Scones: Towards Conversational Authoring of Sketches

Forrest Huang
University of California, Berkeley
Berkeley, California, U.S.A.
forrest_huang@berkeley.edu

Eldon Schoop
University of California, Berkeley
Berkeley, California, U.S.A.
eschoop@berkeley.edu

David Ha
Google Brain
Tokyo, Japan
hadavid@google.com

John Canny
University of California, Berkeley
Berkeley, California, U.S.A.
canny@berkeley.edu

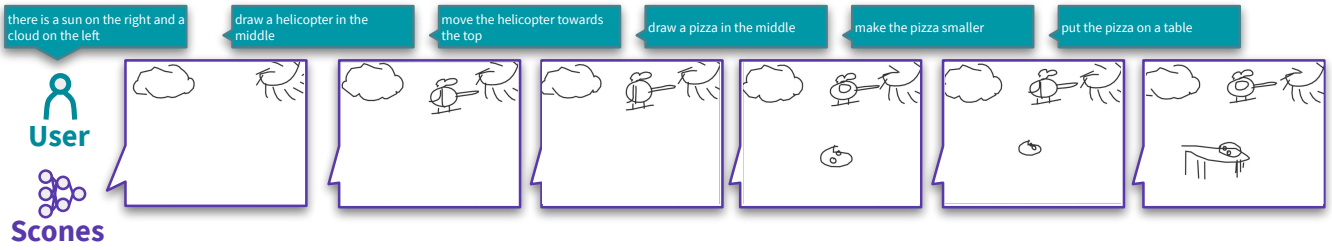

Figure 1: Example interaction between Scones and a human user. Scones can iteratively generate and refine sketched scenes given users' text instructions.

ABSTRACT

Iteratively refining and critiquing sketches are crucial steps to developing effective designs. We introduce Scones, a mixed-initiative, machine-learning-driven system that enables users to iteratively author sketches from text instructions. Scones is a novel deep-learning-based system that iteratively generates *scenes* of *sketched objects* composed with semantic specifications from natural language. Scones exceeds state-of-the-art performance on a text-based scene modification task, and introduces a mask-conditioned sketching model that can generate sketches with poses specified by high-level scene information. In an exploratory user evaluation of Scones, participants reported enjoying an iterative drawing task with Scones, and suggested additional features for further applications. We believe Scones is an early step towards automated, intelligent systems that support human-in-the-loop applications for communicating ideas through sketching in art and design.

CCS CONCEPTS

• **Human-centered computing** → **Natural language interfaces**; • **Computing methodologies** → **Neural networks**; *Computer vision tasks.*

IUI '20, March 17–20, 2020, Cagliari, Italy

ACM ISBN 978-1-4503-7118-6/20/03.
https://doi.org/10.1145/3377325.3377485

KEYWORDS

chatbot; neural networks; critique; sketching; generative models

ACM Reference Format:
Forrest Huang, Eldon Schoop, David Ha, and John Canny. 2020. Scones: Towards Conversational Authoring of Sketches. In *25th International Conference on Intelligent User Interfaces (IUI '20), March 17–20, 2020, Cagliari, Italy.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3377325.3377485

1 INTRODUCTION

Sketching is a powerful communication medium, as even rough drawings can richly communicate the intent of artists, designers, and engineers. These practitioners use sketches as a tool to iteratively present, critique and refine ideas. However, creating sketches that effectively communicate ideas visually requires significant training. Furthermore, the use of sketches in an iterative design process, where the sketch itself is annotated or refined, requires additional, specialized expertise.

Recently developed Machine Learning (ML) models have illuminated how intelligent systems can participate in the sketching and critique processes, e.g., by generating sketches for single objects [7], and using natural language to create images [8]. However, the broader interaction of iteratively *critiquing* and *refining* complex sketches comprising multiple objects poses several additional challenges. For this task, a system would need to 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.

In this paper, we introduce Scones, an intelligent system for iteratively generating and modifying scenes of sketched objects through text instructions. Our contribution is three-fold:

- We formulate the novel interaction of iteratively generating and refining sketches with text instructions and present a web-deployable implementation of Scones to support this interaction.
- We contribute a scene composition proposer, a component of our system 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 introduce a novel method for specifying high-level scene semantics within individual object sketches by conditioning sketch generation with mask outlines of target sketches.

We evaluate our intelligent user interface on an iterative sketching task with 50 participants, where each was asked to use text instructions to create a scene matching a target output. Our results show participants enjoyed the task and were generally satisfied with the output of Scones. Participants also provided feedback for improving Scones in future iterations.

Our ultimate goal for Scones is to support creative processes by facilitating the iterative refinement of complex sketches through natural language. Combining these modalities is a fundamental part of our contributions, as this allows users to freely express their intent using abstract, text-based instructions, together with concrete visual media.

2 RELATED WORK

Scones builds upon related work in four key areas: (1) deep neural networks that generate sketches and scenes, and corresponding datasets they were trained on; (2) sketching support tools that refine and augment sketch inputs; (3) machine-learning-based applications that support image generation from drawing input; and, (4) interfaces that use natural language to interact with visual data.

2.1 Neural Sketch Generation and Large-scale Sketch Datasets

Recent advancements in the ML community introduced deep neural networks capable of recognizing and generating sketches. SketchRNN [7] is one of the first RNN-based models that can generate sequential sketch strokes through supervised learning on sketching datasets. Generative Adversarial Networks (GANs) have also been used to translate realistic images into sketches (or edges) at the pixel level by training on paired [18] and unpaired [35] sketch and image data. While these methods are well-suited for generating sketches of individual objects or stylizing images, they do not encode high-level semantic information of a scene. Sketchforme takes a two-stage approach to generate sketches of scenes comprising multiple objects by first generating a high-level scene layout from text input, and, next, filling the layout with object sketches [9].

