Titre: Title:	Investigating User Interface Design Generation Techniques for Designer Inspiration
Auteur: Author:	Mohammad Amin Mozaffari
Date:	2022
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Mozaffari, M. A. (2022). Investigating User Interface Design Generation Techniques for Designer Inspiration [Mémoire de maîtrise, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/10221/

URL de PolyPublie: PolyPublie URL:	https://publications.polymtl.ca/10221/
Directeurs de recherche: Advisors:	Jinghui Cheng
Programme: Program:	Génie informatique

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Investigating User Interface Design Generation Techniques for Designer Inspiration

MOHAMMAD AMIN MOZAFFARI
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Février 2022

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

Investigating User Interface Design Generation Techniques for Designer Inspiration

présenté par **Mohammad Amin MOZAFFARI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Maxime LAMOTHE, président
Jinghui CHENG, membre et directeur de recherche
Guillaume-Alexandre BILODEAU, membre

DEDICATION

To Shaghayegh and My dear family
I am always proud to have you beside me
. . .

ACKNOWLEDGEMENTS

I owe a great many thanks to a great many people who helped me during my journey in Polytechnique Montreal.

First of all, I would like to express my deep sense of gratitude to my research director Prof. Jinghui Cheng. Definitely, this study would not be completed without his support and advice. Besides his kindness, he was always available to help in my study and answer all of my questions. I am deeply thankful for him to have me in HCDLab.

Second of all, I would like to say, I am very fortunate to have such a family beside me. They always have my back in any stressful situation which I face. I will never forget your supports, especially in these two years.

Third of all, I would like to give my warmest thanks to the HCDLab members. I was not lucky to talk and gather with them much often, because of the Covid-19, but I wish our paths will cross again soon.

At the end, I express my gratitude to the jury members who reviewed this thesis.

RÉSUMÉ

L'inspiration est l'une des étapes essentielles du processus de conception. La créativité et l'originalité du travail sont des facteurs importants qui dépendent fortement de l'étape de l'inspiration. Les designers recherchent généralement des exemples existants avant de commencer la conception pour avoir des idées sur les travaux récents et s'en inspirer respectivement. Il existe un nombre limité d'outils ou de sites web pour répondre aux besoins d'inspiration des designers d'interfaces utilisateur. De plus, ces outils peuvent causer certains problèmes de conception comme la dérive ou la fixation de la conception.

Nous proposons une approche basée sur StyleGAN pour résoudre ces problèmes. Le modèle proposé est un Réseau Adversarial Génératif basé sur le style qui est capable de générer un design d'interface utilisateur aléatoire ou un design d'interface utilisateur pertinent pour l'image d'entrée. Nos résultats montrent que le modèle génère non seulement des designs pertinents mais aussi des images diverses. Comme nous le savons, ces deux facteurs (pertinence et diversité) sont très importants pour l'inspiration. Dans notre étude auprès de praticiens UI/UX évaluant la facilité d'utilisation et la performance de notre modèle, nos participants ont déclaré que ce modèle pourrait être une alternative viable aux outils d'inspiration actuels ou aux sites web pour les designers d'interface utilisateur.

Dans la deuxième étape de notre étude, nous avons étendu notre travail en détectant les composants des images générées. La motivation de cette partie de l'étude est d'améliorer la qualité des images générées et de faciliter les méthodes d'édition et d'automatisation de la conception de l'interface utilisateur. Tout d'abord, nous avons entraîné Yolov5 en utilisant le jeu de données VINS et nos résultats montrent que le modèle reconnaît la plupart des composants des images complètes et des images générées. Cependant, certains composants des images générées, comme l'image et l'icône, n'ont pas pu être détectés avec une précision acceptable. Ensuite, en utilisant la technique d'apprentissage par transfert, nous avons affiné le modèle préliminaire pour résoudre le problème de la détection des composants dans les images générées. Les résultats montrent que nous avons atteint une performance satisfaisante pour détecter les éléments de l'interface utilisateur dans les images générées.

ABSTRACT

Inspiration is one of the substantial steps in the user interaction design process. The creativity and originality of the design work are important factors that are highly dependent on the inspiration step. Designers usually look for existing examples before starting the design to be familiar with and get ideas from the recent works. However, there are a limited number of tools exist to fulfill the user interface designers' needs in inspiration. In addition, existing design example retrieval techniques may cause design problems such as design drift or design fixation.

We propose a StyleGAN-based approach to address these issues. The proposed technique is a style-based Generative Adversarial Network that is able to generate a random user interface design or a user interface design relevant to the input image. Our results show that the technique not only generates the relevant designs but also diverse images. In our user study with UI/UX practitioners evaluating the usability and the performance of our technique, our participants stated that this technique could be a feasible alternative for the current design example retrieval techniques for user interface designers.

As the second step of our study, we extended our work by detecting the components of the generated user interface images. The motivation of this part of the study is to enhance the quality of the generated images and to facilitate the ways of UI editing and design automation. First, we trained Yolov5 using the VINS dataset and our results demonstrate that the model recognizes most of the components of both full-fledged images and generated images. However, some of the components in generated images like Image and Icon, could not be detected with acceptable accuracy. Second, by using the transfer learning technique, we fine-tuned the preliminary model to address the issue of detecting components in the generated images. The results illustrated that we achieved a satisfactory performance to detect the UI elements in the generated images.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

UI	User Interface
UX	User Experience
GAN	Generative adversarial network
IoU	Intersection over union
FID	Fréchet inception distance
IS	Inception Score

LIST OF APPENDICES

CHAPTER 1 INTRODUCTION

1.1 Problem Statement and Overall Objective

User interface (UI) design plays an important role to convey the designers' and the developers' goals to the users. UIs are one of the inseparable parts of developing mobile or desktop applications and websites. As such, UI designers and developers often spend a significant amount of time creating attractive user interface designs for the users.

Searching for user interface designs is a daily task for every UI designer. Studies show that almost every designer looks for examples before starting the design process [1]. They search for examples for a couple of reasons such as getting inspired to design their own product and being updated about the recent designs on different platforms. UI designers may face some challenges during the search for examples before starting their design process; two prominent challenges that the designers experience are design fixation and design drift. Design fixation [2] is defined as being biased to the examples that the designers have been exposed to; this could diminish the originality and novelty of the final work. Design drift [3] is defined as, by exposing many examples before starting the design work, designers may shift the work from the original focus and forget about the initial goals and guidelines of their work [4]. Hence, while inspiration is one of the essential and common steps to be creative while keeping the originality of the design, it is not trivial work and requires careful attention [5]. To aggravate the problem, there is limited tool support to facilitate the design inspiration step while addressing design fixation and design drift.

In this study, we aim to take a step forward to explore techniques that can address the above-mentioned problems in order to help designers in their design inspiration process. In order to achieve these goals, we first propose a StyleGAN-based approach, a style-based generative adversarial network, trained on a large-scale dataset of UI images to generate relevant and diverse design examples. Traditional generative adversarial networks(GAN) are generative models using deep learning methods [6] which comprise two sub-networks called the generator and the discriminator. During the training phase of GANs, the generator tries to fool the discriminator, which means the goal of the model is to try to generate a fake output and, then the discriminator classifies it as a real one. StyleGan is driven from traditional GANs by alternating a new generator architecture while keeping the discriminator [7]. Besides traditional GANs capabilities, Style-Based Generative Adversarial Network (StyleGAN) is not only able to generate high-quality images, but also it has control over the style of the input image and can feed the style at different levels of the generation of the output. By using

the StyleGAN-based approach, we are able to generate new design examples with regards to the existing preliminary designs that the designers work on; in other words, the model can take one preliminary design from the designer, as well as other input UI images that the designer wants to have their styles in the results, then the model can apply the styles at different resolution levels. The outcome is a set of images related to the preliminary design, while having the desired styles on them.

