative analysis and a user study, and found that the mock-ups adhere to constraints given by input text descriptions and can potentially support experts' design processes.

[1]Note that there are contemporary definitions that only consider high-fidelity artifacts that are comparable to the final product as *prototypes*. However, there are a large number of prior literature referencing UI mock-ups that consist of abstract graphics as low-fidelity prototypes, and even consider sketches as prototypes. Please see Section 2.1.4 for a detailed discussion.

1.2 Overview

This dissertation will be organized as follows: We begin with reviewing background knowledge and relevant research in Design, HCI, and ML communities about sketching and prototyping. We will then describe in the following chapters sketch-generation and prototype-generation systems that we contributed in detail. Finally, we conclude this dissertation with high-level takeaways from these research projects and outline important future avenues of research. We include a brief overview of all projects and systems presented in this dissertation in Figure 1.1.

Chapter 2 provides background information about creative processes, tools that support these processes, and the application of sketches and prototypes in these processes. It also provides a technical prior of various DL techniques and datasets that could be useful for modeling sketches and prototypes.

Chapter 3 surveys recent work relevant to the research in this dissertation. In the HCI research literature, this includes systems for sketching and prototyping support such as automated sketching tutors and assistants, sketch-based prototyping tools, and generative design applications in specific domains. We additionally review related work in recent ML research literature in this chapter, which includes models that enabled style transfer, natural image generation, sketch generation, scene and document layout generation, 3D model generation, and interactive conditional generation and editing of visual content.

Chapter 4 describes Sketchforme, the first system presented in this dissertation. Sketchforme can generate individual scenes with multiple sketched objects given a single natural language description. This chapter describes the implementation of Sketchforme and presents both quantitative and qualitative analyses of the sketches generated by Sketchforme. It additionally describes a few applications enabled by Sketchforme in detail, validating and showcasing the utility of Sketchforme-generated sketches for language learning and assistive sketching.

Chapter 5 describes Scones, which built upon the text-to-sketch task and system architecture introduced by Sketchforme to perform *iterative* generation of sketches from text instructions. This chapter describes the Transformer-based system architecture of Scones that allows it to consider previous scenes in addition to text instructions as inputs, consequently enabling its new iterative scene sketching ability over Sketchforme. We then report Scones' quantitative and qualitative performance in composing scenes and sketching objects. We additionally describe a web-deployable system supported by Scones towards the end of this chapter. This system was used in our user study with Mechanical Turk users on sketching with Scones iteratively through natural language. We report findings from this study and important features suggested by our users for both Scones and future human-AI co-creative systems to implement.

We then move beyond non-expert sketches to investigate professional design domains in Chapter 6. We introduce several DL methods that can create UI prototypes in the form of design mock-ups from text descriptions. We describe the dataset choices, pre-processing procedure, model architectures, post-processing and filtering techniques, and mock-up ren-

dering pipeline of these methods. We additionally define a novel benchmark that consists of a set of metrics addressing three main aspects of text-to-UI task success: Well-formedness, Relevance, and Diversity, to systematically evaluate our approach against prior and future research in this area. We then report quantitative results of our methods on the newly-defined metrics and present qualitative examples of UI mock-ups created by our models. Towards the end of this chapter, we report expert UI/UX practitioners' preferences and opinions in a user study about the quality and utility of the generated and retrieved mock-ups.

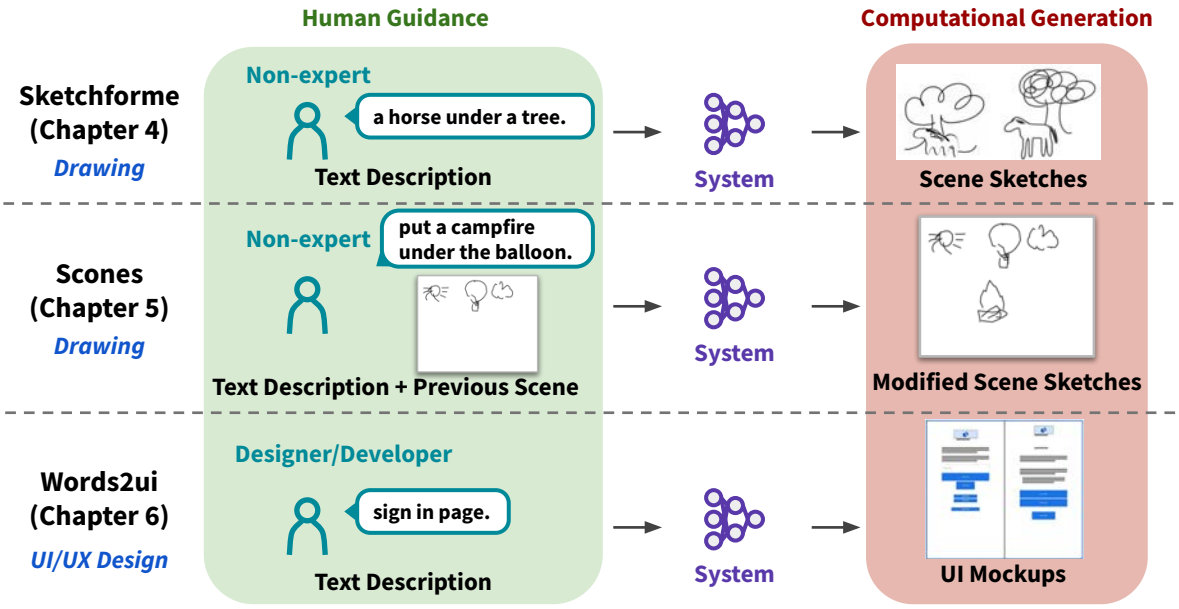


Figure 1.1: Overview of all projects presented in this dissertation.

Finally, in Chapters 7 and 8, we conclude the dissertation with some high-level takeaways and themes that are common across the proposed computational systems. We also further paint the landscape of future research in this exciting area of generative sketching and prototyping.

1.3 Statement of Prior Publications and Authorship

This dissertation contributes multiple projects with core ideas and research progress previously published at various venues. Sketchforme (Chapter 4) was published at the UIST 2019 conference [77] as a full paper. Scones (Chapter 5) was published at the IUI 2020 conference as a long paper [80] and done in collaboration with Google Research. Scones was also included in a chapter on *Sketch-based Creativity Support Tools using Deep Learning* in the *Artificial Intelligence for Human Computer Interaction: A Modern Approach* book

published by Springer in 2022 [81]. Chapter 6 contains work on text-to-UI generative models published at the *Computational Approaches For Understanding, Generating, and Adapting User Interfaces* Workshop at the CHI 2022 conference [79] and done in collaboration with Google Research. I am the first author in all of the aforementioned papers and book chapter, but this research would not have been possible without the assistance of my collaborators. I am especially grateful for the guidance of my faculty advisor—John Canny, and my industry mentors—David Ha and Yang Li, who are also co-authors of most of the aforementioned papers and book chapter. I also appreciate the research and software development support from Eldon Schoop on Scones, and Gang Li and Xin Zhou on text-to-UI generative models, who are also co-authors of the respective papers. Finally, I would like to specially acknowledge my mentee Luming Chen for contributing the TranSketch dataset as an extension to Scones (described in Section 5.6), and Luming had also completed a master's thesis on this topic [23]. I reflected all of my collaborators' support by using 'we' throughout this dissertation, with the exception of Chapter 7 which reflects my personal opinion and speculation on future research opportunities.

Chapter 2

Background

In this chapter, we provide the necessary background for understanding existing types and approaches of support towards creative processes that involve sketching and prototyping, and the technical approach developed recently to model data formats that are commonly found in these processes. We additionally review some datasets, which are important prerequisites for deep-learning (DL) systems to support these processes.

2.1 Sketching and Prototyping in Creative Processes

Sketching and prototyping are central to many creative processes, supporting activities in many creative fields. In the context of this dissertation, we center our discussion on how systems might support the creative processes of non-expert sketching and user interface (UI) design. Note that sketching is often defined as the modern-day practice of creating quick and free-hand drawings that only emphasize general shapes, which is consistent with the usage of this process in design fields. However, the products created by this process are often line drawings (or line art) which can trace their roots to earlier art forms [129].

2.1.1 The History of Sketching and Its Significance for Communication, Self-expression, and Art

Sketching processes, one of the main types of creative processes that this dissertation aims to support, often produce line drawings as their primary artifacts. Line drawings have deep roots as one of the earliest communication media and art forms, manifested as prehistoric cave paintings in the Upper Paleolithic period [69, 71, 144, 182]. They were used by humans at the time for artistic expression of their imaginative worlds [129] and practical, utilitarian records of life events [153]. Some of these paintings are hypothesized by researchers to have been specifically designed to reflect the acoustics of their physical locations within caves, allowing cross-modality information from the physical world to be preserved in the paintings [127].

The pervasiveness of sketches in human society can also be partially reflected by the cognitive development process of humans, particularly of children and infants. Multiple studies have discovered that visual concepts were developed as early as infancy, and sketching has therefore been used by psychologists and cognitive scientists as probes into children's developmental processes [82, 96, 123]. Similarly, drawing has also been recognized as an important outlet of expression among children and can uncover individual stories and experiences that might be hard to verbalize [93].

In artistic fields, sketching and line drawing are considered to be important art forms that aspiring artists practice extensively during their education. Starting from the 18th century, the promotion of line drawings allowed amateurs to participate in producing art now recognized by experts. Line drawings have been recognized as final products that many artists are known for (e.g., John Constable [90], Pablo Picasso [108]). Moreover, the processes of creating many types of more sophisticated art forms often start with artists sketching the general structures and compositions of the final products [49] before details of the art pieces are filled in, making the sketching process important for artistic ideation and production.

2.1.2 Human-Centered Design Process and Artifacts

Perhaps one of the most significant use-cases of sketches and prototypes in contemporary creative processes is in design processes across many domains. In this section, we first provide background about definitions and examples of design processes, and then describe the produced artifacts in various parts of these processes including sketches and prototypes.

The commonly established human-centered design-thinking (HCD) process [5] recommends five steps towards a working solution to a design problem: *Empathize*, *Define*, *Ideate*, *Prototype*, and *Test*. We use a simple example of building a restaurant-searching mobile application for university students to illustrate these steps. We emphasize the definitions of the last three steps in this section due to their high relevance to sketching and prototyping. For completeness, however, we first briefly describe the empathize and define steps. The *Empathize* step aims to be used by designers to fully understand and engage with target users about their problems in their context. In our example, designers might recruit university students and discuss with them about their current pain points and difficulties in using existing solutions for finding local restaurants. Based on the results of the empathize step, the *Define* step aims to clearly frame the design problem and document requirements for the potential solution. In our example, designers would describe requirements such as users' need to see current wait times of restaurants nearby.

Following the empathize and define steps is the *Ideate* step, in which designers are suggested to generate many potential ideas that might solve the concerned design problem. Designers are recommended to diverge and explore a great breadth of concepts and references as starting points and resources for the next steps of the design process. In our example, this might lead to the ideas of using map-based and list-based layouts to display restaurants located around the user.

In the *Prototype* step, designers need to translate concepts and ideas formulated in the previous steps into prototypes. Prototypes are design artifacts that provide certain degrees of resemblance to the final potential solutions. They should embody ideas and concepts converged and chosen from previous steps and allow designers to experience potential solutions, reason about their merits and disadvantages, and compare them between various candidate solutions. In our example, this would represent the process of actually building out map-based and list-based layouts in a mobile form factor to demonstrate their appearance and functionalities. Given the high relevance of this step to targeted design artifacts that our systems aim to support (i.e., *prototypes*), we include a detailed discussion of prototyping in the human-centered design processes in Section 2.1.4.

In the *Test* step, prototypes created in the prototype step are used to solicit feedback from targeted users. Note that on many occasions, these testing activities overlap with the empathize step such that this provides 'another opportunity to gain empathy for the people you are designing for' [5]. In our example, this would represent taking several completed prototypes to the targeted users (e.g., the university student recruited in the empathize step) to solicit their opinions and/or record quantitative statistics with them performing a task representative of the real use-cases of the design. This could potentially lead to another Empathize step, and this overlap highlights the *non-linearity* of the overall design process, such that these activities can repeatedly happen in various orders until a final design solution has been developed and built.

Throughout this design process, numerous artifacts in various modalities would be used by designers to iteratively ideate, create, communicate, reason, and compare design ideas. Some of the most commonly used artifacts in the Ideate step are sketches, which allow designers to quickly express and discuss design ideas without committing to other irrelevant details of the ideas. As they move through the design process, low-fidelity and high-fidelity prototypes[1] that include greater details of proposed design solutions would be used to more comprehensively evaluate the solutions. This leads to our investigation of the possibility for computing systems to generate these important and applicable artifacts to aid in the design process. In addition, natural language is frequently used in the earlier empathize and define steps in interviews and requirement documentations. This leads to natural language being a primary form of human guidance supported by systems proposed in this dissertation.

Before we dive into the specific discussions of these artifacts, it is important to note that this is not the only model for design processes. However, many of the steps defined by this process model are echoed by other processes, including the design process described in influential design literature [134], practical design thinking recommendations by IDEO [37], and a mobile app design process [189] adapted from the famous and widely applicable Total Design method proposed by Pugh [145]. This mobile app design process includes specification, conceptual design, prototyping, and testing phases that share similarities with the

[1]Some definitions of prototypes include sketches as a type of prototypes, widely defining a prototype as 'any representation of a design idea, regardless of medium' [75]. We separate the discussion of sketches due to its distinctiveness in form compared to the mockup-based prototypes that we produce in different systems introduced by this dissertation.

Ideate, Prototype, and Test steps discussed above. Hence, we believe the artifacts produced in the HCD design process discussed above are applicable to design processes in many other domains.

2.1.3 Sketching in Human-Centered Design

Sketching is commonly used by designers today to expand novel ideas, visualize abstract concepts, and rapidly compare alternatives. Its involvement and utility in multiple facets of design have been extensively documented by work in the design research community, including architectural and product design [31, 72, 109]. Sketching, or working in visual media, is considered a core competency and critical skill for designers to master [30]—researchers have shown through interviews and conversations that possessing great sketching skills is one of the most important characteristics of prominent designers [29, 31, 109]. Hence, sketching tutorials have been included in many design textbooks and courses in the training curriculum of designers [19, 160]. The abstract nature of sketches solicits feedback discussed at a high level, without burdening the designer with creating and consuming lower-level details and prevents premature commitment to a particular idea. This allows designers to more comprehensively consider and explore different alternatives, resulting in better final designs. Thus, sketches are often used primarily in the earlier steps in design processes, such as the Ideate step discussed above in Section 2.1.2.

Sketches produced in design processes, while often are earlier manifestations of more detailed and higher fidelity design artifacts and prototypes of the same design idea, are used themselves as artifacts of the design processes [41, 67]. One primary usage of sketches in design is to visually represent the intended design solution in an abstract manner, serving the important purpose of allowing reinterpretation and exploration among designers themselves and among members of design teams [41, 146]. Sketches allow designers to consider potential physical and geometrical configurations of design solutions relevant to their problem contexts. Because of this, sketches on their own can also take different levels of fidelity and *formalities* depending on the levels of completeness of the design solutions and the levels of decision needed to be further made on them, ranging from simpler line drawings depicting overall concepts of the products, to technical sketches that specify constraints for manufacturing prototypes. Note that the definition of sketches can also expand beyond visual representations of final design solutions, to hand-drawn diagrams, symbols, numbers, and texts that explain various aspects of the solutions [30], such as those in system design flowcharts. Such heterogeneity is considered by an early computational system that separately processes each type of sketches to fully support free-form sketching in design [124].

One prolific lens in design research towards sketches is considering them as a *communication* medium, which could be a natural extension of drawings being used to communicate hard-to-explain visual and physical ideas discussed in the previous sections. Ferguson identified three classes of sketches in the context of communication in engineering design: *thinking* sketches for individuals to 'guide nonverbal thinking'; *prescriptive* sketches to dictate instructions towards more complete drawings; and *talking* sketches for interaction among members

of groups and teams [44]. Other researchers have also noted similar use-cases of sketches in design to generate, communicate, analyze, and retain design ideas and solutions [169, 181, 183]. Researchers have additionally noted the important usage of sketches as *boundary objects* [41, 67], which are artifacts that support groups with different levels of expertise and understanding of context to communicate effectively. Most relevant to this dissertation, Neilson and Lee observed complex inter-dependency between natural language and sketches, and that designers frequently use them in combination in design activities [133]. This further motivates our proposed systems in supporting natural language as a form of human guidance for sketching. Overall, all of these definitions of sketching as a communication tool reinforce the pervasiveness of sketches as artifacts in design processes in diverse domains.

Following the documentation of the frequent use of sketches in design processes, researchers have also investigated the relation between design outcomes and the degree of usage of sketches. There is some evidence that points to the positive correlation between levels of sketching and positive design outcomes in university engineering design courses. This is especially due to their roles in encouraging design exploration and lowering cognitive workload when reasoning about potential solutions' effectiveness. There are many other prior research works that describe the important roles of sketching in design that we are unable to include. Review articles written by Eckert et al. [41], and Purcell and Gero [146] are good starting points to these works.

2.1.4 Prototyping in Human-Centered Design

Prototyping usually follows after different design ideas are sketched out and explored, and when the team has decided on a more concrete design (or few candidates) to proceed with. This is also reflected by the Prototype step defined in the human-centered process example in Section 2.1.2 that follows the Define and Ideate steps.

Prototypes are the primary artifact created during prototyping (as the name obviously specifies). There are many definitions, properties, and taxonomies of prototypes available and published, and we discuss several that are relevant to human-centered design. Beaudouin-Lafon and Mackay define a prototype to be a tangible artifact that concretely represents individual parts or the entirety of an interactive system, and that participants in the design process can use these artifacts to *envision and reflect* on the final system [12]. It is thus clear that prototypes usually refer to artifacts that are of greater detail and with more realized properties than sketches, and are used for evaluating and reasoning about the final concrete effects that certain design solutions have on their problems.

To further understand the types of prototypes used in design processes, we consider Beaudouin-Lafon and Mackay's definition of four dimensions of prototypes [12]:

- *Representation*, which is the form that the prototype is presented in (e.g., sketches, CAD model renderings, or user interface (UI) mock-ups with abstract colored graphics).

- *Precision*, which is the level of details that the prototype was created with. This could be thought of as similar to the definition of *level of fidelity* commonly used in design literature (e.g., low-fidelity wireframes vs. high-fidelity screenshots for a mobile app)[2].
- *Interactivity*, which describes the amount of interactive behavior implemented in the prototypes and the degree that users can interact with the prototype.
- *Evolution*, which describes the expected life cycle of the prototype. This refers to whether it is designed to be reused and redesigned iteratively or to be thrown away after a single use.

Using this framework, we discuss several types of prototypes commonly used in design. To begin with prototypes with the lowest fidelity and with great connection to sketches, *paper prototypes* are commonly used by designers to quickly turn design sketches into 'interactive' prototypes to evaluate their design solutions on realistic tasks [177]. A paper prototype consists of a hand-drawn (often in sketch-like form) or printed version of the prototype on paper. The user would perform the target task on pieces of paper with the printed or hand-drawn interface (e.g., tapping on the paper if the target task is tapping on a mobile device in an app), and the designer or practitioner usually accompanies the user in the whole process and plays the role of the 'computer' to rearrange the pieces of paper of the prototype to reflect interactive behavior of the app. Paper prototypes have the advantage of being low-cost and can be quickly created, leading to quick iterations and explorations of many design ideas. Paper prototypes can thus be considered to have low precision, moderate interactivity, and are designed to be thrown away after single or very few uses.

Following paper prototypes, low-fidelity (Lo-fi) *wireframes* and *mock-ups* are prototypes[3] frequently used in the UI design process [9, 177, 189]. A mock-up defines the layout of a certain page in a UI, describing the exact positions of various types of UI elements and widgets (e.g., buttons, text fields, etc.). These elements are contemporarily represented with graphics that depict various types of elements and template text, giving a mock-up a slightly more polished and refined quality than a sketch. One main purpose of mock-ups is to concretely define the relative and absolute sizes and positions of various elements, which affect the aesthetics and usability of the UIs that the mock-ups reflect. Hence, the mock-up representation allows designers and testers to more concretely analyze and evaluate their design solutions and envision final end-products produced from them. Mock-ups and wireframes themselves are often not directly interactive, but they can be augmented with the paper prototyping technique to add interactivity by printing out the mock-ups or wireframes and having the designer act as 'computers' [177].

[2]The definition of fidelity, however, has the additional reference of comparing to final systems, while precision only describes the state of content in prototypes concerned.

[3]There are contemporary definitions that consider prototypes only as high-fidelity artifacts that are comparable to the final product for differentiating them against low-fidelity prototypes in the form of sketches or abstract graphics. However, we follow the definition of prototypes mentioned above and consider wireframes and mock-ups to be prototypes that help designers envision the final to-be-built products.

Towards the end of the prototyping process, high-fidelity prototypes are produced. These prototypes could be complete UI screenshots with partially programmed behavior of a mobile app [189], or a physical prototype for a mechanical structure [20]. These prototypes have similar or equal properties in the four dimensions discussed above as the final design solution. Some of these prototypes might even be reused as final design solutions, given the high effort typically associated with creating them. As such, the design research community has explored and employed various techniques to reduce the effort of coming up with relevant high-fidelity prototypes [20, 78], aiming to encourage further design exploration in this high level of fidelity.

The latter part of this dissertation explores the generation of low-fidelity UI prototypes in the form of mock-ups, with a moderate level of details, and no interactivity since these wireframes primarily serve the purpose of evaluating the aesthetics and usability associated with the placement and sizing of UI elements. Nevertheless, it is important to note that there are many other types of prototypes, creating numerous future research opportunities in this area to be pursued. We would also like to highlight an orthogonal perspective of modeling the characteristics of prototypes based on their *purposes*, which can be useful for ensuring the generated prototypes achieve their intended objectives in design processes [75].

2.2 Creativity Support

To support creative activities which include the aforementioned design and artistic activities that involve sketching and prototyping, HCI researchers have developed a class of *Creativity Support Tools* and has since been established as a prolific research sub-field. In this section, we review some fundamental concepts and frameworks about creativity and their relations to creativity support tools that have been built upon these definitions. We further discuss the applicability of the larger theme of human-machine collaboration to these tools given the highly human-driven nature of creative activities in the past, and our dissertation can be considered as an investigation of human-AI collaboration in this space. Note that this section focuses primarily on the background of creativity support tools. We will further describe specific creativity support tools developed by researchers relevant to this dissertation in Chapter 3.

2.2.1 Definitions of Creativity

The fundamental paper on Creativity Support Tools written by Shneiderman presents three definitions of creativity that influenced the types of support tools being developed. We provide a brief overview of these definitions in this section and refer interested readers to the original article [175] for an in-depth discussion on this topic.

The first definition of creativity originates from *structuralists*, which 'believes people can be creative if they follow an orderly method'. This definition results in creativity support tools that help people manage their information and artifacts explored in creative processes,

and help them quantify their progresses towards solutions. An example tool that falls into this category would be a structured exhaustive search and optimization tool for the parameters of light-bulb designs.

Inspirationalists instead define creativity to originate from departure from familiar structures. Exploration of unrelated problems and viewing of random stimuli (such as photos) are encouraged by this definition of creativity. Some example tools that are inspired by this school of thought include design screenshot libraries and sketch-based interfaces that afford free-hand drawing.

The final school of thought on creativity originates from *situationalists*, which considers creativity to be primarily social, and investigates the social connection of creative individuals to friends, family members, and mentors. This definition of creativity has encouraged the development of social creativity tools such as emails and blogs, facilitating collaboration on creative problems.

2.2.2 Target Applications and Users

Given these definitions of creativity, Shneiderman further defines a genex (GENeration of EXcellence) framework on four phases of creative activities where support is needed [174]:

- *Collect*, in which users 'learn from previous work stored in libraries, the Web, etc.'.
- *Relate*, in which users 'consult with peers and mentors at early, middle and late stages'.
- *Create*, in which users 'explore, compose and evaluate possible solutions'.
- *Donate*, in which users 'disseminate the results and contribute to the libraries'.

With this framework of different phases of creative activities, we review some general and notable applications that support each of these phases proposed by Shneiderman. Note that as the area of creativity support tools is closely relevant to our dissertation, we include related work specific to sketching and prototyping processes in Chapter 3.

Examples of tools supporting the *Collect* phase include the families of browsing and searching tools through digital libraries. More recently, search engines (e.g., Google and Bing) have been frequently used as the first sources of inspiration for many creative activities.

In the *Relate* phase, emails and discussion threads can serve as great asynchronous tools to solicit feedback from the users' peers and mentors. These tools have lowered the time and cost of communication compared to non-digital methods, hence supporting and encouraging users to consult with peers and mentors more frequently.

A large number and variety of tools have historically been designed to support the *Create* phase. Authoring tools are a major type of creativity support tools aiming to support this phase. Some representative examples of these tools are Adobe Photoshop, Microsoft Office, and Apple Final Cut Pro. These tools improve workflows for composing creative artifacts including poems, illustrations, photos, and videos. Moreover, 'What-If' tools, such as

spreadsheets with macro support, are a family of software tools that support the exploration of possible solutions based on the structuralist perspective of creativity. For example, business planners and analysts can author macros in spreadsheets to quickly explore a variety of potential scenarios and solutions.

To help disseminate results created in creative activities, communication tools were designed to support the *Relate* phase, such as Email and Listservs. In addition, digital libraries, scientific journals, and newsletters can all be used to effectively distribute creative work.

The systems presented in this dissertation mostly support the *Create* phase, by modeling the creation process itself using DL models and systems. These systems allow users to more easily create sketches and prototypes with alternative methods of specifying their requirements (i.e., natural language). However, some of the related work also contributed by the author of this dissertation in UI and 3D model retrieval presented in Chapter 3, can also support the *Collect* phase and help users find relevant prior examples of designs.

Finally, we would like to note that while it appears the discussion about creativity support tools in this chapter primarily surrounds professional applications, amateurs also need to engage in creative activities. There have been prior works exploring tools supporting amateurs' needs which are often different from experts [32]. One notable area of research is end-user programming support [100]. This area aims to support non-experts to create usable computer programs customized to their own needs by themselves, and this paradigm has been extended and applied to creative use-cases [11]. Several systems introduced in this dissertation also aim to support non-experts' sketching and drawing process, such as Sketchforme (Chapter 4) and Scones (Chapter 5).

2.2.3 Human-Machine Collaboration and Mixed-Initiative Interfaces

Earlier creativity support tools were often developed in a way that these tools primarily take assistive and passive roles in users' creative processes. In the examples listed above, such as authoring tools (e.g., Adobe Photoshop) and communication tools (e.g., email), users typically take the *initiative* in performing the task, meaning that they are usually the sole party that proactively creates [38] and contributes to solutions [118] that drive the task forward [135].

On the other hand, humans and computational systems have been more actively collaborating in many domains, with machines contributing significant knowledge and effort to collaborative tasks at hand with the *Human-Machine Collaboration* paradigm. One of the earliest manifestations of this paradigm can be found in AURA (AUtomatic Reasoning Assistant), a system that assisted mathematicians in finding proofs automatically [40]. While AURA was originally designed to be an assistive system that validates humans' intuition in mathematical proofs, the actual workflow in-place showed that AURA had contributed new knowledge with its behavior in computational experiments. In more recent research literature, human-in-the-loop machine learning (ML) has gained extensive attention—researchers

have adopted this method to improve data quality of object annotations [163] and to design novel model architectures [171]. *Interactive machine teaching* was introduced as a new human-in-the-loop ML paradigm that allows humans to directly 'teach' target task knowledge to complex ML models, resulting in semantic and debuggable models [152]. Beyond making theoretical and technical advancements in the fields of Mathematics and Computer Science, the human-machine collaboration paradigm has also been applied to domain-specific computational systems to assist experts in diverse fields in their tasks. These systems have aid in performing surgery [137], monitoring health of building structures [131], moderating discussion content [86], and composing music [159]. We recommend interested readers to review proposed frameworks of human-machine collaboration [59, 125], and a taxonomy for these collaborative systems [176].

With such broad application of the human-machine collaboration paradigm, Human-Computer Interaction (HCI) researchers have built *mixed-initiative interfaces* [65, 74] that foster a human-machine collaborative approach towards interacting with their users. The LookOut system is an early example that can intelligently help users manage appointments between calendar scheduling and email correspondence [74]. With algorithmic and technical advancements in computational systems, recent approaches have built upon the mixed-initiative framework to develop mixed-initiative creative interfaces [38], supporting creative activities such as game-level-building [38], icon design [118], information discovery [97], and storytelling [118]. In these activities, humans and machines collaborate to determine the exploration process without either humans or machines making decisions alone and both contributing to the final outcome. In an example game-level-building tool, the AI would determine the playability of certain levels and hence steering the creation process.

This dissertation was inspired by the thread of research in both human-machine collaboration and mixed-initiative creative interfaces. With recent technical advancements in DL, we can introduce more effective and powerful mixed-initiative interfaces that can further stimulate users' creativity through *human-AI collaboration*. For example, the generation of a large variety of free-hand sketches was only made effective after DL was introduced, and tying these sketches to text descriptions also requires advancements in DL-based natural language processing techniques. We believe the introduction of text-to-sketch tools (i.e., Sketchforme in Chapter 4 and Scones in Chapter 5) in this dissertation can elevate the level of human-AI collaboration, providing users with further creative inspiration through the generation of realistic, coherent, and relevant sketches.

2.3 Transformers

Advancements in DL have led to new effective model architectures for handling *sequential* data and artifacts common to sketching and prototyping. The recent state-of-the-art family of DL models for handling sequence data in many domains is Transformers [184] that utilize the *attention mechanism*. This section will describe the attention mechanism and other technical components used in the Transformer architecture by walking through a concrete

example of modeling a sketch artifact. We will additionally describe several other common applications in the DL research community that are built upon Transformers. Note that this section only describes the core components of the Transformer architecture, and we refer interested readers to a complete description of the architecture in the original paper [184].

2.3.1 Notation and Task

We first introduce notations and the example task that help us better explain the technical details of Transformer models. Sketches and prototypes are commonly represented *sequentially*. For instance, one common way of encoding the sketching process is considering it as a sequence of pen events in a chronological order, where each event is the state of the pen including the displacement from the previous event and whether the pen was kept on the canvas so that it forms a drawn stroke on the canvas from the previous point. Alternatively, this event could be just a movement towards the starting point of the next stroke without the pen touching the canvas, or the end of the whole sketch sequence. Prototypes can also be considered as a sequence of semantic elements (e.g., UI elements including text fields and buttons, for UI/UX design) of various positions and classes on the canvas.

Without loss of generality, we consider a sketch with multiple pen events. As mentioned above, each pen event e can be considered as the displacement $(\delta x, \delta y)$ from the previous point, and a pen action event ($p \in$ **{down, up, end}**). We then encode a sketch S as a sequence of n sequential events:

$$S = \{e_i | i \in 1...n\}, \quad e_i = [\delta x_i, \delta y_i, p_i]$$

If we normalize the displacement coordinates against the size of the canvas, we can bound δx and δy between 0 and 1. The pen event p can be considered as a categorical attribute encoded with a one-hot vector. Alternatively, for a prototyping task, we can similarly consider a sequence of UI elements by replacing $\delta x, \delta y$ with the absolute coordinates of the elements on screen x, y, and replacing the action event with the semantic classification of the particular UI element (e.g., text field, button).

One critical example task we can then consider is the autoregressive modeling of pen events, which corresponds to the *next-event prediction* task. This task is important because solving it allows us to generate sketches from scratch by generating pen events sequentially, a novel capability first solved by DL in [62]. To formalize this task, we represent the probability of a certain pen event at time-step $i + 1$ (the next event) given all the previous pen events up to time-step i as follows:

$$P(e_{i+1} | e_1 ... e_i)$$

If we consider a Transformer model with the parameters θ, and attempt to model this distribution with the model, the definition of this distribution becomes:

$$p(e_{i+1} | e_1 ... e_i) = W_{\text{out}}(b_i), b_i = \textbf{Transformer}_\theta(e_1 ... e_i) \tag{2.1}$$

where b_i is the output embedding of the final layer at the last time-step (i) in the Transformer network, and W_{out} is the output projection layer that projects this embedding to form a distribution of the next pen events[4].

2.3.2 Building Blocks of the Transformer Network

We further explain the process of computing b_i by the Transformer model in Equation 2.1. Each Transformer model consists of l blocks/layers (**TBlock**). These blocks are stacked against each other and take the outputs from the immediate previous blocks as inputs. We use an additional index (k) to represent the k-th block in the Transformer, this is combined with the time-step index (i) to represent data at a certain layer for a certain time-step.

The first block in a Transformer ($\mathbf{TBlock}_1$) uses a projection layer W_{in} to convert raw inputs e to an embedding. It additionally adds a *Position Embedding* (**PE**) to the input to indicate the time-step position of the current input. This embedding is only computed based on the index value (i) of the input token. Formally, this is recursively defined as:

$$o_{i,k} = \mathbf{TBlock}_k(i_{i,k}), i_{i,k} = \begin{cases} W_{\text{in}}(e_i) + \mathbf{PE}(i) & \text{if } k = 1; \\ o_{i,k-1} & \text{otherwise (k = 2...l)} \end{cases} \tag{2.2}$$

Then, for the entire Transformer, the output of the entire network would be the output of the last layer at a certain time-step (i):

$$b_i = \mathbf{Transformer}_\theta(e_1 ... e_i) = o_{i,l} \tag{2.3}$$

We would like to remind the readers that we have so far only been considering the output of a single time-step (b_i), and as shown in the above formula, the only direct input dependency of this time-step is the raw input from the previous time-step at the first layer (e_i). We will further address how Transformer models consider other time-steps' inputs with the following description of a Transformer Block.

From the above formulas, it is thus apparent that the main structural building blocks of the Transformer model would be the input embedding layer (W_{in}), the position embedding layer (**PE**), and the Transformer Block (**TBlock**). The input embedding layer is usually modeled as a learnable, linear fully-connected layer. We explain the Transformer Block and the Position Embedding in detail in the following Sections 2.3.3 and 2.3.4.

2.3.3 Transformer Block

Each Transformer block consists of two main sub-components: 1) the multi-head attention layer (**Attn**), and 2) a feed-forward network layer (**FFN**). These two blocks are

[4]It is also reasonable in many cases to consider W_{out} as a part of the Transformer architecture, we separate this layer out for the better discussion of various choices of output parametrizations, such as those discussed in Section 2.4.

combined by stacking the feed-forward network layer on top of the multi-head attention layer:

$$\mathbf{TBlock}(i_i) = \mathbf{FFN}(\mathbf{Attn}(i_i)) \tag{2.4}$$

Note that the indexing of a Transform Block could be applied further to each of these sub-components, and that these blocks in a Transformer model learn l different sets of weights.

Multi-headed Attention

The crucial aspect of handling sequence data is to model the relation and structure between various data points in the sequence. In our example described above, this would be the relations between various pen events at different time-steps in the sketch sequence. Prior to the introduction of Transformers, recurrent neural networks (RNNs) have been handling such relations using a single, fixed-length internal state that is carried forward through the encoding of data at different time-steps in the entire sequence. However, as shown in the equations defined in Section 2.3.2, such dependency is not directly modeled as inputs to any of the main components of a Transformer model.

In Transformers, this relation is handled using key-query-value attention mechanisms. At a high level, the inputs of each time-step in the sequence (i_i) will be used to find other similar inputs (i_j) based on a similarity function learned by the model, and the subsequence output (u_i) at that time-step is determined by a weighted average of similar inputs and itself:

$$u_i = \sum_{j \in 1...i} A(i_i, i_j) \tag{2.5}$$

This allows the Transformer to model inter-dependence and structure in the sequential data by grouping relevant data points together. The specific mechanism (A) works as follows. Given all input data in the sequence up to i ($i_1...i_i$), the attention layer A computes three types of learned embeddings for a single input i_i. They are considered as *key* k_i with dimension d_k, *query* q_i with dimension d_k, and *value* v_i with dimension d_v specifically with their corresponding embedding layers. Note that this embedding layer is shared across all events.

Then, for a certain input i_i that produces the query q_i, other inputs are attended to by taking the dot-product between each query q_i against all other keys k_j (including itself). This will be considered as the similarity between that particular input (i_i) and other inputs ($i_j, j \in \{1...n\}$). These similarity values are then rescaled using a softmax layer (and the constant $\sqrt{d_k}$ for numerical stability) to form a probability distribution. The distribution parameters will then be used as weights to be multiplied with all the value embeddings v_j. Expanding this into a formula:

$$A(i_i, i_j) = \frac{\exp(q_i (k_j)^T / \sqrt{d_k}) v_j}{\sum_{k \in 1...n} \exp(q_i (k_k)^T / \sqrt{d_k})} \tag{2.6}$$

We can parallelize this computation by collecting the K matrix for all keys, where each row k_j is generated from i_j, $j \in \{1...n\}$, and similarly V matrix where each row is $v_j, j \in \{1...n\}$, forming the final output (u_i):

$$u_i = \textbf{softmax}(\frac{q_i K^T}{\sqrt{d_k}})V \tag{2.7}$$

$$q_i \in \mathbb{R}^{1\times d_k}, K \in \mathbb{R}^{n\times d_k}, V \in \mathbb{R}^{n\times d_v}$$

This computation is repeated on inputs at each time-step (e_i) across the entire sequence, forming the final output matrix O composed of all outputs corresponding to each time-step in the sequence (o_i). Note that this computation can be further parallelized by feeding in the stacked Q matrix in the previous equation. In practice, each event is also composed of multiple m keys, query and value projections (i.e., $k_{i,a}, q_{i,a}, v_{i,1}, a \in \{1..m\}$) referred to as *$m$ attention heads*, but this can also be parallelized in modern computational architectures. We omit the description of both parallelization steps for clarity of our explanation of the attention mechanism.

This weighted output is then added to the original inputs (i.e., residual connection [63]) and then normalized using Layer Normalization [8], generating the final output of the attention layer (a_i):

$$a_i = \textbf{Attn}(i_i) = \textbf{LayerNorm}(u_i + i_i) \tag{2.8}$$

Feed-Forward Network

The outputs of the multi-headed attention mechanism (a_i) are then further processed through a feed-forward network. This network consists of typical fully-connected layers of various configurations. The original Transformer uses two linear transformations and a single ReLU layer, followed by residual connection and layer normalization. Using this as an example, the final outputs from the feed-forward network (o_i) are:

$$o_i = \textbf{FFN}(a_i) = \textbf{LayerNorm}(\max(0, a_i W_1 + b_1)W_2 + b_2 + a_i) \tag{2.9}$$

Other activation functions and configurations of normalization layers have also been explored. Prior works have added normalization at the input side instead of at the output side and used GeLU as the activation function instead of ReLU [148].

2.3.4 Position Embeddings

A significant characteristic of only using attention mechanisms to model sequence dependencies in Transformers is its lack of the notion of ordering. This is because all inputs are fed through the same embedding layers and correlated with dot-product-based attention layers, which is an inherently unordered computation. This is in stark contrast to prior

approaches, where orderings are either preserved through the hidden state in RNN-based approaches with inputs processed sequentially, or are preserved through the model architecture in Convolutional Neural Networks using fixed windows. However, the ordering information remains important in solving many tasks.

To alleviate this issue, the ordering information is injected into Transformers using a position embedding layer which gets added to the other inputs of the Transformer. More recent architectures have implemented a learned position embedding, meaning that it learns an embedding layer ($W_{\mathbf{PE}}$) that converts each position in the sequence to a unique and learned d_e-dimensional vector. This means the embedding layer would have learnable weights of size $n \times d_e$, with n being the maximum length of the sequence data (or maximum 'context window' length for a particular task). We can then consider the conversion as retrieving the row embedding weights at row i for each time-step i:

$$\mathbf{PE}(i) = \text{row}_i(W_{\mathbf{PE}}), \quad W_{\mathbf{PE}} \in \mathbb{R}^{n \times d_e} \tag{2.10}$$

Other options of position embeddings have also been investigated. Some notable examples include fixed multi-frequency sine-cosine functions [184] and relative position embeddings [173].

One important aspect of the position embedding component is that the embeddings do not have to be assigned with time-step index values of the inputs. Other relevant information to the attention mechanism, such as the absolute x and y coordinates of pen events in the sketch sequence, can be similarly used for position embeddings to encode similarity and proximity in spaces other than time. Moreover, a special case of this consideration of position embeddings is the *lack* of position embeddings, in which we do not consider any ordering of the inputs. This could be useful for encoding set-based inputs in a *permutation invariant* manner.

2.3.5 Practical Concerns

There are several practical concerns and guidelines towards successfully training Transformer networks. We document some of these practices that we found to be helpful during our investigation in this section. We found that a linear learning-rate warm-up and the learning rate guidelines introduced by the original Transformer paper [184] are often helpful for training. We use the Adam optimizer [99] for most of our trained models, and found that label smoothing and weight decay regularization are also helpful for developing a robust model.

2.3.6 Large Language Models

Building upon the Transformer model architecture described above, researchers have scaled up this general architecture and have trained gigantic models on massive amounts of data, resulting in impressive results in the direct modeling task, downstream tasks supported

by pre-trained embeddings, and even in zero-shot tasks more recently. We believe this is the result of widely available text data from the internet, the low label requirement of *self-supervised* training paradigms employed by these large language models, and the great scaling ability of the Transformer architecture. We survey some of these models and training paradigms in the remainder of this section. Many of these models are trained to model language data, which we would use as an example to explain the core ideas in this body of research.