These ML techniques rely heavily on large-scale sketching datasets. The Quick, Draw! [12] and TU-Berlin [4] datasets consist of human-drawn sketches for 345 and 250 object classes respectively. SketchyDB provides paired images and simple sketches for retrieval tasks [25]. 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 [36].

Scones builds upon the Sketch-RNN model and Sketchforme's generation process to support progressive, iterative generation and conditional modification of sketched scenes from natural language, a novel machine-learning-driven user interaction. Scones uses a Transformer network [30] with a shared natural language and scene information embedding, and is trained on the CoDraw dataset [15] to learn high-level relationships between objects in scenes and text instructions.

2.2 Interactive Sketching Tools

Research in the Human-Computer Interaction (HCI) community has produced interfaces that use drawing input for creating interactive media and prototypes. SILK enables users to annotate user interface mockup sketches to create interactive prototypes [16]. Other work adapts these metaphors for creating interactive, animated images from drawings [14]. Closely related to our domain, DrawAFriend uses crowdsourced data through a Game With a Purpose (GWAP) [31] to refine and correct users' sketch strokes [19]. PixelTone additionally uses natural language speech input to apply filters to annotated images [17]. Most relevant to Scones, Ribeiro and Igarashi introduced a two-way sketch-based communication game for users to iteratively edit sketches using a direct manipulation interface [24]. Scones uses natural language input to create and modify sketches, rather than requiring direct pen stroke input.

2.3 Interactive Image Generation

Researchers have also explored interactive image generation, particularly for GAN-based methods. iGAN fills user-provided outlines with generated image textures [34]. More recent work has enabled finer-grained control of the output by providing tools for drawing semantic maps for generating artwork [3] and photorealistic images [20]. These approaches have also been extended to fill users' drawn outlines with image textures [6] and to generate realistic clothing items in a user-specific recommender system [13]. While these methods allow users to control the content of and iteratively add to generated images, they rely on a direct *visual-to-visual* mapping between input and output media to transfer user intent to the canvas. In contrast, Scones uses a *language-to-visual* mapping, enabling users to add to and modify sketches using natural language, a higher-level medium that allows for variation within user specifications.

2.4 Interfaces Supporting Natural Language Interactions with Visual Data

Several novel user interfaces and ML models have been developed to use natural language input in interactive visual tasks, with a *language-to-visual* mapping. An example ML challenge in this domain is Visual Question Answering (VQA), where a model is provided with a target image and a natural language question as input, and outputs a response predicated on visual, lingual, and commonsense knowledge [1]. Other challenges extend this by requiring *justification* of the response [11, 33], or the truthfulness of an input statement relating two images [28]. These questions in these challenges can be answered by ML architectures such as Relation Networks (RNs), which infer object relationships from the outputs of RNNs and Convolutional Neural Networks (CNNs) [26].

Alternatively, VizWiz deploys just-in-time crowdsourcing tasks to answer open-ended questions about an image for visually impaired users [2].

Deep learning models have been used for image retrieval from natural language captions [5], and algorithmic approaches have been used for searching within videos [21]. Adaptive interfaces can also draw correspondence between language and visual block manipulation tasks [32]. Fashion interfaces which recommend items from natural language specifications [27] or by connecting users to stylists through a chatbot [29] require knowledge of items' semantic and visual features, as well as highly variant user preferences.

Scones also uses a *language-to-visual* mapping, adapting a state-of-the-art deep neural network to use natural language input for a novel visual output task: interactive sketch creation and modification.

3 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 ML systems after this high-to-low-level workflow has been beneficial for transfer learning from other visual domains and for supporting interactive interfaces for human users [9]. 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 *Composition Proposer*, then generates individual object sketches, conditioned on the scene-level information, using its *Object Generators* (Figure 2).

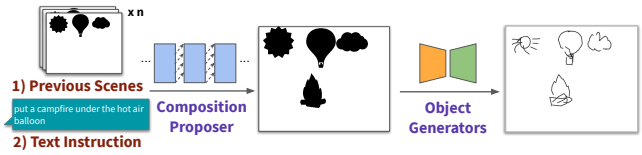

Figure 2: Overall Architecture of Scones. Scones takes a two-stage approach towards generating and modifying sketched scenes based on users' instructions.

3.1 Composition Proposer

The Composition Proposer 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 composition proposal as a sequence modeling task. We use a decoder-only Transformer model architecture similar to GPT-2 [23], a recent deep-learning-based model with high performance.

To produce the output scene S_i at turn i, the Composition Proposer 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 consists of words and punctuation marks.