After developing and evaluating our StyleGAN-based approach with UI design practitioners, we found that if we are able to enhance the quality of the generated images, the results would be much more useful for designers. In this work, we aim to detect the UI components from the generated images, with the goal of enhancing their qualities and laying the foundation for them to be more useful in the designers' works. Because in this way, we can modify or justify the noisy or inappropriate components to make the whole design visually better. Enhancing the quality of the images is not the only application of UIs components detection; there are several other applications such as design automation [8], UI testing [9], code generation [10, 11], and UI editing [12]. After searching for a proper object detector to address our problem in detecting the UIs components, we decided to use Yolov5. Yolos' object detectors are well-known because of their accuracy and speed in detection.

1.2 Research Questions

When investigating and evaluating our work, we pose the following research questions:

- **RQ1**: How do StyleGAN-based approaches compare with random examples and similarity-based examples in quantitative metrics indicating the ability of inspiration support?
- **RQ2**: How do UI/UX practitioners perceive the output of StyleGAN-based approaches in comparison to random examples and similarity-based examples?
- **RQ3**: How does Yolov5, trained with full-fledged UIs, perform on detecting the UI components on generated images?
- **RQ4**: How does fine-tuned Yolov5 with generated images perform on detecting the UI components on generated images?

To answer RQ1 and RQ2, we developed the StyleGAN-based approach which extends the StyleGAN architecture trained on a large-scale dataset including 58,040 screenshots of Android applications. For evaluating our StyleGAN-based approach, we first proposed two quantitative metrics for evaluating the ability of inspiration support of a set of UI images: (1) *similarity* of the images to the input image and (2) *diversity* of the set of UI images. We

found that the StyleGAN-based approach methods provide much more diverse design examples than a similarity-based method, and they provide more similar examples to the input image than a random example selection approach, indicating a balance between diversity and relevance. Through a user study with five professional UI/UX practitioners, we found that the participants perceived the StyleGAN-based approach methods as a viable way to gain inspiration to modify a UI design. We believe that the ideas presented in the StyleGAN-based approach will encourage and influence more research efforts towards the pragmatic use of generative models in the creative, yet constraint, the task of user interface design.

To answer RQ3 and RQ4, we trained the Yolov5 model using the VINS dataset to detect the elements of the UI. The VINS consists of 4800 images of UI screens. In the VINS dataset, there are 257 images of wireframes and the rest are full-fledged UI screenshots on different platforms: Android, iPhone, and websites. We found that the trained Yolov5 can detect almost all of the components of existing UIs with high mean average precision (mAP). This model also was able to detect some of the components of the generated images with satisfactory accuracy compared to literature, except for a few component types such as Image and Icon. So, we fine-tuned the model to improve the performance of detecting the elements of the generated images. To this end, we manually annotated 200 generated UIs to fine-tune the preliminary model for transfer learning. Our results show that not only can the final model detect the UI elements of existing full-fledged UI images and wireframes but also the elements of the generated images with an acceptable mAP.

1.3 Thesis Plan

The structure of the thesis is as follows. In Chapter 2, we review the literature related to our study; it comprises four different parts, regarding each step of the study. In Chapter 3, we describe the StyleGAN-based approach and the evaluation studies; particularly, we propose the model's architecture, describe the training phase, and explain the quantitative evaluation and the user evaluation. In Chapter 4, we explain the UI component detection model and compare the results of different conditions of training and testing. In Chapter 5, we bring a summary of our work, discuss its limitations, and layout future directions.

CHAPTER 2 LITERATURE REVIEW

Our work is most closely related to previous studies that focused on (1) design inspiration, (2) techniques for managing design artifacts, (3) generative machine learning models and StyleGAN, and (4) UI components detection in particular. We briefly review each group of literature in the following sections.

2.1 Design Inspiration

Thrash et al. [13, 14] were among the first who empirically studied inspiration as a psychological construct. They have identified that human inspiration is categorized by motivation (i.e. goal-oriented self-initiation), evocation (i.e. an impulsive reaction to stimuli), and transcendence (i.e. feeling of gaining superior ideas that are "more elegant or novel than those generated willfully").

The problem of inspiration has been then investigated in a wide design community, beyond user interaction design. These previous studies were mostly conducted from the perspectives of how designers get access to and use existing design artifacts. For example, focusing on knitwear design, Eckert and Stacey [15] have identified that designers used a wide variety of sources of inspiration, including artifacts with intriguing shapes, patterns, and colors, as well as their own previous design, to not only concretize the otherwise abstract design ideas but also to create "shortcuts" to help them recall and communicate using these visuospatial "chunks;" i.e. inspirational sources served as "a language of design."

In the HCI community, researchers have explored ways industrial and user interaction designers get inspired by existing design artifacts. For example, Bonnardel [16] has identified that, in the context of product design, "the emergence of new ideas results from analogy-making." From an in-depth interview study with web, graphic, and product designers, Herring et al. [4] identified the common approaches they used and the challenges they faced when they retrieve, store, and disseminate design examples. Based on a glossary of design ideation methods, Gonçalves et al. [5] have also conducted a survey with students and professional industrial designers to understand their sources and methods of inspiration. They found that, compared to students, professional industrial designers adopted a wider variety of inspirational approaches.

The literature has also identified several problems and issues about the common inspirational methods. Notably, many studies have pointed to the fact that over-exposure to a

homogeneous set of design examples may result in "design fixation," which will limit the inspirational power of the examples and result in less creative ideas [3, 4, 17]. Particularly, Marsh et al. [3] identified that exposure to a greater number of examples that share common critical characteristics would increase the fixation issue. The timing of the example exposure can affect the quantity and quality of ideas as well. For example, Siangliulue et al. [18] found that receiving examples when their participants seemed to have run out of ideas have allowed the participants to produce a larger number of ideas, whereas explicitly requesting examples when needed have allowed the participants to produce more novel ideas. In this study, we build on this body of literature to investigate techniques for supporting effective inspiration in user interface design, while avoiding design fixation.

2.2 Managing UI Design Artifacts

While we are aware that there is abundant recent work focused on extracting UI elements, including their hierarchical design information, from design artifacts such as mockups (e.g. [19–24]), they are not directly related to our current work, which focuses on providing inspirational design examples. So we omit the detailed review of this body of literature here. In this section, we focus on reviewing related work that investigates the management of UI design artifacts for the purpose of design inspiration.

Towards this direction, some previous studies have focused on techniques that retrieve UI design examples based on an input UI screenshot. For example, Lee et al. [25] have proposed an "Adaptive Ideas" web design tool, which allows users to view examples similar to their current design work. In their tool, the users could control the dimensions (including background color, primary font, number of columns, and visual density) used to compare design similarities. Similarly, Behrang et al. [26] proposed a technique that combined keyword search and image-based search to retrieve apps (along with their code) with similar screenshots as an input design. Hashimoto et al. [27] have also introduced a technique that aims to help inexperienced designers retrieve similar design examples based on an input in the form of sketch or wireframe. Ritchie et al. [28] have proposed a design exploration tool that allows its users to query design examples by descriptive text including color keywords or style terms; the tool can also search by style similarity.

The recent development of deep neural networks has enabled more powerful techniques for similarity-based design example retrieval. For example, Huang et al. [29] introduced Swire, a sketch-based neural network-driven technique for retrieving user interface designs. The core component of Swire is a deep convolutional neural network (CNN) [30] that calculates the "embedding" (i.e. a numerical representation) of a design artifact (e.g. a sketch or a

screenshot). Once trained, Swire could retrieve UI design artifacts that are similar to an input sketch or screenshot. More recently, Bunian et al. [31] proposed VINS to retrieve the most structurally similar UI screenshots to the input using object detection models to identify the UI components of the UI screenshots or wireframes. Based on the components and their layout, an image retrieval model helps to find similar UI screenshots in the reference dataset.

Notably, most previous studies relied on similarity when retrieving design artifacts, which can result in design fixation and may hinder the creative design process. Our study addresses this issue by focusing on a generative model-based style change approach that balances the targeted and serendipitous aspects of design artifact retrieval for inspiration.