Training Paradigms

There are two general training paradigms commonly utilized by large language models: masked language modeling and autoregressive language modeling. We first consider a data sample that consists of a sequence of words (e_i), forming a text piece S (we used the same notation as the sketch example above for simplicity).

The first paradigm is *masked language modeling*, which randomly masks out a certain percentage of time-steps. We first consider a set of masked time-steps M such that $e_j = \mathbf{0} | j \in M$. Then, at a high level, the model is trained to maximize the probability of all the masked tokens given the non-masked tokens. Formally, the Transformer is trained to model the distribution of text at time-steps in M using the information from all other unmasked tokens, which is to maximize:

$$\mathbb{E}_{j \in M}[P(e_j | \{e_i | i \notin M\})]$$

The first representative example using this training paradigm is the BERT model [39].

The second paradigm is *autoregressive modeling*, which follows the more classic notion of language modeling where the model is trained to recurrently predict the next word given all the previous words in a particular data sample (i.e., the same piece of text). Our example task of sketch prediction (described in Section 2.3.1) also follows this training paradigm. The Transformer is trained to model the distribution of text at a particular time-step i with all information of inputs in previous time-steps $\{e_j | 0 \leq j < i\}$. Formally, given a text sample of n tokens (time-steps), the model is trained to maximize:

$$\mathbb{E}_{i \in 1 \ldots n}[P(e_i | \{e_j | 0 \leq j < i)]$$

Note that to achieve this "backwards-looking-only" dependency, the keys of future time-steps are masked (commonly named as masked self-attention) during training so the model could not 'cheat', as the ground-truth is available if the sequence is not masked. The outputs of this model are also shifted to the future by one time-step, in order to train the model to predict data at the immediate future time-step given all data of past time-steps. Representative examples that use this training paradigm include the original Transformer model [184] and the GPT-series of models [16, 148].

The advantage of these two training paradigms is that they do not require any additional explicit 'labels' of downstream tasks, such that the models are self-supervised by large

amounts of plain text data that is already available from the internet. Thus, crawling text data from the internet has become a common method for developing datasets for training these models.

Large-Scale Models

With the training paradigms outlined above, researchers scaled up the model architecture to include an extremely large number of parameters, and trained them on large amounts of data. One representative model, GPT-3, contains 170 billion parameters and is trained on 300 billion text tokens of web text data [16]. The underlying dataset is even larger than the amount of data used to train the model, which means the model had not seen all data examples once (or had not gone through a single epoch[5]). GPT-3 shows highly impressive performance on numerous downstream applications without explicit supervision, and larger models (up to 540 billion parameters [26]) and models with training data of different focuses (e.g., conversational data [180]) have since been trained and have successfully demonstrated the effectiveness in wider use-cases of this class of models.

Downstream Applications

From large models trained on both masked language modeling and autoregressive language modeling, researchers have either fine-tuned or performed few-shot prompting to solve a number of downstream tasks. Fine-tuning refers to starting with a trained large model and continuing to train it using specific objectives and datasets that a particular downstream application requires. Few-shot learning, on the other hand, works with an already-trained model without further training it, but changes the specific input format (usually referred to as 'prompts') and extracts solutions to tasks from the same text generation task as the original language model.

Taking the example of a downstream task of sentiment classification of whether a sentence is positive or negative, a fine-tuning approach involves taking the output embedding of the last word in the sentence, and training a classification head (and potentially modifying the original large model's weight) to output a single positive or negative label for the sentence as the final output. Few-shot learning, on the other hand, involves first giving a few examples to the model in the format '(sentence A) is (positive / negative). (sentence B) is (positive / negative).', then inputting '(sentence C is)' along with the examples to the model, and measuring whether the probability of the word 'positive' or 'negative' is higher for the next word. This measurement will effectively become the classification result at the end, and usually no model weights are modified in this paradigm.

Using these paradigms, researchers have first explored a wide variety of Natural Language Processing (NLP) tasks, including sentence completion [138], story-ending selection [130] and open-domain question-answering [92]. BERT [39] and GPT-2 [148], two earlier large language models using the two different aforementioned training paradigms, had exceeded the then

[5]Epoch is defined as a single pass over the entire training dataset by ML models.

state-of-the-art performance on almost all tasks through fine-tuning. More impressively, the larger GPT-3, which was trained with more data, outperformed fine-tuned models in some tasks without fine-tuning using the few-shot learning paradigm, demonstrating the great ability of these models in achieving their original design goal—learning general knowledge about natural language—with merely a single self-supervised training objective and a large amount of data. Beyond natural language, most recent works have expanded the large-language model paradigm to other domains. These domains include image modeling [24] and text-to-image generation [151], with Transformers demonstrating impressive results in both cases.

2.4 Mixture Density Networks

Following the consideration of model architectures, the choice of output parameterization is also important for solving generation tasks with neural networks. While other sequence modeling tasks, such as natural language processing, frequently use categorical distribution and cross-entropy loss since these tasks involve picking discrete text tokens out from a vocabulary, our task is slightly different: part of our task involves modeling and generating continuous coordinates (i.e., displacement of pen events).

A naive approach is to directly predict these coordinates using deterministic, real-valued outputs and train our model with mean-squared error. However, our task is inherently *under-specified*—there could be many possible outputs given a single input, and using a deterministic output will resolve to the mean coordinates. This means the deterministic outputs might not be representative of actual sketching actions (i.e., two options of pen strokes with similar length but opposite directions would resolve to a near-zero prediction).

One of the approaches we take is to model our strokes with the well-established Gaussian Mixture Models (GMMs). At a high level, the method constructs multiple Gaussian distributions around a single prediction (e.g., a continuous displacement value) and models the final distribution by taking the learned weighted average of multiple learned Gaussian distributions. This ensures the model is able to handle variations in the output space while leaving flexibility on the magnitudes and directions of the variations through mixtures of potentially different Gaussian distributions. Most importantly, GMMs allow us to reasonably bias the model to consider pen events and strokes with comparable displacement values as similar. However, since the task itself is still highly complex, we parameterize these GMMs with the outputs of our trained neural networks. This is an established approach in prior work named *Mixture Density Network* [13] and has been shown to successfully model sketches and handwriting.

Using the same example we established earlier, we consider a single ground-truth data point e_i and the associated predicted output embedding from the Transformer $b_i = \mathbf{Transformer}_\theta(e)$. Note that this is the same output embeddings computed from the previous Equation 2.3. From this output embedding (b_i), we need to model both a categorical distribution for the action event (1 of 3 options, which has 3 parameters) and mixtures

of Gaussians for two displacement values (i.e., δx and δy which are the displacement on the X- and Y-axis). We found that it is often sufficiently complex and more numerically stable to model mixtures of univariate Gaussian distributions for each axis independently, hence ignoring co-variances between δx and δy (practically, the co-variances might already be modeled by the Transformer itself). For each coordinate, we model its distribution with r different weighted Gaussian distributions (i.e., mean, variance, and mixture weight for each Gaussian distribution). Hence, the likelihood of δx and δy can be defined as follows:

$$p(\delta x, \delta y) = p(\delta x)p(\delta y) \tag{2.11}$$

$$p(\delta x) = \sum_{j=1}^{R} \Pi_{x,j} \mathcal{N}(\delta x | \mu_{x,j}, \sigma_{x,j}), \quad [\Pi, \mu, \sigma] = g_i = a(W_{\text{out}}(b_i)) \tag{2.12}$$

This results in three parameters for each distribution (mean μ, variance σ, mixture weight Π), with two coordinates $\delta x, \delta y$ and r mixtures to be modeled, resulting in $(2 \times 3)r = 6r$ parameters for the GMMs of coordinates and three parameters for the categorical distribution of the action event. Thus, we project our Transformer outputs b_i to $g_i \in \mathbb{R}^{6r+3}$ using the projection layer W_{out} as shown in Equation 2.12. Note that we add special activation functions (a) for each type of parameters, such that variances (σ) are passed through an exponential activation function to ensure they are always positive. The mixture weights (Π) and the categorical distribution parameters are passed through a softmax function to ensure they form valid probability distributions. The means were either passed through a *tanh* function or taken linearly depending on the stability of training.

We then train the entire neural network with the objective to maximize the log-likelihood of ground-truth coordinates $\log p(\delta x, \delta y)$ and the action events $\log p(p_i)$. To train on the joint distribution of multiple continuous coordinates ($\delta x, \delta y$ in this case), we assume that they are independent and take the sum of the log-likelihoods:

$$\log p(\delta x) + \log p(\delta y) = \log(p(\delta x)p(\delta y)) = \log p(\delta x, \delta y) \tag{2.13}$$

This parameterization allows us to model and reflect multiple different stroke options for sketching the same high-level object. We refer interested readers to a more detailed discussion of Mixture Density Networks [13], application examples, and training processes of them in [62].

2.5 Contrastive Learning

Relevant to modeling targeted artifacts from scratch, it is often also important to study the problem of explicitly drawing correspondence between the two modalities of interest. As an example, in the well-studied problem of text-based retrieval, researchers have used this method to draw correspondences between text captions and images. Understanding such

correspondences and subsequently developing quantifiable models of distances between artifacts in multiple modalities allow us to better measure performance of generative models that additionally need to follow certain multi-modal conditions (e.g., text-to-sketch generation). These models themselves also allow us to enable the retrieval of artifacts relevant to conditions, a similarly important interaction in supporting creative activities [70, 110, 157]. This is because using other artifacts as reference examples and sources of inspiration are important and common creative practices.

One recent area of research that directly develops such correspondence is *Contrastive Learning*. Instead of training DL models to classify certain data examples or to directly model distributions of data examples, contrastive learning defines training objectives over *paired* artifact data (these pairs could also be in two modalities of interest, e.g., image-caption pairs), and trains models to embed pairs of related artifacts similarly in their embedding spaces. This paradigm is explored by our novel text-to-UI retrieval model in Chapter 6 in this dissertation.

To formalize the contrastive learning paradigm, we use an example problem of drawing correspondence between UI layouts and text descriptions. Assume we have a UI layout (U_i) and its related text description (t_i), the direct goal of contrastive learning is to develop a model f such that $f(U_i)$ is more (numerically) similar to $f(t_i)$ than any other unrelated text descriptions. A more ambitious but indirect goal is for the model f to learn semantic similarity, so that groups of UIs and text descriptions perceived as similar in some aspects can be modeled by training on merely paired data.

2.5.1 Recent Approaches

To achieve the goal(s) mentioned above, one of the most popular recent approaches is InfoNCE [136, 178], or batch-softmax contrastive loss [54, 140]. We consider the same problem statement as the above section and extend our model (f) to have two sub-parts which are encoders dedicated for each of the modalities: f_U for a UI encoder and f_t for a text encoder. This framework is agnostic to any specific encoder architectures, but one can consider these encoders to be also implemented with Transformers. We then obtain the embeddings $f_U(U_i) = e_{U,i}$ for the UI, and $f_t(t_i) = e_{t,i}$ for the text description. Note that each e is a fixed-length, numerical embedding.

After having these embeddings, we consider a mini-batch $N - 1$ of other paired UIs and text descriptions:

$$\{(U_j, t_j) | j \in 1...N, j \neq i\}$$

This produces a corresponding set of embeddings $\{e_{U,j}, e_{t,j}\}$. At a high level, we train the encoders f_U and f_t to produce similar embeddings for all paired U_i and t_i, while maximizing the distances between the embeddings of non-pairs (as mentioned by our direct goal earlier). We minimize the following loss function to train the encoders:

$$L(f_U, f_t) = -\frac{1}{N}\sum_{i=1}^{N}\Big(S(e_{U,i}, e_{t,i}) - \log \sum_{j=1, j\neq i}^{N} \exp\big(S(e_{U,i}, e_{t,j})\big)\Big) - \frac{1}{N}\sum_{i=1}^{N}\Big(S(e_{t,i}, e_{U,i}) - \log \sum_{j=1, j\neq i}^{N} \exp\big(S(e_{t,i}, e_{U,j})\big)\Big) \quad (2.14)$$

where $S(x, y)$ is the dot-product[6] between x and y. One advantage of this method is that it allows the models to consider many pairs of dissimilar embeddings. This improves the efficiency of training and final performance of trained models.

2.5.2 Applications

This method has since been applied to many domains. Most notably, CLIP [149] uses this training paradigm to correspond between text and images from Transformer-encoded images and text pairs. Subsequent research has found CLIP to be highly effective in encoding not only natural images, but also sketches and artistic images. Moreover, CLIP has been shown to be effective in guiding unconditional generation models with its embeddings to follow certain conditions in the form of CLIP embeddings, demonstrating the great coverage and representation power of CLIP.

2.6 Sketch and Prototype Datasets

All of the aforementioned DL advancements would not be effective without the support of corresponding datasets. Towards the end of this chapter, we survey several important datasets that are relevant to our areas of interest: sketching, text-guidance, and UI layouts.

2.6.1 Sketching Datasets

To support DL-based approaches towards sketch comprehension and generation that rely heavily on large amounts of data, researchers have crowdsourced sketch datasets of individual objects. Most of these datasets have focused on individual instances that correspond to either natural images or general semantic classes. The Quick, Draw! [91] and TU-Berlin [42] datasets consist of human-drawn sketches for 345 and 250 object classes respectively. SketchyDB provides paired images and simple sketches for retrieval tasks [168]. These datasets have been used to enable sketch generation [62] from scratch. They have also been used to support sketch-based image retrieval [166], a well-researched modeling task and interaction of finding corresponding visual content to sketches also relevant to UI design [78].

[6]Dot-product in this formulation measures similarity, which is the opposite of distance, and hence we maximize it in the loss function to minimize the distance between corresponding pairs.

Researchers have also explored beyond individual sketched instances to compile datasets of sketched multi-object scenes. The SketchyScene dataset consists of sketched scenes of pre-drawn objects transformed and resized by humans, as scene sketches are highly demanding for users to create from scratch [198]. The developers of SketchyCOCO started with the MSCOCO dataset and retrieved relevant instances from the Sketchy dataset, compiling them into complex scenes [48]. Nevertheless, none of these datasets consist of sketched scenes drawn by humans entirely from scratch. Most recently, the FS-COCO dataset consists of complete human-drawn sketches from scratch corresponding to photos in COCO [27]. This new dataset opens up great future opportunities in investigating research in end-to-end recognition and generation of complex scene sketches.

Related to complex sketches consisting of multiple artistic objects, the DiDi dataset introduced drawings of flowcharts and functional diagrams traced entirely by human users [52]. While users did not design the diagrams from scratch, they generated each individual sketch stroke based on procedurally generated diagram templates. In addition to general-purpose and non-expert sketches, the OpenSketch [58] dataset offers densely annotated sketches of product designs drawn by professional designers.

2.6.2 Abstract Objects with Text Descriptions and Conversations

Many research projects in this dissertation have employed a multi-stage approach, with the first-stage model only generating abstract object representations, followed by a second-stage model that generates sketches that fit individual object specifications. This approach was developed to utilize the existing parallel efforts in the research community in supporting abstract visual reasoning with natural language.

The three primary datasets that we have been able to repurpose are Abstract Scenes [197], CoDraw [98], and Visual Genome [103]. The Abstract Scenes dataset consists of pairs of text descriptions and scene representations, where each scene is represented by a set of abstract objects with their attributes and classes. These scenes generally are composed of natural objects such as animals, toys, and plants. They are created using a clip-art-based interface by crowd-workers based on the given text descriptions.

The CoDraw dataset was developed upon scenes with the same graphical style as those in the Abstract Scenes dataset [98]. However, instead of single text descriptions, this dataset contains editing and interaction sequences of scenes corresponding to conversation rounds between a *drawer* and a *teller*. In each round, the *drawer* maintains the same canvas and can choose and manipulate scene objects on and off the canvas, while the *teller* sees a ground-truth scene that they need to guide the *drawer* to recreate with natural language instructions in a conversational interface. This creates the sequence of instruction/scene action rounds for the conversational authoring of abstract scenes. We will describe this dataset in detail in Chapter 5 when we describe our system Scones.

In an attempt to support a greater number of classes and a larger variety of object

specifications, we also investigated the use of Visual Genome, a natural image dataset, to extract text and scene object pairs from these image-caption pairs. Each image in the Visual Genome [103] is divided into regions, and each region contains scene-graph information of the position of various objects in the graph and the corresponding text description (e.g., "a boat under a bridge"). Thus, this allows us to extract relevant objects and text descriptions that originate from natural images and use them to support the first-stage models in our early Sketchforme system to generate object specifications from text descriptions. We will similarly describe our usage of this dataset in detail in Chapter 4.

2.6.3 UI Layout Datasets

To explore the benefit of DL models in various UI/UX design tasks, several large-scale datasets of UI layouts have been recently developed. The primary dataset of mobile UI designs used in many fields is Rico [34], which contains more than 72,219 UIs from 9,772 Android apps. Each UI in Rico is collected from crowd-workers' interaction with a particular Android app (on a real device hosted in a research lab) captured using the techniques introduced in ERICA [33]. For each UI, the dataset includes the screenshot, the user action that led to the current and next UI in the sequence of interaction (since these are real user interactions by crowd-workers), and the complete view hierarchy that contains attributes of each UI element as well as the tree structure between them in the UI.

Since the introduction of Rico, many additions to the dataset have then been introduced. Researchers have annotated interactable UI elements as one of the semantic types using neural networks [122], detected and removed mismatched view hierarchies in RicoSCA [117], collected paired sketches to UI screenshots in Swire [78], and crowdsourced text captions of UIs [186]. Most recently, the first large-scale mobile app dataset for iOS apps has also been introduced [194]. All of these datasets have supported recent research progress towards conditional and unconditional UI understanding and generation tasks. We will further discuss the datasets that are used by our proposed text-to-UI system in Section 3.4.

Chapter 3

Related Work

The processes of generating creative artifacts, including sketches and prototypes, have been extensively studied by prior related works in multiple research communities. The Human-Computer Interaction community has a long history of prior work in assisting and educating users to create better sketches and prototypes, and the Machine Learning (ML) community has applied most recent advancements in deep-learning (DL) model architectures to the generation of images, sketches, vector graphics, and 3D models. In this chapter, we review some representative works from each of these research communities on each of these topics.

3.1 Natural and Artistic Image Generation

While generating realistic natural and artistic images has been a longstanding research goal, it is only until recent advancements in DL model architectures have these models been able to generate realistic images almost indistinguishable by human raters. We begin this section with reviewing the more constrained task of style transfer which follows concrete content and style input images, and expand our survey to unconditional and text-conditional image generation from scratch.

3.1.1 Neural Style Transfer

The discovery of *Neural Style Transfer* [50] has created a myriad of new research directions and artistic applications due to its impressive transfer performance. This proposed method is able to take a *'content'* image and a *'style'* image and synthesize a new image that retains the content in the 'content' image, such as scene objects, while presenting it in the artistic style of the 'style' image. An example of this task would be to stylize a photograph ('content' image) of a famous landmark in the style of an oil painting ('style' image). This allows artists and non-expert users to easily re-express visual content artistically, which previously required significant fine-art expertise and/or manual effort.

The core component of this method is a Convolutional Neural Network (CNN) typically trained for classification tasks. To perform transfer, both 'content' and 'style' images are first passed through the CNN, in which 'content' and 'style' losses are computed based on internal features of the CNN of respective images. Then, instead of optimizing the weights of the CNN, Neural Style Transfer optimizes the content image while freezing the CNN's weights, so that the optimized image now contains similar feature statistics (according to the 'content' and 'style' loss) to both the 'content' and 'style' images. We refer interested readers to the original Neural Style Transfer paper [50] for the technical details of this process.

Based on this framework, subsequent works have introduced better formulation of losses, applied neural methods to improve the optimization process, and transferred beyond natural and fine-art images as content and style artifacts [89]. Most related to this dissertation are recent works that transfer styles to Fashion designs [88] and UI designs [46]. Each of these works offers improvements to the structure and content of output images to accommodate design-specific needs, such as maintaining structural integrity of 'content' UI screenshots in the generated UI screenshots. One of the more significant improvements in solving the style transfer task comes from the novel model architecture of Generative Adversarial Networks, which presents a more general formulation of the task that we will cover in detail in the following section.

3.1.2 Generating Images with Generative Adversarial Networks (GANs)

Although Neural Style Transfer offers fresh perspectives of existing visual content, its optimization process relies on a 'starting point' (which is typically the 'content' image) for the final product to look reasonable, which means it relies on the real input content and artistic images. To produce images *truly unseen* in the datasets, researchers developed Generative Adversarial Networks (GANs) [56], a model architecture that has gained significant popularity in the research community for generating highly realistic yet novel images.

The core architecture of GANs contains two neural networks, the *generator* and the *discriminator*. The generator takes a random vector and generates an image that is also the final output of the model. The discriminator takes the image generated by the generator, along with a number of real images, and tries to differentiate real images from the generated ones by outputting a probability of 'realism' for each image. During training, the generator is trained to maximize the probability of its output image being considered as real by the discriminator, while the discriminator is trained to minimize the generator's output image's realism probability while maximizing the realism probability of the real images. As such, the only data required to train models of this architecture are large numbers of images from the target distribution of the generated images, such as natural images, with no labeling requirements for the images. These neural networks are optimized with conflicting objectives and analogically act as a counterfeiter and a cop, such that the generator learns to generate better 'counterfeit' images while the discriminator 'cop' improves its ability to detect

counterfeits. Another novel advantage of this training paradigm over existing work is that it does not explicitly define a loss objective directly over generated images during training, such that this objective could be difficult to define for data with complex, high-dimensional correlations (e.g., pixel data of natural images). This paradigm only relies on a discriminator to serve as the 'loss function', in which the discriminator learns to output single realism probability values for the artifacts from seeing other real artifacts.

The GAN architecture has opened up a new avenue of research work, particularly for image generation since the image distributions are typically difficult to model with prior approaches. The most straightforward application is unconditional image generation, where images have random characteristics that could be related to any aspects of the original dataset. DCGAN is one of the earlier practical works that introduced architectural improvements over GANs to enable the generation of natural images [147]. These improvements include novel activation functions and novel tuning of convolution patterns over the ones used in the original GAN experiments. Since then, many further improvements have been made to the GAN architecture to improve training stability, generation resolution, and output quality. Most recently, multi-stage architectures with large-scale training have been used to achieve state-of-the-art performance [15]. While systems introduced in this dissertation do not utilize the GAN architecture for generating visual content, they have been used by other researchers to generate sketches [51] and UI layouts [115] (further described in Section 3.4.2), with competitive performance that could serve as strong baselines and alternative technical approaches for current and future work on sketch and prototype generation.

3.1.3 Conditional Image Generation with GANs

A more relevant variant of the GAN architecture to this dissertation than the unconditional image generators introduced in the previous section are *conditional* models, in which users can have some form of control and guidance over the content and/or style of the generated artifacts. Towards this goal, *conditional GANs* (CGANs) have been developed by adding the desired form(s) of condition as additional inputs to the generator. Label-based generators use class annotations that exist in large-scale image datasets (e.g., ImageNet [36]), such that given a class (e.g., an orange), models that are trained to generate multiple types of objects can be controlled to only generate images of that particular class (e.g., images of oranges) [126]. Text-to-image systems, which address highly similar tasks to all proposed systems in this dissertation, have also been developed to generate images from text captions [155] using image-caption datasets (e.g., MSCOCO [121], Conceptual Captions [172]). Text descriptions offer more flexible yet fine-grained control over the image content, such as 'a small boat under a bridge' could dictate both object types, sizes, positions, and inter-object relations in the target images. The text-to-image generation process has also been further improved by combining contrastive learning (Section 2.5) to achieve state-of-the-art performance [193].

One possible approach for text-to-sketch generation, a primary task investigated by this dissertation, that arises from text-conditional GANs and Neural Style Transfer (Section 3.1.1)

is to combine these two approaches: applying style transfer to synthetic images generated by CGANs based on text descriptions. However, this approach likely results in realistic, detailed sketch-style images which contain distracting artifacts. On the other hand, our proposed systems in Chapters 4 and 5 focus on synthesizing abstract sketched scenes from scratch that capture fundamental ideas from messages communicated by the scenes.

Label-based generation and text-based generation are examples of using conditions that represent high-level semantics of the target images. Meanwhile, researchers have also developed models that enable low-level controls over image generation processes. One common method of realizing low-level control is through a sketching process, which allows direct annotation of semantic details of the generated artifact. GauGAN is a recent architecture that allows users to sketch out specific parts of the to-be-generated images as particular semantic classes (e.g., sky, water) through the use of specially color-coded strokes, then conditionally generate realistic images based on these annotations [141]. Other than conditioning on single modalities, researchers have also combined multiple methods of controls introduced above, such as both text and sketches, using datasets with multi-modal information to achieve multi-modal control over image generation [102].

For an even finer-grain control for generation, pixel-level conditional GANs were extensively explored by researchers, which can be considered as a more general case of style transfer such that this type of GANs can translate any types of input images to any types of output images with sufficient data support. The pix2pix network trains a U-net model according to the GAN paradigm to generate a new image conditioned on another image of identical resolution [84]. This can be applied to not only style transfer applications to stylize images, but also in transformations between more disparate domains such as transferring dense labels of objects in street scenes to images, sketches to photos, day-time photos to night-time photos, and aerial satellite images to map annotations. Subsequent work also combined pix2pix with class-conditions to enable multi-class sketch-to-image translation [53], and color-conditions to control the colors of generated images [167].

This initial pix2pix framework requires paired images since it is trained to perform one-to-one translation. CycleGAN significantly relaxes this requirement to perform translation without paired data [195]. While pix2pix still optimizes the generated output to match the ground-truth output of paired data, CycleGAN merely enforces correspondence using two generators that translate between the domains in each of the directions. This allows the use of a cycle-consistency loss, such that an image only needs to be consistent with itself after having been translated from the source to the target domain, and then back to the source domain. This allows transfer processes between domains to be learned only with unpaired image data in each of the domains, such that the model learns simultaneously to translate and follow the respected domain information without the use of paired data. In the case of CycleGAN, it only needs ImageNet (for natural images) and artistic paintings retrieved from online repositories to enable natural-image-to-art and art-to-natural-image applications. Subsequent works also use similar paradigms to learn image translation between more than two domains [25] and enforce consistency in the embedding spaces instead of the image spaces [162].

3.1.4 Transformer-Based Image Generation

More recently, another model architecture, Transformers [184], has gained significant popularity among the DL community. Transformers are originally and specifically designed to handle sequence-based data, such that it encodes inter-element relations among items within a sequence using repeatedly stacked self-attention blocks. This results in high flexibility in encoding any correlations between data items and removes the need for hand-crafted connections and memory modules, such that the strengths of connections between data are entirely learned by Transformers. We refer interested readers to a technical description of Transformer models in Section 2.3 and the original Transformer paper [184]. Beyond domains where sequences naturally emerge which are suitable to be modeled by Transformers (e.g., natural language processing and sketch modeling), researchers have also found success in using Transformers to learn image distributions by treating images as flattened sequences of pixel data [24].

Building upon Transformers' success in modeling image data, DALL-E is a recent Transformer-based architecture that demonstrates significant success in generating realistic images from text descriptions [150, 151]. Given a pair of text description and image, DALL-E first discretizes the image pixel data and flattens it into a long sequence of image 'tokens'. Then, it combines both text and image tokens into a common vocabulary and autoregressively decodes image tokens. Some innovations required in DALL-E include a specially hand-crafted attention pattern that imitates spatial convolutions on the image, and an improved training process for stability. It is shown to be able to generate not only realistic images, but also abstract graphics that already exist in the large dataset that DALL-E is trained on, such as paintings and sketches. This approach is similar to the majority of the proposed systems in this dissertation, which utilize Transformer models to generate sketches and UIs from text descriptions.

3.1.5 Generative Image Diffusion Models

Diffusion Models have recently gained significant popularity given the introduction of DALL-E 2. This is a family of models that can serve as alternatives to GAN-based and Transformer-based models presented above for generating images from scratch. Recent experiments have shown that this family of models has been able to surpass prior approaches and achieve state-of-the-art performance in various image generation tasks, sometimes with even fewer model parameters.

The general idea of diffusion models can be explained by learning a progressive denoising process from random noise. During training, noise is progressively added to a training example using a Markov chain of *diffusion* steps, until the input is highly similar to random noise. The model is then trained to denoise (i.e., taking the backward steps) stepwise in this process, effectively learning how to re-create the input image. After this model has been trained, at test-time we can then use the model to generate realistic images from a randomly sampled noise input similar to the ones the model has seen during training time (which do

not require any image data). Note that diffusion models only refer to the general paradigm of training, and do not constrain the exact model architectures used to perform the denoising steps.

This family of models has since been applied to enable state-of-the-art unconditional image generation and conditional text-to-image generation [150, 165], outperforming existing methods by a wide margin, presenting itself as a strong technical approach for future generative systems for visual content to utilize as their backbone models.

3.2 Sketch and Vector Graphics Generation

There are a number of prior works that explored the generation of sketches and abstract graphics closely related to the topic of this dissertation. These works take a different form of generation processes than GAN-based approaches, where sketches and vector graphics are treated as sequences of pen events or vector element parameters. Modeling graphics in this sequential form has the additional benefit of modeling the temporal aspect of creative processes that these sketches and graphics are created in, such that pen strokes and graphical elements are generated in chronological order. As described in the earlier Section 3.1.4, such sequence modeling paradigm also lends itself to the recent success of Transformers, which have shown to be particularly effective for modeling sequential data.

3.2.1 Sketch Generation

Sketches are some of the most commonly used visual artifacts in art, science, design, and engineering, due to their abstract yet highly expressive nature allowing the communication and materialization of artistic and functional ideas. The abilities of computational systems to understand and generate free-form sketches have been significantly augmented by recent advances in DL due to the highly abstract and free-form nature of sketches that still retains core patterns in the strokes about the expressed ideas.

The most significant recent advancement in sketch generation is Sketch-RNN [62], which models sketches from Quick, Draw! [91], a dataset that contains free-form sketches of various individual concept classes drawn by crowd-workers in under 20 seconds. Sketch-RNN models sketches as temporal sequences of geometric offset points and pen action events using LSTMs. The outputs of the LSTMs are used to parameterize Gaussian Mixture Models (GMMs) so that a distribution of offsets is created for each point, in order to model variations of sketches. Using similar paradigms, Sketchformer improved upon Sketch-RNN by replacing the LSTMs in Sketch-RNN with Transformers as the core model architecture and investigated various discretization strategies for the offset points [156]. CoSE introduced a novel hierarchical architecture for generating sketched diagrams [3]. Two systems introduced in this dissertation, Sketchforme and Scones, extend beyond generating individual objects to investigate text-conditioned generation of sketched scenes. These systems will be described in detail in Chapters 4 and 5.

Furthermore, other than training explicitly on sketch data, researchers have developed differentiable renderers that constrain the model output space to only be sketch strokes while being able to optimize against pixel-based data. A notable recent example of this is CLIPDraw, which uses a CLIP-based architecture to compute a loss value between a generated 'sketch' and a particular text condition [47]. The gradients of this loss are propagated back to the sketch strokes and subsequently model weights, so the models can learn to generate sketches conditioned on text descriptions without any sketch data.

3.2.2 Vector Graphics Generation

Apart from sketches, other important elements of visual design such as fonts, icons, and symbols can take a vector-based format (e.g., SVG) that can be represented as sequential data. DeepSVG is a Transformer-based model that is able to generate SVGs from scratch and interpolate between various vector graphics in its latent space [21]. It also introduces the SVG-Icons8 dataset which consists of 100,000 SVG icons. The model learns to generate these icons by modeling each SVG element as individual parts in a hierarchical Transformer-based architecture, and subsequently generating each of the parts separately. Similarly, scale-invariant representations are developed for fonts in vector formats. Further, to bridge vector and pixel-based data formats, Im2Vec introduced a novel method (similar to CLIPDraw [47]) to allow generated vector-based graphics to be differentiable and optimizable against pixel-based graphics [154]. This allows models to learn to reconstruct pixel-based graphics in a vector-based format and enables interpolation between various vector graphics in its latent space, without requiring any supervision of vector-based drawings. Vector graphics are often modeled in a similar manner as sketches as sequences of composition events on the canvas. Hence, we believe some of these research contributions towards vector graphics generation might be repurposed to augment sketch generation in the future and vice versa.

3.3 Automatic Sketching Tutorial and Assistance

Beyond modeling the sketching processes with novel DL models, prior works have also studied humans' sketching processes, and attempted to augment them with automatically-generated and crowdsourced drawing guidance. ShadowDraw [112] and EZ-sketching [179] use edge images traced from natural images to suggest realistic sketch strokes to users. The Drawing Assistant [83] extracts geometric structure guides to help users construct accurate drawings. PortraitSketch [190] provides sketching assistance specifically for facial sketches by adjusting geometric and stroke parameters of user-created sketches. Researchers also developed crowdsourced web applications to provide real-time feedback for users to correct and improve sketched strokes [119].

In addition to assisted sketching tools, researchers also developed sketching tutorial systems to improve users' sketching proficiency. How2Sketch [68] automatically generates multi-step tutorials for sketching 3D objects. Sketch-sketch revolution [45] provides first-hand

experiences created by sketch experts for novice sketchers. We believe the interactions developed by these tools and assistants can be augmented by technical advances we make in this dissertation, and help inform potential improvements to and applications of systems introduced in this dissertation.

3.4 User Interface Prototype Retrieval and Generation

Beyond studying non-expert sketches and general natural and artistic images, researchers have also investigated domain-specific creative applications. One of the most studied areas is UI design, with significant research conducted on dataset creation, example retrieval, and layout generation.

3.4.1 User Interface Retrieval

UI retrieval is an important interaction that can be applied to design workflows. Designers retrieve UI examples to gain inspiration, explore viable alternatives, and evaluate potential solutions [14, 70]. Recent approaches towards UI retrieval have utilized DL models to understand design-related concepts within UIs. These approaches are often advantageous over simpler methods (e.g., keyword matching), since features that correspond to design-related concepts are often hidden from surfaced text and/or UI attributes. The application of DL models towards enabling effective retrieval typically involves two main steps: 1) developing a DL-based encoder that generates meaningful embeddings for UIs and relevant artifacts, such as text or sketches; 2) performing retrieval via nearest-neighbor-search between the embedding of the query and the pre-computed embeddings of examples in the corpus. The second step can be trivially accomplished using common numerical distances (e.g., Euclidean Distance), and optimized using general nearest-neighbor-search techniques such as locality-sensitive hashing. Hence, we review several approaches towards encoding UI examples as fixed-length, numerical embeddings for retrieval corresponding to the first step of the process outlined above.

The original Rico paper proposed an auto-encoder-based approach on heatmaps of clickable and non-clickable UI elements as an early example of searchable semantic embeddings of UI designs. screen2vec proposed a multi-modal approach that uses text information, element attributes, and app descriptions of the source app of the UIs to learn embeddings that capture information in UIs holistically [116]. UIBert proposed to train Transformer encoders with various novel pre-training tasks, such as real/fake UI prediction and masked image prediction, to learn predictive features that compose particular UIs [10]. ActionBert, in contrast, uses data from neighboring UIs in the user interaction sequence in Rico to compute action-context-based embeddings of the UIs [64].

The development of embeddings can be extended beyond only using intrinsic information of the UIs to correlating with other relevant artifacts such as sketches and text descriptions

of the UIs. Swire, a work completed by the author of this dissertation, collects a dataset of paired sketches and UIs to learn sketch-semantic-centric embeddings through contrastive learning. This allows the model to inject semantic meanings of the UIs often contained in designer-drawn sketches into the UI embeddings [78]. Words2ui introduced in Chapter 6 of this dissertation explores a similar cross-modal retrieval paradigm, but with high-level text descriptions instead of sketches to enable the retrieval and creation of UI mock-ups from text descriptions.

3.4.2 Generating UI Designs

Generating UI designs is a more difficult task than retrieving UIs for computational models, because it requires the models to not only understand but also synthesize novel designs that are plausible and reasonable. Similar to generating images and sketches, recent advances of GANs and Transformers have enabled DL models to generate novel UI designs. Most DL-based methods consider UIs as sequences of element classes and positions, and train models to output semantically meaningful and interactable elements included in semantic annotations of Rico [122]. This removes distractors in full UI hierarchies while retaining fundamental and meaningful parts of the UI generation task. LayoutGAN [115] and LayoutTransformer [60] adapted GANs and Transformers respectively, to generate not only UI designs but also other types of document layouts. Variational Transformer Networks improved upon LayoutTransformer by using a VAE-based approach to model UIs in its latent space, enabling generation and interpolation simultaneously [7]. Alternatively, Neural Design Network considers UIs as graphs with nodes as elements and edges to be constraints between elements. This enables unconditional and constraint-conditioned UI generation, which can be helpful in realistic design applications where designers specify partial inter-element constraints [111]. In Chapter 6 of this dissertation, we will present Words2ui which further adds text conditions from the screen2words dataset in its generative model, to enable the novel interaction of generating UIs based on user-provided high-level text descriptions [79].

3.5 Sketch-Based Prototyping Tools

One type of tools that bridges between sketching and prototyping processes are rapid prototyping tools that support the generation of prototypes from sketches. This could be considered similar in spirit to paper prototyping [177], where prototypes can be quickly developed using elements drawn by hand on paper. The early work SILK [106] is the first system that allows designers to author interactive, low-fidelity UI prototypes by sketching on a software interface. DENIM [120] further enables web designers to prototype with sketches at multiple detail levels. More recently, researchers have integrated sketches with verbal text descriptions in crowdsourced sketch-based UI prototyping tools, a combination of modalities that this dissertation also investigates. Aiming to support more high-fidelity prototypes, researchers have developed tools that aim to directly generate UI interface code by detecting

sketched elements drawn by users [161, 187]. A potential future research goal for Words2ui (Chapter 6) is to add support for the sketch-to-UI interaction beyond the current text-to-UI interaction. This sketch-to-UI task was partially explored by the author of this dissertation in the form of sketch-to-UI retrieval in Swire [78].

Beyond the domain of UI/UX design, researchers have also developed sketch-based interactive systems supporting other design processes. DreamSketch [95] introduces a 3D sketch-based design interface that allows users to couple generative algorithms with sketch contexts for solving mechanical engineering design problems. Sketchsoup [6] automatically generates variations of users' input sketches that differ in perspective and/or structure, which helps industrial designers more exhaustively explore the design space during the ideation process.

3.6 3D Model Retrieval and Generation

Other than UI designs, researchers have also developed methods to retrieve and generate 3D models, a modality that is frequently used in industrial and mechanical design domains. These are two similarly important creative domains as UI design and sketching investigated in this dissertation. The first step towards supporting these applications with DL is by building large-scale datasets for these models. ShapeNet is an early large-scale dataset that contains publicly available mesh-only data of over 50,000 3D models [22]. The ABC dataset extends the scale to contain over 1 million models' original CAD files, curves, and patches [101], but does not contain semantic annotations and categories of the 3D models as ShapeNet does. PartNet focuses on offering multi-part annotations within each 3D model which enables part-based applications such as instance segmentation [128]. Most recently, the Fusion 360 Gallery is a smaller-scale dataset but contains action sequences designers originally used to create 3D models in the dataset, and additionally presents an environment for AI agents to interact with these 3D models [188].

These and other similar 3D modeling datasets have subsequently been used to develop retrieval and generation applications for 3D models. Various 3D model retrieval models have utilized contrastive learning to query for relevant 3D models in the dataset given a 3D model provided by the user. The most common features learned by these models are *appearance-based* that originate from considering multiple perspectives of each 3D model independently. Recent research work has also attempted to learn from the context [76] of the 3D models that can return results that are more functionally similar to the input query. Similar to UIs, retrieval modalities beyond original 3D models, such as sketch-based retrieval, have also been explored to allow users to query for 3D models using the versatile modality of sketching [114, 192]. Building upon these retrieval-based models that accept various modalities, the author of this dissertation has co-published a multi-modal search application for 3D models, supporting the use of natural language, individual 3D examples, and 3D assemblies to search for relevant 3D models in the dataset [105]. Beyond retrieval, composition processes of 3D models, including generation, have also been explored in prior works. These works include Sketch-to-3D-model generation [35], text-to-3D-object-generation [85], and reverse-

engineering of modeling sequences from 3D models [191], which could all be applicable to engineering design.

Chapter 4

Sketchforme: Sketch Generation from Individual Text Descriptions

Towards the goal of encouraging the wider adoption of sketches for various applications, we first investigate the generation of non-expert sketches guided by natural language. Having these systems produce diverse sets of sketches can significantly improve applications when it is time-consuming or difficult for users to create sketches. These interactions could be useful for domains such as language learning and communication.

Recent advances in neural-network-based generative models drastically increased machines' ability to generate convincing graphical content, including sketches, from high-level concepts. The Sketch-RNN model demonstrates that recurrent neural networks (RNNs) trained on crowd-sourced data can generate original sketches of various concept classes [62].