We represent each object o as a 102-dimensional vector $o = [\mathbb{1}_s, \mathbb{1}_e, e^{(o)}, e^{(u)}, e^{(s)}, e^{(f)}, x, y]$. 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[1] 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 objects. $e^{(s)}$ 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, described in detail in Section 4.1. For each text token t, we use a 300-dimensional GLoVe vector trained on 42B tokens from the Common Crawl dataset [22] 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 adopt o and t to this representation by simply padding additional dimensions in the representations with zeros as shown in Equation 1.

$$o'_{i,j} = [o_{i,j}, \vec{0}_{(300)}] \quad t'_{i,j} = [\vec{0}_{(102)}, t_{i,j}] \tag{1}$$

We interleave text instructions and scene objects chronologically to form a long sequence $[C_{(i-n)}, S_{(i-n)}, \ldots, C_{(i-1)}, S_{(i-1)}, C_i]$ 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 multimodal 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 2 and in Figure 3.

$$S_i = [o_{(i,1),\ldots,(i,l)}] = \textbf{Transformer}([o'_s, o'_{(i-n,1)}, \ldots o'_{(i-n,l_{(i-n)})}, o'_e, t'_{(i-n,1)}, \ldots, t'_{(i-n,m_{(i-n)})}, \ldots, t'_{(i,1)}, \ldots t'_{(i,l_i)}, o'_s]) \tag{2}$$

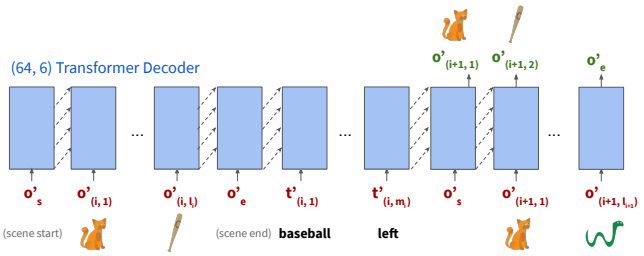

Figure 3: The Scene Layout Generation Process using the Transformer Model of the *Composition Proposer*.

[1] 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.

3.2 Object Generators

Since the outputs of the Composition Proposer 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 Generators*. We adopt 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 $q_{1\ldots h}$. The strokes are encoded using the *Stroke-5* format [7]. 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 moves 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 Generators generate sketches with appearances that follow the object specifications generated by the Composition Proposer. For each object sketch, we compute the aspect ratio $r = \frac{\Delta y}{\Delta x}$ 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 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 Composition Proposer) to generate sketches with poses matching the Composition Proposer's object representations.

$$m_{(x,y)} = \begin{cases} 1 & \text{if } py_{xmax} \geq x \geq py_{xmin}, \text{or;} \\ 1 & \text{if } px_{ymax} \geq y \geq px_{ymin} \\ 0 & \text{otherwise} \end{cases} \quad (3)$$

We adopt the Variational-Autoencoder(VAE)-based conditional variant of Sketch-RNN to enable generating and editing of sketch objects. Our adopted conditional Sketch-RNN encodes input sketches with a Bi-directional LSTM to a latent vector z. The Hyper-LSTM decoder then recreates sketch strokes $q'_{1\ldots h}$ from z, and m, r described above during training, as defined in Equation 4 and shown in Figure 4. Since the latent space is also trained to match a multivariate Gaussian distribution, the Object Generator can support sketch generation when the objects are first added to the scene by randomly sampling $z \sim N(0,1)^{128}$.

$$q'_{1\ldots h} = \textbf{Sketch-RNN Decoder}([m, r, z]), z \sim N(0,1)^{128}$$
$$z = \textbf{Sketch-RNN Encoder}(q_{1\ldots h}) \quad (4)$$

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. 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 4, and are described in the Sketch-RNN paper [7].

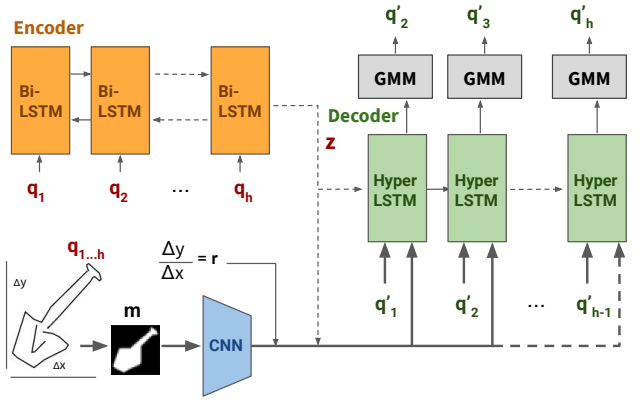

Figure 4: Sketch-RNN Model Architecture of the Object Generators.

4 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.

4.1 CoDraw Dataset

We used the CoDraw dataset to train the Composition Proposer 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 9993 sessions of conversation records, scene modifications, and ground-truth scenes.

Using this dataset, we trained the Composition Proposer 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:

$$L_{cm} = L_c + \lambda_{\textbf{sub}} L_{\textbf{sub}} + \lambda_{\textbf{flip}} L_{\textbf{flip}} + \lambda_{\textbf{size}} L_{\textbf{size}} + \lambda_{xy} L_{xy} \quad (5)$$

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.

4.2 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. We trained our 34 Object Generators on 34 categories of Quick, Draw! data to create sketches of individual sketched 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 75000/2500/2500 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}:

$$L_s = \lambda_{KL} L_{KL} + L_R \tag{6}$$

The KL loss L_{KL} is the KL 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 strokes under the GMM and a categorical distribution parametrized by the model. We refer interested readers to a detailed description of L_s in the original Sketch-RNN paper [7]. 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 at a rate of 0.99995. The models were also trained with gradient clipping of 1.0.

5 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 various stages in Scones.

5.1 Composition Modification State-of-the-art

To evaluate the output of the Composition Proposer 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 paper [15]. 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 a scene composition) 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 Composition Proposer on the CoDraw test set, we achieved an average similarity metric of 3.55. This exceeded existing state-of-the-art performance (Table 1) on the iterative scene authoring task using replayed text instructions from CoDraw.