2.3 StyleGAN and its Application on UI Design

StyleGAN, or Style-Based Generative Adversarial Network [7], extends the traditional GAN [6] architecture on a style-based generator model inspired by the style transfer literature. GANs are generative models using deep learning methods. They typically include two sub-models that are trained at the same time, i.e., a generator and a discriminator. Given an input vector, normally referred to as a latent code, the generator synthesizes an image that resembles particular domain properties of the training dataset. Then a discriminator is used as a binary classifier that predicts if an input image is synthesized by the generator or is from the original training set. Through a back-propagation-based training process [32], the generator learns to produce instances in the high dimensional feature space that are close to the distribution of the original training dataset. Meanwhile, the discriminator learns to classify if an input image is from the original training set or is synthesized by the generator. After training, the generator is saved to generate new samples for the target domain and the discriminator is generally discarded. GANs are often applied for image data and Convolutional Neural Networks (CNNs) [30] are typically used as the generator and discriminator models.

Based on the GAN architecture, StyleGAN proposes a style-based generator that focuses on explicitly transfered 'styles' on an input image at different resolution levels during the synthesis process. Concretely, the input latent code is transformed first into an intermediate vector. After an affine transformation and adaptive instance normalization (AdaIN) [33], the intermediate vector controls the synthesis at each convolutional layer in the generator. StyleGAN can also achieve style mixing by using two distinct latent codes controlling the style at higher and lower convolutional layers respectively. This results in a synthesized image with one latent code dominating its overall features and the other latent code contributing mostly to the details of the image.

Since published, StyleGAN has obtained great attention given its capacity for generating high-resolution and realistic-looking images. The improved version, StyleGAN2 [34] proposed several notable changes to the original StyleGAN approach. By revising the generator architecture and the model training procedure, StyleGAN2 effectively removes the blob-like artifacts commonly found in the images generated by StyleGAN, overcomes the shift-invariance issue during progressive image synthesis (from low resolution to high resolution), and more reliably maps real images into the latent space. For a more detailed explanation of the changes, we refer the interested readers to the work by Karras et al. [34]

The application of GAN for UI design is still in its infancy. The only previous work that used generative models for providing UI examples is a very recent study done by Zhao et al. [35]. They developed a technique to generate UI structures and reused UI components collected from existing mobile apps to fill in the generated structure in order to create UI examples. While interesting, their study only focused on the quality of the generated UIs, in terms of metrics such as color harmony and structure rationality. Additionally, their evaluation was not done with professional designers who are familiar with real-life design practices. Instead, we aim to understand the ability of the generative models in providing inspiration and design support. Our techniques also address style-based design transformation, which is not explored in the literature.

As we discussed earlier, StyleGAN has been improved since its publication. While our work is based on the original StyleGAN, we expect the performance of the StyleGAN-based approach can be enhanced by using more advanced generative models, such as StyleGAN2. Our contribution, however, is not on using the most recent models, but instead on illustrating the potential of applying this line of work for a novel but important problem, i.e., generating design examples with high diversity and relevance for effective inspiration.

2.4 UI Components Detection

Many object detection techniques have been proposed in different domains to detect the desired objects. In the domain of user interface design, researchers also have been investigating techniques that are able to detect the UI elements. Detection of UI components can be used for different objectives in design areas such as UI testing, designing automation, searching for UI, code generation, etc. There are a couple of recent works focused on the detection of UI elements. Chen et al. [36] proposed a hybrid approach in which they split their work to detect text components and non-text components. For detecting the text components, they leveraged the EAST [37] detector which uses a deep learning approach. To detect the non-texts elements, they used traditional computer vision techniques. Additionally, they

developed UIED [12], an online toolkit, to help users to change or edit UIs in an interactive dashboard. Narayanan et al. [38] also proposed two models to detect the elements of hand-drawn sketches: a Cascade RCNN model and the Yolov4 model. They compared the mAP (mean Average Precision) of some models with the goal of UI elements detection of hand-drawn sketches. Compared with the baseline Faster RCNN, they achieved a 38.7% improvement in mAP using the Cascade RCNN model, and almost 50% improvement by using the Yolov4 model. As far as we know, we are the first researchers to try to generate UIs and then detect their components. Generated UIs have different characteristics than full-fledged UI images. They are usually more blurry with less clear borders around the UI components; the generated 'texts' are also only visually similar to texts but are not written in any language. We investigate the performance of object detection models trained to identify those components.

CHAPTER 3 STYLEGAN-BASED APPROACH FOR DESIGN EXAMPLE GENERATION

In this chapter, we explain our StyleGAN-based approach. As we mentioned above, Style based generative adversarial network has more capabilities to produce new images than traditional GANs and also it gives. In the following, we will go through the architecture of the model, describe the process of training the model, and evaluate the capabilities of our StyleGAN-based approach. This chapter is based on a paper that is accepted in the 2022 ACM CHI Conference on Human Factors in Computing Systems.

3.1 Architecture of our StyleGAN-based approach

The interaction between the designer and our approach is initiated when the designer has a preliminary design artifact (e.g., a UI mockup image) at hand, related to their design task. The designer sends this image as an input to the StyleGAN-based system. The system will first encode this image to a latent code (i.e., a high-dimension vector). This latent code will then be used to merge with other latent codes, either randomly generated or obtained from other UI images to synthesize a unique set of new example images. The most representative example images will then be selected and displayed to the designer. Depending on the configuration, the designer can also set to return the real UI screenshots from the database that are most close to the generated results. As such, the architecture of our StyleGAN-based approach is primarily comprised of three components to achieve its key functionality: (1) a latent code search component, (2) a new examples synthesizer, and (3) a representative examples selection component. The overall architecture is illustrated in Figure 3.1. Below, we first describe each of the three major components of the architecture in detail. Then we describe the process used to train the StyleGAN that supports these components.

3.1.1 Component 1: Latent Code Search

The New Examples Synthesizer component of the StyleGAN-based approach relies on a condensed representation of images called latent codes, i.e., high dimensional vectors (512 dimensions in our case). The input images, therefore, need to be first encoded from their original format to the latent space that corresponds to a trained StyleGAN model. This is done through latent code searching, which is built upon the work of StyleGAN-Encoder [39] and relies on a trained StyleGAN model. Concretely, a synthesized image img_s is first

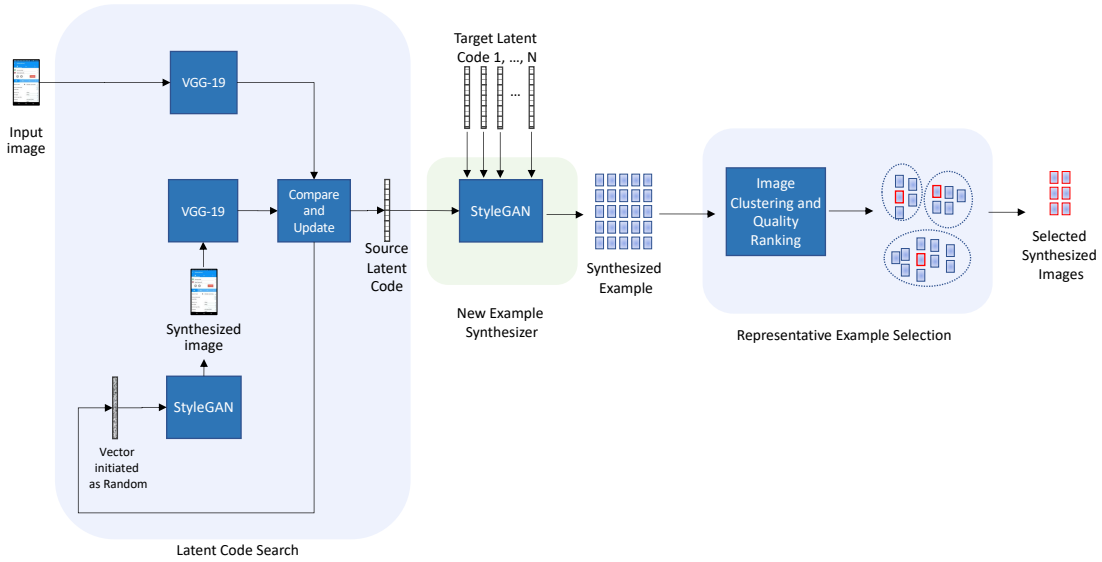


Figure 3.1 Overview of the StyleGAN-based approach. The network takes a preliminary design image as input then transforms it into a latent code. The style of the original input is then merged with a set of target latent codes to synthesize new examples. Image encoding and clustering methods are then used to find the representative examples as output. While not illustrated in this diagram, the output of the synthesized examples can be additionally used to search real UI screenshots for inspiration.