With the advancement in sketch-generation algorithms and the benefits of using sketches as outputs in interactive applications, this chapter introduces Sketchforme[1], the first system that is capable of synthesizing complex sketches for users while allowing them to maintain control over the sketches' content naturally using text descriptions. Sketchforme uses a novel, automated two-step neural method for generating sketched scenes from text descriptions. Sketchforme first uses its *Scene Composer*, a neural network that learned *high-level composition principles* from datasets of human-annotated natural images that contain text captions, bounding boxes of individual objects, and class information of the objects, to generate composition layouts of the scenes. Sketchforme then uses its *Object Sketcher*, a neural network that learned *low-level sketching mechanics* to generate sketches adhering to the objects' aspect ratios in the compositions. Finally, Sketchforme composes these generated objects of certain aspect ratios into meaningful sketched scenes.

We also build and evaluate several applications, including a sketch-based language learning system and an intelligent sketching assistant. These applications illustrate the potential value of Sketchforme in supporting novel sketch-based interactions (Chapter 4.4). In these applications, Sketchforme creates new interactions and user experiences with the interplay

[1]Sketchforme was also published as a conference paper at UIST 2019 [77].

between language and sketches. These features of Sketchforme are highlighted in Figure 4.1.

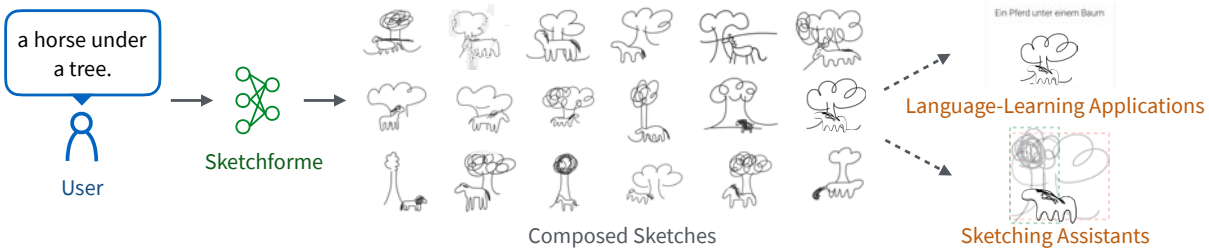


Figure 4.1: Sketchforme synthesizes sketched scenes corresponding to users' text descriptions to support interactive applications.

4.1 System Description

To support applications that afford sketch and natural-language-based interactions, we developed Sketchforme, the system that provides the core capability of synthesizing sketched scenes from natural language descriptions. Sketchforme implements a two-step approach to generate a complete scene from text descriptions as illustrated in Figure 4.2. In the first step, Sketchforme uses its *Scene Composer* to generate composition layouts represented by bounding boxes of individual objects. These bounding boxes dictate locations, sizes, and aspect ratios of objects in the scene. Sketchforme's *Object Sketcher* then uses this information at the second step of the generation process to generate specific sketch strokes of these objects in their corresponding bounding boxes. These steps reflect a fundamental process suggested in many sketching and artmaking tutorials, where the overall composition of the scene is drafted before filling in details that characterize each object [28, 49].

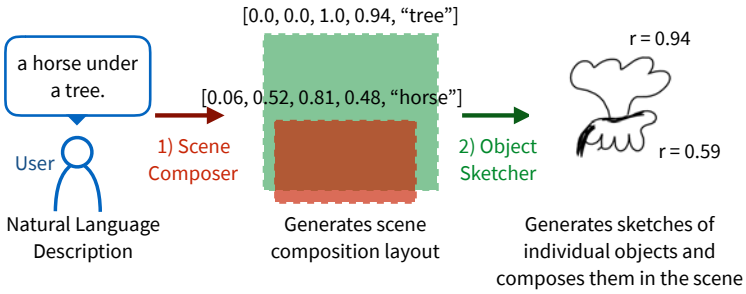


Figure 4.2: Overall system architecture of Sketchforme. Sketchforme consists of two steps in its sketch generation process.

By taking this two-step approach, Sketchforme is able to model high-level object relations critical to composing the scenes, enabling a multitude of applications that require such information. Moreover, this approach overcomes the difficulty for end-to-end sketch generation methods to capture global structures of sequential inputs [62]. End-to-end sketched scene generation also requires datasets of dedicated sketch-caption pairs that are difficult for crowd-workers to create [198].

4.1.1 Scene Composer: Generating Composition Layouts

To generate composition layouts of scenes, we first model composition layouts as a sequence of n objects (and start/end tokens), such that each object generated by the network is represented with 8 values:

$$b_t = [x_t, y_t, w_t, h_t, l_t, \text{box}_t, \text{start}_t, \text{end}_t], t \in [1, n]$$

The first 5 values are fundamental data that describes bounding boxes of objects in the scene: x-position, y-position, width, height, and the class label. The last three values are boolean flags used as extra 'tokens' to mark the actual objects, the beginning of sequences, and the end of sequences.

Using this sequential encoding of scenes, we designed a Transformer-based Mixture Density Network as our *Scene Composer* to generate realistic composition layouts. Transformer models [184] are state-of-the-art neural networks for sequence-to-sequence modeling tasks, such as machine translation and question answering. We use a Transformer Network to perform a novel task: generating a sequence of objects from a text description c, a sequence of words. As multiple scenes can correspond to the same text descriptions, we feed the outputs of the Transformer Network into Gaussian Mixture Models (GMMs) to model the variation of scenes, forming a Mixture Density Network [13] as we described in detail in Section 2.4.

The generation process of the composition layouts involves taking the previous bounding box b_{t-1} (or the start token) as an input and generating the current box b_t. At each timestep, the Transformer Network generates an output t_t conditioned on the text input c and previously generated boxes $b_{1...t-1}$ using self-attention and cross-attention mechanisms built into the architecture. This process is repeated for multiple bounding boxes until an end token is generated:

$$t_t = \text{Transformer}([b_{1...t-1}; c]) \tag{4.1}$$

t_t is then projected to the appropriate dimensionality to parameterize the GMMs with various projection layers W_{xy} and W_{wh} to model $p(x_t, y_t)$, the distribution of the bounding boxes' positions, and $p(w_t, h_t)$, the distribution of the bounding boxes' sizes. The main difference to the generic process described in Section 2.4 is that in the scene composer we model the joint distribution between x_t and y_t, and between w_t and h_t. Sketchforme can then generate bounding boxes $[x_t, y_t, w_t, h_t]$ by sampling from these mixture of Gaussians distributions.

$$p(w_t, h_t) = p(w_t, h_t|\theta), \theta = a(W_{wh}([x_t; y_t; t_t])) \tag{4.2}$$

Another special adaptation to the generic MDN described in Section 2.4 is that while $p(x_t, y_t)$ is modeled only from the first projection layer W_{xy}, we consider $p(w_t, h_t)$ to be conditioned on the position of the boxes similar to [73]. To introduce this condition, we concatenate t_t and $[x_t, y_t]$ as inputs to the second projection layer as described in Equation 4.2. The probability of each generated bounding box being an actual object or a start/end token is generated using a softmax-activated third projection layer W_c from the Transformer output, similar to when generating a pen-event described in Section 2.4:

$$p(\text{box}_t, \text{start}_t, \text{end}_t) = \text{softmax}(W_c t_t) \tag{4.3}$$

In addition, Sketchforme separately uses an LSTM to generate class labels l_t because the class labels given certain descriptions are assumed to not vary across examples. The full architecture of the Scene Composer is shown in Figure 4.3a.

4.1.2 Object Sketcher: Generating Individual Sketches

After obtaining scene layouts from the Scene composer, we designed a modified version of the Sketch-RNN model to generate individual objects in Sketchforme according to the layouts. We adopt the decoder-only Sketch-RNN that is capable of generating sketches of individual objects as sequences of individual strokes. Sketch-RNN's sequential generation process involves generating the current stroke based on previously generated strokes, a method commonly used in sequence modeling tasks. Sketch-RNN also uses a GMM to model variation of sketch strokes, forming a Mixture Density Network similar to the process used by the Scene Composer and described in detail in 2.4 and in the Sketch-RNN paper [62].

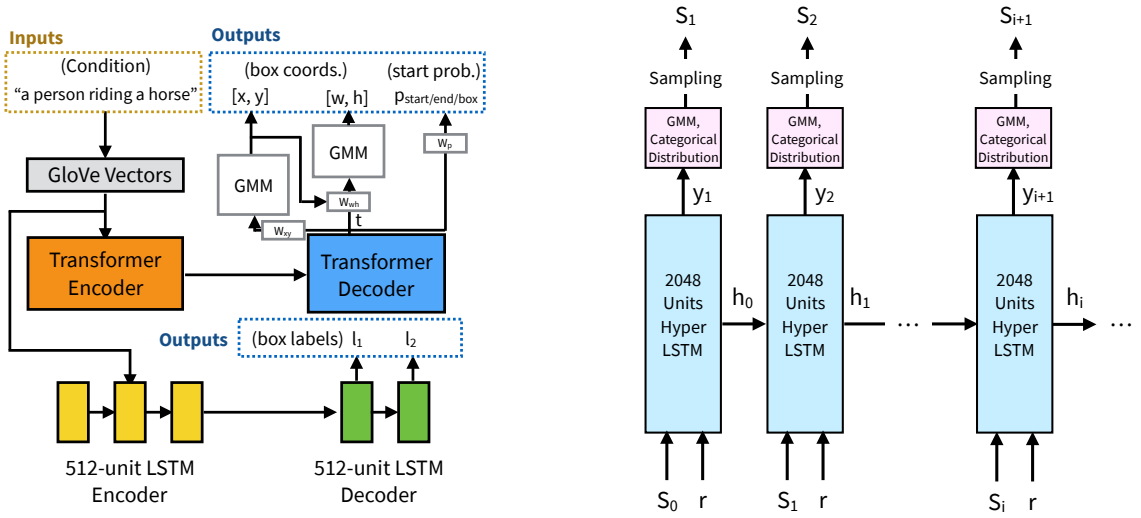
While the decoder-only Sketch-RNN generates realistic sketches of individual objects in certain concept classes, the aspect ratios of the output sketches generated by the original Sketch-RNN cannot be constrained. Hence, sketches generated by the original Sketch-RNN may be unfit for assembling into scene sketches guided by the layouts generated by the Scene Composer. Further, naive direct resizing of the sketches can produce sketches of unsatisfactory quality for complex scenes.

We modified Sketch-RNN as the *Object Sketcher* that factors in the aspect ratios of objects when generating sketches. To incorporate this information in the Sketch-RNN model, we compute the aspect ratios of the training data and concatenate the aspect ratio $r = \frac{\Delta y}{\Delta x}$ of each sketch with the previous stroke as input to our modified Sketch-RNN in the sketch generation process as shown in Figure 4.3b. The new formulation and output of the modified Sketch-RNN for t-th stroke is:

$$[h_t; c_t] = \textbf{LSTM}([S_{t-1}; r; h_{t-1}; c_{t-1}]), \quad y_t = W h_t + b_t \tag{4.4}$$

Since each Sketch-RNN model only handles a single object class, we train multiple modified Sketch-RNN models based on multiple classes and use appropriate models based on class labels in the layouts generated by the Scene Composer for assembling the final sketched scene.

After generating the strokes, the Object Sketcher converts them into SVG paths and fills each object in white using the non-zero rule. The object corresponding to the first bounding box generated by the Scene Composer is then composed as the foreground of the sketched scene, and subsequently generated objects are placed in the background according to the order of generation.



(a) Scene Composer

(b) Object Sketcher

Figure 4.3: Model architecture of (a) the Scene Composer and (b) the Object Sketcher.

4.2 Model Training and Data Sources

Sketchforme's Scene Composer and Object Sketcher are trained on different datasets that encapsulate visual-scene-level knowledge and sketching knowledge separately. This relaxes the requirement for Sketchforme to be trained on natural language annotated datasets of sketched scenes that provide varied scenes corresponding to realistic scene-caption pairs.

We trained the Scene Composer using the Visual Genome dataset [103], which contains natural language region descriptions and object relations of natural images, to demonstrate its flexibility in utilizing various types of scene-layout datasets. Object relations in the dataset each contain a 'subject' (e.g., 'person'), a 'predicate' (e.g., 'on'), and an 'object' (e.g., 'car') represented by class labels and bounding boxes of participating objects in the

image. Natural language region descriptions are represented by bounding boxes of the regions and description texts that correspond to the regions. We reconcile these two types of information using region graphs in the dataset. With the paired data of natural language descriptions and relations, we train the Scene Composer to generate composition layouts. We selected relations that contain subsets of the 100 most commonly used object classes and 70 predicates in the dataset. This dataset of selected object classes and predicates contains 101,968 instances. We split this dataset in the scheme of: 70% training set, 10% validation set, and 20% test set.

The Object Sketcher is trained with the Quick, Draw! [91] dataset that contains 70,000 training sketches, 2,500 validation sketches, and 2,500 test sketches for each of the 345 object categories in the dataset. As mentioned in Chapter 4.1, we preprocess the data by computing the aspect ratios of all sketches as inputs to the Object Sketcher in addition to the original stroke data.

Using these data sources, we train multiple neural networks of various configurations and loss functions in Sketchforme. The LSTM architectures in the Scene Composer for generating composition layouts are stacked with 2 hidden-layers of size 512. Similarly, the Transformer Network has the configuration $(d_{model}, N_{layers}) = (512, 6)$.

The Scene Composer is trained by minimizing the negative log-likelihoods of the position data L_{xy} and size data L_{wh}, and cross-entropy loss for categorical outputs L_p:

$$L_{xy} = -\sum_{i=1}^{n} \log(p(x_i, y_i)) \tag{4.5}$$

$$L_{wh} = -\sum_{i=1}^{n} \log(p(w_i, h_i)) \tag{4.6}$$

$$L_p = -(\sum_{t=1}^{n} p(\text{box}_t) \log(\text{box}_t) + p(\text{start}_t) \log(p(\text{start}_t)) + p(\text{end}_t) \log(p(\text{end}_t))) \tag{4.7}$$

For generating the class labels, each l_t is represented as a 100-dimensional vector in our model, with each value $l_{i,t}$ corresponding to the output probability of the class. L_{class} is thus computed as:

$$L_{\text{class}} = -\sum_{t=1}^{n} \sum_{i=1}^{100} l_{i,t} \log(l_{i,t}) \tag{4.8}$$

We combine these losses with weight hyper-parameters to obtain a general training objective L_{SC} for the Scene Composer:

$$L_{SC} = \lambda_1 L_{xy} + \lambda_2 L_{wh} + \lambda_3 L_p + \lambda_4 L_{\text{class}} \tag{4.9}$$

We set $\lambda_1 = 1.0, \lambda_2 = 1.0, \lambda_3 = 1 \times 10^{-5}, \lambda_4 = 1 \times 10^{-3}$. We used the Adam Optimizer with an initial learning rate of 1×10^{-5} and $\beta_1 = 0.9, \beta_2 = 0.999$ to minimize the loss

function. We used 5 mixtures in each of the GMMs. We chose these hyper-parameters based on empirical experiments.

The Object Sketcher uses an HyperLSTM cell [61] of size 2,048 for the modified Sketch-RNN model. The loss function of the Sketch-RNN model is identical to the reconstruction loss L_R in the original Sketch-RNN model to maximize the log-likelihood of the generated probability distribution for each of the strokes S_t. The model is trained with an initial learning rate of 0.0001 and gradient clipping of 1.0.

4.3 Experiments and Results

Central to evaluating Sketchforme's success is assessing its effectiveness in generating realistic and relevant sketches and layouts from text descriptions. We evaluated the data generated by Sketchforme at each step of the generation process qualitatively and quantitatively to demonstrate its effectiveness of generating sketched scenes. We further conducted two user studies on the overall utility of the generated sketches to explore their potential in supporting real-world applications.

4.3.1 Composition Layout Generation

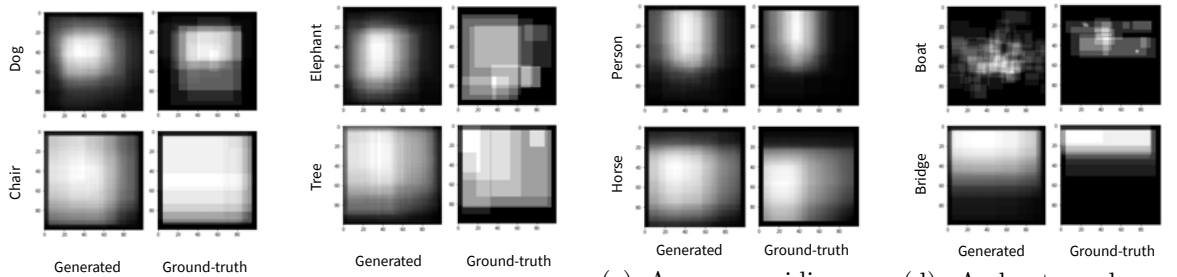
The composition layouts generated by the Scene Composer in the first step of Sketchforme's sketch generation process are represented as bounding boxes of individual objects in the scene. While the Scene Composer already directly maximizes the log-likelihood of the data, we can evaluate the performance of the model by visualizing and comparing heat-maps created by super-positioning instances of real data and generated data.

Because Sketchforme considers the text input when generating the composition layouts, we should only compare the generated bounding boxes with ground-truth bounding boxes from the dataset that are relevant to the text input. We obtain these ground-truth compositions by filtering the subjects, objects, and predicates based on the descriptions. For instance, the composition layouts generated from 'a person riding a horse.' are compared to all ground-truth layouts with 'person' subjects, predicates that are related to riding such as 'on', 'on top of' etc. and 'horse' objects.

Heat-maps in Figure 4.4 show the distributions of Sketchforme-generated bounding boxes and ground-truth bounding boxes from the dataset. From these heat-maps, we can obtain a holistic view of the generation performance of the model by visually evaluating the similarity between the heat-maps. We observe similar distributions between the ground-truth layouts and the generated layouts based on all of the descriptions.

We can further approximate an overlap metric between the distributions using Monte-Carlo simulations to evaluate the model's performance quantitatively. To estimate the degree of overlap between the generated data distribution and the dataset's distribution, we generated 100 composition layouts for each description and randomly sampled 1,000 data points within each bounding box in these layouts. We estimate the overlap between the distributions

by counting the number of data points that lie within the intersections between any generated and ground-truth bounding boxes. We compare Sketchforme's performance with both a heuristic-based bounding box generator and a naive random bounding box generator. The heuristic-based bounding box generator only generates the first bounding boxes above/below the second bounding boxes for descriptions with the above-related/below-related predicates. The random bounding-box generator samples random values that describe the bounding boxes from uniform distributions, which serves as a naive baseline. Table 4.1 shows the percentage of the 1,000 data points that lie in the intersections. The metric value for overlap between real and Sketchforme-generated data is considerably higher than both the value for overlap between real and heuristic-generated data and the value for overlap between real and randomly generated data, which confirms our analysis from qualitative visual inspection of the heat-maps.



(a) A dog on a chair

(b) An elephant under a tree

(c) A person riding a horse

(d) A boat under a bridge

Figure 4.4: Heat-maps generated by super-positioning Sketchforme-generated/ Visual Genome (ground-truth) data. Each horizontal pair of heat-maps corresponds to an object from a description.

Description	**Sketchforme**	Heuristics	Random
a dog on a chair	**89.1%**	64.4%	61.6%
an elephant under a tree	**68.4%**	40.3%	30.6%
a person riding a horse	**94.0%**	57.7%	51.5%
a boat under a bridge	**31.8%**	15.0%	6.85%

Table 4.1: Overlap metric from Monte-Carlo simulations for each description between real data and Sketchforme-generated/heuristics-generated/random data.

4.3.2 Generating Individual Object Sketches at Various Aspect Ratios

The main addition of Sketchforme to the original Sketch-RNN model is a new input that allows the Object Sketcher to generate sketches based on target aspect ratios ($r = \frac{\Delta y}{\Delta x}$) of completed sketches. We evaluate this approach by generating sketches of various aspect ratios. The Object Sketcher is able to adhere to input aspect ratios and generate individual object sketches coherent to the ratios. As shown in Figure 4.5, sketched trees generated with ratio $r = 1.0$ can be perceived as shorter than those generated with $r = 2.0$.

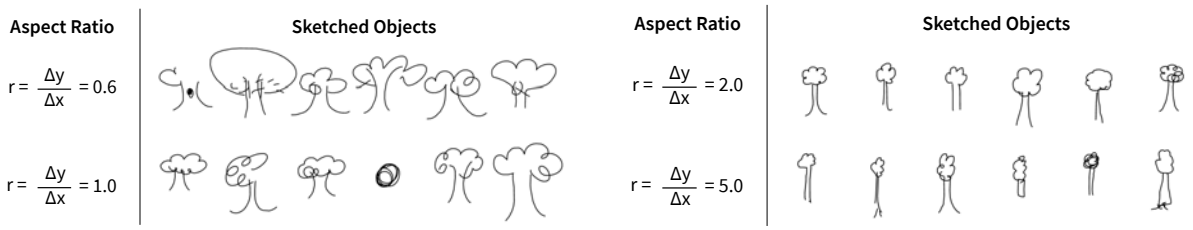


Figure 4.5: Generated sketches of trees with various aspect ratios by the Object Sketcher in Sketchforme.

4.3.3 Complete Scene Sketches

Combining the Scene Composer and the Object Sketcher, Sketchforme generates complete scene sketches directly from text descriptions. Several examples of the sketches are shown in Figure 4.1, Figure 4.6, and Figure 4.8. In these figures, sketches that correspond to 'a boat under a bridge' consist of small boats under bridges, whereas sketches that correspond to 'an apple on a tree' consist of small apples on large trees that follow the actual sizes and proportions of the objects. Moreover, Sketchforme is able to generalize to novel concepts of 'a cat on top of a horse,' such that the only relations involving a cat and a horse in the Visual Genome dataset that the model was trained on correspond to 'a horse facing a cat.' The sizes of cats and horses in these sketches are in proportion to their actual sizes, and the cat is adequately placed on the back of the horse, as shown in Figure 4.8.

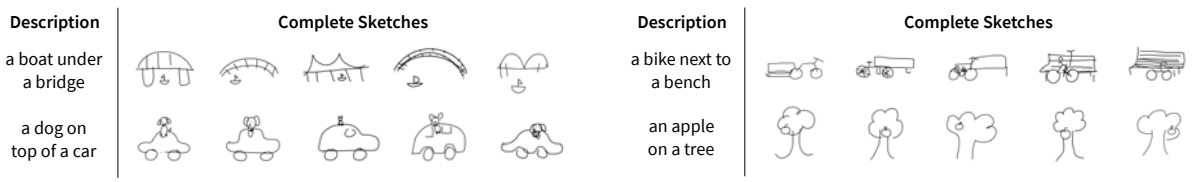


Figure 4.6: Complete scene sketches generated by Sketchforme.

We further trained Sketchforme on the Abstract Scenes dataset [197] to evaluate the approach's generalizability on handling complex scenes. We included examples of sketches generated by Sketchforme based on a) 'Some squirrels near the pond under the trees on a sunny day' (a multi-object multi-relation scene) and b) 'Living room with some paintings on the wall' (a scene consists of an abstract setting, i.e., living room) in Figure 4.7.

Sketchforme was able to generate these scenes adhering to the captions: for description a), Sketchforme was able to locate each object according to inter-object relations dictated by the description; for description b), Sketchforme was able to analyze the phrase 'living room' and generate couches and house plants in the scenes. When Sketchforme needs to handle a large number of objects, we found that it can be further improved by making a minor modification to the Scene Composer: feeding the generated classes into the Transformer network to synchronize the generated bounding boxes with the generated class labels of the objects. Moreover, we observed that the limitation of occlusions (further discussed in Chapter 4.5.1) is more apparent as the scenes become more crowded with objects, such as the multiple overlapping trees in scenes based on description a).

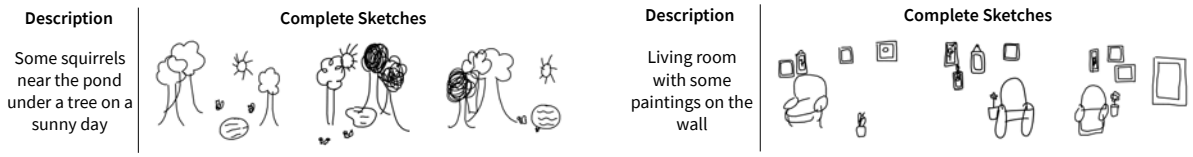


Figure 4.7: Complete scene sketches generated by Sketchforme trained on the Abstract Scenes dataset that contains complex multi-object scenes.

4.3.4 Human Perception User-Study

Sketchforme's high-level goal is to augment users' communication and learning processes by generating realistic, plausible, and coherent sketches for users to interact with. To complement the quantitative and qualitative evaluation of the sketches, we conducted a user study on Amazon Mechanical Turk (AMT) to gauge human subjects' opinions on the sketches' realism and ability to convey the descriptions used to generate them.

Study Procedure

We recruited 51 human subjects on AMT and asked them to each review 50 sketches generated by either humans or Sketchforme. These 50 sketches are generated from five descriptions. The human-generated sketches are obtained from another AMT task prior to this user study based on Quick, Draw! [91]. These human-generated sketches are shown in Figure 4.8. In this study, subjects are provided with complete sketched scenes and descriptions that the scenes are based on. Subjects are required to respond to the following questions:

1. Do you think this sketch was generated by a computer (AI) or a human?
2. On a scale of 1-5 (1 represents that description conveyed very poorly, 5 represents that description conveyed very well), how well did you think the message is conveyed by the sketch?

The subjects are given 10 sketches as trial questions with answers to the first question at the beginning of the task. After completing the trial tasks, the subjects' answers to the remaining 40 sketches are aggregated as the study results. This study protocol is similar to perception studies commonly used to evaluate synthetic visual content generation techniques in the deep-learning community [84]. In addition, we collected comments from the users (if any) and their perceived overall difficulty of the task at the end of the task.

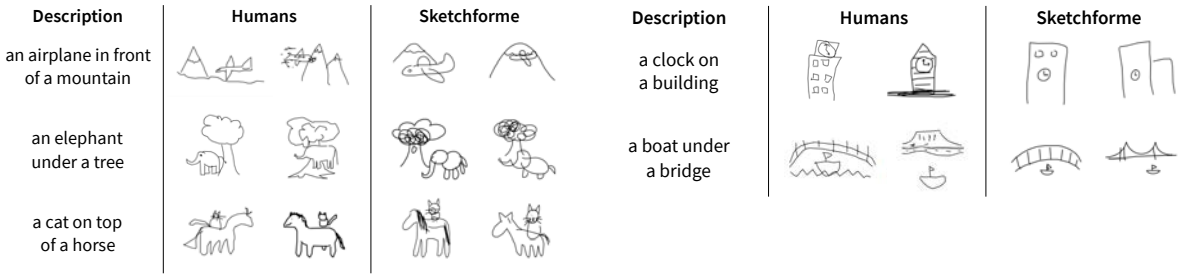


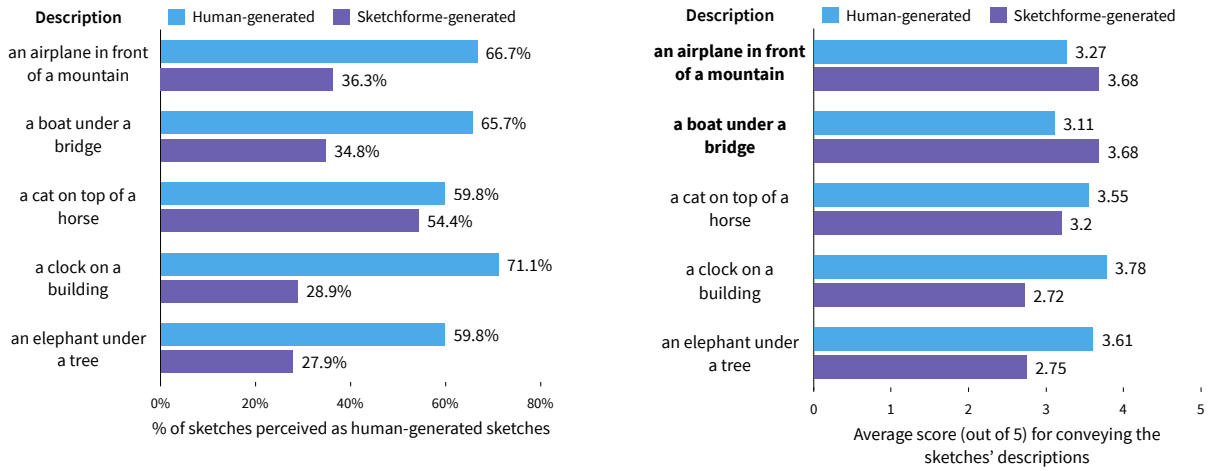
Figure 4.8: Samples of sketches produced by humans and Sketchforme used in the AMT user study.

Results

The first question probes the realism of the sketches with a Turing-test-style question asking the subjects to determine whether the sketches are created by humans. As shown in Figure 4.9a, subjects on average considered 64.6% of the human-generated sketches as generated by humans, while they considered 36.5% of Sketchforme-generated sketches as generated by humans. Although the percentage of Sketchforme-generated sketches considered as generated by humans is significantly lower ($p < 1.05 \times 10^{-10}$, paired t-test) than that of human-generated sketches, individual participants commented in the study that it was difficult to distinguish between human-generated and Sketchforme-generated sketches. P2 mentioned that they "really couldn't tell the difference in most images." P6 commented that they "didn't know if it was (a) human or a computer (that generated the sketches)." These results demonstrate the potential for Sketchforme in generating realistic sketched scenes.

We hypothesize one of the possible reasons for the lower percentage of Sketchforme-generated sketches to be considered as human-drawn is that the curves of the synthetic sketches are in general less jittery than human-drawn sketches. We suggest future work to explore introducing stroke variation to generate more realistic sketches.

The results for the second question reflect the ability of the sketches to communicate the underlying descriptions that they are based on. The average score for human-generated sketches is $\mu = 3.46$, whereas the average score for Sketchforme-generated sketches is $\mu = 3.21$ as shown in Figure 4.9b. Although Sketchforme-generated sketches achieved lower scores overall, Sketchforme-generated sketches achieved statistically better average scores for sketches based on two of the descriptions: 'a boat under a bridge' and 'an airplane in front of a mountain' ($p < 0.0005$, paired t-test). This shows the competitive performance of Sketchforme-generated sketches in communicating the underlying descriptions for some scenes.



(a) Percentage of sketches considered by users as human-generated.

(b) Average score for conveying the sketches' descriptions.

Figure 4.9: Results of the human perception user-study on Sketchforme. 64.6% of human-generated sketches and 36.5% of Sketchforme-generated sketches are perceived as human-generated in (a). In (b), Sketchforme-generated sketches were considered more expressive than human-generated sketches for sketches of 'a boat under a bridge.' and 'an airplane in front of a mountain.'

4.3.5 Sketch Interpretation User Study

To further evaluate Sketchforme's ability to deliver messages through sketches, we conducted an exploratory user study to gauge users' ability to translate Sketchforme-generated sketches back into natural language descriptions.

Study Procedure

We recruited 10 participants on AMT. Each participant was provided with 5 sketches generated by humans and 5 sketches generated by Sketchforme from the same descriptions, but without the original text descriptions. The sketches are reused from the previous study, but the order and selection of the sketches from the set of sketches in the previous study are randomized for each participant. Participants are then asked to produce text descriptions that represent the meaning of each of the sketches. We ensured that the participants had not participated in any of our other studies.

Results

We aggregated the text descriptions produced by each of the participants and compared the subjects, objects, and predicates included in their text descriptions with the original descriptions used to generate the sketches. The percentage of user-generated descriptions that match the subjects/objects/predicates of the original descriptions are respectively 86%/72%/46% for human-generated sketches, and 86%/76%/38% for Sketchforme-generated sketches. While we did not observe a significant difference between the communication ability of both sets of sketches ($p > 0.269$, each paired t-test for subjects/objects/predicates), one novel insight brought by this study is that these sketches are particularly weak in conveying the predicates. Multiple descriptions provided by the users on both sets of sketches did not mention the predicates at all, such as 'an elephant and a tree,' or mentioned more general predicates, such as 'elephant walking near a tree,' for sketches based on 'an elephant under the tree.' Effectively conveying the descriptions' predicates through sketches can be an interesting issue for further research investigation.

4.4 Applications

In this chapter, we explore several applications that can benefit from Sketchforme's ability to generate compelling sketches from natural language descriptions.

4.4.1 Sketch-Assisted Language Learning

Sketches have been shown to improve memory [185]. As language learning is a memory-intensive task, Sketchforme could support language education applications based on sketches. These sketches can potentially create engaging and effective learning processes and avoid rote learning.

Language Learning Application

To explore the possibility of Sketchforme in supporting language learning, we built a basic language-learning application that aims to educate learners with a translation task

from German to English. In this application, learners are presented with German phrases and asked to translate them to English in the form of multiple-choice questions similar to the process of learning term definitions from flash-cards. This application also implements the Leitner system [113] with three bins that repeat phrases which learners make the most mistakes on most frequently. Under this system, the phrases are moved to different bins depending on the participants' familiarity with the translations.

We gathered 10 pairs of German-English sentences from a native German speaker and formed 2 sets of 5 translations each. In addition, deceptive English sentences are added as other choices in the multiple-choice test to be selected by the learners in the application. We deployed this application on AMT to test the improvement of learning performance by presenting Sketchforme-generated sketches along with the phrases. The UI of the full application with a sketch presented to the users is shown in Figure 4.10a.

Study Procedure

The study consists of a training phase and a test phase for each participant. In the training phase, participants are presented with correct answers after answering each question. The participant can only advance to the next phase when they answer all questions correctly consecutively for all translations according to the Leitner system. In the test phase, participants are given one chance to provide their answers to all translations without seeing the correct answers. The participants are divided into two conditions, with the 'control' group only receiving phrases on their interface during training, and the 'treatment' group that receives both phrases and sketches generated by Sketchforme on their interface during the training phase. Both groups receive only the phrases on their interface during the test phase. Moreover, we use our two sets of translations for training and test phases alternatively, such that the participants will not get consecutive training and test phases for the same set of descriptions.

The performances of the participants during the study are monitored with multiple analytical metrics, including completion time of each phase, and scores in the test phase, etc. At the end of the study, we also provide surveys for them to rate the difficulty of the task and the usefulness of the sketches (if applicable) on five-point Likert scales, and ask them to provide any additional suggestions to the interface.

Results

We recruited 38 participants on AMT to participate in the study. While we did not see a significant difference ($p = 0.132$, unpaired t-test) in the correctness of answers in the test phase of the phrases between the 'control' and 'treatment' groups of participants, we discovered that the *time taken to complete the learning task* for the 'treatment' group (**246** seconds on average) was significantly less ($p = 0.011$, unpaired t-test) than the control group (**338** seconds on average). The 'treatment' group also generally found the sketches to be helpful for learning (rated 4.58 out of 5 in the post-study survey).

As Sketchforme is an automated system that is capable of generating sketches from free-form text descriptions, and with these promising results on sketch-assisted language learning, we envision Sketchforme to support and improve large-scale language learning applications in the future.

4.4.2 Intelligent Sketching Assistant

Since Sketchforme uses sequence models and a multi-step generation process to generate sketches, by design it can support interactive human-in-the-loop sketching systems. To demonstrate such capability, we built a prototype of an intelligent sketching assistant reflective of two potential use-cases:

Auto-completion of scenes

As Sketchforme's Scene Composer consists of the Transformer Network, a sequence model that attends to previous objects in the scene to generate the upcoming object, we can complete unfinished user scenes instead of starting with a blank canvas by starting the generation with both the start token b_1 and an existing object in the scene created by the user b_2. Figure 4.10b shows examples of Sketchforme completing users' sketch of a horse in step a) by adding potential sketched trees involved in the scene.

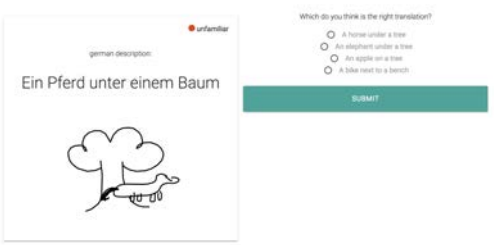
User-Steerable generation

Sketchforme's Scene Composer is capable of generating multiple potential candidate sketched objects at each step of composing the final sketched scenes. As such, users can select their preferred scene layout from the candidates. Figure 4.10b shows multiple candidates proposed by Sketchforme based on a text description in step b). Moreover, since Sketch-RNN is also capable of generating a variety of sketches, the users can select their preferred sketches of each individual object in the scene.

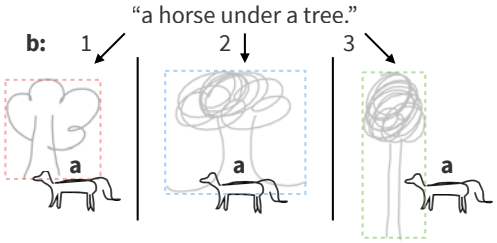
4.5 Limitations

4.5.1 Occlusions and Layer Order

Sketchforme is trained to model scene compositions from a natural image dataset. In natural images, objects might occlude each other, hence affecting sizes and positions of the bounding boxes in the composition layouts. Figure 4.11a shows several boats that were inadequately placed in front of parts of the bridges that should have occluded the boats. To overcome this limitation, future systems can augment Sketchforme by including advanced vision models to determine the objects' layer order in the original natural image. The current Sketchforme system only considers a naive layer order determined by the generation sequence of the composition layout, which is 'subject' then 'object' from the dataset.



(a) Sketch-augmented language-learning application.



(b) Intelligent sketching assistant.

Figure 4.10: Applications enabled by Sketchforme. Sketchforme can augment (a) language-learning applications and significantly reduced the time taken for users to achieve similar learning outcomes. With the (b) intelligent sketching assistant powered by Sketchforme, the user can create a partial sketch for Sketchforme to suggest multiple candidates for them to choose the adequate sketch they prefer from the description 'a horse under a tree.'

Moreover, novel methods should be developed to handle overlapping sketched objects generated from occluded compositions. For instance, the model that generates composition layouts could enforce constraints to avoid overlaps in the sketches or follow user-specified heuristics to handle overlaps.

(a) Occluded Objects

(b) Incoherent Poses

Figure 4.11: Limitations of Sketchforme's sketch generation process. In (a), the boats are significantly occluded by the bridges. In (b), the elephants were represented with square bounding boxes which guided the system to sketch only the faces of the elephants.

4.5.2 Aspect Ratios might be Weak Signals for Object Poses

Sketchforme uses aspect ratios of bounding boxes as the primary signal to inform the shapes of sketches of individual objects. Although these shapes can be sufficient to determine the correct poses for objects in some classes, such as the 'tree' class, merely constraining the shapes might be weak signals for objects of other classes. These shapes can suggest

incoherent perspectives or incomplete sketches, such as examples shown in Figure 4.11b. In Figure 4.11b, only faces of the elephants were sketched due to aspect ratios provided to the Object Sketcher, which is inappropriate for composing sketched scenes. To mitigate this limitation, future work could model the poses of objects in sketches and natural images more closely using other cues, such as complete masks of the objects.

Chapter 5

Scones: Sketch Generation and Iterative Refinement in Critique Cycles

While Sketchforme enabled the new affordance of generating complex sketched scenes from natural language descriptions, the creation of sketches is often an internalized or externalized iterative process. Hence, it would be preferable for systems that can continuously accept users' instructions and improve and edit the generated sketch. Towards this goal, we develop a system that supports *iterative generation* of sketches given users' text instructions to support refinement and critique. The goal is to mimic a user trying to verbally convey a visual idea to an expert sketcher.

The use of sketches in an iterative design and/or artistic process, where the sketch itself is annotated or refined, requires additional, specialized expertise. Inspired by recent development of deep-learning (DL) QA systems and generative sketching models, we develop Scones, a DL-based sketching system that can progressively construct a sketched scene based on multiple natural language instructions across multiple turns, an interaction analogous to an iterative sketch/critique process. This system must unify knowledge of the *low-level* mechanics for generating sketch strokes and natural language modification instructions with a *high-level* understanding of composition and object relationships in scenes.

We formulate the novel task of iteratively generating and refining sketches with text instructions and present a web-deployable implementation of Scones[1]. Scones contains a scene composer that takes a novel approach in creating and editing scenes of objects using natural language. It adapts a recent neural network architecture and improves state-of-the-art performance on the scene modification task. We also introduce in Scones a novel method for specifying high-level scene semantics within individual object sketches by conditioning sketch generation with mask outlines of target sketches. Using Scones, we hope to enable users of all levels of sketch expertise to freely express their intent using abstract, text-based

[1]Scones was also published as a long conference paper at IUI 2020 [80].

instructions, together with concrete visual media.

5.1 System Architecture

The creation of complex sketches often begins with semantic planning of scene objects. Sketchers often construct high-level scene layouts before filling in low-level details. Modeling machine-learning systems after this high-to-low-level workflow has been shown to be beneficial for transfer learning from other visual domains and for supporting interactive interfaces for human users [77]. Inspired by this high-to-low-level process, Scones adopts a hierarchical workflow that first proposes a scene-level composition layout of objects using its *Scene Composer*, then generates individual object sketches, conditioned on the scene-level information, using its *Object Sketchers* (Figure 5.1).