Table 1: Performance of Various Models on CoDraw Task

Teller	Drawer	Similarity ↑ (out of 5)
Script	**Scones**	**3.55**
Script	Neural Network [15]	3.39
Script	Nearest-Neighbour [15]	0.94
Script	Human	**3.83**

To provide an illustrative example of our Composition Proposer's output on this task, we visualize two example scenes generated from the CoDraw validation set in Figure 5. In the scene a), the Composition Proposer 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 note the Composition Proposer 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 unchanged while other attributes of the scenes were modified.

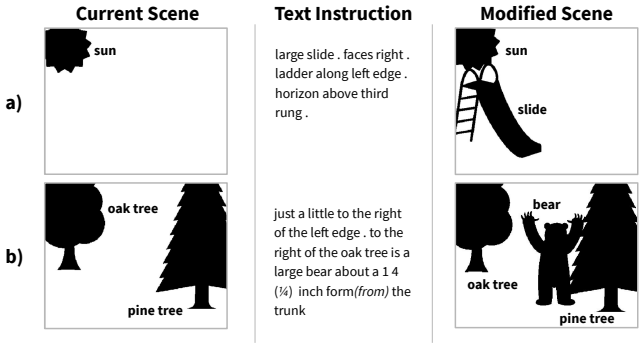

Figure 5: Example Scenes for the Scene Layout Modification Task. The Composition Proposer was able to achieve state-of-the-art performance for modifying object representations in scene compositions.

5.2 Sketches with Clip Art Objects as Mask and Ratio Guidance

The Object Generators are designed to generate sketches which 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 3.1. We demonstrate the Object Generator's performance in two important scenarios that allow Scones to adapt to specific pose and subclass contexts.

5.2.1 Generating objects for closely related classes. While the Composition Proposer classifies objects as one distinct class out of 58, some of these classes are closely related and are not differentiated by the Object Generators. In these cases, object masks can be used by an Object Generator to effectively disambiguate the desired output subclass. For instance, the Composition Proposer 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 Generator 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 Generator to generate appropriate tree objects (by sampling $z \sim N(0, 1)^{128}$). The Object Generator was able to sketch trees with configurations corresponding to input masks from clip art objects (Figure 6). The generated sketches for pine trees were pointy; for apple trees, had round leaves; and for oak trees, had curvy edges.

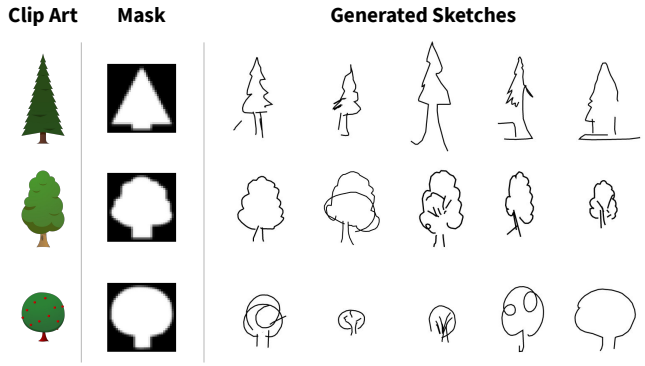

Figure 6: Sketch Generation Results of Trees Conditioned on Masks. The Object Generator was able to sketch trees of three different classes based on mask and aspect ratio inputs.

5.2.2 Generating objects with direction-specific poses. The Composition Proposer can specify the horizontal orientation of the objects (pointing left or right). As such, the Object Generators are required to sketch horizontally asymmetric objects (e.g., racquets, airplanes) with a specific pose to follow users' instructions. We show the ability of Object Generators to produce racquets at various orientations in Figure 7. The generated racquet sketches conformed to the orientation of the mask, facing the specified direction at similar angles.

5.3 Complete Sessions with Composition Layouts and Sketches

We show the usage of Scones in six turns of conversation from multiple sessions in Figure 8 and Figure 1. 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

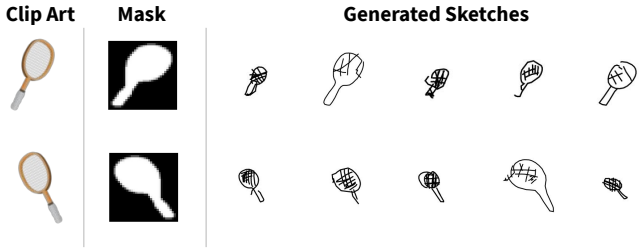

Figure 7: Sketch Generation Results of Racquets Conditioned on Masks. The Object Generator was able to sketch racquets at two orientations consistent to the masks.

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 "flip the plane to point to the right instead". This flipping action demonstrates the Object Generator's ability to generate objects with the require 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, and basketball under the tree in session b).

Moreover, we discover that Scones was able to handle more advanced instructions, such as generating multiple objects at once. In Figure 1, other than basic enlarging and moving commands for the pizza and the helicopter, Scones was able to sketch both the sun and the cloud onto the scene, at positions satisfying the first instruction. It was also able to sketch a table directly under the pizza with the instruction 'put the pizza on a table'.

5.4 Interpreting Transformer's Attention Maps

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

The Transformer model in the Composition Proposer uses masked self-attention to attend to scene objects and instructions from previous time steps most relevant to generating an object specification at the current turn. 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. In the second turn, the user instructed the model to place a slide facing right under the plane. The model similarly attended to the "slide" token the most, while also significantly attended to the "under", and "plane" text tokens, and the plane object, which are useful for situating the slide object at the desired location relative to an existing airplane object (Figure 10).