generated from the trained StyleGAN model using a latent code z initialized with zeros in all dimensions. Both img_s and the original image img_i is processed using a pre-trained VGG19 model [40], a deep convolutional neural network that is pretrained for the classification task using the ImageNet database [41]. The output from VGG19 is then used to calculate the perceptual difference *loss* between the original image and the synthesized image, using the Learned Perceptual Image Patch Similarity (LPIPS) metric [42]. The LPIPS metric is used then to update the latent code z through gradient descent. This process iterates until the number of maximum iteration is reached or the *loss* stops to decrease. The latent code obtained is then returned to represent the input image.

3.1.2 Component 2: New Examples Synthesizer

New Examples Synthesizer merges the style of the original image as source with a set of target images or latent codes to produce a new set of examples. We use one source and one target as an example to illustrate the process below (also see Figure 3.2). When multiple

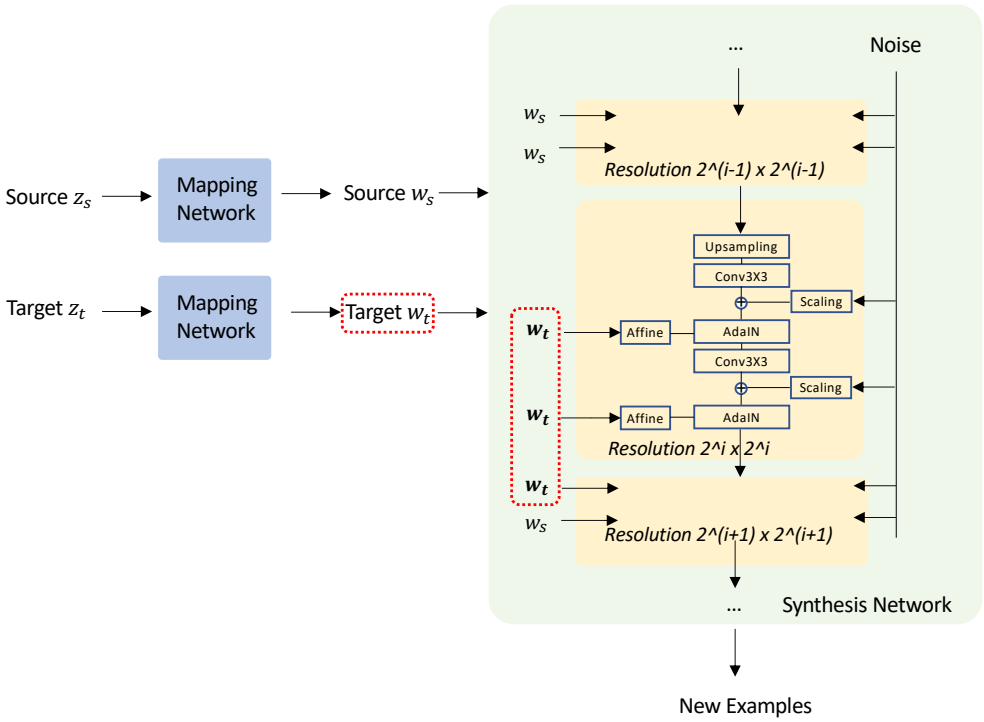


Figure 3.2 The detailed architecture for merging styles of two latent codes using StyleGAN. In this case, the target latent code is used for three style input locations and the source latent code is used for the rest of the style input locations.

target images are involved, they are treated in the same way. We use z_s to represent the source image latent code, which is obtained from the previous step. The target latent code z_t can be any vector from the latent space that will be merged with the source image; for example, it can be a latent code obtained from a target image or a latent code that is directly sampled from the latent space.

Both latent codes are first transformed into intermediate vectors w_s and w_t that have the same number of dimensions as the original latent codes through a mapping network, i.e., an 8-layer feed-forward network. The two intermediate vectors w_s and w_t are then used in the StyleGAN image synthesizer at different layers to control the degree and level of style mix between the source and the target. In particular, as shown in Figure 3.2, each resolution level during the image synthesis is comprised of two style inputs for the AdaIN operations [33], which is an image style transfer algorithm. The synthesizer is configured to include eight levels of image resolutions (i.e., from 8×8 to 1024×1024 in steps of powers of two), resulting in a total of 16 locations of style inputs. We used the source vector w_s as the default style input for all the style input locations and replace the style input locations from loc_m to loc_n with the target vector w_t. The example shown in Figure 3.2 illustrates the case of using w_t

for three style input locations from the resolution $2^i \times 2^i$ and $2^{(i+1)} \times 2^{(i+1)}$. We iterated loc_m from 1 to 16 and loc_n from loc_m to 16 to obtain a total of $16 \times (16 + 1)/2 = 136$ merged images for each pair of source and target inputs, which covers all possible granularity levels of influence (e.g., from the structure to the details) of the target image to the source image.

3.1.3 Component 3: Representative Examples Selection

In the previous step, we synthesized a large number of new examples for each pair of inputs, which can be overwhelming for the designers to examine. Moreover, since our style merging process is very fine-grained, it also introduces redundancies in the set of synthesized examples. In this component, we apply a clustering method to pick a smaller sample of representative images from the set of generated images. In particular, we adopt the DBSCAN method [43] to cluster the images, as it is independent of the number of clusters that prevents limited clustering of the synthesized examples. DBSCAN uses perceptual similarity [42] calculated between any two images from the synthesized example set. Image perceptual similarity represents the perceptual distance which measures the similarity between two images in a way that to be close to human judgment, which is why we decided to use a pre-trained AlexNet [44] to calculate the metric. For clustering, we set the threshold ϵ as 0.9, i.e., the maximum distance between two samples to be considered as neighbors of each other. Within each cluster, we used the discriminator of the trained StyleGAN as a proxy to evaluate the quality of each image. For each cluster, we selected the one with the highest quality according to the discriminator as the representative example, indicating that the output is the closest to the real UI screenshot images used in the training process.

3.2 StyleGAN Training

To use the capacity of StyleGAN in the context of UI design, we need to train the StyleGAN model with a dataset of user interface design artifacts. In this section, we describe the dataset we used and our training process of StyleGAN for our approach. The trained StyleGAN model was then used in the components described above.