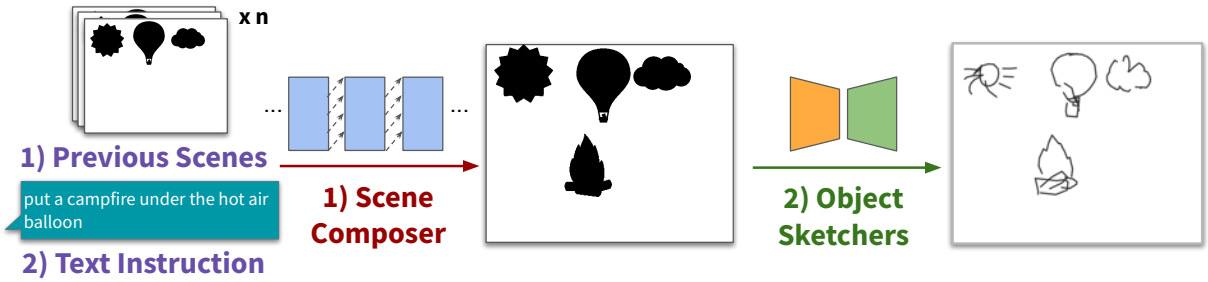


Figure 5.1: Overall architecture of Scones. Scones takes a two-stage approach towards generating and modifying sketched scenes based on users' instructions.

5.1.1 Scene Composer

The Scene Composer in Scones uses text instructions to place and configure objects in the scene. It also considers recent past iterations of text instructions and scene *context* at each conversation turn. As text instructions and sketch components occur sequentially in time, each with a variable length of tokens and objects, respectively, we formulate scene composition proposal as a sequence modeling task. We use a self-attention-only decoder component of the Transformer [184], a recent DL model architecture with high performance for this task.

To produce the output scene S_i at turn i, the Scene Composer takes inputs of $n = 10$ previous scenes $S_{(i-n),\ldots,(i-1)}$ and text instructions $C_{(i-n),\ldots,(i-1)}$ as recent *context* of the conversation. Each output scene S_i contains l_i objects $o_{(i,1),\ldots,(i,l_i)} \in S_i$ and special tokens o_s marking the beginning and o_e marking the end of the scene. Each text instruction C_i contains m_i text tokens $t_{(i,1),\ldots,(i,m_i)} \in C_i$ that consist of words and punctuation marks.

We represent each object o as a 102-dimensional vector o:

$$o = [\mathbb{1}_s, \mathbb{1}_e, e^{(o)}, e^{(u)}, e^{(s)}, e^{(f)}, x, y]$$

The first two dimensions $\mathbb{1}_s, \mathbb{1}_e$ are Boolean attributes reserved for the start and end of the scene object sequences. $e^{(o)}$ is a 58-dimensional one-hot vector[2] representing one of 58 classes of the scene objects $e^{(u)}$ is a 35-dimensional one-hot vector representing one of 35 sub-types (minor variants) of the scene object. $e^{(s)}$ is a three-dimensional one-hot vector representing one of three sizes of the scene objects. $e^{(f)}$ is a two-dimensional one-hot vector representing the horizontal orientation of whether the object is flipped in the x-direction. The last two dimensions $x, y \in [0, 1]$ represents the x and y position of the center of the object. This representation is very similar to that of the CoDraw dataset the model was trained on, which is described in detail in Section 5.2.1. For each text token t, we use a 300-dimensional GLoVe vector trained on 42B tokens from the Common Crawl dataset [142] to semantically represent these words in the instructions.

To train the Transformer network with the heterogeneous inputs of o and t across the two modalities, we create a unified representation of cardinality $|o| + |t| = 402$ and adapt o and t to this representation by simply padding additional dimensions in the representations with zeros as shown in Equation 5.1.

$$o'_{(i,j)} = [o_{(i,j)}, \mathbf{0}_{(300)}] \quad t'_{(i,j)} = [\mathbf{0}_{(102)}, t_{(i,j)}] \tag{5.1}$$

We interleave text instructions and scene objects chronologically to form a long sequence $[C_{(i-n)}, S_{(i-n)}\ , ..., C_{(i-1)}, S_{(i-1)}, C_i]$ as input to the model for generating an output scene representation S_i. We expand the sequential elements within C and S, and add separators to them to obtain the full input sequence to a single Transformer Decoder. To adapt the Transformer model to our multi-modal inputs t' and o' and produce new scene objects o, we employ a 402-dimensional input embedding layer and 102-dimensional output embedding layer in the Transformer model. The outputs from the network are then passed to sigmoid and softmax activations for object position and other properties respectively. We show this generation process in Equation 5.2 and in Figure 5.2.

$$\begin{aligned} S_i = [o_{(i,1),...,(i,l)}] = \textbf{Transformer}([o'_s, o'_{(i-n,1)}, ...o'_{(i-n,l_{(i-n)})}, \\ o'_e, t'_{(i-n,1)}, ..., t'_{(i-n,m_{(i-n)})}, ..., t'_{(i,1)}, ...t'_{(i,l_i)}, o'_s]) \end{aligned} \tag{5.2}$$

5.1.2 Object Sketchers

Since the outputs of the Scene Composer are scene layouts consisting of high-level object specifications, we generate the final raw sketch strokes for each of these objects based on their specifications with *Object Sketchers*. We adapt Sketch-RNN to generate sketches of individual object classes to present to users for evaluation and revision in the next conversation turn. Each sketched object Q consists of h strokes $q_{1...h}$. The strokes are encoded

[2] an encoding of class information that is an array of bits where only the corresponding position for the class to be encoded is 1, and all other bits are 0s.

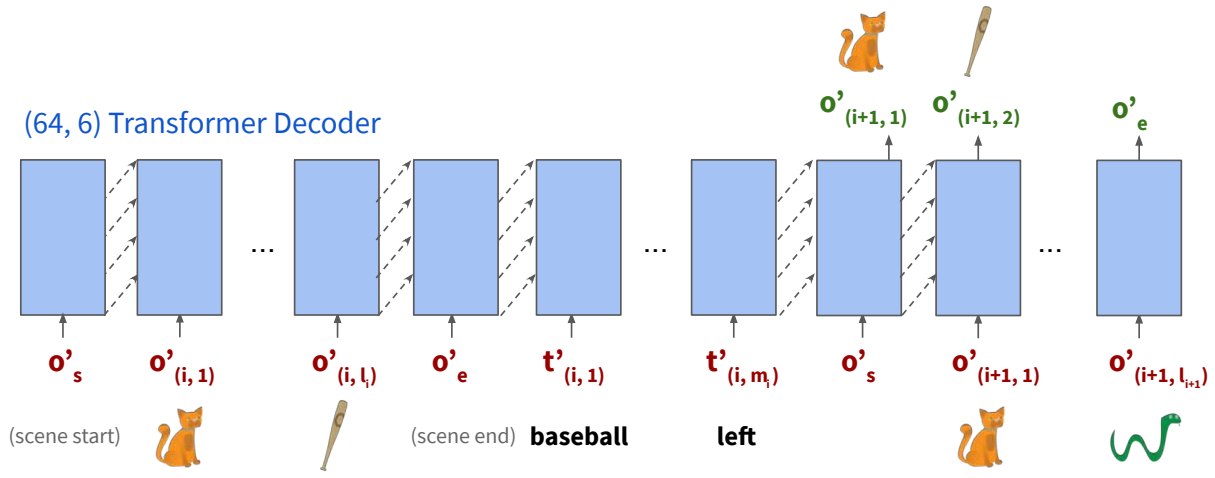


Figure 5.2: The scene layout generation process using the Transformer Model of the *Scene Composer*.

using the *Stroke-5* format [62]. Each stroke $q = [\Delta x, \Delta y, p_d, p_u, p_e]$ represents states of a pen performing the sketching process. The first two properties Δx and Δy are offsets from the previous point that the pen moved from. The last three elements $[p_d, p_u, p_e]$ are a one-hot vector representing the state of the pen after the current point (pen down, pen up, end of sketch, respectively). All sketches begin with the initial stroke $q_1 = [0, 0, 1, 0, 0]$.

Since Sketch-RNN does not constrain aspect ratios, directions and poses of its output sketches, we introduce two additional conditions for the sketch generation process: masks m and aspect ratios r. These conditions ensure our Object Sketchers generate sketches with appearances that follow the object specifications generated by the Scene Composer. For each object sketch, we compute the aspect ratio $r = \dfrac{\Delta y}{\Delta x}$ by taking the distance between the leftmost and rightmost stroke as Δx and the distance between topmost and bottommost stroke as Δy. To compute the object mask m, we first render the strokes into a pixel bitmap, then mark all pixels as 1 if they are in between the leftmost pixel py_{xmin} and rightmost pixel py_{xmax} that are passed through by any strokes for each row y, or if they are in between the bottommost pixel px_{ymin} and topmost pixel px_{ymax} that are passed through by any strokes for each column x (Equation 5.3). As this mask-building algorithm only involves pixel computations, we can use the same method to build masks for clip art objects (used to train the Scene Composer) to generate sketches with poses matching the Scene Composer's object representations.

$$m_{(x,y)} = \begin{cases} 1 & \text{if } py_{xmax} \geq x \geq py_{xmin}, \text{or;} \\ 1 & \text{if } px_{ymax} \geq y \geq px_{ymin} \\ 0 & \text{otherwise} \end{cases} \tag{5.3}$$

We adapt the Variational-Autoencoder(VAE)-based conditional variant of Sketch-RNN

to enable generating and editing of sketch objects. Our adapted conditional Sketch-RNN encodes input sketches with a Bi-directional LSTM to a latent vector z. The Hyper-LSTM decoder then recreates sketch strokes $q'_{1...h}$ from z, and m, r described above during training, as defined in Equation 5.4 and shown in Figure 5.3. Since the latent space is also trained to match a multivariate Gaussian distribution, the Object Sketchers can support sketch generation when the objects are first added to the scene by randomly sampling $z \sim N(0,1)^{128}$.

$$q'_{1...h} = \textbf{Sketch-RNN Decoder}([m, r, z]), z \sim N(0,1)^{128}$$
$$z = \textbf{Sketch-RNN Encoder}(q_{1...h}) \quad (5.4)$$

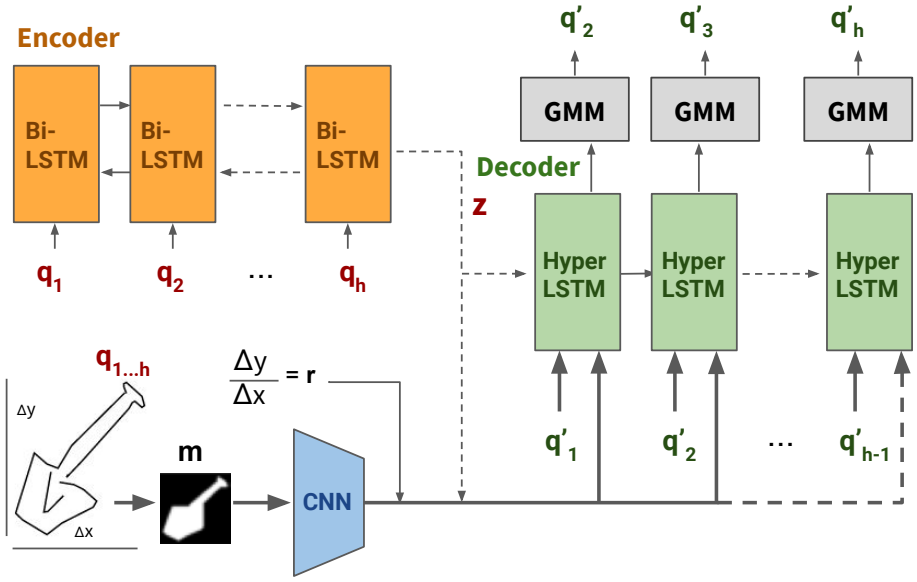


Figure 5.3: Sketch-RNN model architecture of the Object Sketchers.

As m is a two-dimensional mask, we encode m using a small convolutional neural network into a flattened embedding to be concatenated with z, r, and q_i as inputs to the decoder. The decoder then outputs parameters for a Gaussian Mixture Model (GMM) which will be sampled to obtain Δx and Δy, forming a Mixture Density Network. It also outputs probabilities for a categorical distribution that will be sampled to obtain p_d, p_u and p_e. This generation process and the architecture of the model are illustrated in Figure 5.3, and are described in detail in Section 2.4 and in the Sketch-RNN paper [62].

5.2 Datasets and Model Training

As Scones uses two components to generate scenes of sketched objects, it is trained on two datasets that correspond to the tasks these components perform.

5.2.1 CoDraw Dataset

We used the CoDraw dataset [98] to train the Scene Composer to generate high-level scene layout proposals from text instructions. The task used to collect this data involves two human users taking on the roles of *Drawer* and *Teller* in each session. First, the Teller is presented with an abstract scene containing multiple clip art objects in certain configurations, and the Drawer is given a blank canvas. The Teller provides instructions using only text in a chat interface to instruct the Drawer on how to modify clip art objects in the scene. The Teller has no access to the Drawer's canvas in most conversation turns, except in one of the turns when they can decide to 'peek' at the Drawer's canvas. The dataset consists of 9,993 sessions of conversation records, scene modifications, and ground-truth scenes.

Using this dataset, we trained the Scene Composer to respond to users' instructions given past instructions and scenes. We used the same training/validation/test split as the original dataset. Our model is trained to optimize the loss function L_{cm} that corresponds to various attributes of the scene objects in the training set:

$$L_{cm} = L_c + \lambda_{\textbf{sub}} L_{\textbf{sub}} + \lambda_{\textbf{flip}} L_{\textbf{flip}} + \lambda_{\textbf{size}} L_{\textbf{size}} + \lambda_{xy} L_{xy} \tag{5.5}$$

L_c is the cross-entropy loss between the one-hot vector of the true class label and the predicted output probabilities by the model. Similarly $L_{\textbf{flip}}$ and $L_{\textbf{size}}$ are cross-entropy losses for the horizontal orientation and size of the object. L_{xy} is the Euclidean Distance between predicted position and true position of the scene object. We trained the model using an Adam Optimizer with the learning rate of $lr = 1 \times 10^{-4}$ for 200 epochs. We set $\lambda_{\textbf{sub}} = 5.0 \times 10^{-2}$, $\lambda_{\textbf{flip}} = 5.0 \times 10^{-2}$, $\lambda_{\textbf{size}} = 5.0 \times 10^{-2}$, $\lambda_{xy} = 1.0$. These hyper-parameters were tuned based on empirical experiments on the validation split of the dataset.

5.2.2 Quick, Draw! Dataset

The Quick, Draw! dataset consists of sketch strokes of 345 concept categories created by human users in a game in 20 seconds [91]. We trained our 34 Object Sketchers on 34 categories of Quick, Draw! data to create sketches of individual objects.

Each sketch stroke in Quick, Draw! was first converted to the Stroke-5 format. Δxs and Δys of the sketch strokes were normalized with their standard deviations for all sketches in their respective categories. Each category consists of 75,000/2,500/2,500 sketches in the training/validation/test set.

The loss function of the conditional Sketch-RNN L_s consists of the reconstruction loss L_R and KL loss L_{KL}:

$$L_s = \lambda_{KL} L_{KL} + L_R \tag{5.6}$$

The KL loss L_{KL} is the Kullback-Leibler divergence between the encoded z from the encoder and $N(0,1)$[128]. The reconstruction loss L_R is the negative log-likelihood of the ground-truth strokes under the GMM and a categorical distribution parameterized by the model. We refer interested readers to a detailed description of L_s in the original Sketch-RNN paper [62]. The initial learning rate of the training procedure was $lr = 1.0 \times 10^{-3}$ and exponentially decayed to 1.0×10^{-5} at a rate of 0.9999. λ_{KL} was initially 0.01 and exponentially increased to 0.5[3] at a rate of 0.99995. The models were also trained with gradient clipping of 1.0.

5.3 Results

To compare the effectiveness of Scones at generating scene sketches with existing models and human-level performance, we quantitatively evaluated its performance in an iterative scene authoring task. Moreover, as Scones uses generative models to produce object sketches, we qualitatively evaluated a large number of examples generated by the two components of Scones.

5.3.1 Scene Composition Modification State-of-the-Art

To evaluate the output of the Scene Composer against the models introduced with the CoDraw dataset, we adapted its output to match that expected by the well-defined evaluation metrics proposed by the original CoDraw paper [98]. The original task described in the CoDraw paper involves only proposing and modifying high-level object representations in scenes agnostic to their appearance. The performance of a "Drawer" (a human or machine which generates scene compositions) can be quantified by a similarity metric constrained between 0 and 5 (higher is more similar) by comparing properties of and relations between objects in the generated scene and objects in the ground truth from the dataset.

Running our Scene Composer on the CoDraw test set, we achieved an average similarity metric of 3.55. This exceeded existing state-of-the-art performance (Table 5.1) on the iterative scene authoring task using replayed text instructions (script) from CoDraw.

To provide an illustrative example of our Scene Composer's output on this task, we visualize two example scenes generated from the CoDraw validation set in Figure 5.4. In scene a), the Scene Composer extracted the class (slide), direction (faces right), and position relative to parts of the object (ladder along left edge) from the text instruction, to place a slide in the scene. Similarly, it was able to place the bear in between the oak and pine trees in scene b), with the bear touching the left edge of the pine tree. It is important to

[3]For some object categories, we found that increasing the KL weight to 1.0 improves the authors' perceived quality of generated sketches.

Table 5.1: Test set performance of various models on the CoDraw task.

Teller	Drawer	Similarity ↑ (out of 5)
Script	**Scones**	**3.55**
Script	Neural Network [98]	3.39
Script	Nearest-Neighbor [98]	0.94
Script	Human	**3.83**

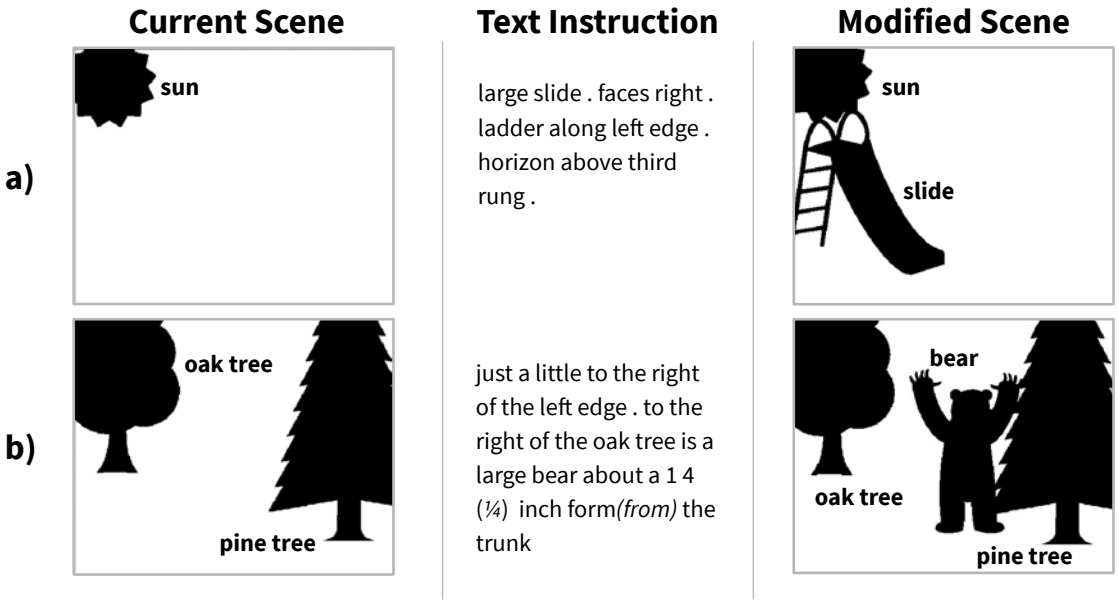


Figure 5.4: Example generated scenes for the scene layout modification task. The Scene Composer was able to improve state-of-the-art performance for modifying object representations in scene compositions.

note the Scene Composer completely regenerates the entire scene at each conversation turn. This means it correctly preserved object attributes from previous scenes while making the requested modifications from the current turn. In these instances, the sun in scene a) and the trees in scene b) were left mostly unchanged while other attributes of the scenes were modified.

5.3.2 Sketches with Clip Art Objects as Mask and Ratio Guidance

The Object Sketchers are designed to generate sketches that respect high-level scene layout information under the guidance of the mask and aspect ratio conditions. To inform generated object sketches with pose suggestions from scene composition layouts, we built outline masks from clip art objects and computed aspect ratios using the same method as building them for training sketches described in Section 5.1.2. We demonstrate the Object Sketchers' performance in two important scenarios that allow Scones to adapt to specific subclass and pose contexts.

Generating objects for closely related classes While the Scene Composer classifies objects as one distinct class out of 58, some of these classes are closely related and are not differentiated by the Object Sketchers. In these cases, object masks can be used by an Object Sketcher to effectively disambiguate the desired output subclass. For instance, Scene Composer generates trees as one of three classes: Oak tree (tall and with curly edges), Apple tree (round and short), and Pine tree (tall and pointy); while there is only a single Object Sketcher trained on a general class of all types of tree objects. We generated three different masks and aspect ratios based on three clip art images and used them as inputs to a single tree-based Object Sketcher to generate appropriate tree objects (by sampling $z \sim N(0,1)^{128}$). The Object Sketcher was able to sketch trees with configurations corresponding to input masks from clip art objects (Figure 5.5). The generated sketches for pine trees were pointy; for apple trees, had round leaves; and for oak trees, had curved edges.

Generating objects with direction-specific poses The Scene Composer can specify the horizontal orientation of the objects (pointing left or right). As such, the Object Sketchers are required to sketch horizontally asymmetric objects (e.g., racquets, airplanes) with specific poses to follow users' instructions. We show the ability of an Object Sketcher to produce racquets at various orientations in Figure 5.6. The generated racquet sketches conformed to the orientation of the mask, facing the specified direction at similar angles.

5.3.3 Complete Sessions with Composition Layouts and Sketches

We show the usage of Scones in six turns of conversation from multiple sessions in Figure 5.7. We curated these sessions by interacting with the system ourselves to demonstrate various capabilities of Scones. In session a), Scones was able to draw and move the duck to the left, sketch a cloud in the middle, and place and enlarge the tree on the right, following instructions issued by the user. In session b), Scones was similarly able to place and move a cat, a tree, a basketball and an airplane, but at different positions from session a). For instance, the tree was placed on the left as opposed to the right, and the basketball was moved to the bottom. We also show the ability of Scones to flip objects horizontally in session b), such that the plane was flipped horizontally and regenerated given the instructions of

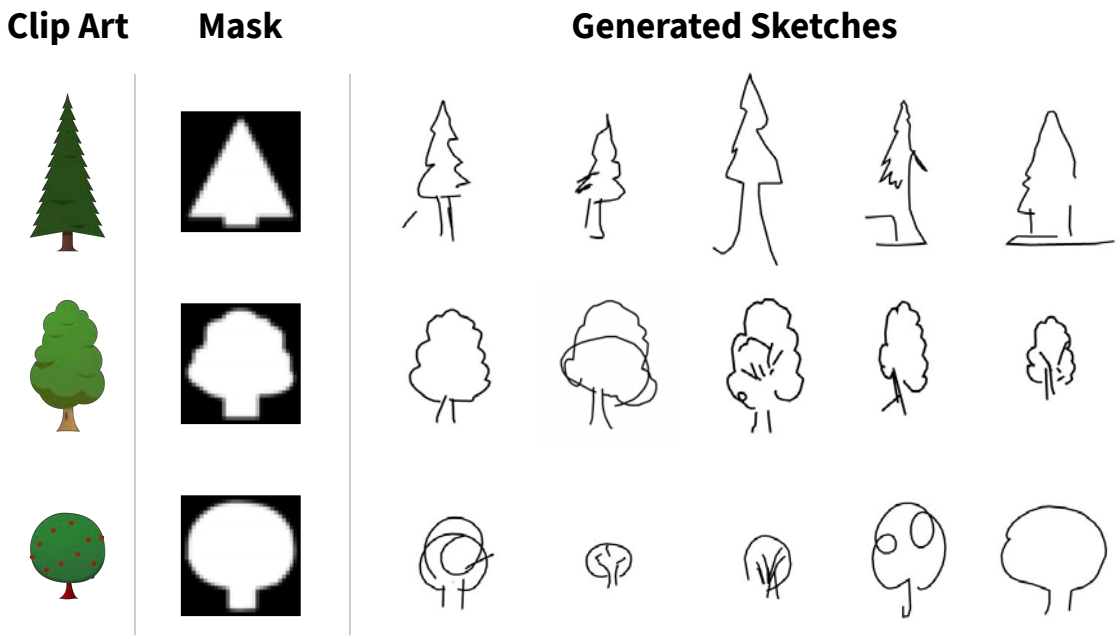


Figure 5.5: Sketch generation results of trees conditioned on masks. The Object Sketcher was able to sketch trees of three different classes based on mask and aspect ratio inputs.

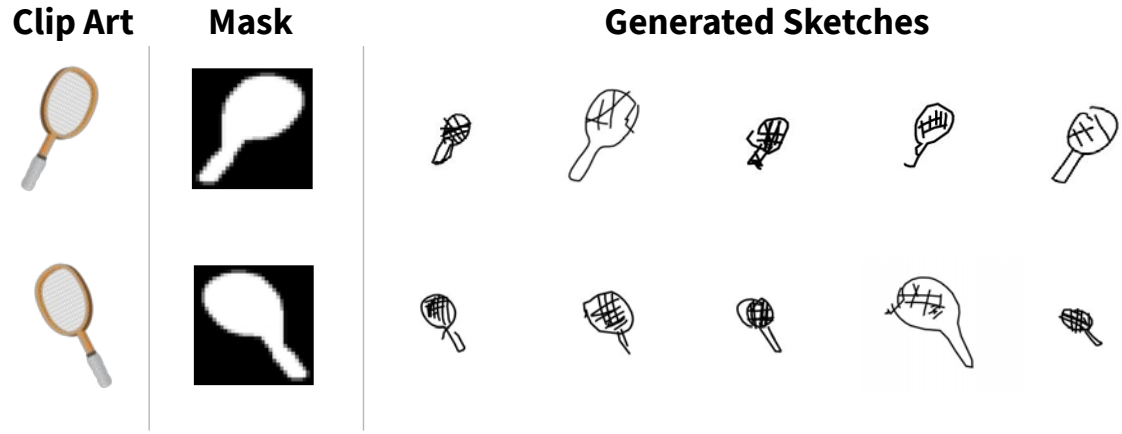


Figure 5.6: Sketch generation results of racquets conditioned on masks. The Object Sketcher was able to sketch racquets at two orientations consistent to the masks.

"flip the plane to point to the right instead". This flipping action demonstrates the Object Sketcher's ability to generate objects with the required poses by only sharing the latent vectors z, such that the flipped airplane exhibits similar characteristics as the original airplane. In both sessions, Scones was able to correlate multiple scene objects, such as placing the owl on the tree in session (a), and basketball under the tree in session b).

5.3.4 Interpreting Transformer's Attention Maps

We can further verify the relationships between text and object representations learned by the model by visualizing attention weights computed by the Transformer model of the Scene Composer. These weights also create the unique possibility of generalizing and prompting for sketches of new objects specified by users.

The Transformer model in the Scene Composer uses masked self-attention to attend to scene objects and instructions from previous time steps most relevant to generating the object specification at the current time step. We explore the attention weights of the first two turns of a conversation from the CoDraw validation set. In the first turn, the user instructed the system, "top left is an airplane medium size pointing left". When the model generated the first object, it attended to the "airplane" and "medium" text tokens to select class and output size (Figure 5.8). In the second turn, the user instructed the model to place a slide facing right under the airplane. The model similarly attended to the "slide" token the most, it also significantly attended to the "under" and "plane" text tokens, and the airplane object. These objects and tokens are important for situating the slide object at the desired location relative to the existing airplane object (Figure 5.9).

These attention weights could potentially be used to handle unknown scene objects encountered in instructions. When the model does not output any scene objects, but only a o_e (scene end) token, we can inspect the attention weights for generating this token to identify a potentially unknown object class, and ask the user for clarification. For example, when a user requests an unsupported class, such as a 'sandwich' or 'parrot' (Figure 5.10), Scones could identify this unknown object by taking the text token with the highest attention weight, and prompting the user to sketch it by name.

5.4 Exploratory User Evaluation

To determine how effectively Scones can assist users in creating sketches from natural language, we conducted an exploratory evaluation of Scones. We recruited 50 participants from English-speaking countries on Amazon Mechanical Turk (AMT) for our study. We collected quantitative and qualitative results from user trials with Scones, as well as suggestions for improving Scones. Participants were given a maximum of 20 minutes to complete the study and were compensated $3.00 USD. Participants were only allowed to complete the task once.

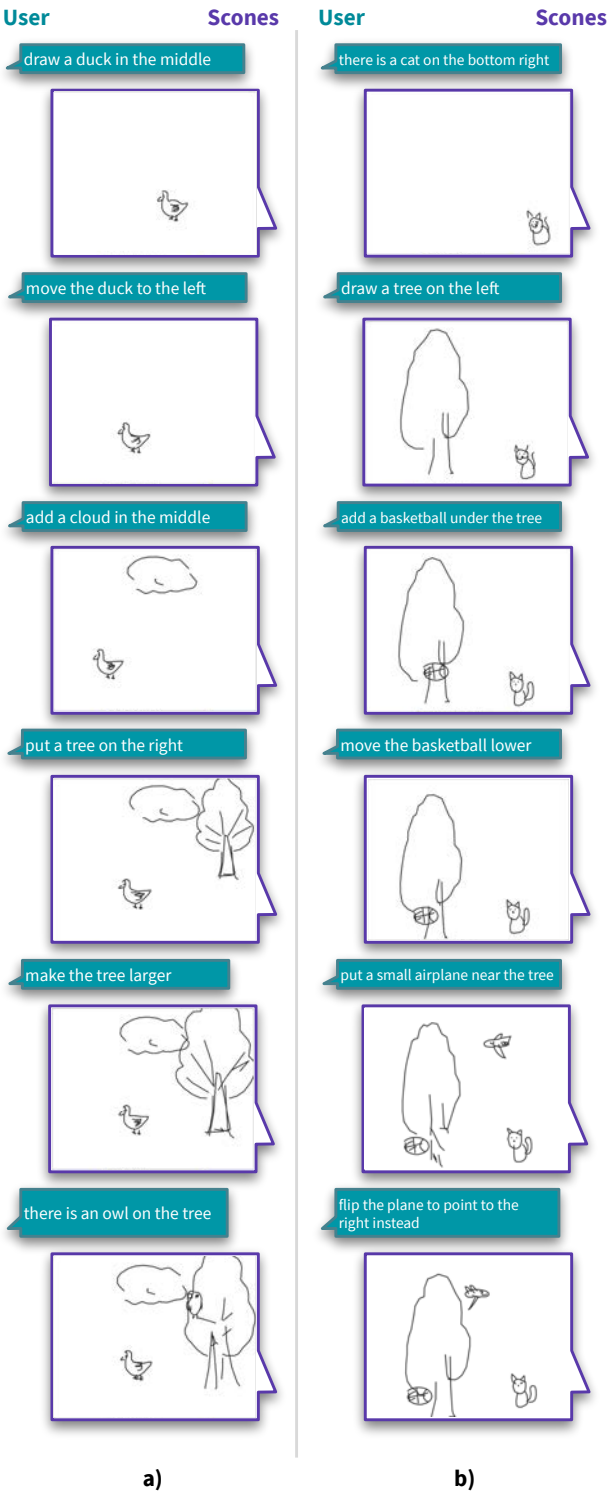


Figure 5.7: Complete sketching sessions with Scones curated by the authors.

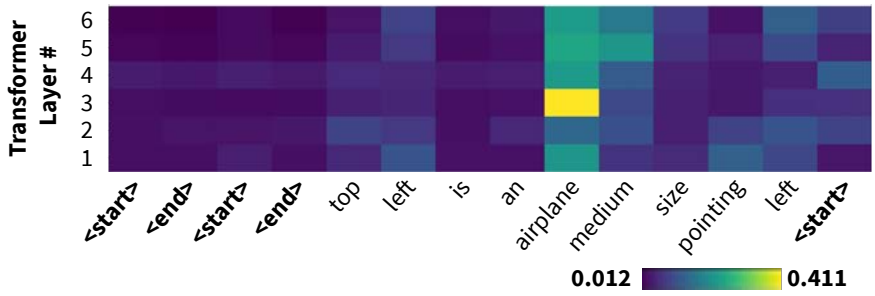


Figure 5.8: Attention map of the Transformer across object and text tokens for the generation of an airplane, the first object in the scene.

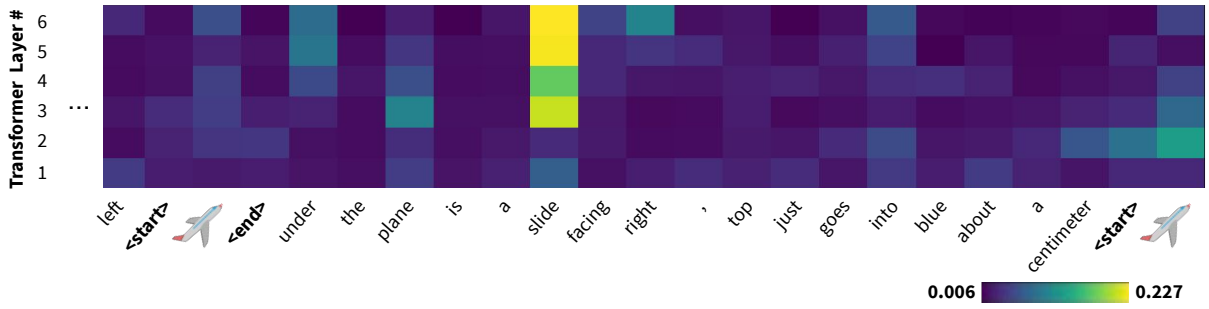


Figure 5.9: Attention map of the Transformer across object and text tokens for the generation of slide in the second turn of conversation. We observed that the Transformer model attended to the corresponding words and objects related to the newly generated 'slide' object.

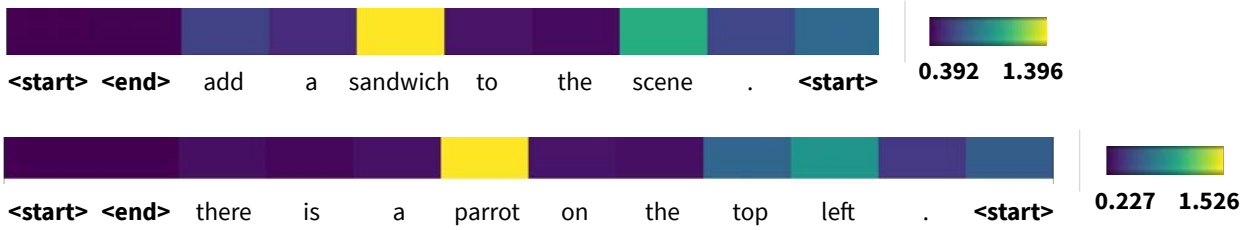


Figure 5.10: Attention map of the Transformer for text instructions that specify unseen objects.

5.4.1 Method

The participants were asked to recreate one of five randomly chosen target scene sketches by providing text instructions to Scones in the chat window. Each target scene had between four and five target objects from a set of 17 possible scene objects. Participants were informed that the final result did not have to be pixel perfect to the target scene, and to mark the sketch as complete once they were happy with the result. Instructions supplied in the chat window were limited to 500 characters, and submitting an instruction was considered as taking a "turn". The participants were only given the sketch strokes of the target scene without class labels, to elicit natural instructions.

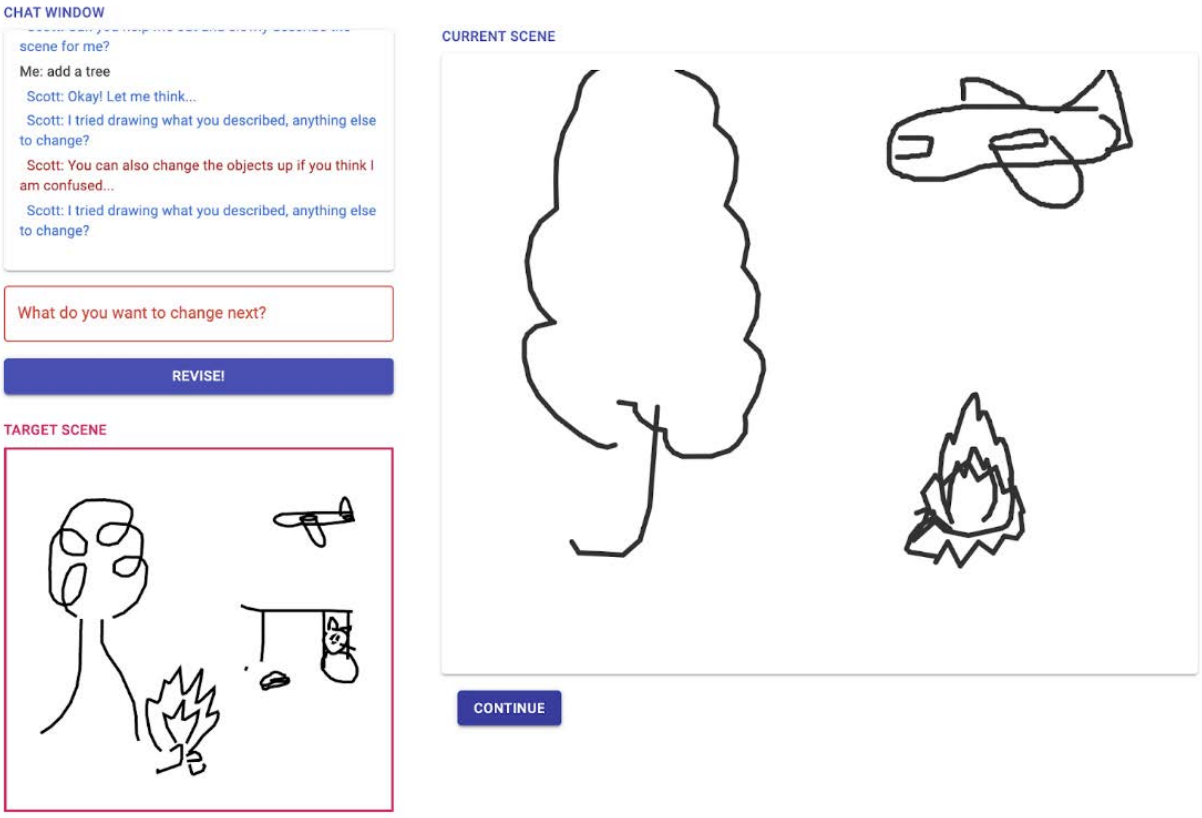


Figure 5.11: Screenshot of Scones' evaluation user interface.

Participants were first shown a short tutorial describing the canvas, chat interface, and target scene in the Scones interface (Figure 5.11), and were asked to give simple instructions in the chat window to recreate the target scene. Only two sample instructions were given in the tutorial: "add a tree", and "add a cat next to the table". At each turn, participants were given the option to redraw objects which remained in the scene for over three turns using a paintbrush-based interface. After completing the sketch, participants filled out an

exit survey with Likert-scale questions on their satisfaction with the sketch and enjoyment of the system, and open-ended feedback on the system.

5.4.2 Results

Participants Satisfied with Sketches, Enjoyment Was Bimodal Participants were generally satisfied with their final sketches ($\mu = 3.38, \sigma = 1.18$), and enjoyed the task ($\mu = 4.0, \sigma = 1.12$). In open-ended feedback, participants praised Scones's ability to parse their instructions: *"it was able to similarly recreate the image with commands that I typed" (P25); "I liked that it would draw what I said. it was simple and fun to use" (P40).* Some participants even felt Scones was able to *intuitively* understand their instructions. P15 remarked, *"I thought it was cool how quickly and intuitively it responded,"* while P35 said, *"It had an intuitive sense of what to draw, and I did not feel constrained in the language I used".*

While enjoyment was high on average, we found responses to enjoyment followed a bimodal distribution (Figure 5.12). By reviewing qualitative feedback and instructions to Scones, we observe that many instances of low enjoyment (score ≤ 2) come from class confusion in target scene sketches. Some participants confused the tent in a target scene as a "pyramid" in their instructions, which Scones does not support: *"There is a pyramid on the left side a little ways up from the bottom" (P44).* P49 tried five times to add a "pyramid" to the scene.

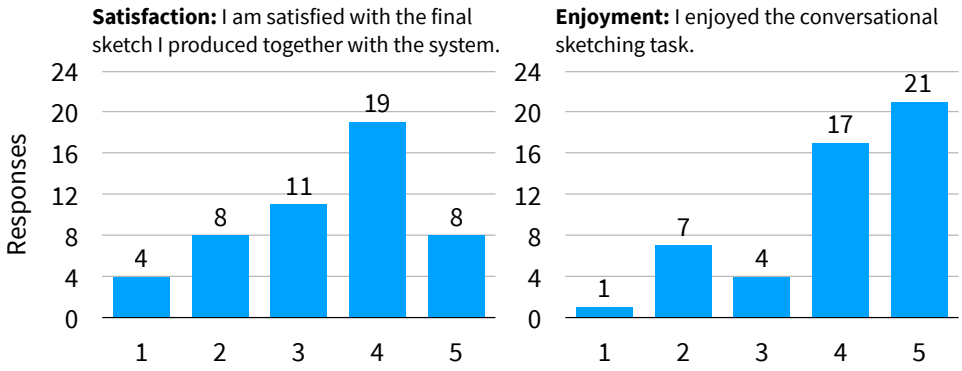


Figure 5.12: Survey results from user sessions with Scones.