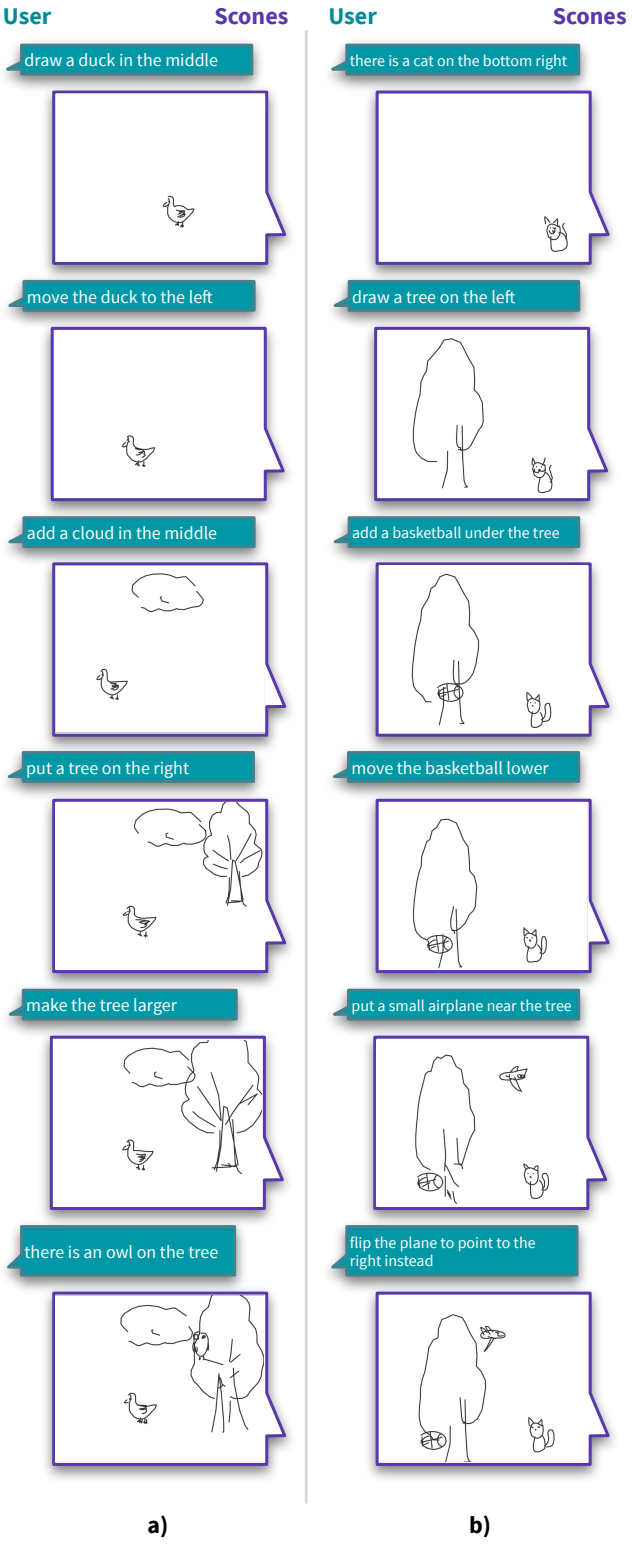

Figure 8: Complete Sketching Sessions with Scones curated by the authors.

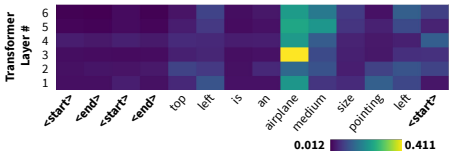

Figure 9: Attention Map of the Transformer across Object and Text Tokens for the Generation of an Airplane, the First Object in the Scene.

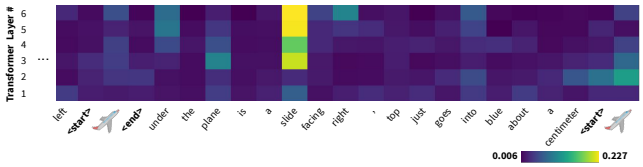

Figure 10: Attention Map of the Transformer across Object and Text Tokens for the Generation of Slide in the Second Turn of Conversation. We observed that the Transformer model attended to the corresponding words and objects that describe objects in the scene related to the newly generated 'slide' object.

These attention weights could 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 users request unsupported classes, such as a 'sandwich' or 'parrot' (Figure 11), 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.

Figure 11: Attention Map of the Transformer for Text Instructions that Specifies Unseen Objects.

6 EXPLORATORY 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.

6.1 Method

Participants 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 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.

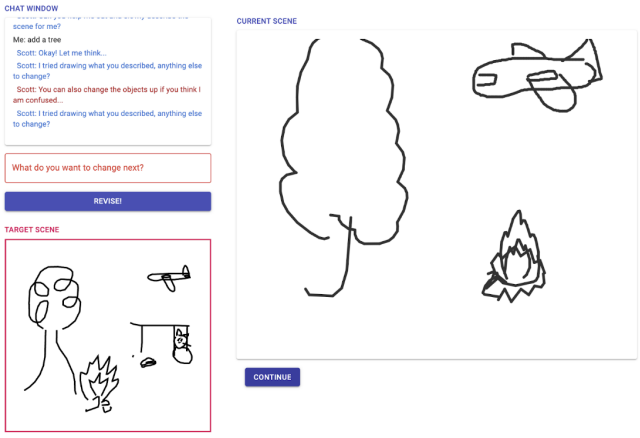

Figure 12: Screenshot of Scones's Evaluation User Interface.