3.2.1 Dataset

We used the Rico dataset [45] for StyleGAN training. It contains 66,261 unique UI screenshots of Android apps and serves as one of the largest repositories of mobile app designs to date. The dataset includes a diverse set of UI screenshots with varied complexity that contains various types of UI components. In order to obtain a high-quality dataset for inspiration

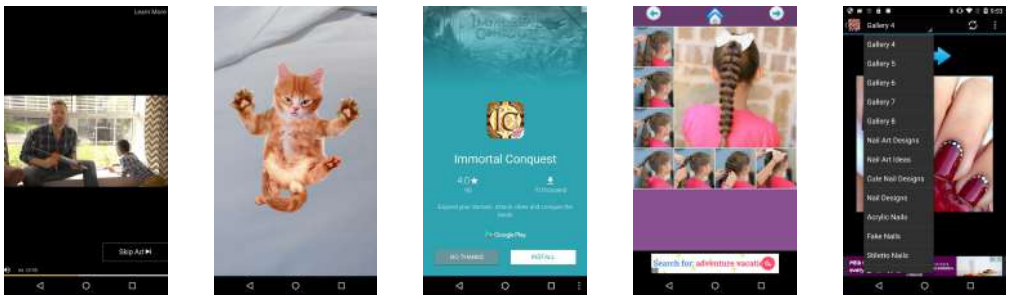


Figure 3.3 Samples of UI screenshots removed because they only have one or two unique component labels. Most of the screenshots with less than three unique components contain splash screen, video screenshot, full-screen image, and web view in their design, adding unnecessary visual complexity.

purposes, we removed UI screenshots that have only one or two unique component labels from the Rico dataset. This is because we found in a manual inspection that the UIs with less than three unique component types usually do not contain enough interaction elements to support inspiration or benefit from our StyleGAN-based approach. They also often introduce unnecessary visual complexity that only exists in singular instances (e.g., a splash screen, video screenshot, advertisement, or full-screen image, see Figure 3.3) that affects the training performance; in other words, these instances make it difficult for StyleGAN to generate similar components and even if generated, the visual presentation of those components are too uncommon to be useful for inspiration. We used the UI view hierarchy data in the Rico dataset to calculate the number of unique component types in each screenshot. In total, we removed 8,221 images that had only one or two unique component types, resulting in 58,040 images in the dataset. Figure 3.4 shows the distribution of the number of unique component labels in the UI screenshots in the dataset. We then resized each image into 1024×1024 for the training process.

3.2.2 Training process

We built upon the official TensorFlow implementation of StyleGAN[1] and used our preprocessed dataset to train the model. While we did not perform a formal hyperparameter search, we explored a few changes of hyperparameters reported in Karras et al. [7], including the initial learning rates of the generator and the discriminator, the number of times the discriminator is trained per generator iteration, and the number of minibatches to run before adjusting the training parameters. We eventually used the following hyperparameters because they achieved the best performance according to the Fréchet inception distance (FID) [46]

[1]https://github.com/NVlabs/stylegan

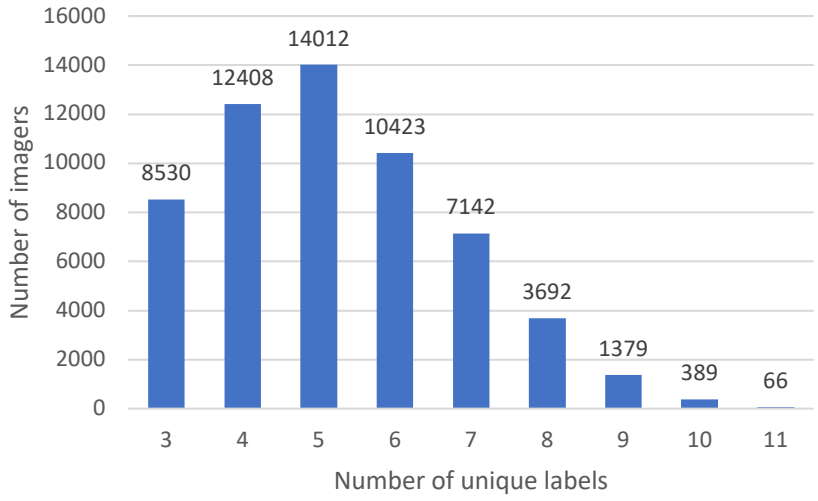


Figure 3.4 The distribution of the number of unique component labels in the UI screenshots in our dataset.

in our exploration: for both the generator and the discriminator, the learning rate was set to 0.0015 for resolution levels equal or less than 128×128, 0.002 for resolution levels 256×256 and 512×512, and 0.003 for 1024×1024; the discriminator was trained at the same frequency as the generator; and training parameters were adjusted after every four minibatches. Additionally, mirror data augmentation was not enabled during training due to the asymmetric nature of UI images. We used a server that contained four NVidia V100SXM2 GPUs to train the model. Training terminates when the FID value increases (i.e., the generation quality deteriorates) for three consecutive iterations. The entire training process lasted 162 hours. The best performance measured with FID was achieved at 42.91. Although this performance is not ideal comparing to the typical face generation tasks, by manual inspection, we found that the generated images can already represent certain layout features and visual details that can help design inspiration. This difference in performance may be due to the extraordinary complexity and diversity of screenshots of UI design. It is worth noting that our focus and contribution in this paper are not to achieve a higher performance in the generative model. Instead, we focus on evaluating the potential of this line of techniques for supporting the challenging task of design inspiration, even with less-than-perfect image generation.

3.3 Quantitative Evaluation

We conducted an experiment that focused on evaluating the generative methods against two other techniques that suggest design examples (i.e., random suggestion and similarity-based

suggestion). The evaluation is based on the scenario where a designer inputs a preliminary interface design to the system and gets a set of suggested design examples for inspiration. While we used two quantitative metrics (i.e., similarity and diversity) for measuring the ability of inspiration support of the system outputs, we also focus on discussing our observations and insights on how the outputs might have contributed to the metrics.

3.3.1 Data sampling

To sample a diverse set of UI screenshots as inputs, we grouped the Rico dataset according to the number of unique component types on the UIs. We considered the number of unique component types as an indicator of the complexity of the UI, thus the complexity of the design task. Our dataset contained UI screenshots that contain 3 to 11 unique component types, resulting in nine unique groups (see Figure 3.4). We randomly selected three UI screenshots from each unique component type count group, resulting in 27 screenshots as inputs in the evaluation scenario; these images are shown in Figure 3.5. This strategy ensures the coverage of different levels of complexity in the UI inputs, thus the coverage of the complexity of the design task.

3.3.2 Quantitative metrics

We derived two metrics to measure the ability of inspiration support of a set of UI images. These metrics are inspired by previous work about inspiration in design [25, 47] and collectively focused on both targeted and serendipitous inspiration. Both metrics rely on a measure of distance between images, particularly a measure of perceptual distance that is aimed to approximate human visual perception; this measure is the same as the one used in the clustering task in the representative examples selection component of our approach (see section 3.1.3).

In the following equations, $E_i^{(D)}$ $(i = 1, 2, ..., n)$ denotes the ith output example image for an input design D, $O^{(D)} = \{E_1^{(D)}, E_2^{(D)}, ...E_n^{(D)}\}$ denotes the set of output images for an input D, and $dist(A, B)$ denotes the perceptual distance between images A and B measured using the technique described above. The two metrics we used are:

- **Similarity** of the suggested examples to the input design. This metric is double-sided. A sufficient similarity may indicate the relevance of the suggested examples, which is important to provide targeted inspiration. However, a high similarity indicates the potential of design fixation. We use the mean similarity (i.e., the complement of the perceptual distance) between the output images and the input image to evaluate the