P17, who strongly disagreed with enjoying the task (1/5), faced repeated class confusion issues, mentioning, *"it was very frustrating that it wouldn't draw the circle by the cloud . . . It wouldn't draw anything besides the plane, cloud, tent, and fire. Was that not a person up by the cloud?"* Scones does not support "circle" or "person" classes—the target sketch had the sun next to the cloud. When Scones is asked to draw an unsupported object, the canvas will be left unchanged. Providing participants with an explicit list of classes in the target image

or adding error messages could mitigate these frustrations. Furthermore, attention-based methods mentioned in Section 5.3.4 could be used when an unrecognized class is detected to prompt users to provide sketch strokes with corresponding labels.

Participants Communicate with Scones at Varying Concept Abstraction Levels On average, participants completed the sketching task in under 8 turns ($\mu = 7.56$, $\sigma = 3.42$), with a varied number of tokens (words in instructions) per turn ($\mu = 7.66$, $\sigma = 3.35$). Several participants only asked for the objects themselves (turns delimited by commas): *"helicopter, cloud, swing, add basketball" (P25).* Other participants made highly detailed requests: *"There is a sun in the top left, There is an airplane flying to the right in the top right corner, There is a cat standing on it's hind legs in the bottom right corner, Move the cat a little to the right, please, . . . " (P14).* Participants who gave instructions at the expected high-level detail produced satisfying results, *"draw a tree in the middle, Draw a sun in the top left corner, A plane in the top right, A cat with a pizza under the tree" (P32).* The recreation of this participant is shown on the top right of Figure 5.13.

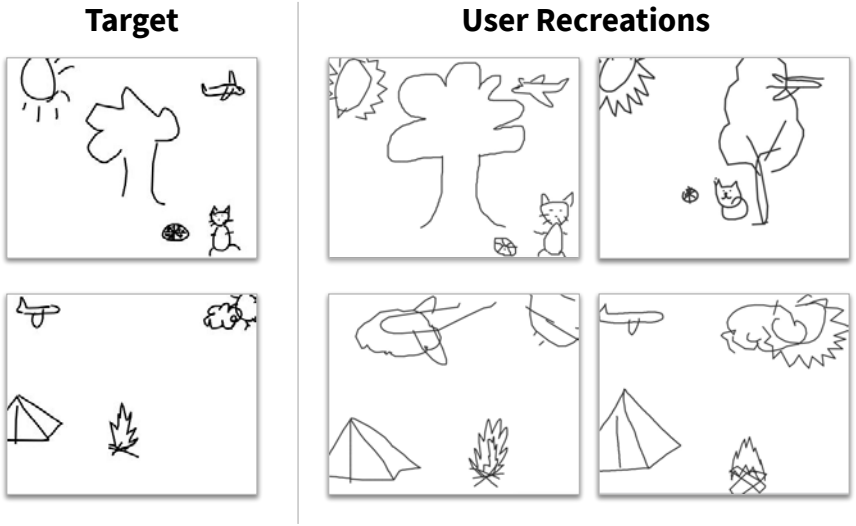


Figure 5.13: Recreated scenes during the user study. Users combined Scones-generated outputs with their own sketch strokes to reproduce the target scenes presented to them.

The longest conversations were often from participants with mismatched expectations for Scones, who repeated commands: *"Draw a cloud in the upper left corner with three round edges., Change the cloud to have 3 round edges., Draw only 3 round waves around the edge of the cloud., . . . Draw a snowman to the left of the table., . . . Draw a circle touching the middle circle., . . . " (P23).* This trial reflects the need for Scones to make clearer expectations of input to users. P23's 16-instruction session contains expectations for the system to modify low-level aspects of the sketches (changing the number of edges in the cloud), exhibits

class confusion (snowman and circles with shovel), and has mismatched concept abstraction levels (drawing a shovel versus constructing a shovel from visual primitives, i.e., circles). A potentially simple mitigation method for these hurdles would be to introduce more detailed tutorial content for a wider deployment of Scones.

Scones as a Tool for Collecting Iterative Sketching Data The results of our study show significant potential for Scones to be used as a Game With a Purpose (GWAP) [1] to collect sketch critiques (natural-language-specified modifications to an input sketch to match a target sketch) and user-generated sketch strokes. 26 (52% of) participants redrew objects in their sketches when prompted ($\mu = 0.98$, $\sigma = 1.19$), and participants who redrew objects expressed their appreciation for this feature: *"I liked that I could redraw the image" (P48); "I liked being able to draw parts myself because it was relaxing and I felt I was more accurate" (P11).* Most participants who redrew objects also kept output from Scones in their final sketches, reflecting Scones's potential as a mixed-initiative design tool. Redrawing was voluntary in our task, and these results suggest Scones may be useful for collecting user-generated sketches in addition to natural language critique in a GWAP. Further motivating this application, 14 participants described the task as "fun" in open-ended feedback, e.g., *"This was a very fun task" (P23); "[I liked] Playing the game and describing the drawing. It was fun!" (P42).*

5.4.3 Participants' Feedback for Improving Scones

Participants offered suggestions for how they would improve Scones, providing avenues for future work.

Object Translations and Spatial Relationships A major theme of dissatisfaction came from the limited ability of our system to respond to spatial relationships and translation-related instructions at times: *"It does not appear to understand spatial relationships that well" (P35); "you are not able to use directional commands very easily" (P11).* These situations largely originate from the CoDraw dataset [98], in which users had a restricted view of the canvas, resulting in limited relative spatial instructions. This limitation is discussed further in Section 5.5.3.

To improve the usability of Scones, participants suggest its interface could benefit from the addition of direct manipulation features, such as selecting and manually transforming objects in the scene: *"I think that I would maybe change how different items are selected in order to change of modify an object in the picture. (P33); "maybe there should be a move function, where we keep the drawing the same but move it" (P40).* Moreover, some participants also recommended adding an undo feature, *"Maybe a separate button to get back" (P31)*, or the ability to manually invoke Scones to redraw an object, *"I'd like a way to ask the computer to redraw a specific object" (P3).* These features could help participants

express corrective feedback to Scones, potentially creating sketches that better match their intent.

More Communicative Output Some participants expected Scones to provide natural language output and feedback to their instructions. Some participants asked questions directly to elicit Scones's capabilities: *"In the foreground is a table, with a salad bowl and a jug of what may be lemonade. In the upper-left is a roughly-sketched sun. Drifting down from the top-center is a box, tethered to a parachute., Did you need me to feed you smaller sentences? . . . " (P38).* P23 explicitly suggested users should be able to ask Scones questions to refine their intentions: *"I would like the system to ask more questions if it does not understand or if I asked for several revisions. I feel that could help narrow down what I am asking to be drawn".* Other participants used praise between their sketching instructions, which could be used as a cue to preserve the sketch output and guide further iteration: *". . . Draw an airplane, Good try, Draw a table . . . " (P1); "Draw a sun in the upper left corner, The sun looks good! Can you draw a hot air balloon in the middle of the page, near the top? . . . " (P15).* Providing additional natural language output and prompts from Scones could enable users to refine Scones's understanding of their intent and understand the system's capabilities. A truly *conversational* interface with a sketching support tool could pave the way for advanced mixed-initiative collaborative design tools.

5.5 Limitations

5.5.1 Underspecified Masks

While mask conditioning effectively guides the Object Sketchers in creating sketches with desired configurations, they can be underspecified for the poses exhibited by objects of some classes. As shown in Figure 5.14, the mask of the right-facing body of a sitting cat can be similar to the face of a cat. The current mask generation algorithm is also not able to capture all the curves of the snake, resulting in ambiguous sketches of snakes. Future iterations of Scones can improve on the mask generation algorithms with more advanced techniques.

5.5.2 Limited Variation of Sketches

Scones currently supports a limited number of sketched object classes and poses due to its discrete representation of object configurations used by the Scene Composer. Future work should explore models conditioned on continuous representations of classes and poses from word embeddings for a flexible number of object classes. Moreover, Scones currently supports only limited stylistic modifications (i.e., it may not support 'sketch the leaves on the tree with more details'). A future iteration of the Scene Composer could output a continuous embedding that contains objects' class, pose, and stylistic information to fully support a wide range of sketches.

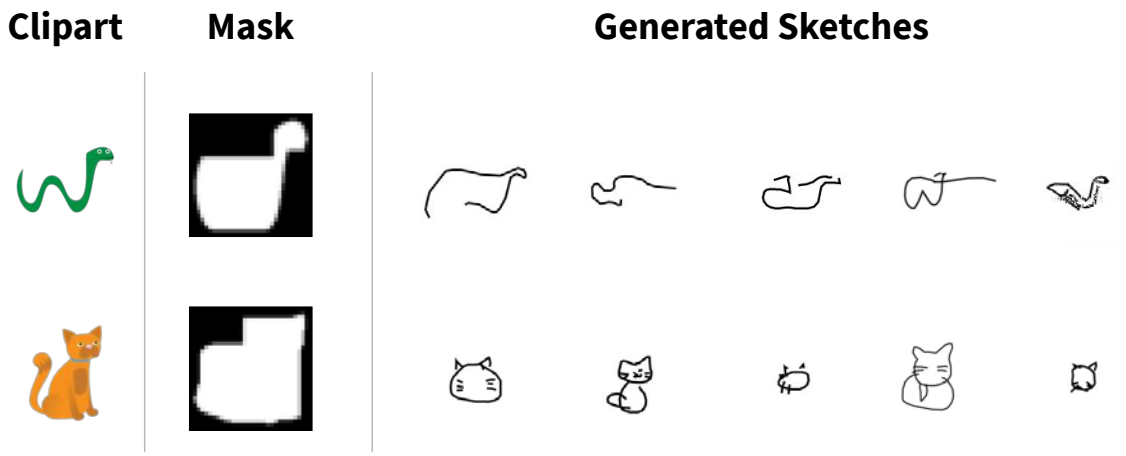


Figure 5.14: Sketches generated by the Object Sketcher with underspecified masks of the Snake and Cat classes.

5.5.3 Data Mismatch Between CoDraw and Target Task

There are differences between the task protocol used to collect the CoDraw dataset and the user interactions in Scones. The conversation in CoDraw only offers the Teller one chance to 'peek' at the Drawer's canvas, which significantly decreases the number of modifications to existing scene objects. As a result, Scones performs well at adding objects of correct classes at appropriate sizes, but is not as advanced at modifying or removing objects. Future work can explore data augmentation techniques, such as super-sampling randomly-perturbed rounds with modifications, or adding removal rounds that mirror the addition of scene objects, to improve the ability of Scones to handle these tasks.

5.6 Towards End-to-End Generation

Two of the main limitations of Scones mentioned in Section 5.5.2—underspecification of masks and limited fine-grained variation of sketches—originates from its two-stage system design and that these stages were not trained end-to-end. As a result, modifications to details of the scene objects are constrained by the representation of objects that are shared between the Scene Composer and the Object Sketchers, which currently only includes several discrete sizes, orientations, masks, and aspect ratios beyond continuous positions for each object.

Towards advancing Scones to become truly flexible in considering and editing objects at the stroke and point level, we conducted additional research in an attempt to train Scones-like models in an end-to-end manner. We collected *TranSketch*, a new dataset of text-based fine-grained modifications for the tree class, and combined it with CoDraw [98] and Quick,

Draw! [91] datasets at the stroke level and investigated behaviors of Transformer models trained directly to generate new strokes given all existing strokes in the original scene and text tokens. In this section, we describe in detail the development of the dataset, models, and metrics, and present some baseline results for this task. We hope that this investigation can serve as a strong starting point for future research towards end-to-end sketch generation and modification based on natural language critique.

5.6.1 Dataset Development

While there are no end-to-end scene modification datasets currently publicly available, we synthesized such a dataset in two steps to mimic partial features of this dataset. We first collected TranSketch, a new dataset that consists of text descriptions of pairs of individual tree sketches. This creates some data examples in our final combined dataset that needs to be modified at the stroke level. We then created the sketched scenes by taking the composition of CoDraw, and placing various Quick, Draw! sketch objects onto the canvas according to that composition. We finally combined all the strokes in a single long sequence for both the original and the final scene to enable end-to-end scene sketch modification.

TranSketch Dataset

The TranSketch dataset contains unique pairs of sketched trees from the Quick, Draw! dataset. Each individual tree is encoded in the same pen-event-based vector format as in Quick, Draw!, and each pair consists of a source sketch and a target sketch. These pairs are presented to crowd-workers on Amazon Mechanical Turk, and the task for crowd-workers is to come up with a short text description summarizing the changes required to modify the source sketch to the target sketch for each sketch pair. The source and target sketch differs at the stroke level but are both trees, hence representing a semantically complex sketch modification task.

We collected 20,032 triplets of a source sketch, a target sketch, and a text description. Each text description in the dataset contains 13.2 words on average, and we filter out outlier descriptions that are either too short or too long. This dataset was primarily investigated by my mentee Luming Chen, and we refer interested readers to a detailed description of this dataset in Luming's master's thesis [23].

Combining CoDraw, TranSketch and Quick, Draw Datasets

To build the final dataset for our end-to-end task, we combine scene compositions from the CoDraw dataset used to train the original Scene Composer in Scones, and the object sketches in both the TranSketch dataset and the Quick, Draw! dataset.

The first step is to pre-process the object sketches. We additionally filter out object sketches that are of low quality and/or are not in suitable poses for composition (e.g., a dog sketch that only depicts the head but not the entirety of the dog). To perform filtering

on these criteria, we utilize a pre-trained classification model on all 345 classes of sketches available in the Quick, Draw! dataset. To identify sketches with correct poses, we first hand-labeled a small number of sketches from the particular class of concern to be examples of either 'accepted' or 'rejected' sketches. We then compute model embeddings at the penultimate layer for all of these labeled sketches, and filter out sketches that have embeddings that are closer on average to the 'rejected' sketches than the 'accepted' sketches.

Then, to further filter out low-quality sketches, we consider the classification probability as a proxy for the recognizability and sketch quality of the sketches. We pick the sketches that are both within the top-N (N being the number of sketches needed from that particular class in the CoDraw dataset) classification probabilities from each class and have correct poses based on the filtering procedure above.

After performing these filtering steps, we use compositions from the CoDraw dataset to resize and position sketch strokes on the canvas. Using notations from Section 5.1.1, we expand each object (o) to not only be a fixed-length vector representation, but a sequence of sketch stroke events ($q_{1...n}$) that is used in Section 5.1.2. For each sketched scene, we randomly sample a single sketched object for each type of objects, meaning that the dog sketch in the original scene, for example, would use the exact same strokes as the dog sketch in the final scene (note that the strokes, however, can be resized based on object specifications). We then position and resize the strokes to maximally but completely fit into each object's bounding box in the corresponding scene. This forms a longer final sequence representation of the data such that each stroke is a single unique time-step in the entire scene, as opposed to in a single object. The text representation remains the same in this new dataset, except with shorter padding at the beginning to reflect the smaller size of the pen-event data format of the strokes. We reduce the number of turns (n) to a single turn to fit into common context-window sizes of Transformers in our experiments. We will formally define these transformation steps in the following Section 5.6.2.

We also randomly shuffle the orders of strokes in the input sketch, to remove any bias on the orders of drawing by users on the input side. On the output side, we explore a few random and non-random ordering options for model-generated strokes. We will further discuss these order variants of our data in Section 5.6.3.

For each composition in our final dataset, we sampled 10 different object sketches, resulting in a final dataset with 122,816/42,624/42,624 examples for train/validation/test splits. Figure 5.15 shows some examples from this final combined dataset (along with the predicted scenes by our model).

5.6.2 Task Formulation and Model Architectures

We formally define the learning task and training objectives in this section. As mentioned in Section 5.6.1, each input data example consumed by the model now consists of the full sketch sequence of the entire scene. We first update our index notation of the strokes such that it additionally includes the object index. Each sketched object $Q_{(i,j)}$ is now represented

by $q_{(i,j,1)\ldots(i,j,h_{(i,j)})}$, with $h_{(i,j)}$ being the length of strokes of j-th sketched object at round i, and would correspond to the symbolic object $o_{(i,j)}$.

We similarly have to reconcile text and sketch embeddings as inputs. We base our definition on Equation 5.1, but we now update the stroke representation instead so it becomes:

$$q'_{(i,j,k)} = [q_{(i,j,k)}, \mathbf{0}_{(300)}] \tag{5.7}$$

All strokes from each object within a single scene are then combined and concatenated to represent the full scene. We can re-index the strokes from 1 to $h_{(i,j)}$ (which corresponds to the original object $o_{(i,j)}$), to 1 to $h_i = \sum_j h_{(i,j)}$ to include combined lengths of all objects' strokes. The sketches of the entire scene now become $q'_{(i,1)\ldots(i,h_i)}$. Note that the new index can be ordered in various ways, meaning that the strokes in the new scene can be drawn in different orders which we will discuss in detail in Section 5.6.3. With this new notation, we define the new model inputs and outputs to be:

$$S'_i = [q_{(i,1)\ldots(i,h_i)}] = \textbf{Transformer}([q'_{(i-1,1)}, \ldots q'_{(i-1,h_{(i-1)})}, t'_{(i,1)}, \ldots t'_{(i,l_i)}, q'_{(i,1)}]) \tag{5.8}$$

In this experiment, we also only consider the previous turn ($n = 1$ according to the original notation) due to limitations of computational resources which affects the maximum context size in the Transformer that we were able to support.

To train our model, since the output now is in the raw stroke format that was defined by the Object Sketchers, we use the same negative log-likelihood loss function:

$$L = -\log p(S'_i) = \sum_{k=1}^{h_i} -\log p(q_{(i,k)}) \tag{5.9}$$

The likelihood P is defined by either Gaussian Mixture Models (GMMs) or categorical distributions that we parameterized with the specific Transformer's outputs. This GMM paradigm is the same as the Object Sketcher and prior work in sketch modeling, and was described in detail in Section 2.4. Each of these choices is considered with either continuous or discrete coordinates mentioned in Section 5.6.3.

The Transformer architecture we used in this set of experiments is a Transformer with masked self-attention similar to GPT-2 similar to the Scene Composer (Section 5.1.1). However, we scaled up the Transformer to have 512 hidden units and 8 attention blocks and 8 attention heads, since this model needs to additionally handle the raw sketching mechanics that were previously handled by the Object Sketchers. There are 10 distributions in the Gaussian Mixture Models we used (only applicable to models with continuous coordinates). The model was trained using an Adam Optimizer [99] with an initial learning rate of $lr = 6 \times 10^{-4}$, and had a linear warm-up and decay schedule similar to that of the original Transformer model [184]. The model was trained on our newly compiled dataset for 125 epochs in which the validation loss converged by the end of training.

5.6.3 Task Variants

We have conducted experiments on several variants of the task. These variants primarily differ by the order of input and output strokes given that there is ambiguity on how strokes of different objects are concatenated into a single long sequence that represents the sketch scene. The variants also differ by the representation of geometric coordinates of the pen events.

We first consider different input and output orders of pen events in our model variants' learning task. For all variants, we group pen events into strokes, such that all pen events (time-steps) within the same strokes are ordered identically as they were in individual object sketches. For input strokes of all variants, we first randomly shuffle the order of strokes so that there are no prior assumptions on how users might draw various strokes of various objects in the scenes. We then experimented with several different orders for output strokes. The first order we explored is obtained by first grouping the strokes by their original objects, and then sorting the objects by their y coordinates, finally tie-breaking by their x coordinates. This corresponds to the procedure where top-left objects are drawn first before objects at the bottom right of the canvas. We also experimented with an order where the objects are randomly shuffled, but the orders of strokes within each object were preserved.

We then experimented with various representations of pen event coordinates: absolute coordinates with reference to the top left of the canvas, and relative coordinates with reference to the end of the previous pen events. Note that in all variants, the absolute coordinates are provided as additional information to the model regardless of the actual coordinate representation.

We further experimented with discrete and continuous coordinates, and trained models with continuous coordinate data with GMMs and models with discrete coordinate data with categorical distributions.

5.6.4 Metrics

To effectively measure task success, we adapted a well-known metric to our context to ensure both the nature of our task is reflected, and this metric is compatible with various stroke output formats of our model variants. We additionally prefer metrics that are extensible to future pixel-based models, given the high performance of pixel-based, general text-to-image methods such as DALL-E [150, 151] and Imagen [165].

Our metrics are centered around the Chamfer Distance, which given a set of points to be measured and a set of reference points, for each point in the sequence to be measured we pair it up with the reference point that produces the least Euclidean Distance to that particular measured point. We then take the average over all measured points' minimum Euclidean Distances to be one measurement value. This process is repeated by swapping the reference and measured sequence. The final metric value is the sum of the two values in the two measurements. In our application, when we consider the ground-truth pen events

($Q = \{q\}$) and the predicted pen events ($\hat{Q} = \{\hat{q}\}$), the definition of Chamfer Distance would be:

$$\textbf{Chamfer}(Q, \hat{Q}) = \frac{1}{|Q|} \sum_{q \in Q} \min_{\hat{q} \in \hat{Q}} d(q, \hat{q}) + \frac{1}{|\hat{Q}|} \sum_{\hat{q} \in \hat{Q}} \min_{q \in Q} d(\hat{q}, q) \tag{5.10}$$

with d being the distance function. This is the Euclidean Distance between the absolute coordinates of the end-points of the two pen events in our computation in our case.

At a high level, this metric represents how well the reference points are represented by the measured points and in the opposite direction symmetrically. We extend this computation to two choices and aggregations of Q and $\hat{Q}$. We compute Chamfer Distance when Q and $\hat{Q}$ are either 1) all pen events in the sequence, or 2) all starting points of the strokes in the sequence. We also aggregate these values differently across the sequence, including taking the average value per point and the average value per stroke.

5.6.5 Baseline Results

Quantitative Metrics

For each of the proposed task variants in Section 5.6.3, we compute each aggregation variant of the metrics described in Section 5.6.4. Table 5.2 summarizes the metric values on the test set of our proposed dataset in this task. We found that the model that uses **D**iscrete and **R**elative coordinates, with a fixed, **S**orted output order of the predicted strokes (i.e., predicting the top left object first) yields the best Chamfer Distance in all aggregation variants. With Chamfer Distance at 0.0784 when aggregated and averaged across all points in the predicted and ground-truth pen event coordinates, this model outperformed other model variants with continuous and absolute coordinates and those with a random output order.

Table 5.2: Chamfer Distances for end-to-end sketch critique, lower is better.
D - **D**iscrete Coordinates, **C** - **C**ontinuous Coordinates,
A - **A**bsolute Coordinates, **R** - **R**elative Coordinates
N - Ra**N**domized Output Objects, **S** - **S**orted Output Objects

	Chamfer - All Points	Chamfer - Stroke Start Points	Chamfer - Average by Stroke
C + R + S	0.176	0.275	0.159
D + R + N	0.174	0.263	0.164
D + A + S	0.258	0.296	0.217
D + R + S	**0.156**	**0.242**	**0.148**

We hope all statistics of the aforementioned model variants that we included in Table 5.2 can serve as baselines for future end-to-end scene sketch modification models. We would also like to acknowledge that the usage of Chamfer Distance allowed us to fairly compare models that output continuous and discrete coordinates, given that they are trained with different losses which can yield significantly different values.

We additionally split the set of evaluated points into points that are 'copied' from the original scene belonging to objects that already exist in the original scene (note that these points can move or scale in the new scene, and these include the 'TranSketch turns' which contain trees in the original scene), and points that are 'added' which did not exist in the original scene. However, since we only have this point partition knowledge for ground-truth points, we are only able to compute an *asymmetric* part of the Chamfer Distance outlined in Equation 5.10. We define this as *partial* Chamfer Distance in the following equation:

$$\textbf{Partial\_Chamfer}(Q, \hat{Q}) = \frac{1}{|Q|} \sum_{q \in Q} \min_{\hat{q} \in \hat{Q}} d(q, \hat{q}) \tag{5.11}$$

Table 5.3 shows the results separately for the 'copied' and 'added' points based on the Partial Chamfer Distance defined above. We observe that the discrete model is better at copying points, while the continuous model is better at generating new points. Therefore, we believe one promising future research direction is to combine the strengths of a continuous model in creating new objects with the accuracy of a discrete model in copying original objects. We can potentially achieve this by introducing the inductive bias in the continuous models' Gaussian distributions, into the discrete model by changing its target categorical distributions to be Gaussian-like.

Table 5.3: Partial Chamfer Distances (from ground-truth to predicted points) for copy and added objects, lower is better. Both models used **R**elative coordinates and **S**orted output object order.

	Partial Chamfer - All Points - Copy	Partial Chamfer - All Points - Add
Continuous Coordinates	0.0558	**0.116**
Discrete Coordinates	**0.0540**	0.141

We finally analyze Chamfer Distances separately for examples that correspond to the new TranSketch dataset and the other examples that correspond to the original CoDraw dataset. Recall that the examples in the TranSketch dataset require stroke-level modifications of tree objects in the scenes based on text descriptions. Table 5.4 shows the quantitative results separately for TranSketch and non-TranSketch examples. We observe that TranSketch examples appear to have significantly lower Chamfer Distance compared to other examples. We believe this might be related to the fact that the positions of trees are already defined

in the original scenes in TranSketch turns, and that the only task of the model is to redraw the tree at the same location but with different strokes. We believe such statistics reflect that additional metrics might be required to further understand stroke-level modification performance. At the very least, however, an effective evaluation might require researchers to separately compare TranSketch and non-TranSketch examples similar to this part of the analysis.

Table 5.4: Chamfer Distances for TranSketch rounds and Non-TranSketch rounds, lower is better. Both models used **R**elative coordinates and **S**orted output object order.

	Chamfer - All Points - TranSketch	Chamfer - All Points - Others
Continuous Coordinates	0.137	0.193
Discrete Coordinates	**0.0864**	**0.186**

Qualitative Exploration

We also reflect qualitatively the performance of our models in this end-to-end sketch modification task. Figure 5.15 shows various examples that include original scenes, ground-truth target edited scenes, and the predictions made by our model. We included the ground-truth scene to demonstrate the resulting scenes in this dataset generated by our proposed procedures in Section 5.6.1. The model that performed the prediction in Figure 5.15 was the best-performing variant quantitatively—the model with discrete and relative coordinates, and a sorted output order. In a), b), and c), we see that the model is capable of generating individual objects at appropriate positions, with the sun at the top right in a) and the cloud on the top left for b). We additionally see that the model is able to handle more complex sketches with many strokes (i.e., pen lifting and re-drawing again) with the raindrops in the rain-cloud in c) differentiated against a simpler cloud shown in b). In d) and e), we observe the model is able to copy original scenes with high accuracy, while adding the appropriate new classes of objects such as tents, tables, and trees.

Finally, in f), we observe that the model is sometimes able to handle stroke-level modifications for trees that we extracted from the TranSketch dataset. This example refers to the stroke-level modification task where the model needs to make the trunk skinnier and the leaves less hand-like, and we observe the model was able to make such stroke-level changes in the predicted scene in this case. Nevertheless, we have also observed that the model failed to follow more complex text instructions in other cases, prompting future research on the consistency of modeling correspondences between fine-grained text and strokes.

Since all of these scenes are consumed and generated from scratch in an end-to-end manner by a single Transformer with masked self-attention, with no explicit or hand-crafted copying mechanisms, this reflects the current ability of our trained model to handle sketches

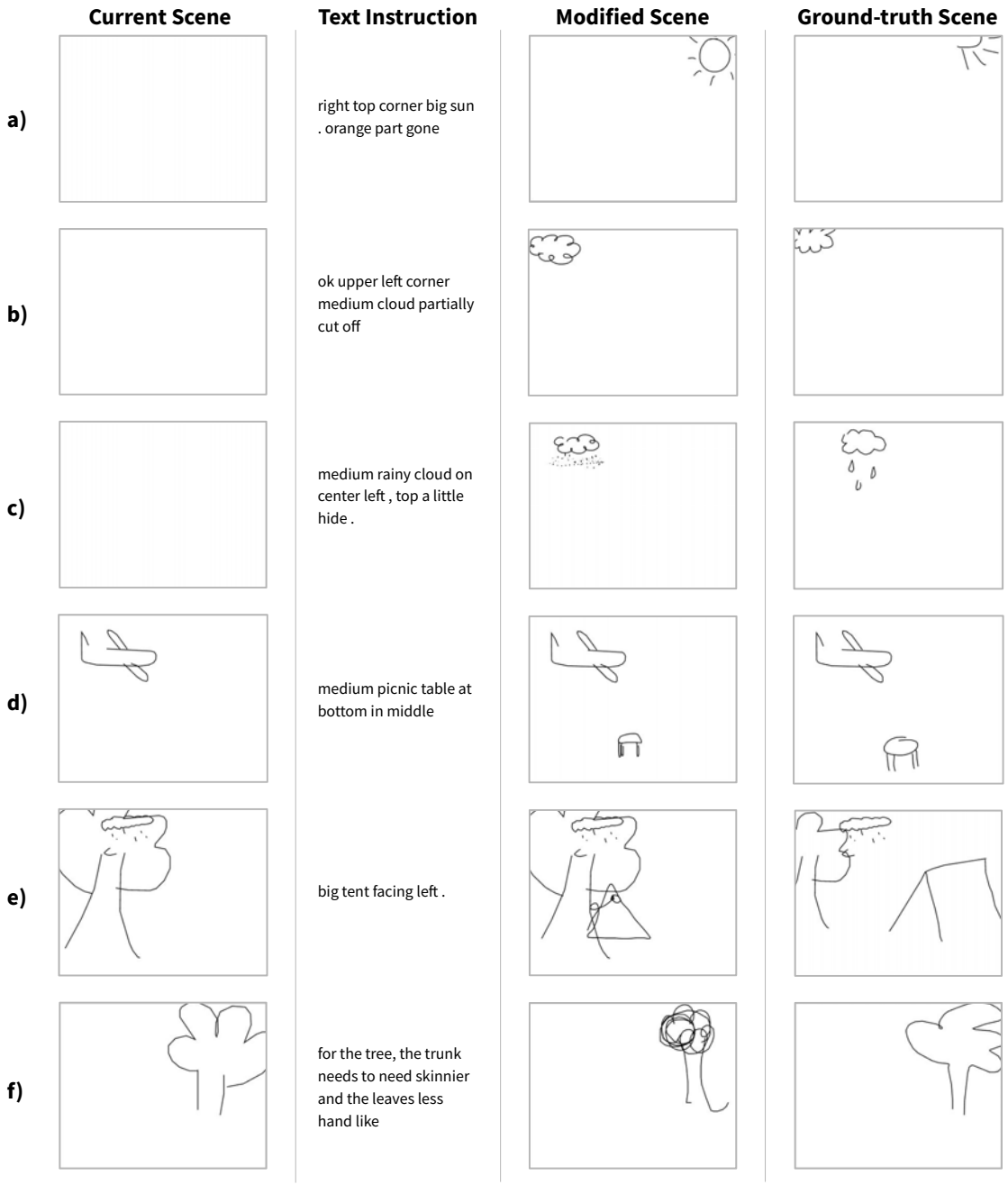


Figure 5.15: Generated and ground-truth scenes for the end-to-end sketch modification task. The end-to-end model is able to copy sketched objects from the original scene, and modify and add sketched objects into the output predicted scene.

of various concept classes and to copy original scene objects. This also shows the potential for Transformers to support more complex, stroke-level edits of scene sketches shall the data become available in the future.

We additionally qualitatively review the correspondence between our primary metric—Chamfer Distance, and the similarity between predicted and ground-truth sketched scenes. Figure 5.16 shows two example pairs of predicted and ground-truth sketched scenes, one with a low Chamfer Distance (Figure 5.16a) at 1.40×10^{-3}, and another one with high Chamfer Distance (Figure 5.16b) at 0.449. The scene pair in Figure 5.16a was taken from our preliminary model which only learns to copy exactly from the ground-truth scene. We observe a high degree of similarity between the copied scene and the original scene in this case, and this is reflected by the low Chamfer Distance. Conversely, we observe in Figure 5.16b that when the model mispredicts and leaves one important object (the tree) off the scene, the Chamfer Distance between this pair of scenes is high. This reflects a reasonable ability of Chamfer Distance to represent perceptual scene similarity, especially when some objects and/or strokes are obviously missing.

(a) Chamfer Distance = 1.40×10^{-3} (Low)

(b) Chamfer Distance = 0.449 (High)

Figure 5.16: Predicted and ground-truth scenes with (a) low and (b) high Chamfer Distances.

5.6.6 Key Research Challenges

Through developing these baseline models for end-to-end sketch modification, we reflect on several key research challenges that we believe future research in this area could focus on. The first important challenge is the appropriate re-use of original strokes in the scene. We observe that some models have encountered difficulty in even copying over existing sketch strokes that do not need to be modified. While we intentionally shuffle the input stroke orders to reflect realistic use-cases that users might sketch the scene in any orders, this creates great difficulty for some models (especially the model that uses continuous coordinates) to copy over identical objects—some strokes were missing, and others were repeated in the output scenes generated by these models. This issue becomes even more apparent when the output scenes do not need an exact copy of certain objects, but require resized and/or moved copies of existing sketched objects from the original scenes.

Another key research challenge is related to the original limitation of the CoDraw dataset, which was unbalanced with various classes of objects and consists of significantly more turns that add new objects into the scene than turns that only move or edit existing objects in the scene (discussed in detail in Section 5.5.3).

Finally, we believe while Chamfer Distance is a good starting point to fairly compare models with various architectures and output formats on this task, we believe additional normalization would be helpful across scenes with different numbers of points: scenes with higher numbers of ground-truth points are denser, hence it would be easier for even randomly plotted points to achieve low Chamfer Distances. A potential normalization can use sets of randomly sampled points of similar lengths to the ground-truth points as prediction baselines to more fairly evaluate similarities between predicted scenes and ground-truth scenes.

Chapter 6

Words2ui: User Interface Prototype Generation and Retrieval from Text

Beyond building systems to support sketching for non-experts, we believe the benefits of deep-learning (DL) models can be extended to building systems that support professional design domains. Our first attempt is to support user interface (UI) design, a complex process that requires significant domain-specific expertise gathered from years of education and experience. One critical step of this process is to produce low-fidelity mock-up prototypes from high-level specifications of requirements (as described in Section 2.1.2 and prior literature [5, 134, 145, 189]), and this step often involves professional designers dedicating extensive time and effort. Many small-scale app developers and amateur designers, however, might not have the resources and/or expertise for such a design task, leading to lower-quality end products. While producing these mock-ups from scratch might be difficult, specifying high-level requirements for these UIs in natural language requires less design expertise. These specifications might even already exist in these practitioners' current development processes.

As such, we believe computational systems that can create concrete UI design mock-ups guided by high-level text descriptions can greatly benefit UI design practitioners. Moreover, these computational systems can be helpful in communicating and discussing designs with clients and/or other non-experts, as the generated design artifacts can be useful for them to ground their discussions on (see Section 2.1.3 for a detailed discussion on sketches' role as a communication medium). Producing diverse sets of UIs relevant to specifications still takes significant effort and time even for professional designers, and such a system can help reduce professional designers' workload, allowing them to make more important decisions in the design process.

The first step towards such a system is developing methods that can robustly, coherently, and diversely generate plausible UI design mock-ups from brief text descriptions about the desired UIs. Coupling recent advancements in the DL community on effective text-to-image retrieval and generation models with large datasets of pairs of UIs and captions [186], we

introduce two DL-based retrieval and generation models[1] that are first of their classes to be able to create UI mock-ups based on a wide range of natural language descriptions (see also a brief overview in Figure 6.1):

- We introduce *UI Generator*, the first deep generative model that is able to generate UI mock-ups from scratch with only a high-level text description about the desired UI, and a set of post-processing techniques to filter and present high-quality UI designs to users.
- We introduce *Multi-modal Retriever*, the first DL model that learns cross-modality correspondence and latent representation to retrieve design examples from a large UI corpus using high-level text descriptions about the desired UIs.

In this chapter, we first describe our methods towards developing each of these models. We then developed a set of benchmark metrics to better assess this novel task effectively and quantitatively, especially when comparing to existing and future approaches. We then present quantitative and qualitative results of our generative and retrieval models. Towards the end of this chapter, we report feedback provided by experts after inspecting artifacts generated by our system. We additionally outline several applications that we envision to support in the future based on this feedback.

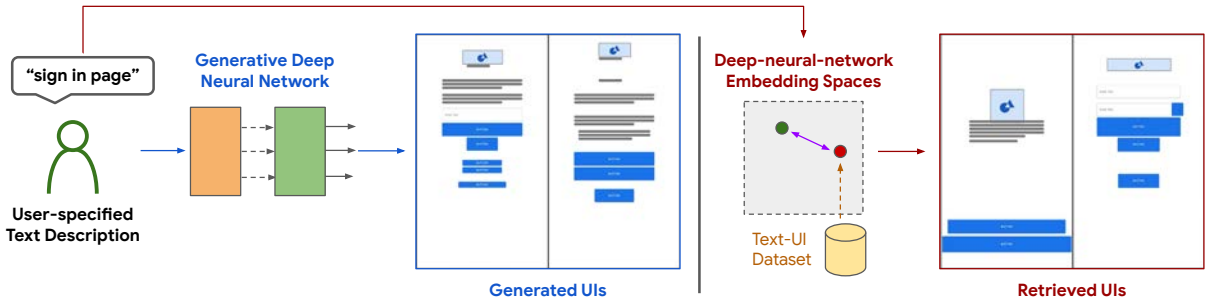


Figure 6.1: High-level overview of the two proposed retrieval and generative methods for creating UI mock-ups from text descriptions with deep neural networks.

6.1 UI Generation and Retrieval Methods

In this section, we describe our two proposed methods towards generating and retrieving UI mock-ups from text descriptions in detail. Figure 6.2 illustrates the commonalities and differences between the two methods. The *UI Generator* is a generative model that learns

[1]Part of the work presented in this Chapter was presented as a workshop paper at the Computational UI workshop at CHI 2022 [79].

to synthesize UIs entirely from scratch only from high-level text descriptions. The *Multi-modal Retriever* includes a dual-encoder [140] that is capable of processing (embedding) a text description provided by a user into a common embedding space that is populated with both text and UI embeddings. We can then directly retrieve the nearest UIs in this space. We describe and discuss different processes of dataset selection and processing, model architectures, training configurations, and resources required by each of these methods.

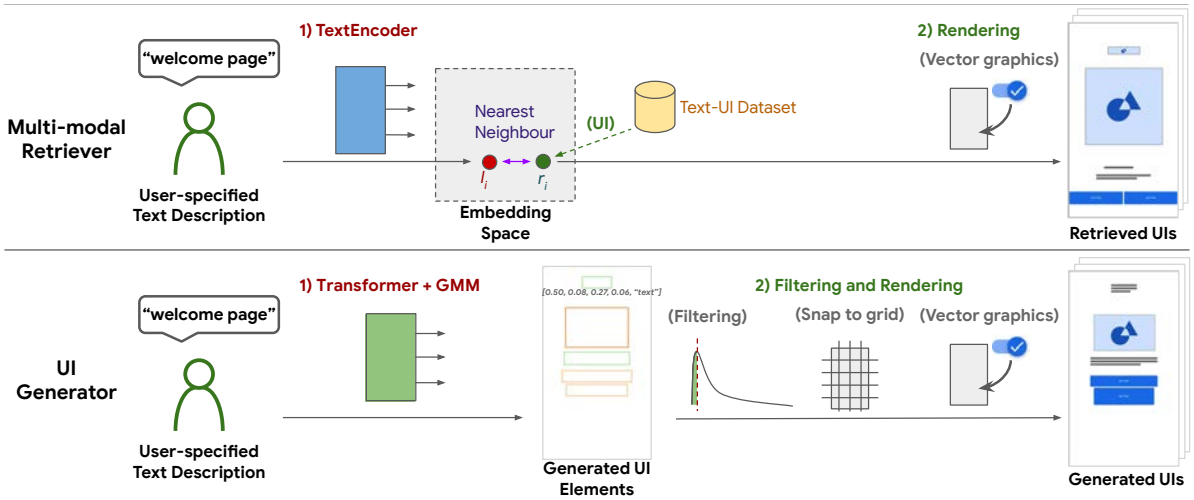


Figure 6.2: Side-by-side comparison of overall workflows of each of our proposed methods.

6.1.1 Datasets

While both methods use different modalities and features for training, they all commonly use the screen2words dataset[2] [186], a large-scale dataset with more than 112,000 high-level text descriptions corresponding to detail attributes and screenshots of more than 22,000 UIs in the RicoSCA dataset [117]. To our knowledge, it is the only large-scale dataset that contains pairs of high-level text descriptions and UI examples. These pairs are required by our target task of text-based creation of UI mock-ups.

We used the original dataset splits with 15,712[3]/2,364/4,310 UIs in the training/ validation/test sets. The UIs in each of the splits are captured from different apps. Each UI is captioned by five crowd-workers, producing five English sentences with under 10 text tokens. We also embed text descriptions using a pre-trained BERT model [39]. We extract one BERT embedding t_i for each token e_i in a text description sequence.

[2] `https://github.com/google-research/google-research/tree/master/screen2words`

[3] The original screen2words training set contains 15,743 UIs. However, there are 31 UIs that we were unable to process, and hence we excluded them in our training set.