Participants were first shown a short tutorial describing the canvas, chat interface, and target scene in the Scones interface (Figure 12), and were asked to give simple instructions in the chat window to recreate the target scene. Only two sample instructions were given in the background image of 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 at the sketch and enjoyment of the system, and open-ended feedback on the system.

6.2 Results

6.2.1 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 13). By reviewing qualitative feedback and instructions to Scones, we observe that many instances of low enjoyment (score ≤ 2) come from class confusion in the target scene sketch. 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.

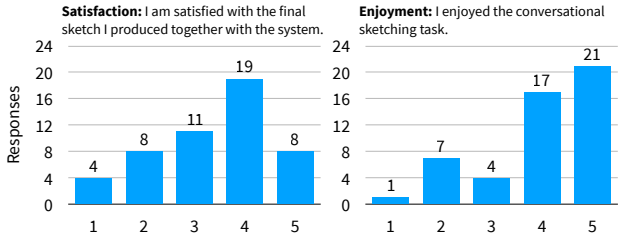

Figure 13: Survey Results from User Sessions with Scones.

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 5.4 could be used when an unrecognized class is detected to prompt users to provide sketch strokes with a corresponding label.

6.2.2 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 14.

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

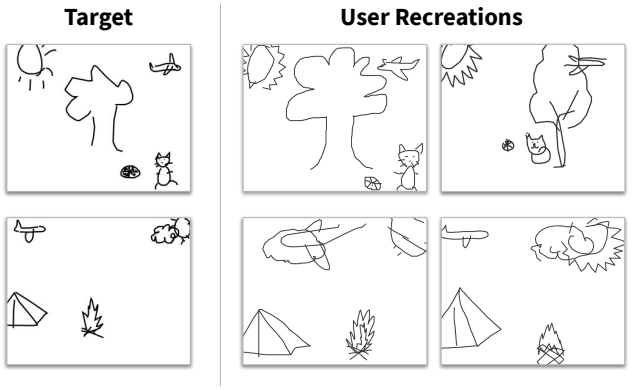

Figure 14: Recreated Scenes during the User Study. Users combined Scones-generated outputs with their own sketch strokes to reproduce the target scenes presented to them.

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 for these hurdles would be to introduce more detailed tutorial content for a wider deployment of Scones.

6.2.3 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) [31] to collect sketch critiques (natural language specified modifications to an input sketch to match a target sketch) and user-generated sketching 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).*

6.3 Participants' Feedback for Improving Scones

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

6.3.1 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 [15], in which users had a restricted view of the canvas, resulting in limited relative spatial instructions. This limitation is discussed further in Section 7.3.

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.

6.3.2 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 learn about system capabilities. A truly *conversational* interface with a sketching support tool could pave the way for advanced mixed-initiative collaborative design tools.

7 LIMITATIONS

7.1 Underspecified Masks

While mask conditioning effectively guides the Object Generators in creating sketches with desired configurations, they can be underspecified for the poses exhibited by objects of some classes. As shown in Figure 15, the mask of a 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.

7.2 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 Composition Proposer. Future work should explore models conditioned on continuous representations of classes and poses from word embeddings for a flexible number of object

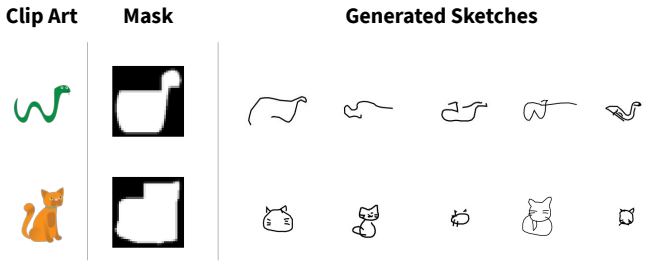

Figure 15: Sketches Generated by the Object Generator with Underspecified Masks of the Snake and Cat Classes.

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 Composition Proposer could output a continuous embedding that contains objects' class, pose, and stylistic information to fully support a wide range of sketches.

7.3 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.

8 DISCUSSION AND FUTURE WORK

8.1 Scones-supported Games With a Purpose (GWAP)

While Scones demonstrates a plausible system architecture for composing scenes of limited scenarios, the limitations of object classes and modification capability are mainly due to the lack of large-scale datasets of multimodal sketch modification. These datasets are considered to be difficult to collect due to the sketching skill requirement of crowdworkers [36]. We believe Scones can be used as a gateway towards creating such a dataset. By decomposing each scene into object components, crowdworkers would only need to sketch a single object in context, which was shown to be possible from the Quick, Draw! dataset. While the models currently are restricted to handling objects of a small set of poses and aspect ratios, we can prompt users to generate these objects freely, in turn expanding the variety of sketches in our dataset. Since Scones can automate sketch generation for other parts of the scene, this significantly improves the scalability of the game and makes it possible for Scones to be used as a data-collection system. Moreover, we also collect text instructions that could help to build a critique model for providing text-based sketch modification suggestions to users.