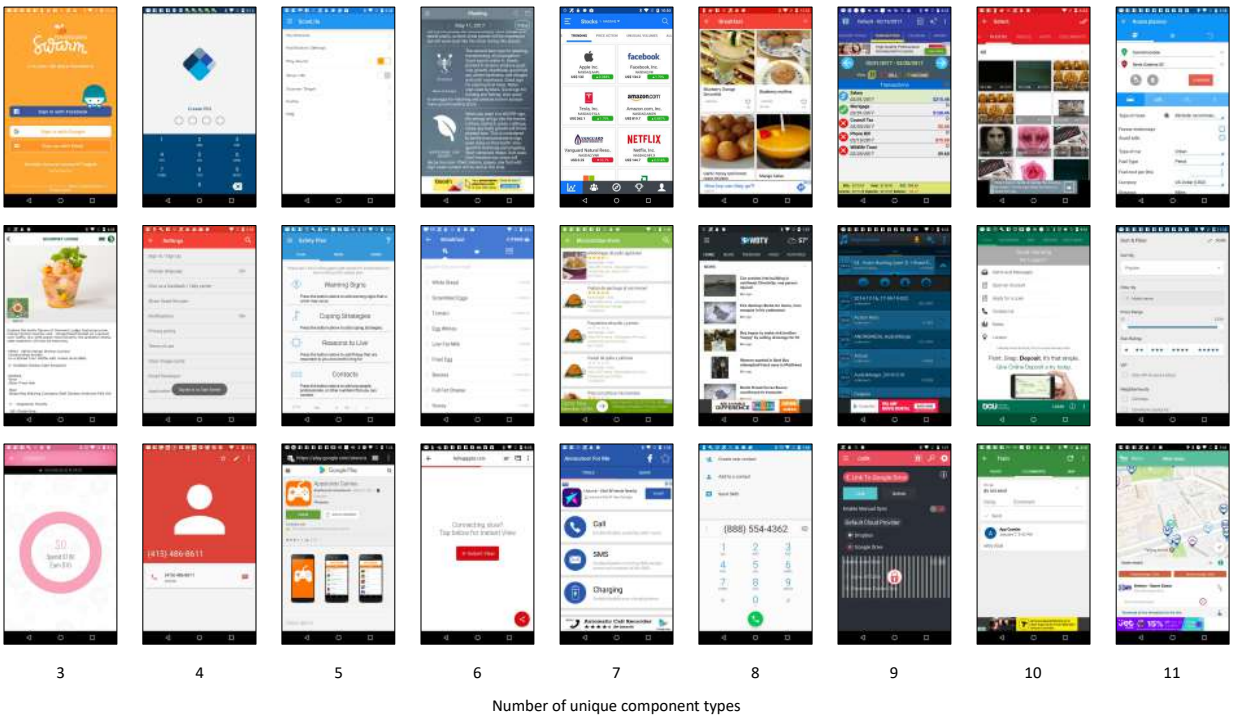


Figure 3.5 Sample of UI screenshots used in the evaluation study, ordered by the number of unique component types.

overall similarity of the output examples to an input design:

$$Rel(O^{(D)}) = 1 - \frac{1}{n}\sum_{i=1}^{n} dist(D, E_i^{(D)})$$

- **Diversity** of the suggested examples indicates how different and varied the outputs are, given an input design. It plays an important role in preventing fixation. We use the mean pairwise distance among all output examples of an input design to evaluate the diversity of the output set:

$$Div(O^{(D)}) = \frac{1}{n(n-1)/2}\sum_{j=i}^{n}\sum_{i=1}^{n} dist(E_i^{(D)}, E_j^{(D)})$$

3.3.3 Experimental design

In the experiment, we considered the following conditions for suggesting a set of images for inspiration. Particularly, conditions 1, 2, 3, and 4 are four variants of the StyleGAN-based method. Figure 3.6 summarizes these conditions.

Condition 1. In this condition, we used the trained StyleGAN model to merge the input

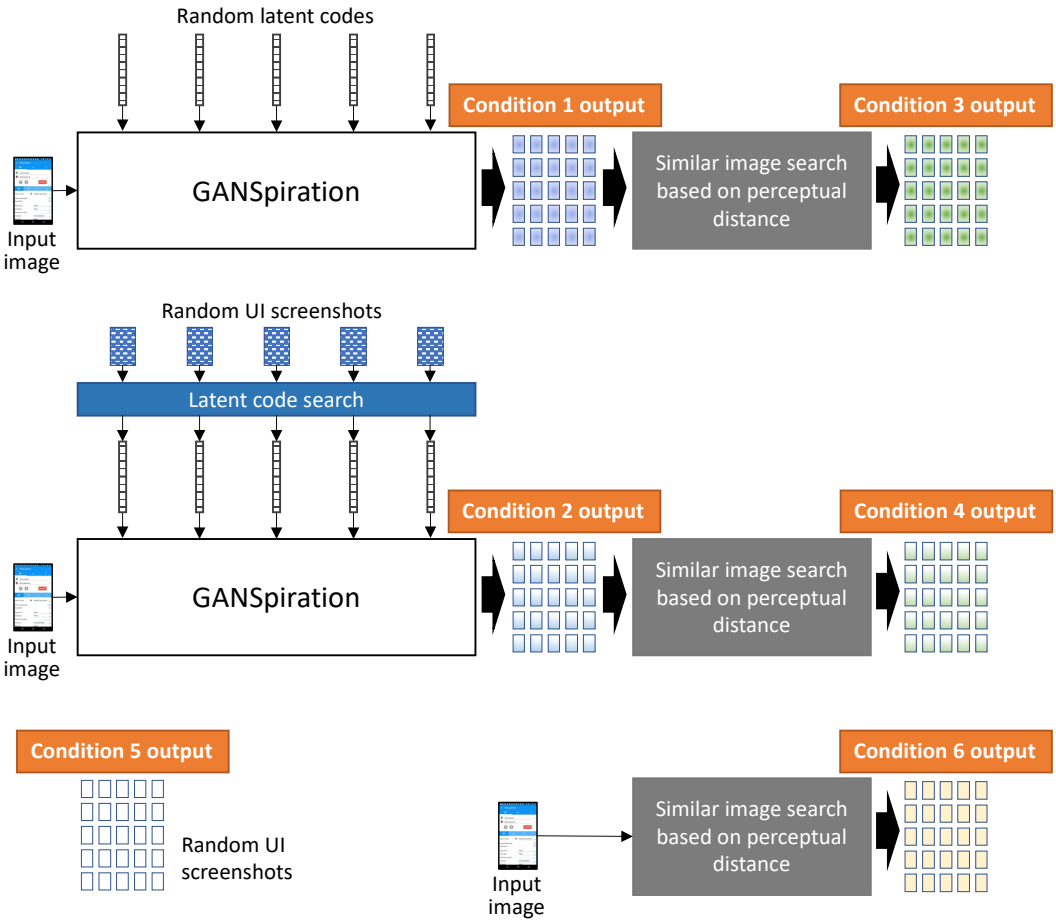


Figure 3.6 The six conditions used in the experiment. The condition outputs were then used to calculate the metrics and in user studies.

image with five random latent codes, each generated a set of examples according to the StyleGAN-based architecture (see Section 3.1). We then combined these examples as the output.

Condition 2. This condition is similar to Condition 1; however, instead of using five random latent codes, we randomly selected five images from the prepossessed dataset and obtained their latent codes for style merging. The examples generated from each merge were then combined as the output. Conditions 1 and 2 are created to evaluate two different variants for using directly generated images for inspiration.

Condition 3. In this condition, we first obtained the output of Condition 1. Then for each initial output image, we searched for the most similar image, using the perceptual distance described in section 3.3.2, from a dataset of real UI screenshots. Because of the computational cost of this search, we used a smaller dataset created by Huang et al. [29]

as the search space; it contained 2201 high-quality UI screenshots sampled from the Rico dataset. Real UI screenshots obtained from this search were then combined as the output.

Condition 4. This condition is similar to Condition 3 except that the initial output was obtained from Condition 2 instead of Condition 1. Conditions 3 and 4 are created to examine the effects of the realisticity of the design examples on inspiration.

Condition 5. In this condition, we randomly selected 25 images from the prepossessed dataset as the output.

Condition 6. In this condition, we performed a search on a subset of Rico dataset created by Huang et al. [29] to obtain the most similar images to the input image, based on the perceptual distance described in section 3.3.2. The top 25 similar images were combined as the output.