6.1.2 UI Generator

Generative models are central to our investigation because they can learn from existing designs to synthesize novel UIs. These generated UIs are potentially unconsidered by human designers based on unseen and flexible text descriptions. This ability of a successfully trained text-to-UI generative model represents a deep understanding of the design space and can enable new applications for UI Design. Therefore, we built Transformer-based generative models to generate UI element attributes directly from text descriptions.

Additional Pre-processing

While each UI example in the combined RicoSCA and screen2words datasets includes detailed attributes of all UI elements, most prior work uses a more semantically meaningful subset of UI elements with additional annotations from [122]. This subset reduces the task difficulty yet retains the most meaningful part of the UI generation task, so we adopt this common workflow. These UI elements are then flattened within each UI by sorting them based on their approximate Y coordinates, and tie-broken with X coordinates. We also filter out UIs with more than 128 elements, like in [60].

Task Formulation and Model Training

Once we have pairs of clean and sorted UI element attribute sequences and text descriptions, we formulate the machine learning task as follows. Our model takes k BERT vectors of a text description $t_{1...k}$ and generates n UI elements $u_{1...n}$. Each UI element is represented by combining normalized X and Y coordinates and width and height values $[x, y, w, h]$ relative to the screen dimensions, with a one-hot vector $e^{(c)}$ that is 1 at the index of the semantic class number that the UI element belongs to, and 0 anywhere else. This gives us one $u_i = [x_i, y_i, w_i, h_i, e_i^{(c)}]$ vector for each UI element.

We can then consider this problem as a typical sequence-to-sequence translation problem where we 'translate' a sequence of text tokens to a sequence of UI elements. We explore two Transformer-based methods for UI elements to interact with input text tokens.

The first method is implemented as a Transformer Encoder-Decoder model, where text descriptions t are encoded with an independent, separate set of encoder weights into final layer embeddings (one for each text token). Similarly, the UI elements are also decoded using a decoder exclusive to UI element inputs u, but they interact with the encoder embeddings using cross-attention at each attention block. This approach is more common in sequence-to-sequence modeling problems, especially those that operate on two separate modalities such as image captioning. Figure 6.3 illustrates this model architecture variant in detail, including the employed pre-processing and loss computation steps.

The second method is implemented as a Transformer decoder-only model, where text token inputs t are prepended to UI element inputs u (after passing through embedding layers specific to UI or text) and masked attention is applied at each time-step. In this case, all inputs are modeled with the same set of weights, and each attention block has access to

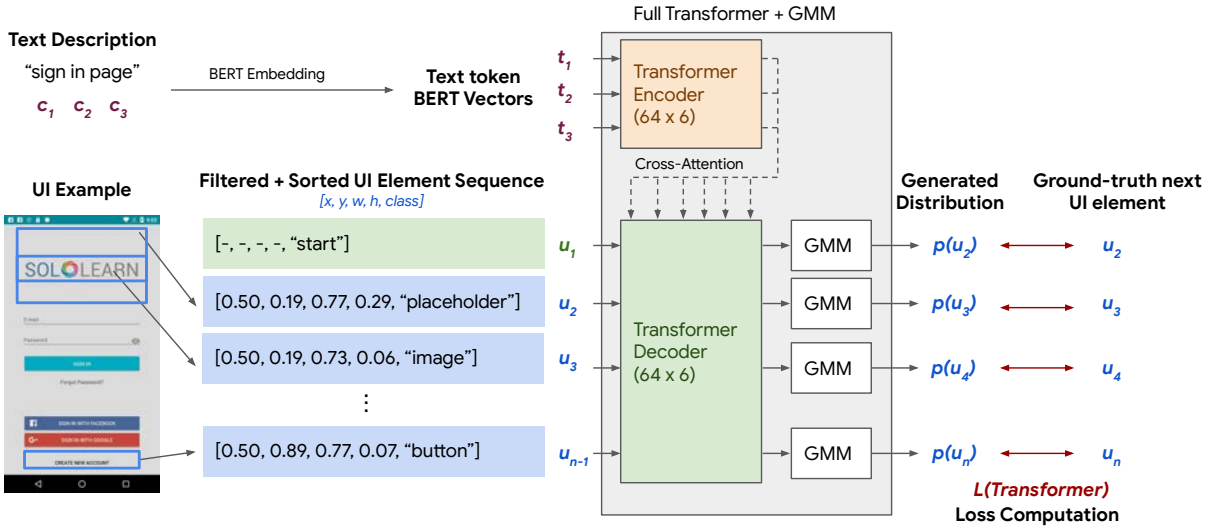


Figure 6.3: Model architecture of the UI Generator (Transformer Encoder-Decoder variant).

all intermediate representations of both text and UI element inputs. This approach is similar to language modeling tasks that are typically used to model data in a single modality [16], but have recently also found success in cross-modality modeling tasks [151].

While we can train the decoder to generate exact coordinates and classes of UI elements, a generative model commonly outputs a distribution instead of a specific token in the inputs' format for each time-step. Because each UI element has continuous attributes x, y, w, h and a discrete attribute $e^{(c)}$, we split the output of the decoder so that one part of it is used to parameterize distributions for continuous attributes while the other is to parameterize distributions for discrete class attributes. For the first part that parameterizes the distribution of continuous attributes, we explore two options: 1) Gaussian Mixture Models (GMM), which forms a Mixture Density Network [13] with the Transformer (described in detail in Section 2.4); 2) categorical distribution with discretized coordinates. The other part of the outputs is treated as logits of a categorical distribution of the UI element class.

At a high level, this allows us to generate a distribution of output UIs instead of a single prediction. This is especially important for our problem because the text descriptions are often underspecified—a single text description, such as 'login page', can correspond to many potential candidate UI designs. Overall the Transformers generate a probability distribution of each predicted output UI element:

$$p(u_i) = p(u_i|\theta), u_i = [x_i, y_i, w_i, h_i, e^{(c)}], \theta = \textbf{Transformer}(u_{1...i-1}, t_{1...k}) \tag{6.1}$$

To train these generative models, we minimize the total negative log-likelihood of the ground-truth UI element sequence given the probability distribution above (note that we don't have to train $p(u_1)$ as it is always the start token).

$$L(\textbf{Transformer}) = -\sum_{i=2}^{n} \log(p(u_i)) \tag{6.2}$$

This is a typical training process for generating continuous, low-dimensional attributes with a sequence model. The specific definition of the likelihood $p(u_i)$ depends on whether discrete or GMM-based parametrization is used to model the continuous positional attributes of the UI elements. A detailed description of the optimization process for GMMs can be found in Section 2.4, and discrete distributions are optimized with typical cross-entropy loss functions (equivalent to optimizing Equation 6.2 directly with corresponding model outputs as likelihoods).

We choose hyperparameters ($d_{\text{model}} \in \{128, 64, 32\}$, $n_{\text{layers}} \in \{6, 4\}$) of our models based on performance on the validation set. We trained our models using an effective batch size of 1024 with an Adam optimizer with a starting learning rate of 10^{-3} and we manually decreased it to 10^{-4} when the validation loss plateaued. We implemented our model using Trax and trained them on Google Cloud TPU v3 with 32 cores.

Sampling UIs

Once a generative model is trained, we can generate all the elements on a UI given a text description autoregressively. At each time-step we feed the Transformer all previously generated elements along with the text description, and sample the Transformer for the next predicted UI element $\hat{u_i}$ from the distribution $p(u_i|\textbf{Transformer}(u_{1\ldots i-1}, t_{1\ldots k}))$. We repeat the process until the model outputs a special token of `EOS` to indicate the end of the generation process. For our methods that model continuous attributes with GMMs, we use greedy sampling and apply the temperature parameter of $\tau = 1.0$ or 0.1 to both the categorical distribution of UI element classes (by rescaling the logits) and the GMM distribution (by rescaling the variances). This temperature parameter controls the stochasticity of the model that translates to diversity of generated UIs for a given description. For models that discretize all UI attributes, we use nucleus sampling with cutoff $p = 0.9$ for a fair comparison with prior work. The sequences of numeric and categorical attributes of the generated UI elements are taken as final outputs of the model.

6.1.3 Multi-modal Retriever

While generative models enable novel designs to be generated, one important interaction that is similarly important in the UI design process is to search and find inspirational designs from large corpora of real designs [70, 110, 157]. This allows designers to understand the design landscape and comparatively evaluate their design against prior art. Towards this goal, we developed a *Multi-modal Retriever* that enables designers to retrieve existing UIs by providing text descriptions. This method embeds both text descriptions and UIs into a shared embedding space using a contrastive-learning method, and then performs retrieval

across the modalities directly in this space. To our knowledge, this is the first cross-modality text-to-UI DL retrieval model.

Investigating cross-modality retrieval also aids in the evaluation of generative models that we developed in Section 6.1.2. This is because a crucial part of evaluating text-to-UI task success is measuring the relevance between the generated UIs to their input text descriptions. However, this correspondence is hard to be directly quantified with simple rules or metrics. Hence, a cross-modality retrieval model that learns correspondence and distances between artifacts in the two modalities (text and UIs) needs to be developed and used to evaluate text-UI relevance of the generative models. Some of the most established metrics in text-to-image generation (e.g., R-precision) similarly require the development of a text-image cross-modality retrieval model [140].

Additional Pre-processing

We included a large amount of information from both the text descriptions and the paired UIs from the screen2words dataset to obtain cross-modality embeddings that encode more nuanced and detailed UI knowledge in the Multi-modal Retriever. We flattened UIs to element attribute sequences and added a start token, an end token, and a token that contains pooled BERT embedding of the text description of the UI's app to each sequence. For each element in the UI (including intermediate elements), we embed the dimensions (x, y, w, h), the element type (same as the ones used in the screen2words dataset), and a single pooled BERT embedding of the element's text content. This gives us detailed content-based, semantic, and geometric information of each element in the UIs. We also filter out UIs that are longer than 512 tokens.

Note that to support the evaluation of the UI Generator, we train an additional variant of the Multi-modal Retriever that uses the same pre-processing method as the UI Generator described in 6.1.2. This dual-encoder is based on only UI layout attributes without the content of the UIs, so that generated UIs can be used as inputs to this model. This variant is employed instead of the full Multi-modal Retriever in Section 6.3.1 since it takes the same input format as the output format of the UI Generator.

Model Architecture and Training

The Multi-modal Retriever encodes text descriptions and UIs respectively using two sub-modules **TextEncoder** and **UIEncoder**, which is adapted from a dual-encoder [140] used for text-to-image retrieval. Each of **TextEncoder** and **UIEncoder** is a Transformer Encoder with hidden-layer size of 64, intermediate size of 256, and 4 layers (chosen by performance on the validation set).

Given a text description, we obtain the BERT embeddings $t_{1...k}$ for each of the k text tokens from the pre-processing step. From these embeddings we can use the **TextEncoder** to obtain a single, fixed-length text embedding $e_t = \textbf{TextEncoder}(t_{1...k})$ by taking only the output of the special start token as the 'pooled vector' of the sequence. Similarly, from the

pre-processing step we obtain the flattened sequence of n UI element attribute vectors $u_{1...n}$. From this we can obtain a single, fixed-length UI embedding $e_U = \textbf{UIEncoder}(u_{1...n})$ at the start token position similar to the **TextEncoder**.

With pairs of corresponding embeddings e_t and e_U, we obtain a mini-batch of K pairs of embeddings $e_{t,1...K}$ and $e_{U,1...K}$. We follow the Contrastive Learning paradigm (described in detail in Section 2.5) and formulate a loss function that would minimize the distance between matching pairs of e_t and e_U and maximize the distance between unmatching pairs. We minimize the following bidirectional in-batch sampled softmax loss defined in Equation 2.14. Translating into the original notations in Equation 2.14, we define $f_U = \textbf{UIEncoder}$ and $f_t = \textbf{TextEncoder}$, that the distance function $S(e_t, e_U)$ is the dot product between the text and UI embeddings.

We use an Adam Optimizer [99] with a constant learning rate of 0.001 to train our model with this loss. We implemented our model using Trax[4] and trained it on Google Cloud TPU v3 with 32 cores with mini-batches of 64 samples per replica.

Embedding and Retrieval

To retrieve UIs given a user's text description t_d, we compute $e_{t,d} = \textbf{TextEncoder}(t_d)$ and find the nearest neighboring UI embeddings e_U in the set of UIs to be retrieved using the dot-product similarity metric. The sequences of numeric and categorical attributes of the UI elements of the retrieved UIs are taken as final outputs of the model.

6.1.4 Rendering

The final step of all proposed methods is to render the produced UIs in the representation of low-fidelity mock-ups. Low-fidelity mock-ups are commonly used in early stages of UI design because they enable designers to focus on essential design ideas instead of being distracted by details. Since the UIs outputted by both methods are sequences of numeric and categorical attributes, converting them into the mock-up format more familiar to designers enables them to be potentially more effective in supporting design applications.

We first adjust the numeric outputs from the UI Generator that uses GMMs for continuous attributes to better aligned values. We snap each UI element to one of 32 discrete grids along each dimension to remove minor alignment discrepancies as the model estimates continuous values for each of the continuous variables (x, y, w, h).

For both methods, we then curated a set of SVG elements from the Material Design guidelines [57], and programmatically modified specific graphic attributes within these SVG elements for each individual UI element (e.g., only resizing the length of the bar of a slider instead of stretching the circle selector). These enhancements allow us to accommodate different aspect ratios of the UI elements to avoid stretching, producing coherent final UI mock-ups compiled as HTML documents. Note that for retrieval-based methods, we can

[4] `https://github.com/google/trax`

optionally provide the original screenshots of the UIs in addition to the mock-ups as these UIs are taken directly from the dataset.

6.2 Benchmark Metrics for Generative UI Models

Prior to measuring the performance of our proposed methods quantitatively, we investigate and define novel quantitative metrics specifically for generative methods. This is because while measuring the performance of the Multi-modal Retriever can be done with standard information-retrieval-based metrics (e.g., Top-k Accuracy, Mean Average Precision), measuring a generative model's performance is much more difficult and non-trivial, especially for such a new task as text-to-UI generation as ours.

Our novel benchmark for the UI Generator consists of a set of metrics that are adopted or modified from related work for quantitatively comparing and contrasting various approaches. Specifically, we define success of our text-conditioned UI generation task to have three critical aspects:

Well-formedness *What is the quality and realism of the generated UIs on their own?* This can be measured using existing metrics for UI quality used in prior unconditional models and serves as a check that our model is able to generate convincing UIs comparable to prior models.

Relevance *How relevant are the UIs to the text descriptions?* This aspect is novel to our task since we have the additional requirement of conditioning the generated UIs on text. We adapt several metrics from the text-to-image literature and outline important reasons for the required modifications for these metrics to be applied to our problem.

Diversity *How diverse (while still being relevant to the text description) are the generated UIs?* This is typically measured with the similarity of the distributions of the generated UIs and the dataset distributions. We additionally condition these distribution similarity metrics on text-UI relevance to ensure the metrics' relevance to our task.

6.2.1 Well-formedness

One of the most important properties of a UI generation model is to be able to generate convincing and realistic UI layouts of interest to design applications. We adopted three existing well-formedness metrics defined in literature that are used as evaluation metrics to effectively compare our models with prior work [7, 60, 111]. This also serves as a check that our model is comparable to the 'baselines' in UI generation performance regardless of the text conditions. These metrics are respectively:

- **Overlap**—the percentage of area on the entire UI that is occupied by at least two elements.
- **IOU**—the average intersection-over-union between any two elements in the UI.

- **Alignment**—a metric introduced in [111] that approximates the average inter-element alignment distance between elements in the UI.

6.2.2 Relevance

The most important contribution in terms of evaluation of our work is the introduction of several new text-UI metrics modified from prior work. This is because an important new requirement for our novel task is to have the generated UIs be relevant to text descriptions. Moreover, there are significant differences between text descriptions used to describe UIs over ones used to describe natural images.

R-precision

R-precision (RP) is a major existing metric to measure cross-modality relevance commonly used in text-to-image generation models. It relies on a cross-domain distance function that measures the distance between a particular output (in prior work this could be an image, and in our use case this would be a UI) and a text description. This function is typically implemented using neural networks because of the complex correspondences between text and UIs. Typically, a retrieval model between UI and text (i.e., a *dual-encoder* [140]) similar to the Multi-modal Retriever will be used. Therefore, our previously developed Multi-modal Retriever can be repurposed to help with evaluating the generative model as mentioned in Section 6.1.3. With this trained model, RP reports the proportion of generated UIs $\hat{u}$ that rank their corresponding text prompt t_p as the nearest (top-1) neighbor in the dual-encoder's embedding space.

Limitations of R-precision for Text-to-UI Generation

One primary limitation that we found with using the original RP for text-to-UI generation is that RP only rewards cases when the paired text of the UI is ranked as the top example by the dual-encoder. This means the text needs to be uniquely qualified to describe the particular paired UI. In our domain, however, a single text description can correspond to many UIs, and the same UI can correspond to many text descriptions. This is demonstrated by our observation that screen2words text descriptions are more high-level and could describe more contextual and/or background information about the UIs, especially when compared to text descriptions of natural images. For instance, a login-screen UI is often paired with the description 'login screen of an app', instead of being specific at the element level where it describes the number and/or positions of certain types of UI elements.

To quantitatively verify our hypothesis of non-unique matching between UIs and their paired text descriptions, we present two findings: 1) a quantitative performance comparison between text-UI and text-image models; and 2) the average text embedding distance between captions within the dataset.

For 1) a quantitative performance comparison between text-UI and text-image models, we randomly sample 100 pairs of different text and images and report top-1 and top-10 accuracies of the retrieval performance of a dual-encoder trained on text-UI pairs in screen2words in Table 6.1. We also used the same code to train a dual-encoder on the COCO dataset [121], a standard dataset for image captioning. We observe that the model trained on the COCO dataset has a significantly higher top-1 accuracy. This observation further verifies our hypothesis of the prevalence of non-unique text-UI matches specific to the screen2words dataset.

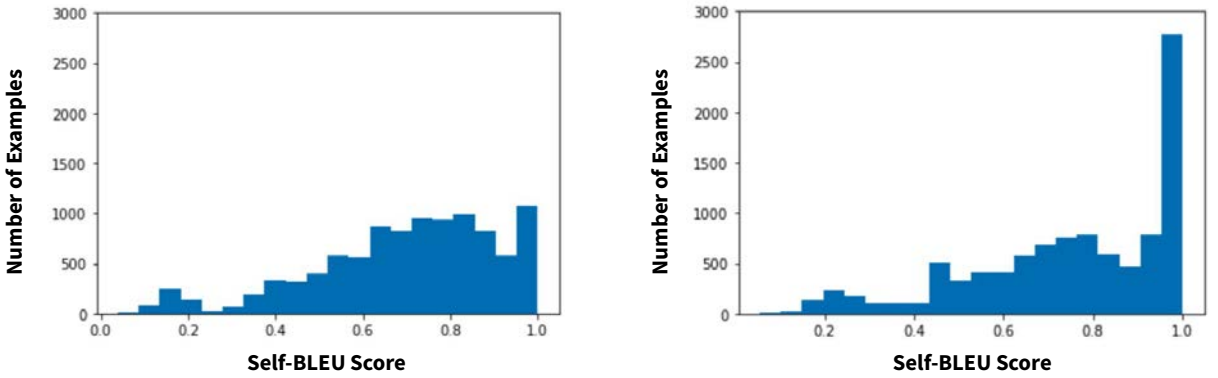
Table 6.1: Dual-encoder accuracies for models trained on COCO and screen2words datasets.

	Top-1	Top-10
screen2words (Multi-modal Retriever)	13.1%	50.7%
COCO	64.1%	97.9%

For 2) the average text embedding distance between captions within the dataset, we measure the self-similarity (using BLEU score [139]) between any particular text description in the dataset against all other text descriptions that describe other artifacts for both the COCO and the screen2words dataset. This metric is called self-BLEU and is commonly used in generative models' literature to measure generation diversity [196]. In addition, we also take the average BERT pooled embedding distance [39] of the top-10 nearest neighbors of each caption to verify the self-BLEU metric. For our computation, we took a subset of 10,000 captions from each dataset, and each of the captions corresponds to different UIs or natural images. Despite this attempt to minimize duplication, we found that screen2words captions are less specific than COCO, and such specificity is distributed more irregularly.

Figures 6.4a and 6.4b show distributions of self-BLEU metrics for captions of the COCO and screen2words dataset. We observe the average self-BLEU score for COCO of 0.699 to be lower than the average of screen2words of 0.755. In addition, we observe that a substantial number of captions (examples) in screen2words have self-BLEU scores closer to 1.0, and the distribution is more irregular over the sampled captions. This provided additional evidence for our hypothesis that screen2words captions are less specific than COCO.

We believe these observations are consistent with the data-collection instructions of the screen2words dataset, such that the annotators are instructed to provide short (less than 10 words) and high-level descriptions. We believe this is also more aligned to the language used in certain types of design applications where designers would only want to specify high-level text descriptions of the desired UIs, and that more specific specifications will be better described using direct manipulation tools where designers can add elements directly onto a draft UI canvas. However, as mentioned above, this results in a significant underestimation of the model's performance by R-precision, i.e., in many cases, even the paired ground-truth text would not be ranked as top-1 of a particular ground-truth UI.



(a) COCO (b) screen2words

Figure 6.4: Self-BLEU distributions for examples in COCO and screen2words datasets.

Hardness-normalized R-precision

To overcome the limitations described above, we introduce **Hardness-normalized R-Precision (HRP)**, a novel relevance metric that builds upon RP. This metric normalizes against the performance of the dual-encoder on particular UIs and allows us to adapt to the 'hardness' of the UIs[5]. We first define the distance computed by the dual-encoder between a pair of text and UI in the following equation:

$$\mathbf{DE}(t, u) = d(e_t, e_U), \quad e_t = f_t(t), \quad e_U = f_U(u) \tag{6.3}$$

with the definitions of the text encoder f_t and UI encoder f_U following Section 6.1.3, and d as a distance function. We then relax the top-1 requirement of RP to also 'accept' a generated UI if it is closer to the text prompt than that prompt's corresponding ground-truth UI pairing, according to the dual-encoder distance ($\mathbf{DE}(t, u)$):

$$\mathbf{HRP} = \frac{c}{\#\text{ of test set examples}} \tag{6.4}$$

where c is the number of examples with $\mathbf{DE}(t_p, \hat{u}) \leq \mathbf{DE}(t_p, u_p)$[6] **or** t_p is the nearest neighbor of $\hat{u}$.

[5]We consider the 'hardness' of a certain UI to be the distance between a text prompt and its corresponding ground-truth UI.

[6]This definition assumes d defined in Equation 6.3 is a distance function. The condition should be flipped if d is a similarity function, such as the dot-product (S) used in definitions in Sections 6.1.3 and 2.5.

6.2.3 Diversity

To effectively measure diversity, distributional distances between real data and synthesized data are typically used, such as Wasserstein Distance (WD). In our use case, however, the distributions of UIs need to be additionally conditioned to the text prompt used to generate them. Existing work compares the distribution of UIs in the entire test set $U_{\mathcal{T}}$ with a distribution of generated UIs $\{\hat{u}\} = \hat{U}$ using the Wasserstein distance $W(\hat{U}, U_{\mathcal{T}})$. We derive a conditional version of Wasserstein distance (**CWD**) where we consider the distribution of UIs in the test set to be weighted by the distance between their corresponding text description and the prompt text description that the generated UIs are conditioned on.

As an example, for a particular text prompt t_p, we sample a distribution of generated UIs from the model $\{\hat{u}|t_p\} = \hat{U}|t_p$, then we compare this distribution with the weighted distribution of ground-truth UIs: $U_w = \{(1 - D_{\text{text}}(t', t_p)u'\}, (t', u') \in \mathcal{T}$. We compute the final conditional wasserstein distance to be:

$$\textbf{CWD} = W(\hat{U}|t_p, U_w) \tag{6.5}$$

To make this computation tractable, we consider all coordinates and the class attributes as multiple 1D distributions. We also only sample $2n$ closest neighbors to a particular prompt to measure for $n = 10$ generated UIs from the same prompt. We use BERT pooled embedding distance normalized between $[0, 1]$ against the dataset as D_{text}.

6.3 Results

The analysis of our results consists of two parts. First, we separately analyze the UI Generator and the Multi-modal Retriever with different appropriate quantitative metrics. We then consider the final products of both models rendered by the common rendering pipeline for a qualitative analysis to compare the characteristics of both models. Such comparison also resonates with the user study we then conduct with experts on various text-to-UI methods in Section 6.4.

6.3.1 UI Generator Quantitative Results

We computed the aforementioned quantitative metrics in our benchmark (Section 6.2) on the test set for different variants of the UI Generator. These variants uses either an encoder-decoder (Enc.-Dec.) or decoder-only (Dec.-only) architecture, and the continuous coordinates are either discretized (**D**) or modeled by GMMs continuously (**C**) as mentioned in Section 6.1.2.

We first compare our model's performance on the well-formedness metrics against prior work. This serves as a check that our conditional model is still able to generate reasonable UIs compared to prior models. Table 6.2 shows that all of our variants are able to obtain comparable scores for both IoU and overlap metrics. Our model performs slightly worse

Table 6.2: Benchmark for Text-to-UI Generation (*Lower is better for all metrics, except HRP and RP*), **D** - **D**iscretized coordinates, **C** - **C**ontinuous coordinates (modeled with GMMs), 10 samples per caption in screen2words test set, * - real data, ** - published performance on Rico

	IoU	**Overlap**	**Alignment**	**HRP ↑**	**RP ↑**
Ours (Enc.-Dec., **D**)	0.101	0.133	0.508	29.2%	2.85%
Ours (Enc.-Dec., **C**)	0.102	0.0891	0.483	30.8%	2.92%
Ours (Dec.-only, **D**)	**0.0705**	0.0634	0.512	22.7%	1.04%
Ours (Dec.-only, **C**)	0.0980	**0.0573**	**0.445**	**30.9%**	**3.16%**
Ours (Dec.-only, **C**, $\tau = 0.1$)	**0.0699**	**0.0261**	**0.355**	**40.1%**	**5.69%**
LayoutTransformer**	0.086	0.145	0.366	-	-
VTN**	0.115	0.165	0.373	-	-
screen2words*	0.0307	0.0500	0.407	-	-
Rico*	0.0496	0.170	0.406	-	-

	$\mathbf{CWD_{bbox}}$	$\mathbf{CWD_{class}}$	$\mathbf{WD_{bbox}}$	$\mathbf{WD_{class}}$
Ours (Enc.-Dec., **D**)	0.0114	0.0135	0.0348	0.00849
Ours (Enc.-Dec., **C**)	0.0118	**0.0126**	0.0211	0.00597
Ours (Dec.-only, **D**)	0.0157	0.0150	0.0336	0.0115
Ours (Dec.-only, **C**)	**0.0111**	0.0129	**0.0186**	**0.00479**
Ours (Dec.-only, **C**, $\tau = 0.1$)	0.0207	0.0285	0.0433	0.0162
LayoutTransformer**	-	-	0.023	0.004
VTN**	-	-	0.018	0.007
screen2words*	-	-	-	-
Rico*	-	-	-	-

at the alignment metric, which might be due to alignment being harder to learn from the smaller screen2words dataset.

We also compare the performance between all variants of our models on benchmark metrics that reflect our target task of text-to-UI generation. In our experiments, we found that models that model continuous coordinates with GMMs typically outperform similar variants that use discretized coordinates. We believe this is also due to the small dataset size of screen2words, such that modeling continuous coordinates with GMMs offers useful

inductive biases for models to bootstrap learning with. We also observe that decoder-only models tend to perform better than encoder-decoder models. We believe this might be due to the flexibility for decoder-only models to allocate different numbers of weights to model either text-only, UI-only, or joint distributions in the data. As an additional experiment, we lowered the sampling temperature of our best-performing model, and the generated UIs from the low-temperature model have improved quality and relevance but reduced diversity. This trade-off matched our intuition about the effects of the temperature parameter on the model.

Moreover, the proposed benchmark offers a deeper view of the generative models' relevance and diversity performance compared to prior metrics (also reported in Table 6.2). For relevance, HRP varies less than RP as expected, since requiring a similar text-UI pair to be ranked top-1 would introduce significant variance for spaces where many-to-many pairings are plausible (i.e., many 'login screens'). We further observe that while RP reflects our decoder-only model with discretized coordinates to be significantly inferior in relevance to other models, HRP reflects that additional investigation into individual cases might be necessary since the differences were not as large. Similarly, we observe CWD does not exactly follow the trends of WD. This could mean models follow specific conditional distributions to a different degree than the overall prior distribution of UIs. Finally, comparing WD of our proposed methods with prior work, we attain similar performance in covering the entire dataset distribution with the distribution of generated UIs. We hope all of these quantitative results can serve as a benchmark and baseline for future work in this area.

In-domain and Out-of-domain Analysis

Beyond holistically considering our results for the dataset as a whole, we further consider descriptions from the *'in-domain'* test set that are most represented by the training set, measured by average top-10 neighboring distances in BERT embedding-space. Similarly, the captions that have the highest BERT embedding average distances are considered 'out-of-domain'. The 'in-domain' examples have an average BERT embedding distance of 0.161, while the 'out-of-domain' examples have an average distance of 2.04. We sample 10% of each set of examples and compute their well-formedness and relevance metrics. We found that the 'out-of-domain' examples only perform slightly worse in most metrics of the benchmarks. In our Decoder-only model with $\tau = 0.1$, the 'in-domain' examples obtained 0.0621/0.0224/0.333 (for IoU/Overlap/Alignment) over 'out-of-domain' examples' 0.0710/0.0268/0.362 for well-formedness metrics. For relevance, 'in-domain' and 'out-of-domain' examples respectively obtained 6.02/42.0 and 5.24/40.6 (RP/HRP), which increases our confidence in the model's generalization ability.

6.3.2 Multi-modal Retriever Quantitative Results

To evaluate the performance of our Multi-modal Retriever quantitatively, we used a standard information-retrieval metric commonly used to evaluate retrieval models in general:

Table 6.3: Cross-modality retrieval accuracy results.

	Top-1	Top-10
Multi-modal Retriever (5 subsets avg.)	23.2%	65.0%
Swire	15.9%	60.9%
Multi-modal Retriever (entire test set)	2.80%	4.84%

Top-k Accuracy. As our dataset consists of ground-truth pairs of query text and UIs, this accuracy represents the number of pairs where the paired UIs are considered to be within the k-nearest neighbors of the text descriptions, measured with the embeddings created by our Multi-modal Retriever.

Following this metric, we compute the embeddings of text descriptions and UIs in the test set and show retrieval accuracy in Table 6.3. While the overall performance is relatively low, this is because the number of text and UI pairs in the entire dataset is quite high (4310 UIs $\times$ 5 descriptions each) and similar descriptions might be paired with different UIs as investigated in Section 6.2.2, increasing the difficulty of this evaluation scenario. However, when comparing our method with an established cross-modality sketch-to-UI retrieval method—Swire [78], we obtain better performance than Swire from the same number of candidates—over the average of five random subsets of the same size as Swire's test set (276 candidates).

6.3.3 Qualitative and Comparative Analysis

After reporting quantitative metrics for each set of methods as a concrete reference, we qualitatively and comparatively investigated the characteristics of results of our proposed methods. Overall, we observe that most UI mock-ups created by our methods are well-formed and comprehensible, and are within the general categories of UIs that we would expect based on the text descriptions. In this section, we analyze the results acquired by each method for two text descriptions from the test set[7]: *"screen displaying list of topics under pocket physics"* (see Figure 6.5) and *"pop up displaying an image and other options"* (see Figure 6.6).

UI Generator Qualitative Results

For the results synthesized from scratch by the UI Generator, we observe that this method is good at making small but coherent variations within the same theme for both cases in Figures 6.5 and 6.6. Figure 6.5 shows minor changes the UI Generator made to vary

[7]These captions are also within the set that was used in Section 6.4, which were chosen with mock-up results unseen to us prior to selection.

list structures. Similarly, we observe popups with varying layouts that follow the same theme in Figure 6.6. Nevertheless, such subtle variations can lead to the lack of large thematic variations and lower diversity of UIs in these sets of results. However, this issue can be addressed by hyper-parameter tuning: the UI Generator contains a tunable sampling temperature hyper-parameter (set to a fixed value of 0.1 for the presented results) that can be explored in the future to vary the balance between diversity and well-formedness of the UIs. It is worth noting that while it is easy for retrieval-based methods to return well-formed UIs, it is non-trivial for a generative method to return sensible UIs because such a method needs to synthesize complete UIs from scratch instead of using an existing UI.

Multi-modal Retriever Qualitative Results

The Multi-modal Retriever, on the other hand, excels at matching specific details described in text. For instance, only this method was able to retrieve pop-ups with emphasized large images specified in *"pop up displaying an image..."*. We believe this is because the Multi-modal Retriever has access to text-based features of individual elements and text descriptions in the UIs. Nevertheless, this method also has access to content-specific text details that might be confusing for retrieval given some text descriptions. For example, the model retrieved a UI with a large block of text that seemingly conflicts with *"a list of topics"* at Figure 6.5(c). However, this UI originated from an app presenting business topics and ideas which contain text content related to "lists". This provides an example where matching text content on the screen and within the app might not necessarily lead to matching design concepts.

6.4 Expert Feedback

To further solidify our assessments of the characteristics of results gathered by each of our proposed methods in Section 6.3.3, we conducted a user study to gauge expert UI designers and practitioners' preferences on mock-ups created by our methods.

6.4.1 Procedure

To solicit our participants' preferences for the mock-ups our methods have created, we manually chose five text descriptions from the test dataset that cover diverse design scenarios (two of the descriptions were included in the qualitative results in Section 6.3.3). We then used each of our proposed methods to create 5 UI mock-ups for each of these descriptions. We *did not inspect* the mock-ups created by these methods prior to selecting these descriptions. In addition to the two methods introduced in this chapter, we included a baseline *Text-only* Retriever, which uses the pre-trained BERT embeddings of the paired text descriptions of the UIs in our dataset to perform retrieval in only the text-BERT embedding space.

These baseline and proposed methods allow us to form 3 sets of 5 created UI mock-ups for each of the 5 text descriptions. In the study, we frame each text description as a design

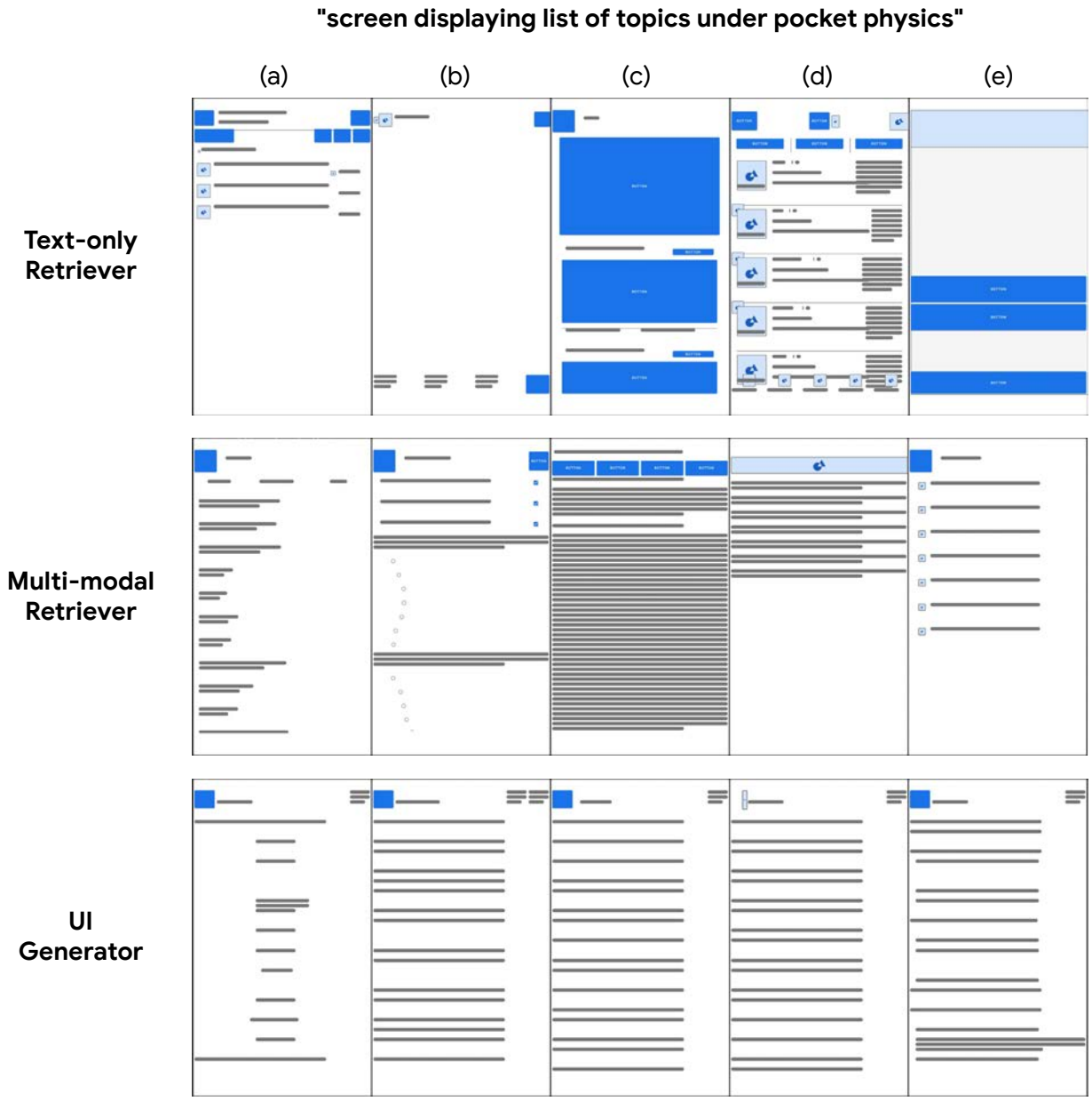


Figure 6.5: UI mock-ups created by the baseline Text-only Retriever and our methods in response to the text description *"screen displaying list of topics under pocket physics"*.

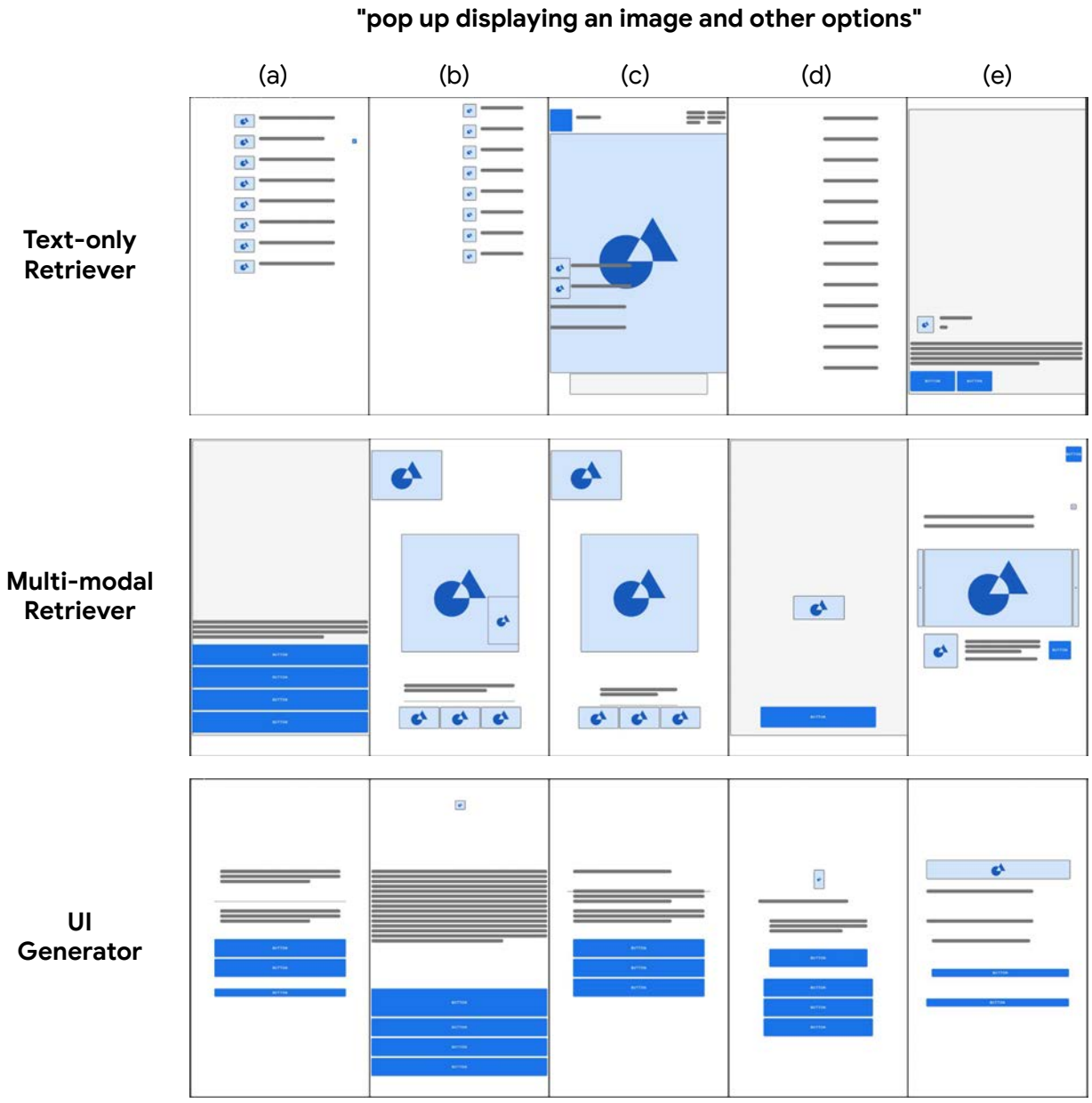


Figure 6.6: UI mock-ups created by the baseline Text-only Retriever and our methods in response to the text description *"pop up displaying an image and other options"*.

goal. The participants were first presented with a ground-truth high-fidelity screenshot corresponding to the design goal in text. They were then presented with a scrambled 3x5 grid of all 15 UI designs generated across 3 methods. The positions of these design mock-ups are randomized. Based on these UIs, they would then answer the following two questions by choosing a specific mock-up for each and explaining their rationales.