8.2 Application to Professional Domains

We believe the system architecture of Scones can be applied to professional domains if object and scene data for these domains become available. For instance, Scones could participate in the UI/UX design process by iteratively modifying UI design sketches according to design critique. To enable this interaction, we could consider complete UI sketches as 'scenes' and UI components as 'scene objects'. Scones could be trained on this data along with text critiques of UI designs to iteratively generate and modify UI mockups from text. While datasets of UI layouts and components, such as those presented in Swire [10], suggest this as a near possibility, this approach may generalize to other domains as well, such as industrial design. Future work in adapting our system to new domains could benefit from fine-tuning pre-trained models in the current implementation of Scones.

9 CONCLUSION

In this paper, we introduced Scones, a machine-learning-driven system that generates scenes of sketched objects from text instructions. Scones consists of two stages, a *Composition Proposer* and a set of *Object Generators*, to compose sketched scenes of multiple objects that encode semantic relationships specified by natural language instructions. We establish state-of-the-art performance for the text-based scene modification task, and introduce mask conditioning as a novel component in the Object Generators, which enables finer-grained control of object poses in sketch output. With Scones, users can interactively add and modify objects in sketches with inferred operations (e.g., transforming, moving, reflecting).

In an exploratory user evaluation, we found participants enjoyed working with Scones and were satisfied with the output sketches it produced. Most participants contributed hand-drawn sketches during the activity, motivating the potential for Scones to be used as a Game With A Purpose (GWAP) for collecting end-to-end sketching critique and modification data.

We see Scones as a step towards design support interfaces with tight human-in-the-loop coupling, providing an entirely new means for creative expression and rapid ideation. We are excited to continue designing for this future of design, art, and engineering.

ACKNOWLEDGMENTS

The authors would like to thank Douglas Eck, Heiga Zen, Yingtao Tian, Nathaniel Weinman, J.D. Zamfirescu-Pereira, Philippe Laban, David M. Chan, Roshan Rao, and all anonymous reviewers for providing valuable comments on the system and the paper. The authors would also like to thank all workers on Amazon Mechanical Turk for participating in the study and providing valuable feedback and comments on the system. This research is supported by Google Cloud.

REFERENCES

[1] Aishwarya Agrawal, Jiasen Lu, Stanislaw Antol, Margaret Mitchell, C. Lawrence Zitnick, Devi Parikh, and Dhruv Batra. 2017. VQA: Visual Question Answering. *Int. J. Comput. Vision* 123, 1 (May 2017), 4–31. https://doi.org/10.1007/s11263-016-0966-6

[2] Jeffrey P. Bigham, Chandrika Jayant, Hanjie Ji, Greg Little, Andrew Miller, Robert C. Miller, Robin Miller, Aubrey Tatarowicz, Brandyn White, Samual White, and Tom Yeh. 2010. VizWiz: Nearly Real-time Answers to Visual

Questions. In *Proceedings of the 23nd Annual ACM Symposium on User Interface Software and Technology (UIST '10)*. ACM, New York, NY, USA, 333–342. https://doi.org/10.1145/1866029.1866080

[3] Alex J. Champandard. 2016. Semantic Style Transfer and Turning Two-Bit Doodles into Fine Artworks. *CoRR* abs/1603.01768 (2016). arXiv:1603.01768 http://arxiv.org/abs/1603.01768

[4] Mathias Eitz, James Hays, and Marc Alexa. 2012. How Do Humans Sketch Objects? *ACM Trans. Graph. (Proc. SIGGRAPH)* 31, 4 (2012), 44:1–44:10.

[5] Fartash Faghri, David J Fleet, Jamie Ryan Kiros, and Sanja Fidler. 2018. VSE++: Improving Visual-Semantic Embeddings with Hard Negatives. (2018). https://github.com/fartashf/vsepp

[6] Arnab Ghosh, Richard Zhang, Puneet K. Dokania, Oliver Wang, Alexei A. Efros, Philip H. S. Torr, and Eli Shechtman. 2019. Interactive Sketch & Fill: Multiclass Sketch-to-Image Translation. In *Proceedings of the IEEE international conference on computer vision*.

[7] David Ha and Douglas Eck. 2018. 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*. https://openreview.net/forum?id=Hy6GHpkCW

[8] Seunghoon Hong, Dingdong Yang, Jongwook Choi, and Honglak Lee. 2018. Inferring Semantic Layout for Hierarchical Text-to-Image Synthesis. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 7986–7994.

[9] Forrest Huang and John F. Canny. 2019. 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)*. Association for Computing Machinery, New York, NY, USA, 209–220. https://doi.org/10.1145/3332165.3347878

[10] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. Swire: Sketch-Based User Interface Retrieval. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems (CHI '19)*. Association for Computing Machinery, New York, NY, USA, Article Paper 104, 10 pages. https://doi.org/10.1145/3290605.3300334

[11] J. Johnson, B. Hariharan, L. v. d. Maaten, L. Fei-Fei, C. L. Zitnick, and R. Girshick. 2017. CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1988–1997. https://doi.org/10.1109/CVPR.2017.215

[12] Jonas Jongejan, Henry Rowley, Takashi Kawashima, Jongmin Kim, and Nick Fox-Gieg. 2016. The Quick, Draw! - A.I. Experiment. https://quickdraw.withgoogle.com/

[13] Wang-Cheng Kang, Chen Fang, Zhaowen Wang, and Julian McAuley. 2017. Visually-Aware Fashion Recommendation and Design with Generative Image Models. In *2017 IEEE International Conference on Data Mining (ICDM)*. IEEE, 207–216.