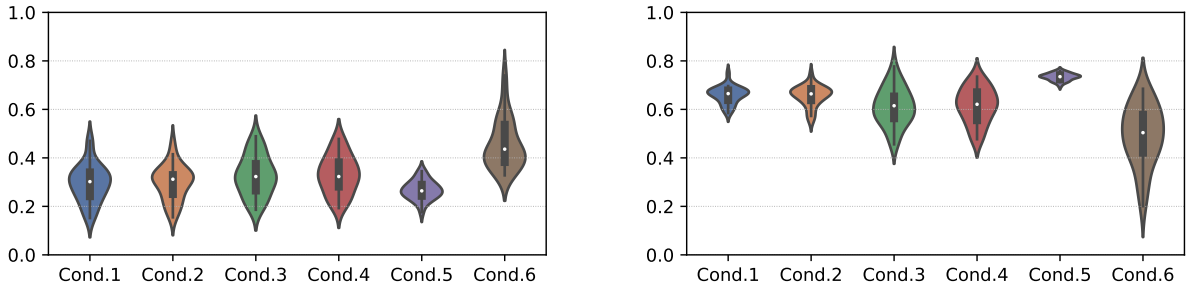
3.3.4 Results

Among all the sampled inputs, both Condition 1 and Condition 2 have resulted in an average of 19 output images ($SD = 7.2$ and 5.15, respectively). A comparison of samples of five outputs from each experimental condition for one input image is shown in Figure 3.7. Figure 3.8 shows the distributions of the similarity and diversity metrics on the six conditions. Kruskal-Wallis tests indicated statistically significant differences among the experimental conditions with respect to both metrics ($p < 0.001$). We then conducted posthoc pairwise analyses using the Mann-Whitney U test with Bonferroni correction to identify the conditions that contributed to the difference; the Bonferroni correction was used to address multiple comparisons. The posthoc analyses revealed the following results.

- The StyleGAN-based methods (Conditions 1, 2, 3, and 4) have resulted in significantly lower similarity ($p < 0.001$) than Condition 6 (i.e., search-based approach). Conditions 3 and 4 (i.e., generation + search) also resulted in significantly higher similarity ($p < 0.05$), thus relevance, than random examples (Condition 5).
- While the StyleGAN-based methods (Conditions 1, 2, 3, and 4) have resulted in significantly lower diversity ($p < 0.001$) than Condition 5 (i.e., random examples), they have also achieved significantly higher diversity ($p < 0.01$) than similarity-based approach (Condition 6).

To understand how the complexity of the input UI design can influence the similarity and diversity metrics of the six experiment conditions, we separated the 27 sampled inputs into

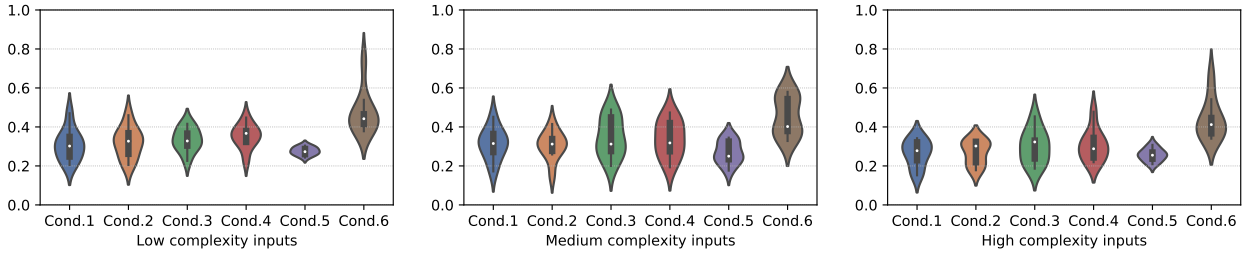
Figure 3.7 Samples of five outputs from each experimental condition for the input image. Samples from Condition 3 are matched with those from Condition 1, and samples from Condition 4 are matched with those from Condition 2.



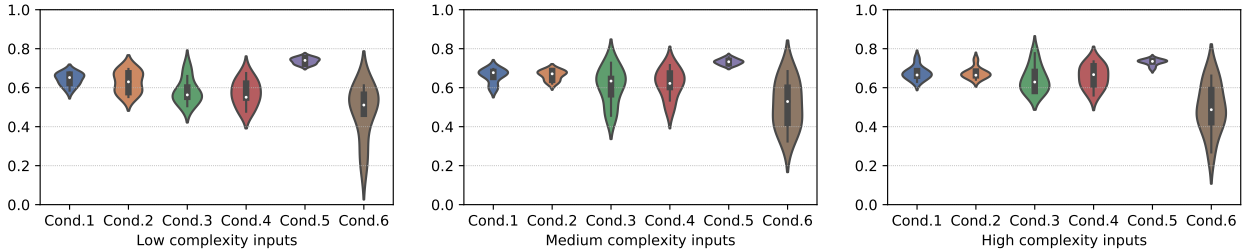
(a) similarity

(b) diversity

Figure 3.8 Distribution of the similarity and diversity metrics on the experimental conditions.



(a) similarity by input complexity



(b) diversity by input complexity

Figure 3.9 Distribution of the similarity and diversity metrics on the experimental conditions, analyzed by input complexity.

three groups: (1) *Low complexity inputs* contain less than six unique component types. In our sample, they often represent login screens, setting menu screens, or screens that communicate a single piece of information (see Figure 3.5). (2) *Medium complexity inputs* contain between six to eight unique component types. In our sample, they often represent screens that contain heterogeneous information (usually presented in lists or cards) or screens that provides multiple options to users. (3) *High complexity inputs* contain more than eight unique component types. In our sample, they often represent screens that contain complex interaction mechanisms, including tabs, multiple options on list items, maps or complex forms. Figure 3.9 presents the distributions of the similarity and diversity metrics in each input complexity group. Kruskal-Wallis tests indicated statistically significant differences among the experimental conditions with respect to both metrics in all three groups ($p < 0.05$). Table 3.1 and Table 3.2 present the posthoc analysis results based on pairwise Mann-Whitney U tests with Bonferroni correction. Results indicated that certain conditions StyleGAN-based conditions (particularly Conditions 1 and 2) achieved a significantly lower similarity than Condition 6 (i.e., search-based approach) for low and high complexity inputs, but not in medium complexity inputs. Additionally, for high complexity inputs, the StyleGAN-based approach can achieve a similar level of diversity to random suggestions (Condition 5), and

significantly higher diversity than the search-based suggestions (Condition 6).

Table 3.1 Mean difference (row label minus column label) of the similarity metric among the six conditions, by input complexity.

(a) Low complexity inputs

	C.2	C.3	C.4	C.5	C.6
Cond.1	-0.01	-0.03	-0.04	0.03	*-0.17
Cond.2		-0.01	-0.03	0.04	*-0.15
Cond.3			-0.02	0.06	*-0.14
Cond.4				0.08	-0.12
Cond.5					**-0.20

(b) Medium complexity inputs

	C.2	C.3	C.4	C.5	C.6
Cond.1	0.01	-0.03	-0.03	0.04	-0.14
Cond.2		-0.04	-0.03	0.04	-0.15
Cond.3			0.01	0.08	-0.11
Cond.4				0.07	-0.12
Cond.5					*-0.19

(c) High complexity inputs

	C.2	C.3	C.4	C.5	C.6
Cond.1	0.00	-0.03	-0.04	0.01	**-0.17
Cond.2		-0.03	-0.04	0.01	**-0.17
Cond.3			-0.01	0.04	-0.14
Cond.4				0.05	-0.13
Cond.5					**-0.18

Using pairwise Mann-Whitney U test with Bonferroni correction: * $p < 0.05$, ** $p < 0.01$

Table 3.2 Mean difference (row label minus column label) of the diversity metric among the six conditions, by input complexity.

(a) Low complexity inputs

	C.2	C.3	C.4	C.5	C.6
Cond.1	0.01	0.06	0.07	**-0.09	**0.17
Cond.2		0.05	0.06	**-0.11	*0.15
Cond.3			0.01	**-0.16	0.10
Cond.4				**-0.17	0.09
Cond.5					**0.26

(b) Medium complexity inputs

	C.2	C.3	C.4	C.5	C.6
Cond.1	0.00	0.05	0.04	**-0.07	0.15
Cond.2		0.06	0.04	**-0.07	0.15
Cond.3			-0.01	*-0.13	0.10
Cond.4				**-0.11	0.11
Cond.5					**0.22

(c) High complexity inputs

	C.2	C.3	C.4	C.5	C.6
Cond.1	0.00	0.03	0.02	-0.06	*0.18
Cond.2		0.03	0.02	-0.06	*0.18
Cond.3			-0.01	-0.09	0.15
Cond.4				*-0.08	0.16
Cond.5					**0.24

Using pairwise Mann-Whitney U test with Bonferroni correction: * $p < 0.05$, ** $p < 0.01$

We manually inspected the outputs of the approaches used in the six experimental conditions to identify their risks and potential to support design inspiration. We found the following themes through our inspection.