- Q1. Which mock-up in the collection best resembles the hi-fidelity design?
- Q2. Select the mock-up in the collection that presents the best design alternative for the given design goal.

At the end of the survey, we also obtained the participants' opinions on the following questions on AI-assisted design in general, followed by them providing open-ended comments or suggestions for Words2ui.

- Qa. If an AI system is built to suggest UIs from text description inputs just as the ones presented in this study, how might it help / not help with your design workflow?
- Qb. Following the above question, if such an AI system that suggests UIs from text is developed, what do you think are the most important features this system should have?

6.4.2 Participants

We recruited 15 professional User Interface and Experience (UI/UX) practitioners to complete this survey from a mailing-list. The participants self-reported an average of 9.8 years of industrial UI/UX design experience. Their professions include UI/UX Designer, and UX Engineer, Interaction Designer, UX Researcher, and are located in the United States and the United Kingdom. All participants completed the survey at their own pace remotely, and were each compensated with a $20 USD gift card upon completion of the survey. Each participant took roughly 20 minutes to complete the survey.

6.4.3 Results and Discussion

We split our analysis of the study results by the five design goals (i.e., text description conditions). Q1s of each design goal serve as warm-up questions for participants to familiarize themselves with the mock-ups' representation. These questions also gauge the created UIs' relevance and diversity as perceived by designers, because a relevant and diverse set of UI suggestions has a higher chance of covering the high-fidelity UI associated with the design goal. We found that for a given design goal (description), each of our methods is more frequently chosen by participants than other methods (Chi-square goodness-of-fit test, $p < 0.05$ for each case, with equal expected frequency across methods) for exactly one design

goal, while results are insignificant for the other two design goals. This result shows that each of the methods is competitive for specific scenarios.

Q2s in our study are more important for evaluating the overall quality and relevance of UI suggestions. Similar to the first question, each method was more frequently chosen by participants ($p < 0.05$) than the other methods in one of the three descriptions, while the other two descriptions have insignificant results. The particular designs chosen by participants align with our understanding of the characteristics of each of the methods presented in Section 6.3.3. For instance, our participants considered the Multi-modal Retriever to have provided the best design alternatives (Figure 6.6) for *"pop up displaying an image and other options"* (the middle chart in Figure 6.7). They prefer these mock-ups due to their high relevance to the text description for including an image, as commented by a participant (P8): *"[this design] allows users to focus on the main image"*.

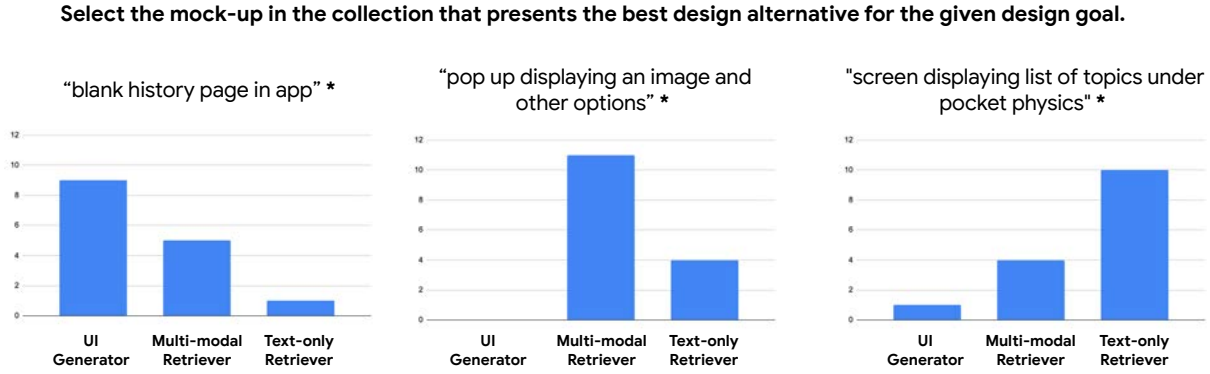


Figure 6.7: The study results for best alternative designs for three of the five design goals (Q2.). We observe that each of our methods is preferred in one of the design goals. The design goals are shown at the top of each chart.

In another example, the results based on the description *"blank history page in app"* (see Figure 6.7) indicate that most participants preferred the mock-ups created by the UI Generator, citing that it is nice to have an empty state illustration instead of nothing (P4). The mock-up generated by the UI Generator still retains the general structure of an empty UI but with an extra-large illustration. This result showed the benefit of the generative method which allows variations under the same theme given a text description.

While we have evaluated a diverse set of UI suggestions and design goals, we acknowledge that our evaluation is limited in that we only investigated a small set of design cases that can potentially benefit from our methods. We suggest future work to more comprehensively understand general trends that exist across large-scale datasets for these methods. The open-ended questions regarding AI-assisted design systems at the end of the survey also provided us with a number of actionable suggestions for developing our methods into AI-driven systems. We discuss participants' responses to these questions in the following Section 6.5 for potential applications.

6.5 Envisioned Applications

Our ultimate goal of developing these methods is to provide intelligent design support for UI designers and developers with relevant design artifacts given high-level text descriptions. We discuss several potential applications that can utilize our methods based on open-ended comments provided by participating UI/UX practitioners in the user study.

6.5.1 Early-Stage Design Sketch Rendering

A major theme emerged from the participants' feedback for Qa. is that the UI suggestions presented to participants would be good for initial stages of design, such that they can be good starting points for higher-fidelity designs in their design processes. P2 mentioned that these designs can be *"providing some quick, initial mocks that I could leverage for higher fidelity design solutions early on in my design process."* P9 commented that *"this can in turn help in brainstorming and generating a greater variety of designs."* As such, one potential application is to further develop systems catered specifically for early-stage design. For instance, we can adjust our rendering engine to display sketch-like mock-ups which could encourage more high-level early feedback. Sketch-based rendering would allow designers to discuss these generated designs as early-stage artifacts and interact with them directly with pen strokes. As mock-ups created by our methods contain attributes of each UI element, this application can be realized by replacing the set of rendering elements with sketched elements introduced in the UISketch dataset [170].

6.5.2 Interactive and Steerable UI Generation

On the other hand, one of the most requested features by the participants, as reflected by their responses to Qb., is to provide designers with more control over these AI-assisted methods. P7 mentioned that it *"should be able to generate other layouts and substitutions if I pin certain controls or elements."* Similarly, P8 requested these methods to *"allow people to quickly modify the AI-generated design and maybe even control the design directions."* These features can be achieved by further developing the UI Generator as it generates one new UI element at a time based on all previously generated elements. We can simply feed elements that are already designed by the designer, instead of only using the model's previously predicted elements, as inputs to the autoregressive decoder. This enables designers to control the design process with their own designs. Moreover, as the UI Generator outputs UI elements using a sampling-based approach, it can generate multiple candidate elements each time and allow designers to select their preferred elements. Designers can also adjust the candidate diversity of the model by changing the temperature parameter that controls the stochasticity of the sampling process as discussed in Section 6.1.2. These features could provide designers with rich control over the mock-up creation process as requested in the user study.

6.5.3 Combining Multiple UI Suggestion Methods

As we discovered in the previous sections, our two proposed methods are suitable for obtaining designs of varying levels and types of diversity, and hence are suited for different stages of the design process. We envision a final system that combines all of these methods to support various stages of the design process. A designer might start with the Text-only Retriever for high-level design inspirations from the mock-ups and optionally concrete UI screenshots examples that it can retrieve. With the gradual convergence of design details, the system can use the Multi-modal Retriever to retrieve more closely related and specific designs. Finally, the designer can use the UI Generator to innovate on design artifacts based on existing designs they prefer.

6.6 Limitations

While we have shown that our models are competitive in generating and retrieving well-formed, relevant, and diverse UIs based on our quantitative metrics, and have generated mock-ups considered to be potentially helpful by experts, we discuss several limitations of our proposed methods in this section.

6.6.1 Supporting Design-Specific Language

The primary limitation of our work is related to partial misalignment between the descriptions in the dataset that we use and the type of descriptions we aim to support. The screen2words dataset is collected by asking crowd-workers to summarize UI screenshots in natural language. The type of language that crowd-workers use to describe UI screenshots might not be perfectly aligned with how designers would articulate design goals. For example, a text description in this dataset might include content-based details which could be less important to the design goal, or lack information about design constraints that designers might use to describe desired UIs. Nevertheless, our work made a valuable contribution by investigating two novel methods applicable to other paired text-UI datasets that might become available in the future. Investigating the degree of such misalignment and the effectiveness of our methods on design-specific language remain future work.

6.6.2 Addressing Rare and Intermediate UI Elements

Another limitation of our methods originates from the use of DL-driven approaches, such that it is more infrequent for them to generate or retrieve types of UI elements that are rare in the dataset. Only text, image, and button elements are common in the datasets the models were trained on, and as a result, generated/retrieved UIs more frequently contain these types of elements. Other types of elements, such as checkboxes, and sliders, are more rarely seen in the generated UIs. This issue is particularly apparent in the UI Generator method.

Future work could explore data augmentation methods to improve such class imbalance in the dataset.

Moreover, we only considered a relatively flat structure of the UI elements (which originates from the processing of the [122] dataset) that are often leaf nodes with the UI Generator. Some intermediate nodes in the UI hierarchy that group these elements together might also be significant for design processes. We believe effectively extracting these useful elements in addition to the semantically labeled elements is an important direction for future work.

Chapter 7

Discussion and Future Research Opportunities

Throughout this dissertation, we described several methods and systems for generating sketches and prototypes. In this chapter, I outline several directions of future research that I believe are important and impactful towards improved capability of both understanding human guidance and generating artifacts in creative activities.

7.1 Machine Guidance and Tutorial for Sketching and Prototyping

This dissertation explored multiple human-guided interactions of generating sketches and prototypes, such as text-to-sketch and text-to-user-interface (text-to-UI) generation. I believe it would also be interesting to explore the *inverse* interaction, where machines provide guidance to users in the form of tutorials and suggestions, acting as tutors and/or critics to guide users to create better output sketches and prototypes in their creative processes. While direct generation of sketches and prototypes, like interactions supported by our proposed systems, could reduce users' workload, suggestive systems that indirectly steer users (through another modality such as natural language) might help them become better at creating relevant artifacts and/or explore alternative ideas previously unconsidered by themselves.

Pursuing this area of research, however, shall include a thorough investigation of the human-AI collaboration dynamics during sketching and prototyping beyond making technical advancements on models that perform the inverse tasks of generating text from user-provided artifacts. This investigation might include carefully studying the types of advice that the targeted user group might desire, and the modalities and tones to deliver them in, for the final machine-generated guidance to be respectful, productive, and effective. Following this thorough characterization study of guidance and the thoughtful design of such AI-guided interaction, data collection and model development efforts should follow users' demands and opinions in the study, and researchers shall continuously work with users to

support their desired interactions. I believe there are ample research opportunities in all of the aspects mentioned above surrounding the automatic generation of guidance for humans' creative processes.

7.2 From Cross-modality to Multi-modality

As mentioned in previous sections, our current models primarily support *cross-modality* interactions, transforming artifacts in one modality (e.g., natural language descriptions) to those in another modality (e.g., UI prototypes). While Scones (Chapter 5) has touched upon some concepts of being *multi-modal* to flexibly understand multiple modalities (e.g., scene objects and text descriptions), I believe there is further research work required to achieve true *multi-modality*. The investigation of a simplistic system that handles the inverse interaction detailed in the Section 7.1 constitutes merely one aspect of the investigation on multi-modality.

Extending on the example interaction of text-to-sketch generation supported by Sketchforme and Scones, a truly multi-modal system needs to be able to additionally generate text responses that tie to its own or users' sketch-generation process (since users might sketch parts of the scene). These processes might also be ill-formed in turn-taking, such that a system might need to generate a text response given a text instruction by the user (a text-to-text interaction, e.g., asking clarifying questions), and the user might choose to demonstrate as a response to the systems' questions (a sketch-to-sketch interaction, e.g., showing the system how to draw an icon). These interactions would result in a highly heterogeneous and interactive environment, where sketch-to-sketch, text-to-sketch, sketch-to-text, and text-to-text interactions need to be all modeled by the future multi-modal system effectively. This system also needs to select the appropriate interactions to engage users with during the co-creation processes. There is hence a need to model the overall creative process itself beyond just any specific individual interactions listed above.

Achieving the capabilities of a multi-modal system mentioned above requires data, modeling, and interaction support. Clean and balanced datasets of all of the above interactions would be helpful for the development of new models. It is also preferable for models to be compact, interpretable, and effective in understanding and generating data in modalities involved in the interaction. Finally, clearly communicating the systems' limitations and capabilities to the users, and allowing for fall-back interaction modes (e.g., exposing the isolated interaction capability of the system in any of the directions) can help users establish reasonable expectations and prevent frustration when interacting with the system.

7.3 Large Multi-modal Models

One approach towards significantly advancing multi-modal systems' data support and modeling capability required in Section 7.2 above is leveraging recent large language models

that have demonstrated impressive general knowledge in the domain of natural language. These models are shown to be highly capable of modeling text in many topics and could consequently reduce the amount of data needed to support individual domain-specific applications through fine-tuning or few-shot learning (see Section 2.3.6 for a detailed description). To understand the applicability of such knowledge towards reasoning in modalities other than natural language, various experiments have been conducted to understand large language models' knowledge of the visual domain highly relevant to sketching and prototyping. Researchers have found that these models possess a certain degree of knowledge of the visual world, and such visual knowledge can be interacted with if visual data can be first translated (typically via another domain-specific image captioning model) to the natural language domain. Some examples that have gained significant attention include zero-shot VQA by coupling large language models with image captioning models, and UI-generation-from-text experiments via language-based structural representation of UIs in the JSON format.

To gauge these models' proficiency in knowledge in creative domains, we had attempted some informal prompt-engineering experiments with GPT-3 directly on geometric and UI data by asking it to generate and manipulate natural scene layouts and UI layouts in the form of numerical attributes. We have attempted multiple formats (JSON, XML, Plain text) as input representations and few-shot prompts to the model. We observe that while GPT-3 was able to follow the format of our prompts in its responses, these responses do not contain any meaningful natural scenes or UI geometries at all. GPT-3 was unable to generate objects with even simple constraints on the horizontal direction order (from the prompt 'an [X] on the left of [Y]'), or on counts of UI elements (from the prompt 'there are [X] of [Y] type of UI elements in a UI that does [Z]'). Nevertheless, we observe that it is *occasionally* able to generate reasonable semantic *types* of UI elements given a text description of the entire UI (i.e., a description of a map UI as input to the model gives the MapView UI element class as part of the output).

I believe this presents great opportunities for future research to build upon large language models' extraordinary ability in maintaining fluent natural language interactions, to support creative processes. Enabling creative support applications by these models would require in-depth understanding and extraction of creative-content knowledge embedded in these models. Some other research opportunities also include developing novel generative, fine-tuning, and few-shot prompting approaches to leverage such multi-modal knowledge from large language models.

Furthermore, another potential research area is to train large models with multi-modal data and tasks from scratch. There is an increasing amount of research literature on large 'language' models developed with an inherent multi-modal focus, such as DALL-E [151], Flamingo [4], and MUM [132]. These large models can help overcome both data and modeling challenges listed above, by having models that are designed and trained to perform multi-modal tasks without requiring explicit translation. I believe this area of research could benefit from the interactions contributed by this dissertation, along with the accompanying tasks, metrics, and observations. I also believe these large multi-modal models are significant steps towards the final goal of developing multi-task, multi-modal systems that are capable

of supporting sketching and prototyping interactions in diverse creative domains.

7.4 Novel Architecture and Task Design

This dissertation presented multiple novel learning tasks to support sketching and prototyping applications. I believe there are numerous architectural and task improvements that shall be explored to further improve the models' performance in these target applications. This is because as we handle users' guidance in a different modality as the generated artifacts, these modalities are unlikely to require the same amount of representational power. Hence, models that use different numbers of parameters or even types of architectures for each of the guidance and generation modalities should be considered for these cross-modal tasks (i.e., the most effective text-to-sketch models might not have symmetric text and sketch encoders). Moreover, such asymmetric architectural design could be an important method for biasing the models with our knowledge of the target task that it is intended to support, improving both model training and final performance.

One special case of this exploration was the system architecture of Sketchforme (Chapter 4) and Scones (Chapter 5), where we take a two-stage approach to separately train a model to propose composition layouts of various objects, and another model to generate individual sketched objects based on applicable conditions that fit the compositions. Each of these models has a different architecture optimized for each of their distinct learning tasks. These architectures were designed to vaguely model the planning process in sketching and to overcome the lack of end-to-end data from text descriptions/critique to sketch strokes. I believe such exploration should be more extensive in all of the proposed architectures. For our example models in Words2ui (Chapter 6), at a small scale, this could be exploring different numbers of layers (blocks) and numbers of hidden units in each layer for approaches that encode text and UIs/sketches separately with sub-networks; there could also be a broader exploration by customizing attention layers and connections, or building special-purpose modules to consume domain-specific data, such as graph-convolutional-network for UI constraints or hierarchical architectures for UI element trees. Further, I believe one most recent promising direction of research is to augment GAN-based and/or Diffusion-based generative models to handle structural information that often exists in prototypes in their generation and up-sampling processes, to generate prototypes more effectively.

Other than modeling improvements, I believe performing multi-task learning on intermediate representations could also improve generative models' performance even when end-to-end data has become increasingly available. For instance, for a text-to-multi-object-sketch task, I speculate that training the models to additionally predict the intermediate object representation can not only help them better understand structure in data to improve performance, but would also allow users to interact with higher-level composition in the generation process, improving the interpretability and usability of these models in creativity support applications.

7.5 New Domains and Modalities

In this dissertation, we showed that computational systems can support sketching and prototyping for non-experts and professional UI designers. I believe many of our findings can aid in the development of future applications in other creative domains, such as industrial design, architectural design, mechanical engineering, presentation layout design, and storyboarding for visual media. As sketches and prototypes are extensively used in these domains as media for creative problem-solving, I believe the unique challenges in these new domains create great opportunities for innovation in sketch-based design and engineering applications.

I also think there are significant research opportunities in other modalities complementary to sketching and prototyping. For instance, finer details in the generated artifacts, such as colors and animations, are useful for sketching and prototyping applications. I envision Scones and Sketchforme might be extended in the future to consider the colors of sketches, which can be derived from the compositional data in natural images and objects within them. The animations of sketches could also be inferred from dense video segmentation datasets [143] which provide data in the form of animated masks for the sketches. Animations in UIs, which can be obtained from the RICO dataset, are another important modality that relates to an important step in the design workflow and can be explored in the future. One important challenge in these research areas is to maintain coherence between various artifacts generated in the same task and/or interaction, such as maintaining consistent color and animation style across various frames of the same generated drawing.

While it might be currently time-consuming to pursue all of these new domains and modalities from scratch with Deep Learning (DL) such as the proposed methods in this dissertation, I speculate the exploration of these new models and applications would become easier and faster given the improvement of software and hardware infrastructure for ML. I believe such exploration in the future would become a quick and iterative design process on its own, converging to more novel and effective DL-driven interactions. I also believe the use of current Large Language Models (described in Section 2.3.6) and future Large Multi-modal Models (proposed above in Section 7.3) could help bootstrap applications in new domains with their extensive breadth of general knowledge, further shortening the development time of new DL-driven systems. Researchers have already proposed to use Large Language Models as prototyping tools for ML-driven interactive systems [87], and I expect this approach to become more prolific for exploring new application domains given its lower requirement for in-domain data.

7.6 Integration with Application in Real Usage Scenarios

While we have made technical contributions in this dissertation towards the goal of supporting sketching and prototyping processes, I believe a significant area of future research is

to deploy these systems in real-usage scenarios to understand their advantages and limitations in supporting real sketching and prototyping processes. These insights would provide important future directions for further technical improvements to sketching and prototyping models, and help reconcile potential discrepancies between users' mental models and the computational models' reasoning and generation processes. Moreover, given how our systems have created entirely new affordances (e.g., conversational authoring in Scones), researching real-world interactions between the user and these deployed systems allows us to better understand users' preferences, such as choices of modalities across different stages of design and human-AI collaboration dynamics (further discussed in the following Section 7.7). These findings can potentially provide valuable guidance (such as those needed in Section 7.1) towards the design of future creativity support systems with the assistance of the latest machine-learning (ML) techniques.

I also believe real-world deployments could be used to improve current systems' performance by providing in-domain, in-context usage data. For instance, the Words2ui models (Chapter 6) were only trained on UI captioning data. Deploying these models to real design processes could help us simultaneously collect real natural language descriptions that designers use in such a workflow (hence overcoming our limitations mentioned in Section 6.6.1), and consequently improve our proposed models by re-training and/or active learning. These systems already have a certain degree of intelligence, and our users have expressed interest in exploring their utility in their design processes. I believe this provides great opportunities for these models to continually benefit both the proposed systems' users, given the incentives of improving users' workflow that they provide and their further improved future iterations with new user data.

7.7 Dynamics and Research of Future Designer-AI Interaction

As all research systems proposed by this dissertation are heavily motivated by design applications, the discussion of future research opportunities would not be complete without our speculation about the dynamics of Designer-AI interaction in future design processes.

I speculate that the future development of generative systems for design sketches and prototypes, such as Words2ui introduced in this dissertation (Chapter 6), would significantly change the interaction dynamics between designers and these design-support computational systems, especially when these systems become increasingly capable and pervasive in their future iterations. Sketching and drawing had previously been considered as an important skill and ability that is unique to expert designers (further discussed in Section 2.1.3). However, as our proposed systems are novelly capable of automatically generating sketches, I believe designers might offload *some* of the more mundane drawing and/or prototyping tasks to these generative systems. Delegating these tasks would leave designers with extra time and mental capacity to consider the wider context, broader problem definition, and explore newer

directions towards solving the 'wicked' design problem[1] [18, 158], resulting in more comprehensive design considerations and better designs. I also speculate that in the near future, this capability of designing solutions that the clients 'never dreamed [they] wanted' [29] might be difficult for computational systems to achieve, and hence such ability to explore new problem definitions and innovative solutions would remain unique and become a more important role for designers to take in their collaboration with these new generative systems. Therefore, as proposed in previous sections in this chapter, I believe a promising future research direction is to validate whether such a shift in interaction dynamics would exist with the introduction of these generative systems into realistic use-cases and workflows, or if there is a certain requirement of these systems' generative ability for such shift to happen.

Moreover, the usage of these generated artifacts of sketches and prototypes in realistic design processes shall be further explored, such that whether they will be integrated as-is into design documentations as artifacts to replace all manually-created artifacts, or have multiple machine-generated artifacts combined and further modified by designers, or only taken as reference for designers to manually create their final artifacts with. Given sketches and prototypes are important not only as an artifact itself, but also as a thinking tool for problem comprehension and solving in design [5, 29, 55, 177], it would be important to investigate the renewed purpose and intensity of sketching and prototyping activities under these new AI-assisted sketching workflows. Associated with this investigation would be researching the nature of guidance and control that designers hope to have on the computational generation processes of sketches and prototypes. This is also a requirement for future systems suggested by experts who participated in our user studies in Words2ui (Section 6.4). Further, this research might need to additionally consider novice and intermediate users that are now empowered by generative systems to create sketches and prototypes, because I believe these generative systems will promote wider participation in design and introduce a new user group that we shall consider for.

Finally, based on the results of all proposed future investigations above, other types of design-support interactions should be researched beyond those introduced in this dissertation. While in this dissertation we chose to first investigate a conversational process (i.e., elementary sketch/critique cycles introduced by Scones in Chapter 5) because sketching is often characterized as 'dialogues' and 'conversation' for designers [29, 169], I believe more practical and useful interactions would arise in future systems. I believe these systems will contribute diverse and comprehensive design alternatives and ideas, moving beyond their current assistive and tool-based roles to designers, and participate in the often *social* processes of design [17, 29, 175].

[1] Wicked problems are complex problems that are often ill-formed and do not contain singular, correct, and optimal solutions. It has been used extensively to characterize design and planning problems.

Chapter 8

Conclusion

In this dissertation, we explored several methods and systems to **generate non-expert sketches and user interface (UI) prototypes with human guidance.** Such guidance takes the forms of natural language and existing sketches, enabling non-experts and professionals to effortlessly and promptly generate relevant artifacts in their creative activities.

We first developed Sketchforme, which is **the first deep-learning (DL) system to be able to generate complex sketched scenes consisting of multiple objects given a single text description.** The core technical architecture of Sketchforme is a two-stage process, in which we successfully adapted Recurrent-Neural-Networks and Transformers to model both scene compositions and sketching mechanics of individual objects. These models are then connected with each other via relevant conditions based on aspect ratios of the objects. This design avoided the need to train our system with end-to-end text-to-sketch datasets that were previously unavailable. The generated sketches were considered to be realistic and relevant by human raters, and were shown to have successfully supported a language learning application and a sketching assistant.

We then improved upon Sketchforme to develop Scones, which presents a new interaction paradigm of *conversational authoring* of sketches. Scones **continuously iterate on sketched scenes by adding and editing sketched objects in the scenes over multiple turns where users would provide multiple text instructions.** It enables sketch/critique cycles, a more powerful paradigm to customize the output sketches progressively according to users' guidance than only generating individual scenes with a single text description in Sketchforme. Scones also improved upon Sketchforme's technical architecture and adapted Transformer models to achieve new state-of-the-art quantitative performance in an established scene composition task with its first stage. The second stage of Scones now additionally considers object masks that better describe the poses of the object sketches. In our exploratory user study, users enjoyed interacting with Scones and were also generally satisfied with the final sketches that they were instructed to co-create with it.

Beyond non-expert sketches, we developed Words2ui, which consists of **a set of generative and retrieval models for low-fidelity UI prototypes based on high-level text descriptions.** This is a novel task for UI generation models, in which we adapted state-of-

the-art Transformer architectures and developed novel metrics to fully establish the research landscape in this area. We found that these adapted Transformer models were able to generate well-formed UIs of similar quality as those generated by unconditional models, and are qualitatively relevant to the text descriptions that they are based on. The retrieval models also enable quantitative evaluation of the generative models, and by themselves support the important application of exploring inspirational design examples in UI design processes. Experts who reviewed the generated prototypes considered them to be potentially helpful in their current design processes, and provided suggestions for further adoption of Words2ui.

With these three projects, we have demonstrated how DL models can help generate sketches and prototypes (primarily using the Transformer architecture) that users might not be proficient or familiar with, guided by natural language that users can more easily specify. These systems operate at various abstraction levels, including high-level composition and low-level sketching mechanics, to overcome data challenges and to enable practical interactions such as object-level manipulation. The models introduced along with these systems also establish new state-of-the-art performance in existing tasks and enable new affordances and interactions that users enjoy and could benefit from in their creative processes.

We believe all of the proposed systems can encourage wider participation, enable greater self-expression, and reduce required effort in creative activities. With the increasing availability of large-scale datasets and prevalence of more capable language and text-to-visual generative models, we are optimistic about future datasets, models, and interactive applications that can be developed to further advance the state-of-the-art of creativity support tools. We are excited about a future where desirable, realistic, and compelling artifacts can be generated by machines in various creative activities for users of diverse backgrounds and skill levels.

Bibliography

[1] Luis von Ahn and Laura Dabbish. "Designing Games with a Purpose". In: *Commun. ACM* 51.8 (Aug. 2008), pp. 58–67. ISSN: 0001-0782. DOI: 10.1145/1378704.1378719. URL: http://doi.acm.org/10.1145/1378704.1378719.

[2] Shaaron Ainsworth, Vaughan Prain, and Russell Tytler. "Drawing to Learn in Science". In: *Science* 333.6046 (2011), pp. 1096–1097. DOI: 10.1126/science.1204153. eprint: https://www.science.org/doi/pdf/10.1126/science.1204153. URL: https://www.science.org/doi/abs/10.1126/science.1204153.

[3] Emre Aksan et al. "CoSE: Compositional Stroke Embeddings". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 10041–10052. URL: https://proceedings.neurips.cc/paper/2020/file/723e8f97fde15f7a8d5ff8d558ea3f16-Paper.pdf.

[4] Jean-Baptiste Alayrac et al. "Flamingo: a Visual Language Model for Few-Shot Learning". In: *CoRR* abs/2204.14198 (2022). DOI: 10.48550/arXiv.2204.14198. arXiv: 2204.14198. URL: https://doi.org/10.48550/arXiv.2204.14198.

[5] *An Introduction to Design Thinking Process*. Hasso Plattner Institute of Design at Stanford. URL: https://web.stanford.edu/~mshanks/MichaelShanks/files/509554.pdf.

[6] Rahul Arora et al. "SketchSoup: Exploratory Ideation Using Design Sketches". In: *Computer Graphics Forum* (2017). URL: http://www-sop.inria.fr/reves/Basilic/2017/ADNBS17.

[7] Diego Martín Arroyo, Janis Postels, and Federico Tombari. "Variational Transformer Networks for Layout Generation". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 13642–13652. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Arroyo_Variational_Transformer_Networks_for_Layout_Generation_CVPR_2021_paper.html.

[8] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *CoRR* abs/1607.06450 (2016). arXiv: 1607.06450. URL: http://arxiv.org/abs/1607.06450.

[9] Benjamin Bahr. *Prototyping of User Interfaces for Mobile Applications*. 1st. Springer Publishing Company, Incorporated, 2018. ISBN: 3319850911.

[10] Chongyang Bai et al. "UIBert: Learning Generic Multimodal Representations for UI Understanding". In: *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montréal, Canada, 19-27 August 2021*. Ed. by Zhi-Hua Zhou. ijcai.org, 2021, pp. 1705–1712. DOI: 10.24963/ijcai.2021/235. URL: https://doi.org/10.24963/ijcai.2021/235.

[11] Mira Balaban, Eli Barzilay, and Michael Elhadad. "Abstraction as a Means for End-User Computing in Creative Applications". In: *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans* 32.6 (2002), pp. 640–653. DOI: 10.1109/TSMCA.2002.807042.

[12] Michel Beaudouin-Lafon and Wendy Mackay. "Prototyping Tools and Techniques". In: *The Human-Computer Interaction Handbook: Fundamentals, Evolving Technologies and Emerging Applications*. USA: L. Erlbaum Associates Inc., 2002, pp. 1006–1031. ISBN: 0805838384.

[13] Christopher M Bishop. "Mixture Density Networks". In: *Neural Computing Research Group Report* NCRG/94/004 (Feb. 1994). URL: https://publications.aston.ac.uk/id/eprint/373/1/NCRG_94_004.pdf.

[14] Nathalie Bonnardel. "Creativity in design activities: The role of analogies in a constrained cognitive environment". In: *Proceedings of the 3rd Conference on Creativity & Cognition*. C&C '99. Loughborough, United Kingdom: ACM, 1999, pp. 158–165. ISBN: 1-58113-078-3. DOI: 10.1145/317561.317589. URL: http://doi.acm.org/10.1145/317561.317589.

[15] Andrew Brock, Jeff Donahue, and Karen Simonyan. "Large Scale GAN Training for High Fidelity Natural Image Synthesis". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=B1xsqj09Fm.

[16] Tom Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 1877–1901. URL: https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

[17] Louis L. Bucciarelli. *Designing Engineers*. Inside Technology. Cambridge, Mass: MIT Press, 1994. ISBN: 0262023776.

[18] Richard Buchanan. "Wicked Problems in Design Thinking". In: *Design Issues* 8 (1992), p. 5.

[19] Bill Buxton. *Sketching User Experiences: Getting the Design Right and the Right Design*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2007. ISBN: 9780123740373.

[20] Bradley Camburn et al. "Design prototyping methods: state of the art in strategies, techniques, and guidelines". In: *Design Science* 3 (2017), e13. DOI: 10.1017/dsj.2017.10.

[21] Alexandre Carlier et al. "DeepSVG: A Hierarchical Generative Network for Vector Graphics Animation". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 16351–16361. URL: https://proceedings.neurips.cc/paper/2020/file/bcf9d6bd14a2095866ce8c950b702341-Paper.pdf.

[22] Angel X. Chang et al. "ShapeNet: An Information-Rich 3D Model Repository". In: *CoRR* abs/1512.03012 (2015). arXiv: 1512.03012. URL: http://arxiv.org/abs/1512.03012.

[23] Luming Chen. "TranSketch Dataset: Learning to Transform Sketches". Master's thesis. EECS Department, University of California, Berkeley, May 2020. URL: http://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-92.html.

[24] Mark Chen et al. "Generative Pretraining From Pixels". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, July 2020, pp. 1691–1703. URL: https://proceedings.mlr.press/v119/chen20s.html.

[25] Yunjey Choi et al. "StarGAN v2: Diverse Image Synthesis for Multiple Domains". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 8185–8194. DOI: 10.1109/CVPR42600.2020.00821. URL: https://openaccess.thecvf.com/content_CVPR_2020/html/Choi_StarGAN_v2_Diverse_Image_Synthesis_for_Multiple_Domains_CVPR_2020_paper.html.

[26] Aakanksha Chowdhery et al. "PaLM: Scaling Language Modeling with Pathways". In: *CoRR* abs/2204.02311 (2022). DOI: 10.48550/arXiv.2204.02311. arXiv: 2204.02311. URL: https://doi.org/10.48550/arXiv.2204.02311.

[27] Pinaki Nath Chowdhury et al. "FS-COCO: Towards Understanding of Freehand Sketches of Common Objects in Context". In: *CoRR* abs/2203.02113 (2022). DOI: 10.48550/arXiv.2203.02113. arXiv: 2203.02113. URL: https://doi.org/10.48550/arXiv.2203.02113.

[28] Jamie Combs and Brenda Hoddinott. *Drawing for dummies*. Wiley, 2011.

[29] Nigel Cross. *Design Thinking: Understanding How Designers Think and Work*. Berg Publishers, 2011. ISBN: 9781847886361. URL: https://books.google.com/books?id=Ndu48n8Ik9UC.

[30] Nigel Cross. *Designerly Ways of Knowing*. Springer London, 2006. ISBN: 9781846283017. URL: https://books.google.com/books?id=lvWP3jEW6RAC.

[31] R. Davies and R.J. Talbot. "Experiencing ideas: identity, insight and the imago". In: *Design Studies* 8.1 (1987), pp. 17–25. ISSN: 0142-694X. DOI: https://doi.org/10.1016/0142-694X(87)90027-5. URL: https://www.sciencedirect.com/science/article/pii/0142694X87900275.

[32] Stephen Boyd Davis and Magnus Moar. "The Amateur Creator". In: *Proceedings of the 5th Conference on Creativity & Cognition*. C&C '05. London, United Kingdom: Association for Computing Machinery, 2005, pp. 158–165. ISBN: 1595930256. DOI: 10.1145/1056224.1056247. URL: https://doi.org/10.1145/1056224.1056247.

[33] Biplab Deka, Zifeng Huang, and Ranjitha Kumar. "ERICA: Interaction Mining Mobile Apps". In: *Proceedings of the 29th Annual Symposium on User Interface Software and Technology*. UIST '16. Tokyo, Japan: ACM, 2016, pp. 767–776. ISBN: 978-1-4503-4189-9. DOI: 10.1145/2984511.2984581. URL: http://doi.acm.org/10.1145/2984511.2984581.

[34] Biplab Deka et al. "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. Québec City, QC, Canada: Association for Computing Machinery, 2017, pp. 845–854. ISBN: 9781450349819. DOI: 10.1145/3126594.3126651. URL: https://doi.org/10.1145/3126594.3126651.

[35] Johanna Delanoy et al. "3D Sketching Using Multi-View Deep Volumetric Prediction". In: *Proc. ACM Comput. Graph. Interact. Tech.* 1.1 (July 2018). DOI: 10.1145/3203197. URL: https://doi.org/10.1145/3203197.

[36] Jia Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *2009 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2009), 20-25 June 2009, Miami, Florida, USA*. IEEE Computer Society, 2009, pp. 248–255. DOI: 10.1109/CVPR.2009.5206848. URL: https://doi.org/10.1109/CVPR.2009.5206848.

[37] *Design Thinking*. IDEO. URL: https://www.ideou.com/pages/design-thinking.

[38] Sebastian Deterding et al. "Mixed-Initiative Creative Interfaces". In: *Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*. CHI EA '17. Denver, Colorado, USA: Association for Computing Machinery, 2017, pp. 628–635. ISBN: 9781450346566. DOI: 10.1145/3027063.3027072. URL: https://doi.org/10.1145/3027063.3027072.

[39] Jacob Devlin et al. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *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)*. Minneapolis, Minnesota: Association for Computational Linguistics, June 2019, pp. 4171–4186. DOI: 10.18653/v1/N19-1423. URL: https://aclanthology.org/N19-1423.

[40] Stephanie Dick. "AfterMath: The Work of Proof in the Age of Human–Machine Collaboration". In: *Isis* 102 (2011), pp. 494–505.

[41] Claudia Eckert et al. "Sketching across design domains: Roles and formalities". In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 26.3 (2012), pp. 245–266. DOI: 10.1017/S0890060412000133.

[42] Mathias Eitz, James Hays, and Marc Alexa. "How Do Humans Sketch Objects?" In: *ACM Trans. Graph.* 31.4 (July 2012). ISSN: 0730-0301. DOI: 10.1145/2185520.2185540. URL: https://doi.org/10.1145/2185520.2185540.

[43] Judith Fan. "Drawing to Learn: How Producing Graphical Representations enhances Scientific Thinking". In: *Translational Issues in Psychological Science* 1 (June 2015), pp. 170–181. DOI: 10.1037/tps0000037.

[44] Eugene S. Ferguson. *Engineering and the Mind's Eye*. Cambridge, Mass: MIT Press, 1992. ISBN: 0262061473.

[45] Jennifer Fernquist, Tovi Grossman, and George Fitzmaurice. "Sketch-sketch Revolution: An Engaging Tutorial System for Guided Sketching and Application Learning". In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. Santa Barbara, California, USA: ACM, 2011, pp. 373–382. ISBN: 978-1-4503-0716-1. DOI: 10.1145/2047196.2047245. URL: http://doi.acm.org/10.1145/2047196.2047245.

[46] Michael H. Fischer, Richard R. Yang, and Monica S. Lam. "ImagineNet: Restyling Apps Using Neural Style Transfer". In: *CoRR* abs/2001.04932 (2020). arXiv: 2001.04932. URL: https://arxiv.org/abs/2001.04932.

[47] Kevin Frans, Lisa B. Soros, and Olaf Witkowski. "CLIPDraw: Exploring Text-to-Drawing Synthesis through Language-Image Encoders". In: *CoRR* abs/2106.14843 (2021). arXiv: 2106.14843. URL: https://arxiv.org/abs/2106.14843.

[48] Chengying Gao et al. "SketchyCOCO: Image Generation From Freehand Scene Sketches". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 5173–5182. DOI: 10.1109/CVPR42600.2020.00522. URL: https://openaccess.thecvf.com/content_CVPR_2020/html/Gao_SketchyCOCO_Image_Generation_From_Freehand_Scene_Sketches_CVPR_2020_paper.html.

[49] Henry Gasser. "From Pencil Note to Painting". In: *Design* 61.2 (1959), pp. 68–71. DOI: 10.1080/00119253.1959.10744001. eprint: https://doi.org/10.1080/00119253.1959.10744001. URL: https://doi.org/10.1080/00119253.1959.10744001.

[50] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: *CoRR* abs/1508.06576 (2015). arXiv: 1508.06576. URL: http://arxiv.org/abs/1508.06576.

[51] Songwei Ge et al. "Creative Sketch Generation". In: *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=gwnoVHIES05.

[52] Philippe Gervais et al. "The DIDI dataset: Digital Ink Diagram data". In: *CoRR* abs/2002.09303 (2020). arXiv: 2002.09303. URL: https://arxiv.org/abs/2002.09303.

[53] Arnab Ghosh et al. "Interactive Sketch & Fill: Multiclass Sketch-to-Image Translation". In: *2019 IEEE/CVF International Conference on Computer Vision, ICCV 2019, Seoul, Korea (South), October 27 - November 2, 2019*. IEEE, 2019, pp. 1171–1180. DOI: 10.1109/ICCV.2019.00126. URL: https://doi.org/10.1109/ICCV.2019.00126.

[54] Daniel Gillick, Alessandro Presta, and Gaurav Singh Tomar. "End-to-End Retrieval in Continuous Space". In: *CoRR* abs/1811.08008 (2018). arXiv: 1811.08008. URL: http://arxiv.org/abs/1811.08008.