[14] Rubaiat Habib Kazi, Fanny Chevalier, Tovi Grossman, and George Fitzmaurice. 2014. Kitty: Sketching Dynamic and Interactive Illustrations. In *Proceedings of the 27th Annual ACM Symposium on User Interface Software and Technology (UIST '14)*. ACM, New York, NY, USA, 395–405. https://doi.org/10.1145/2642918.2647375

[15] Jin-Hwa Kim, Nikita Kitaev, Xinlei Chen, Marcus Rohrbach, Byoung-Tak Zhang, Yuandong Tian, Dhruv Batra, and Devi Parikh. 2019. CoDraw: Collaborative Drawing as a Testbed for Grounded Goal-driven Communication. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Florence, Italy, 6495–6513. https://doi.org/10.18653/v1/P19-1651

[16] James A. Landay. 1996. SILK: Sketching Interfaces Like Krazy. In *Conference Companion on Human Factors in Computing Systems (CHI '96)*. ACM, New York, NY, USA, 398–399. https://doi.org/10.1145/257089.257396

[17] Gierad P. Laput, Mira Dontcheva, Gregg Wilensky, Walter Chang, Aseem Agarwala, Jason Linder, and Eytan Adar. 2013. PixelTone: A Multimodal Interface for Image Editing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (CHI '13)*. ACM, New York, NY, USA, 2185–2194. https://doi.org/10.1145/2470654.2481301

[18] Mengtian Li, Zhe Lin, Radomir Mech, Ersin Yumer, and Deva Ramanan. 2019. Photo-Sketching: Inferring Contour Drawings from Images. In *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 1403–1412.

[19] Alex Limpaecher, Nicolas Feltman, Adrien Treuille, and Michael Cohen. 2013. Real-time Drawing Assistance Through Crowdsourcing. *ACM Trans. Graph.* 32, 4, Article 54 (July 2013), 8 pages. https://doi.org/10.1145/2461912.2462016

[20] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. 2019. Semantic Image Synthesis with Spatially-Adaptive Normalization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[21] Amy Pavel, Dan B. Goldman, Björn Hartmann, and Maneesh Agrawala. 2015. SceneSkim: Searching and Browsing Movies Using Synchronized Captions, Scripts and Plot Summaries. In *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology (UIST '15)*. ACM, New York, NY, USA, 181–190. https://doi.org/10.1145/2807442.2807502

[22] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing (EMNLP)*. 1532–1543. http://www.aclweb.org/anthology/D14-1162

[23] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language Models are Unsupervised Multitask Learners. (2019).

[24] Andre Ribeiro and Takeo Igarashi. 2012. Sketch-Editing Games: Human-Machine Communication, Game Theory and Applications. In *Proceedings of the 25th Annual ACM Symposium on User Interface Software and Technology (UIST '12)*. Association for Computing Machinery, New York, NY, USA, 287–298. https://doi.org/10.1145/2380116.2380154

[25] Patsorn Sangkloy, Nathan Burnell, Cusuh Ham, and James Hays. 2016. The Sketchy Database: Learning to Retrieve Badly Drawn Bunnies. *ACM Transactions on Graphics (proceedings of SIGGRAPH)* (2016).

[26] Adam Santoro, David Raposo, David G Barrett, Mateusz Malinowski, Razvan Pascanu, Peter Battaglia, and Timothy Lillicrap. 2017. A simple neural network module for relational reasoning. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 4967–4976. http://papers.nips.cc/paper/7082-a-simple-neural-network-module-for-relational-reasoning.pdf

[27] Edward Shen, Henry Lieberman, and Francis Lam. 2007. What Am I Gonna Wear?: Scenario-oriented Recommendation. In *Proceedings of the 12th International Conference on Intelligent User Interfaces (IUI '07)*. ACM, New York, NY, USA, 365–368. https://doi.org/10.1145/1216295.1216368

[28] Alane Suhr, Mike Lewis, James Yeh, and Yoav Artzi. 2017. A Corpus of Natural Language for Visual Reasoning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Vancouver, Canada, 217–223. https://doi.org/10.18653/v1/P17-2034

[29] Kristen Vaccaro, Tanvi Agarwalla, Sunaya Shivakumar, and Ranjitha Kumar. 2018. Designing the Future of Personal Fashion. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems (CHI '18)*. ACM, New York, NY, USA, Article 627, 11 pages. https://doi.org/10.1145/3173574.3174201

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.). Curran Associates, Inc., 5998–6008. http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf

[31] Luis von Ahn and Laura Dabbish. 2008. Designing Games with a Purpose. *Commun. ACM* 51, 8 (Aug. 2008), 58–67. https://doi.org/10.1145/1378704.1378719

[32] Sida I. Wang, Percy Liang, and Christopher D. Manning. 2016. Learning Language Games through Interaction. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, 2368–2378. https://doi.org/10.18653/v1/P16-1224

[33] Rowan Zellers, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. From Recognition to Cognition: Visual Commonsense Reasoning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[34] Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei A. Efros. 2016. Generative Visual Manipulation on the Natural Image Manifold. In *Proceedings of European Conference on Computer Vision (ECCV)*.

[35] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*.

[36] Changqing Zou, Qian Yu, Ruofei Du, Haoran Mo, Yi-Zhe Song, Tao Xiang, Chengying Gao, Baoquan Chen, and Hao Zhang. 2018. SketchyScene: Richly-Annotated Scene Sketches. In *ECCV*. Springer International Publishing, 438–454. https://doi.org/10.1007/978-3-030-01267-0_26