[55] Vinod Goel and Jordan Grafman. "Role of the Right Prefrontal Cortex in Ill-structured Planning". In: *Cognitive Neuropsychology* 17.5 (2000). PMID: 20945189, pp. 415–436. DOI: 10.1080/026432900410775. eprint: https://doi.org/10.1080/026432900410775. URL: https://doi.org/10.1080/026432900410775.

[56] Ian Goodfellow et al. "Generative Adversarial Nets". In: *Advances in Neural Information Processing Systems*. Ed. by Z. Ghahramani et al. Vol. 27. Curran Associates, Inc., 2014. URL: https://proceedings.neurips.cc/paper/2014/file/5ca3e9b122f61f8f06494c97b1afccf3-Paper.pdf.

[57] Google. *Material Design*. 2021. URL: https://material.io/.

[58] Yulia Gryaditskaya et al. "OpenSketch: A Richly-Annotated Dataset of Product Design Sketches". In: *ACM Trans. Graph.* 38.6 (Nov. 2019). ISSN: 0730-0301. DOI: 10.1145/3355089.3356533. URL: https://doi.org/10.1145/3355089.3356533.

[59] Kelleher Guerin et al. "Adjutant: A Framework for Flexible Human-Machine Collaborative Systems". In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems, Chicago, IL, USA, September 14-18, 2014*. IEEE, 2014, pp. 1392–1399. DOI: 10.1109/IROS.2014.6942739. URL: https://doi.org/10.1109/IROS.2014.6942739.

[60] Kamal Gupta et al. "LayoutTransformer: Layout Generation and Completion with Self-attention". In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, pp. 984–994. DOI: 10.1109/ICCV48922.2021.00104. URL: https://doi.org/10.1109/ICCV48922.2021.00104.

[61] David Ha, Andrew M. Dai, and Quoc V. Le. "HyperNetworks". In: *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=rkpACe1lx.

[62] David Ha and Douglas Eck. "A Neural Representation of Sketch Drawings". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. 2018. URL: https://openreview.net/forum?id=Hy6GHpkCW.

[63] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 770–778. DOI: 10.1109/CVPR.2016.90. URL: https://doi.org/10.1109/CVPR.2016.90.

[64] Zecheng He et al. "ActionBert: Leveraging User Actions for Semantic Understanding of User Interfaces". In: *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*. AAAI Press, 2021, pp. 5931–5938. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16741.

[65] Marti A. Hearst. "Mixed-Initiative Interaction". In: *IEEE Intelligent Systems* (Sept. 1999), pp. 14–24.

[66] Kathryn Henderson. "Flexible Sketches and Inflexible Data Bases: Visual Communication, Conscription Devices, and Boundary Objects in Design Engineering". In: *Science, Technology, & Human Values* 16.4 (1991), pp. 448–473. DOI: 10.1177/016224399101600402. eprint: https://doi.org/10.1177/016224399101600402. URL: https://doi.org/10.1177/016224399101600402.

[67] Kathryn Henderson. *On Line and on Paper: Visual Representations, Visual Culture, and Computer Graphics in Design Engineering*. Cambridge, MA, USA: MIT Press, 1998. ISBN: 0262082691.

[68] James W. Hennessey et al. "How2Sketch: Generating Easy-to-Follow Tutorials for Sketching 3D Objects". In: I3D '17. San Francisco, California: Association for Computing Machinery, 2017. ISBN: 9781450348867. DOI: 10.1145/3023368.3023371. URL: https://doi.org/10.1145/3023368.3023371.

[69] Christopher Henshilwood et al. "An abstract drawing from the 73,000-year-old levels at Blombos Cave, South Africa". In: *Nature* 562 (Oct. 2018). DOI: 10.1038/s41586-018-0514-3.

[70] Scarlett R. Herring et al. "Getting Inspired!: Understanding How and Why Examples Are Used in Creative Design Practice". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '09. Boston, MA, USA: ACM, 2009, pp. 87–96. ISBN: 978-1-60558-246-7. DOI: 10.1145/1518701.1518717. URL: http://doi.acm.org/10.1145/1518701.1518717.

[71] D. L. Hoffmann et al. "U-Th dating of carbonate crusts reveals Neandertal origin of Iberian cave art". In: *Science* 359 (Feb. 2018), p. 912. URL: http://science.sciencemag.org/content/359/6378/912.abstract.

[72] Jan Hoftijzer et al. "A typology of design sketches, defined by communication factors: The case study of the Thule Yepp nexxt child bike seat". English. In: *Proceedings of the 20th International Conference on Engineering and Product Design Education, E and PDE 2018*. Ed. by Stephen Green et al. Institution of Engineering Designers, The Design Society, 2018. URL: https://epde.info/epde2018/.

[73] Seunghoon Hong et al. "Inferring Semantic Layout for Hierarchical Text-to-Image Synthesis". In: *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. Computer Vision Foundation / IEEE Computer Society, 2018, pp. 7986–7994. DOI: 10.1109/CVPR.2018.00833. URL: http://openaccess.thecvf.com/content_cvpr_2018/html/Hong_Inferring_Semantic_Layout_CVPR_2018_paper.html.

[74] Eric Horvitz. "Principles of Mixed-Initiative User Interfaces". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '99. Pittsburgh, Pennsylvania, USA: Association for Computing Machinery, 1999, pp. 159–166. ISBN: 0201485591. DOI: 10.1145/302979.303030. URL: https://doi.org/10.1145/302979.303030.

[75] Stephanie Houde and Charles Hill. "Chapter 16 - What do Prototypes Prototype?" In: *Handbook of Human-Computer Interaction (Second Edition)*. Ed. by Marting G. Helander, Thomas K. Landauer, and Prasad V. Prabhu. Second Edition. Amsterdam: North-Holland, 1997, pp. 367–381. ISBN: 978-0-444-81862-1. DOI: https://doi.org/10.1016/B978-044481862-1.50082-0. URL: https://www.sciencedirect.com/science/article/pii/B9780444818621500820.

[76] Ruizhen Hu et al. "Predictive and Generative Neural Networks for Object Functionality". In: *ACM Trans. Graph.* 37.4 (July 2018). ISSN: 0730-0301. DOI: 10.1145/3197517.3201287. URL: https://doi.org/10.1145/3197517.3201287.

[77] Forrest Huang and John F. Canny. "Sketchforme: Composing Sketched Scenes from Text Descriptions for Interactive Applications". In: *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. UIST '19. New Orleans, LA, USA: Association for Computing Machinery, 2019, pp. 209–220. ISBN: 9781450368162. DOI: 10.1145/3332165.3347878. URL: https://doi.org/10.1145/3332165.3347878.

[78] Forrest Huang, John F. Canny, and Jeffrey Nichols. "Swire: Sketch-Based User Interface Retrieval". In: *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. CHI '19. Glasgow, Scotland, UK: Association for Computing Machinery, 2019. ISBN: 9781450359702. DOI: 10.1145/3290605.3300334. URL: https://doi.org/10.1145/3290605.3300334.

[79] Forrest Huang et al. "Creating User Interface Mock-ups from High-Level Text Descriptions with Deep-Learning Models". In: *CoRR* abs/2110.07775 (2021). arXiv: 2110.07775. URL: https://arxiv.org/abs/2110.07775.

[80] Forrest Huang et al. "Scones: Towards Conversational Authoring of Sketches". In: *Proceedings of the 25th International Conference on Intelligent User Interfaces*. IUI '20. Cagliari, Italy: Association for Computing Machinery, 2020, pp. 313–323. ISBN: 9781450371186. DOI: 10.1145/3377325.3377485. URL: https://doi.org/10.1145/3377325.3377485.

[81] Forrest Huang et al. "Sketch-Based Creativity Support Tools Using Deep Learning". In: *Artificial Intelligence for Human Computer Interaction: A Modern Approach*. Ed. by Yang Li and Otmar Hilliges. Cham: Springer International Publishing, 2021. ISBN: 978-3-030-82681-9. DOI: 10.1007/978-3-030-82681-9_12. URL: https://doi.org/10.1007/978-3-030-82681-9_12.

[82] Holly Huey et al. "Developmental Changes in the Semantic Part Structure of Drawn Objects". In: *Proceedings of the Annual Meeting of the Cognitive Science Society*. Vol. 44. 2022.

[83] Emmanuel Iarussi, Adrien Bousseau, and Theophanis Tsandilas. "The Drawing Assistant: Automated Drawing Guidance and Feedback from Photographs". In: *Proceedings of the 26th Annual ACM Symposium on User Interface Software and Technology*. UIST '13. St. Andrews, Scotland, United Kingdom: Association for Computing Machinery, 2013, pp. 183–192. ISBN: 9781450322683. DOI: 10.1145/2501988.2501997. URL: https://doi.org/10.1145/2501988.2501997.

[84] Phillip Isola et al. "Image-to-Image Translation with Conditional Adversarial Networks". In: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*. IEEE Computer Society, 2017, pp. 5967–5976. DOI: 10.1109/CVPR.2017.632. URL: https://doi.org/10.1109/CVPR.2017.632.

[85] Ajay Jain et al. "Zero-Shot Text-Guided Object Generation With Dream Fields". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2022, pp. 867–876.

[86] Shagun Jhaver et al. "Human-Machine Collaboration for Content Regulation: The Case of Reddit Automoderator". In: *ACM Trans. Comput.-Hum. Interact.* 26.5 (July 2019). ISSN: 1073-0516. DOI: 10.1145/3338243. URL: https://doi.org/10.1145/3338243.

[87] Ellen Jiang et al. "PromptMaker: Prompt-Based Prototyping with Large Language Models". In: *Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems*. CHI EA '22. New Orleans, LA, USA: Association for Computing Machinery, 2022. ISBN: 9781450391566. DOI: 10.1145/3491101.3503564. URL: https://doi.org/10.1145/3491101.3503564.

[88] Shuhui Jiang and Yun Fu. "Fashion Style Generator". In: *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI-17*. 2017, pp. 3721–3727. DOI: 10.24963/ijcai.2017/520. URL: https://doi.org/10.24963/ijcai.2017/520.

[89] Y. Jing et al. "Neural Style Transfer: A Review". In: *IEEE Transactions on Visualization & Computer Graphics* 26.11 (Nov. 2020), pp. 3365–3385. ISSN: 1941-0506. DOI: 10.1109/TVCG.2019.2921336.

[90] *John Constable's sketches*. Victoria and Albert Museum. URL: https://www.vam.ac.uk/articles/john-constables-sketches.

[91] Jonas Jongejan et al. *The Quick, Draw! - A.I. Experiment*. 2016. URL: https://quickdraw.withgoogle.com/ (visited on 10/03/2019).

[92] Mandar Joshi et al. "TriviaQA: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension". In: *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Vancouver, Canada: Association for Computational Linguistics, July 2017, pp. 1601–1611. DOI: 10.18653/v1/P17-1147. URL: https://aclanthology.org/P17-1147.

[93] Catherine Kaplun. "Children's drawings speak a thousand words in their transition to school". In: *Australasian Journal of Early Childhood* 44.4 (2019), pp. 392–407. DOI: 10.1177/1836939119870887. eprint: https://doi.org/10.1177/183693911987088 7. URL: https://doi.org/10.1177/1836939119870887.

[94] Phyllis Katz. *Drawing for Science Education: An International Perspective*. SensePublishers, 2017. ISBN: 9789463008754. URL: https://books.google.com/books?id=LqNODgAAQBAJ.

[95] Rubaiat Habib Kazi et al. "DreamSketch: Early Stage 3D Design Explorations with Sketching and Generative Design". In: *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. UIST '17. Québec City, QC, Canada: ACM, 2017, pp. 401–414. ISBN: 978-1-4503-4981-9. DOI: 10.1145/3126594.3126662. URL: http://doi.acm.org/10.1145/3126594.3126662.

[96] Rhoda Kellogg. *Analyzing Children's Art*. Palo Alto, Calif: National Press Books, 1969. ISBN: 0874841968.

[97] Andruid Kerne and Eunyee Koh. "Creativity Support: The Mixed-Initiative Composition Space". In: *Proceedings of the 7th ACM/IEEE-CS Joint Conference on Digital Libraries*. JCDL '07. Vancouver, BC, Canada: Association for Computing Machinery, 2007, p. 509. ISBN: 9781595936448. DOI: 10.1145/1255175.1255309. URL: https://doi.org/10.1145/1255175.1255309.

[98] Jin-Hwa Kim et al. "CoDraw: Collaborative Drawing as a Testbed for Grounded Goal-driven Communication". In: *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Florence, Italy: Association for Computational Linguistics, July 2019, pp. 6495–6513. DOI: 10.18653/v1/P19-1651. URL: https://www.aclweb.org/anthology/P19-1651.

[99] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2015. URL: http://arxiv.org/abs/1412.6980.

[100] Amy J. Ko et al. "The State of the Art in End-User Software Engineering". In: *ACM Comput. Surv.* 43.3 (Apr. 2011). ISSN: 0360-0300. DOI: 10.1145/1922649.1922658. URL: https://doi.org/10.1145/1922649.1922658.

[101] Sebastian Koch et al. "ABC: A Big CAD Model Dataset for Geometric Deep Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 9601–9611. DOI: 10.1109/CVPR.2019.00983. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Koch_ABC_A_Big_CAD_Model_Dataset_for_Geometric_Deep_Learning_CVPR_2019_paper.html.

[102] Jing Yu Koh et al. "Text-to-Image Generation Grounded by Fine-Grained User Attention". In: *IEEE Winter Conference on Applications of Computer Vision, WACV 2021, Waikoloa, HI, USA, January 3-8, 2021*. IEEE, 2021, pp. 237–246. DOI: 10.1109/WACV48630.2021.00028. URL: https://doi.org/10.1109/WACV48630.2021.00028.

[103] Ranjay Krishna et al. "Visual Genome: Connecting Language and Vision Using Crowdsourced Dense Image Annotations". In: *Int. J. Comput. Vision* 123.1 (May 2017), pp. 32–73. ISSN: 0920-5691. DOI: 10.1007/s11263-016-0981-7. URL: https://doi.org/10.1007/s11263-016-0981-7.

[104] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[105] *Multi-Modal Search for Inspirational Examples in Design*. Vol. Volume 6: 33rd International Conference on Design Theory and Methodology (DTM). International Design Engineering Technical Conferences and Computers and Information in Engineering Conference. V006T06A020. Aug. 2021. DOI: 10.1115/DETC2021-71825. eprint: https://asmedigitalcollection.asme.org/IDETC-CIE/proceedings-pdf/IDETC-CIE2021/85420/V006T06A020/6801617/v006t06a020-detc2021-71825.pdf. URL: https://doi.org/10.1115/DETC2021-71825.

[106] James A. Landay. "SILK: Sketching Interfaces Like Krazy". In: *Conference Companion on Human Factors in Computing Systems*. CHI '96. Vancouver, British Columbia, Canada: ACM, 1996, pp. 398–399. ISBN: 0-89791-832-0. DOI: 10.1145/257089.257396. URL: http://doi.acm.org/10.1145/257089.257396.

[107] Walter S. Lasecki et al. "Apparition: Crowdsourced User Interfaces That Come to Life As You Sketch Them". In: *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*. CHI '15. Seoul, Republic of Korea: ACM, 2015, pp. 1925–1934. ISBN: 978-1-4503-3145-6. DOI: 10.1145/2702123.2702565. URL: http://doi.acm.org/10.1145/2702123.2702565.

[108] Irving Lavin. "Picasso's Bull(s): Art History in Reverse". In: *Art in America* 81.3 (1993), pp. 76–93.

[109] Bryan Lawson. *Design in Mind*. Butterworth Architecture, 1994. ISBN: 9780750612111. URL: https://books.google.com/books?id=aQdQAAAAMAAJ.

[110] Brian Lee et al. "Designing with Interactive Example Galleries". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '10. Atlanta, Georgia, USA: ACM, 2010, pp. 2257–2266. ISBN: 978-1-60558-929-9. DOI: 10.1145/1753326.1753667. URL: http://doi.acm.org/10.1145/1753326.1753667.

[111] Hsin-Ying Lee et al. "Neural Design Network: Graphic Layout Generation with Constraints". In: *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part III*. Ed. by Andrea Vedaldi et al. Vol. 12348. Lecture Notes in Computer Science. Springer, 2020, pp. 491–506. DOI: 10.1007/978-3-030-58580-8_29. URL: https://doi.org/10.1007/978-3-030-58580-8_29.

[112] Yong Jae Lee, C. Lawrence Zitnick, and Michael F. Cohen. "ShadowDraw: Real-time User Guidance for Freehand Drawing". In: *ACM Trans. Graph.* 30.4 (July 2011), 27:1–27:10. ISSN: 0730-0301. DOI: 10.1145/2010324.1964922. URL: http://doi.acm.org/10.1145/2010324.1964922.

[113] Sebastian Leitner. *So Lernt Man Lernen*. Angewandte Lernpsychologie ein Weg zum Erfolg. Herder, 1972. ISBN: 9783451162657. URL: https://books.google.com/books?id=sD3AAQAACAAJ.

[114] Bo Li et al. "Sketch-Based 3D Model Retrieval Utilizing Adaptive View Clustering and Semantic Information". In: *Multimedia Tools Appl.* 76.24 (Dec. 2017), pp. 26603–26631. ISSN: 1380-7501. URL: https://doi.org/10.1007/s11042-016-4187-3.

[115] Jianan Li et al. "LayoutGAN: Generating Graphic Layouts with Wireframe Discriminators". In: *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HJxB5sRcFQ.

[116] Toby Jia-Jun Li et al. "Screen2Vec: Semantic Embedding of GUI Screens and GUI Components". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: 10.1145/3411764.3445049. URL: https://doi.org/10.1145/3411764.3445049.

[117] Yang Li et al. "Mapping Natural Language Instructions to Mobile UI Action Sequences". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, July 2020, pp. 8198–8210. DOI: 10.18653/v1/2020.acl-main.729. URL: https://aclanthology.org/2020.acl-main.729.

[118] Antonios Liapis et al. "Can Computers Foster Human Users' Creativity? Theory and Praxis of Mixed-Initiative Co-Creativity". In: *Digital Culture & Education* 8 (2016).

[119] Alex Limpaecher et al. "Real-time Drawing Assistance Through Crowdsourcing". In: *ACM Trans. Graph.* 32.4 (July 2013), 54:1–54:8. ISSN: 0730-0301. DOI: 10.1145/2461912.2462016. URL: http://doi.acm.org/10.1145/2461912.2462016.

[120] James Lin et al. "DENIM: Finding a Tighter Fit Between Tools and Practice for Web Site Design". In: *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*. CHI '00. The Hague, The Netherlands: ACM, 2000, pp. 510–517. ISBN: 1-58113-216-6. DOI: 10.1145/332040.332486. URL: http://doi.acm.org/10.1145/332040.332486.

[121] Tsung-Yi Lin et al. "Microsoft COCO: Common Objects in Context". In: *Computer Vision - ECCV 2014 - 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V*. Ed. by David J. Fleet et al. Vol. 8693. Lecture Notes in Computer Science. Springer, 2014, pp. 740–755. DOI: 10.1007/978-3-319-10602-1_48. URL: https://doi.org/10.1007/978-3-319-10602-1_48.

[122] Thomas F. Liu et al. "Learning Design Semantics for Mobile Apps". In: *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*. UIST '18. Berlin, Germany: Association for Computing Machinery, 2018, pp. 569–579. ISBN: 9781450359481. DOI: 10.1145/3242587.3242650. URL: https://doi.org/10.1145/3242587.3242650.

[123] Bria Long, Judith E. Fan, and Michael C. Frank. "Drawings as a window into developmental changes in object representations". In: *Cognitive Science* (2018).

[124] Ellen Yi-Luen Do. "Design sketches and sketch design tools". In: *Knowledge-Based Systems* 18.8 (2005). Computational Approaches for Early Stages of Design, pp. 383–405. ISSN: 0950-7051. DOI: https://doi.org/10.1016/j.knosys.2005.07.001. URL: https://www.sciencedirect.com/science/article/pii/S0950705105000705.

[125] Tony McCaffrey and Lee Spector. "An approach to human–machine collaboration in innovation". In: *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 32.1 (2018), pp. 1–15. DOI: 10.1017/S0890060416000524.

[126] Mehdi Mirza and Simon Osindero. "Conditional Generative Adversarial Nets". In: *CoRR* abs/1411.1784 (2014). arXiv: 1411.1784. URL: http://arxiv.org/abs/1411.1784.

[127] Shigeru Miyagawa, Cora Lesure, and Vitor A. Nóbrega. "Cross-Modality Information Transfer: A Hypothesis about the Relationship among Prehistoric Cave Paintings, Symbolic Thinking, and the Emergence of Language". In: *Frontiers in Psychology* 9 (2018). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2018.00115. URL: https://www.frontiersin.org/article/10.3389/fpsyg.2018.00115.

[128] Kaichun Mo et al. "PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 909–918. DOI: 10.1109/CVPR.2019.00100. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Mo_PartNet_A_Large-Scale_Benchmark_for_Fine-Grained_and_Hierarchical_Part-Level_3D_CVPR_2019_paper.html.

[129] Gillian M. Morriss-Kay. "The evolution of human artistic creativity". In: *Journal of Anatomy* 216 (2010).

[130] Nasrin Mostafazadeh et al. "A Corpus and Cloze Evaluation for Deeper Understanding of Commonsense Stories". In: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, June 2016, pp. 839–849. DOI: 10.18653/v1/N16-1098. URL: https://aclanthology.org/N16-1098.

[131] Sifat Muin and Khalid Mosalam. "Human-machine collaboration framework for structural health monitoring and resiliency". In: *Engineering Structures* 235 (May 2021), p. 112084. DOI: 10.1016/j.engstruct.2021.112084.

[132] Pandu Nayak. *MUM: A new AI milestone for understanding information*. Google, May 2021. URL: https://blog.google/products/search/introducing-mum/.

[133] Irene Neilson and John Lee. "Conversations with graphics: implications for the design of natural language/graphics interfaces". In: *International Journal of Human-Computer Studies* 40.3 (1994), pp. 509–541. ISSN: 1071-5819. DOI: https://doi.org/10.1006/ijhc.1994.1024. URL: https://www.sciencedirect.com/science/article/pii/S107158198471024X.

[134] Donald A. Norman. *The Design of Everyday Things*. USA: Basic Books, Inc., 2002. ISBN: 9780465067107.

[135] David G. Novick and Stephen Sutton. "What is mixed-initiative interaction". In: *In Procedings of the AAAI Spring Symposium on Computational Models for Mixed Initiative Interaction*. 1997, pp. 114–116.

[136] Aäron van den Oord, Yazhe Li, and Oriol Vinyals. "Representation Learning with Contrastive Predictive Coding". In: *CoRR* abs/1807.03748 (2018). arXiv: 1807.03748. URL: http://arxiv.org/abs/1807.03748.

[137] Nicolas Padoy and Gregory D. Hager. "Human-Machine Collaborative Surgery Using Learned Models". In: *IEEE International Conference on Robotics and Automation, ICRA 2011, Shanghai, China, 9-13 May 2011*. IEEE, 2011, pp. 5285–5292. DOI: 10.1109/ICRA.2011.5980250. URL: https://doi.org/10.1109/ICRA.2011.5980250.

[138] Denis Paperno et al. "The LAMBADA dataset: Word prediction requiring a broad discourse context". In: *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1525–1534. DOI: 10.18653/v1/P16-1144. URL: https://aclanthology.org/P16-1144.

[139] Kishore Papineni et al. "BLEU: a Method for Automatic Evaluation of Machine Translation". In: *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*. Philadelphia, Pennsylvania, USA: Association for Computational Linguistics, July 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. URL: https://aclanthology.org/P02-1040.

[140] Zarana Parekh et al. "Crisscrossed Captions: Extended Intramodal and Intermodal Semantic Similarity Judgments for MS-COCO". In: *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*. Online: Association for Computational Linguistics, Apr. 2021, pp. 2855–2870. URL: https://aclanthology.org/2021.eacl-main.249.

[141] Taesung Park et al. "Semantic Image Synthesis With Spatially-Adaptive Normalization". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. Computer Vision Foundation / IEEE, 2019, pp. 2337–2346. DOI: 10.1109/CVPR.2019.00244. URL: http://openaccess.thecvf.com/content_CVPR_2019/html/Park_Semantic_Image_Synthesis_With_Spatially-Adaptive_Normalization_CVPR_2019_paper.html.

[142] Jeffrey Pennington, Richard Socher, and Christopher Manning. "GloVe: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. URL: https://aclanthology.org/D14-1162.

[143] Federico Perazzi et al. "A Benchmark Dataset and Evaluation Methodology for Video Object Segmentation". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, pp. 724–732. DOI: 10.1109/CVPR.2016.85. URL: https://doi.org/10.1109/CVPR.2016.85.

[144] A.W.G. Pike et al. "U-Series Dating of Paleolithic Art in 11 Caves in Spain". In: *Science (New York, N.Y.)* 336 (June 2012), pp. 1409–13. DOI: 10.1126/science.1219957.

[145] Stuart Pugh, Don Clausing, and Ron Andrade. *Creating Innovative Products Using Total Design: The Living Legacy of Stuart Pugh*. Engineering Design. Addison-Wesley Publishing Company, 1996. ISBN: 9780201634853. URL: https://books.google.com/books?id=sv9TAAAAMAAJ.

[146] A.T. Purcell and J.S. Gero. "Drawings and the design process: A review of protocol studies in design and other disciplines and related research in cognitive psychology". In: *Design Studies* 19.4 (1998), pp. 389–430. ISSN: 0142-694X. DOI: https://doi.org/10.1016/S0142-694X(98)00015-5. URL: https://www.sciencedirect.com/science/article/pii/S0142694X98000155.

[147] Alec Radford, Luke Metz, and Soumith Chintala. "Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks". In: *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*. Ed. by Yoshua Bengio and Yann LeCun. 2016. URL: http://arxiv.org/abs/1511.06434.

[148] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: (2019).

[149] Alec Radford et al. "Learning Transferable Visual Models From Natural Language Supervision". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8748–8763. URL: http://proceedings.mlr.press/v139/radford21a.html.

[150] Aditya Ramesh et al. "Hierarchical Text-Conditional Image Generation with CLIP Latents". In: *CoRR* abs/2204.06125 (2022). DOI: 10.48550/arXiv.2204.06125. arXiv: 2204.06125. URL: https://doi.org/10.48550/arXiv.2204.06125.

[151] Aditya Ramesh et al. "Zero-Shot Text-to-Image Generation". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 8821–8831. URL: http://proceedings.mlr.press/v139/ramesh21a.html.

[152] Gonzalo A. Ramos et al. "Interactive Machine Teaching: a human-centered approach to building machine-learned models". In: *Hum. Comput. Interact.* 35.5-6 (2020), pp. 413–451. DOI: 10.1080/07370024.2020.1734931. URL: https://doi.org/10.1080/07370024.2020.1734931.

[153] James V. Rauff. "Rock Art Tallies: Mathematics on Stone in Western North America". In: *Journal of Humanistic Mathematics* 3.2 (2013), pp. 76–87.

[154] Pradyumna Reddy et al. "Im2Vec: Synthesizing Vector Graphics Without Vector Supervision". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 7342–7351. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Reddy_Im2Vec_Synthesizing_Vector_Graphics_Without_Vector_Supervision_CVPR_2021_paper.html.

[155] Scott Reed et al. "Generative Adversarial Text to Image Synthesis". In: *Proceedings of The 33rd International Conference on Machine Learning*. Ed. by Maria Florina Balcan and Kilian Q. Weinberger. Vol. 48. Proceedings of Machine Learning Research. New York, New York, USA: PMLR, June 2016, pp. 1060–1069. URL: https://proceedings.mlr.press/v48/reed16.html.

[156] Leo Sampaio Ferraz Ribeiro et al. "Sketchformer: Transformer-Based Representation for Sketched Structure". In: *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*. Computer Vision Foundation / IEEE, 2020, pp. 14141–14150. DOI: 10.1109/CVPR42600.2020.01416. URL: https://openaccess.thecvf.com/content_CVPR_2020/html/Ribeiro_Sketchformer_Transformer-Based_Representation_for_Sketched_Structure_CVPR_2020_paper.html.

[157] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R. Klemmer. "d.tour: Style-Based Exploration of Design Example Galleries". In: *Proceedings of the 24th Annual ACM Symposium on User Interface Software and Technology*. UIST '11. Santa Barbara, California, USA: Association for Computing Machinery, 2011, pp. 165–174. ISBN: 9781450307161. DOI: 10.1145/2047196.2047216. URL: https://doi.org/10.1145/2047196.2047216.

[158] Horst W. J. Rittel and Melvin M. Webber. "Dilemmas in a General Theory of Planning". In: *Policy Sciences* 4.2 (June 1973), pp. 155–169. DOI: doi:10.1007/BF01405730. URL: http://dx.doi.org/doi:10.1007/BF01405730.

[159] Adam Roberts et al. "Magenta Studio: Augmenting Creativity with Deep Learning in Ableton Live". In: *Proceedings of the International Workshop on Musical Metacreation (MUME)*. 2019. URL: http://musicalmetacreation.org/buddydrive/file/mume_2019_paper_2/.

[160] Scott Robertson and Thomas Bertling. *How to Draw: Drawing and Sketching Objects and Environments from Your Imagination*. Design Studio Press, 2013. ISBN: 9781933492759. URL: https://books.google.com/books?id=2w40LgEACAAJ.

[161] Alex Robinson. "sketch2code: Generating a website from a paper mockup". In: *CoRR* abs/1905.13750 (2019). arXiv: 1905.13750. URL: http://arxiv.org/abs/1905.13750.

[162] Amélie Royer et al. "XGAN: Unsupervised Image-to-Image Translation for Many-to-Many Mappings". In: *Domain Adaptation for Visual Understanding*. Ed. by Richa Singh et al. Cham: Springer International Publishing, 2020, pp. 33–49. ISBN: 978-3-030-30671-7. DOI: 10.1007/978-3-030-30671-7_3. URL: https://doi.org/10.1007/978-3-030-30671-7_3.

[163] Olga Russakovsky, Li-Jia Li, and Li Fei-Fei. "Best of both worlds: human-machine collaboration for object annotation". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 2121–2131. DOI: 10.1109/CVPR.2015.7298824. URL: https://doi.org/10.1109/CVPR.2015.7298824.

[164] Marco de Sá and Luís Carriço. "Low-Fi Prototyping for Mobile Devices". In: *CHI '06 Extended Abstracts on Human Factors in Computing Systems*. CHI EA '06. Montréal, Québec, Canada: Association for Computing Machinery, 2006, pp. 694–699. ISBN: 1595932984. DOI: 10.1145/1125451.1125592. URL: https://doi.org/10.1145/1125451.1125592.

[165] Chitwan Saharia et al. "Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding". In: *CoRR* abs/2205.11487 (2022). DOI: 10.48550/arXiv.2205.11487. arXiv: 2205.11487. URL: https://doi.org/10.48550/arXiv.2205.11487.

[166] Aneeshan Sain et al. "Cross-Modal Hierarchical Modelling for Fine-Grained Sketch Based Image Retrieval". In: *31st British Machine Vision Conference 2020, BMVC 2020, Virtual Event, UK, September 7-10, 2020*. BMVA Press, 2020. URL: https://www.bmvc2020-conference.com/assets/papers/0102.pdf.

[167] Patsorn Sangkloy et al. "Scribbler: Controlling Deep Image Synthesis with Sketch and Color". In: (2017), pp. 6836–6845. DOI: 10.1109/CVPR.2017.723. URL: https://doi.org/10.1109/CVPR.2017.723.

[168] Patsorn Sangkloy et al. "The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies". In: *ACM Trans. Graph.* 35.4 (July 2016), 119:1–119:12. ISSN: 0730-0301. DOI: 10.1145/2897824.2925954. URL: http://doi.acm.org/10.1145/2897824.2925954.

[169] Donald A. Schön. *The Reflective Practitioner: How Professionals Think in Action*. An Ashgate Book. Ashgate, 1991. ISBN: 9781857423198. URL: https://books.google.com/books?id=E85qAAAAMAAJ.

[170] Vinoth Pandian Sermuga Pandian, Sarah Suleri, and Matthias Jarke. "UISketch: A Large-Scale Dataset of UI Element Sketches". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. New York, NY, USA: Association for Computing Machinery, 2021. ISBN: 9781450380966. URL: https://doi.org/10.1145/3411764.3445784.

[171] Mohammad Javad Shafiee et al. "Human-Machine Collaborative Design for Accelerated Design of Compact Deep Neural Networks for Autonomous Driving". In: *BMVC Workshop on Visual AI and Entrepreneurship* (2019). arXiv: 1909.05587. URL: http://arxiv.org/abs/1909.05587.

[172] Piyush Sharma et al. "Conceptual Captions: A Cleaned, Hypernymed, Image Alt-text Dataset For Automatic Image Captioning". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Melbourne, Australia: Association for Computational Linguistics, July 2018, pp. 2556–2565. DOI: 10.18653/v1/P18-1238. URL: https://aclanthology.org/P18-1238.

[173] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. "Self-Attention with Relative Position Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*. New Orleans, Louisiana: Association for Computational Linguistics, June 2018, pp. 464–468. DOI: 10.18653/v1/N18-2074. URL: https://aclanthology.org/N18-2074.

[174] Ben Shneiderman. "Creating Creativity: User Interfaces for Supporting Innovation". In: *ACM Trans. Comput.-Hum. Interact.* 7.1 (Mar. 2000), pp. 114–138. ISSN: 1073-0516. DOI: 10.1145/344949.345077. URL: https://doi.org/10.1145/344949.345077.

[175] Ben Shneiderman. "Creativity Support Tools: Accelerating Discovery and Innovation". In: *Commun. ACM* 50.12 (Dec. 2007), pp. 20–32. ISSN: 0001-0782. DOI: 10.1145/1323688.1323689. URL: https://doi.org/10.1145/1323688.1323689.

[176] Monika Simmler and Ruth Frischknecht. "A taxonomy of human–machine collaboration: capturing automation and technical autonomy". In: *AI Soc.* 36.1 (Mar. 2021), pp. 239–250. ISSN: 0951-5666. DOI: 10.1007/s00146-020-01004-z. URL: https://doi.org/10.1007/s00146-020-01004-z.

[177] Carolyn Snyder. *Paper Prototyping: The Fast and Easy Way to Design and Refine User Interfaces*. ITPro collection. Elsevier Science, 2003. ISBN: 9781558608702. URL: https://books.google.com/books?id=YgBojJsVLGMC.

[178] Kihyuk Sohn. "Improved Deep Metric Learning with Multi-class N-pair Loss Objective". In: *Advances in Neural Information Processing Systems*. Ed. by D. Lee et al. Vol. 29. Curran Associates, Inc., 2016. URL: https://proceedings.neurips.cc/paper/2016/file/6b180037abbebea991d8b1232f8a8ca9-Paper.pdf.

[179] Qingkun Su et al. "EZ-sketching: Three-level Optimization for Error-tolerant Image Tracing". In: *ACM Trans. Graph.* 33.4 (July 2014), 54:1–54:9. ISSN: 0730-0301. DOI: 10.1145/2601097.2601202. URL: http://doi.acm.org/10.1145/2601097.2601202.

[180] Romal Thoppilan et al. "LaMDA: Language Models for Dialog Applications". In: *CoRR* abs/2201.08239 (2022). arXiv: 2201.08239. URL: https://arxiv.org/abs/2201.08239.

[181] David G. Ullman, Stephen Wood, and David Craig. "The importance of drawing in the mechanical design process". In: *Computers & Graphics* 14.2 (1990), pp. 263–274. ISSN: 0097-8493. DOI: https://doi.org/10.1016/0097-8493(90)90037-X. URL: https://www.sciencedirect.com/science/article/pii/009784939090037X.

[182] Héléne Valladas et al. "Evolution of prehistoric cave art". In: *Nature* 413 (Nov. 2001), p. 479. DOI: 10.1038/35097160.

[183] Remko van der Lugt. "How sketching can affect the idea generation process in design group meetings". In: *Design Studies* 26.2 (2005), pp. 101–122. ISSN: 0142-694X. DOI: https://doi.org/10.1016/j.destud.2004.08.003. URL: https://www.sciencedirect.com/science/article/pii/S0142694X04000778.

[184] Ashish Vaswani et al. "Attention Is All you Need". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[185] Jeffrey D. Wammes, Melissa E. Meade, and Myra A. Fernandes. "The drawing effect: Evidence for reliable and robust memory benefits in free recall". In: *The Quarterly Journal of Experimental Psychology* 69.9 (2016). PMID: 26444654, pp. 1752–1776. DOI: 10.1080/17470218.2015.1094494. eprint: https://doi.org/10.1080/17470218.2015.1094494. URL: https://doi.org/10.1080/17470218.2015.1094494.

[186] Bryan Wang et al. "Screen2Words: Automatic Mobile UI Summarization with Multimodal Learning". In: *The 34th Annual ACM Symposium on User Interface Software and Technology*. New York, NY, USA: Association for Computing Machinery, 2021, pp. 498–510. ISBN: 9781450386357. DOI: 10.1145/3472749.3474765. URL: https://doi.org/10.1145/3472749.3474765.

[187] Benjamin Wilkins. *Sketching Interfaces - Generating code from low fidelity wireframes*. Airbnb. URL: https://airbnb.design/sketching-interfaces/.

[188] Karl D. D. Willis et al. "Fusion 360 Gallery: A Dataset and Environment for Programmatic CAD Construction from Human Design Sequences". In: *ACM Trans. Graph.* 40.4 (July 2021). ISSN: 0730-0301. DOI: 10.1145/3450626.3459818. URL: https://doi.org/10.1145/3450626.3459818.

[189] Chui Yin Wong, Chee Weng Khong, and Kimberly Chu. "Interface Design Practice and Education Towards Mobile Apps Development". In: *Procedia - Social and Behavioral Sciences* 51 (2012). The World Conference on Design, Arts and Education (DAE-2012), May 1-3 2012, Antalya, Turkey, pp. 698–702. ISSN: 1877-0428. DOI: https://doi.org/10.1016/j.sbspro.2012.08.227. URL: https://www.sciencedirect.com/science/article/pii/S1877042812033654.

[190] Jun Xie et al. "PortraitSketch: Face Sketching Assistance for Novices". In: *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology*. UIST '14. Honolulu, Hawaii, USA: ACM, 2014, pp. 407–417. ISBN: 978-1-4503-3069-5. DOI: 10.1145/2642918.2647399. URL: http://doi.acm.org/10.1145/2642918.2647399.

[191] Xianghao Xu et al. "Inferring CAD Modeling Sequences Using Zone Graphs". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 6062–6070. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Xu_Inferring_CAD_Modeling_Sequences_Using_Zone_Graphs_CVPR_2021_paper.html.

[192] Yuxiang Ye, Bo Li, and Yijuan Lu. "3D Sketch-based 3D Model Retrieval with Convolutional Neural Network". In: *23rd International Conference on Pattern Recognition, ICPR 2016, Cancún, Mexico, December 4-8, 2016*. IEEE, 2016, pp. 2936–2941. DOI: 10.1109/ICPR.2016.7900083. URL: https://doi.org/10.1109/ICPR.2016.7900083.

[193] Han Zhang et al. "Cross-Modal Contrastive Learning for Text-to-Image Generation". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2021, virtual, June 19-25, 2021*. Computer Vision Foundation / IEEE, 2021, pp. 833–842. URL: https://openaccess.thecvf.com/content/CVPR2021/html/Zhang_Cross-Modal_Contrastive_Learning_for_Text-to-Image_Generation_CVPR_2021_paper.html.

[194] Xiaoyi Zhang et al. "Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels". In: *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*. CHI '21. Yokohama, Japan: Association for Computing Machinery, 2021. ISBN: 9781450380966. DOI: 10.1145/3411764.3445186. URL: https://doi.org/10.1145/3411764.3445186.

[195] Jun-Yan Zhu et al. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision, ICCV 2017, Venice, Italy, October 22-29, 2017*. IEEE Computer Society, 2017, pp. 2242–2251. DOI: 10.1109/ICCV.2017.244. URL: https://doi.org/10.1109/ICCV.2017.244.

[196] Yaoming Zhu et al. "Texygen: A Benchmarking Platform for Text Generation Models". In: *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. SIGIR '18. Ann Arbor, MI, USA: Association for Computing Machinery, 2018, pp. 1097–1100. ISBN: 9781450356572. DOI: 10.1145/3209978.3210080. URL: https://doi.org/10.1145/3209978.3210080.

[197] C. Lawrence Zitnick and Devi Parikh. "Bringing Semantics into Focus Using Visual Abstraction". In: *2013 IEEE Conference on Computer Vision and Pattern Recognition, Portland, OR, USA, June 23-28, 2013*. IEEE Computer Society, 2013, pp. 3009–3016. DOI: 10.1109/CVPR.2013.387. URL: https://doi.org/10.1109/CVPR.2013.387.

[198] Changqing Zou et al. "SketchyScene: Richly-Annotated Scene Sketches". In: *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XV*. Ed. by Vittorio Ferrari et al. Vol. 11219. Lecture Notes in Computer Science. Springer, 2018, pp. 438–454. DOI: 10.1007/978-3-030-01267-0_26. URL: https://doi.org/10.1007/978-3-030-01267-0_26.