- Conditions 1 and 2 (i.e., generation-only): Examples generated in these two conditions resembled real UI screenshots, but contained a lot of blurry and noisy components. However, some examples generated in these conditions contained interesting variations and alternatives to the input UI. For example, Figure 3.10 shows that the directly generated examples suggested alternatives on the color scheme, layout, and item details.
- Conditions 3 and 4 (i.e., generation + search): Examples generated in these two conditions contained real images that addressed the noise issues in Conditions 1 and 2. Some of the returned examples also contained interesting variations to the input image (see Figure 3.11). However, some of the searched images do not reflect directly the intention of the originally generated images. For example, in Figure 3.12, the generated example

seems to suggest a different color scheme and a layout change, but the search resulted in somewhat irrelevant screens.

- Condition 6 (i.e., search-based approach): The outputs of this condition commonly contained visually similar UI screenshots with the input, both in terms of color and layout (see Figure 3.7, Condition 6).

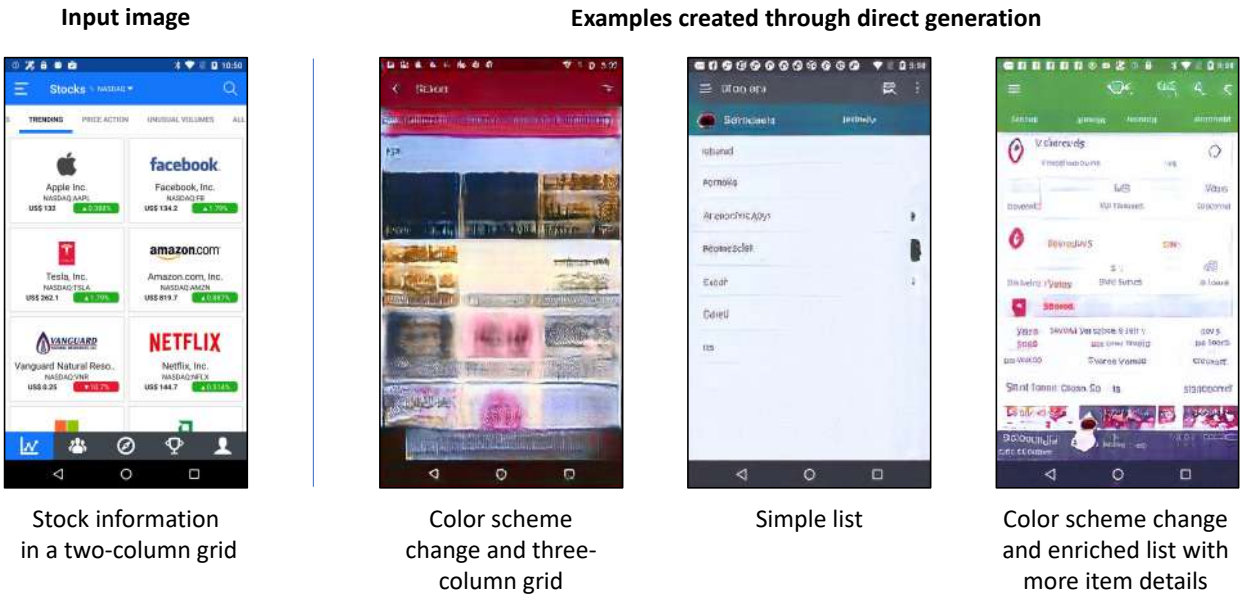


Figure 3.10 Directly generated examples involves noises but can provide useful insights and suggestions.

3.4 User Evaluation

To understand the potential of our StyleGAN-based techniques from the perspective of professional UI/UX practitioners, we conducted a user study focusing on the practitioners' opinions on how well the results of our StyleGAN-based approach may help in their design practice.

3.4.1 Methods

The user studies were conducted in June and July 2021. In this section, we describe our participants, the procedure of the study sessions, the materials used, and the analysis methods. The user study protocol is approved by the ethics committee at [anonymous institution].

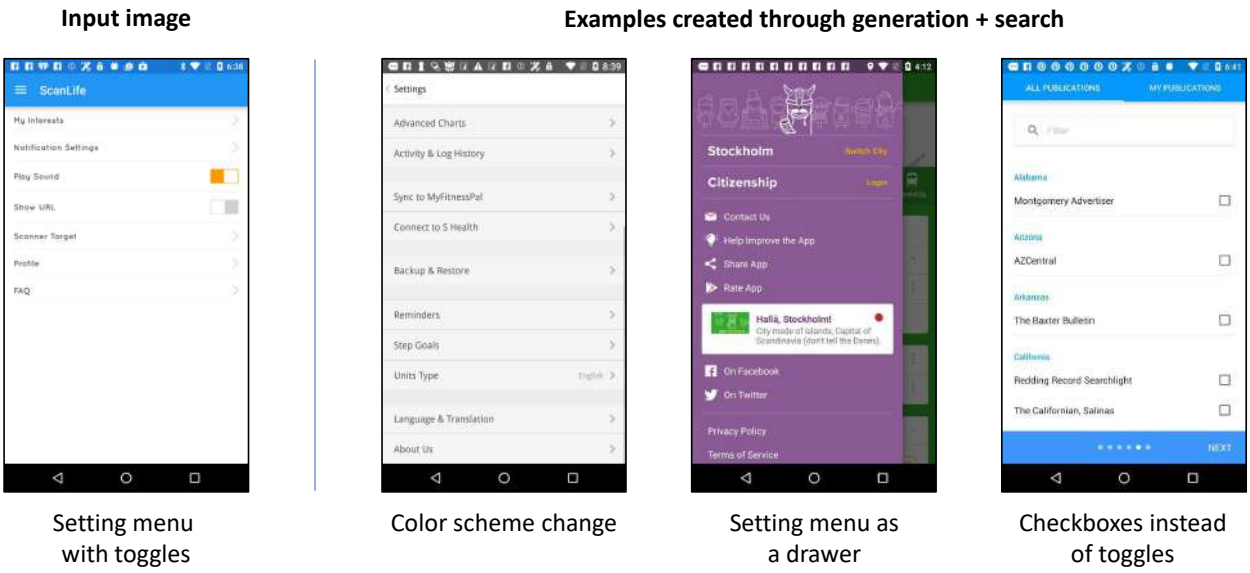


Figure 3.11 Searched examples based on generated images are cleaner and can provide useful insights and suggestions.

Participants. We conducted the user study with five UI/UX professionals. The participants had varying levels of experience, ranging from one to ten years as either UX researcher or UI/UX designer. They worked in different types of organizations, including two freelancers, one in a small start-up company, one in a more established medium-sized company, and one in a large multinational company. Table 3.3 summarizes the participants' characteristics.

Procedure and material. The user studies were conducted online via the Zoom platform. Each participation took about one hour to complete. Each study session began with a semi-structured short interview in which the participants were asked about their professional experiences and their experience of using design examples. We then presented one input UI screenshot from the sample used in the quantitative evaluation (the one that is shown in Figure 3.7) to the participants. We told the participants to consider a scenario in which they want to get inspiration from examples to modify the design of this UI. This UI screenshot contained 11 different types of components, representing a complicated design task. After the participants familiarized themselves with this UI screenshot, we then presented the corresponding output images from all six conditions as the design examples, one after another; a sample of these materials was shown in Figure 3.7. The participants were asked to examine each set of the design examples and provide feedback on (1) their relevance to the design task, (2) the diversity of the design examples, (3) the effectiveness of the design examples for inspiration, and (4) general positive and negative perceptions of the examples. The order of

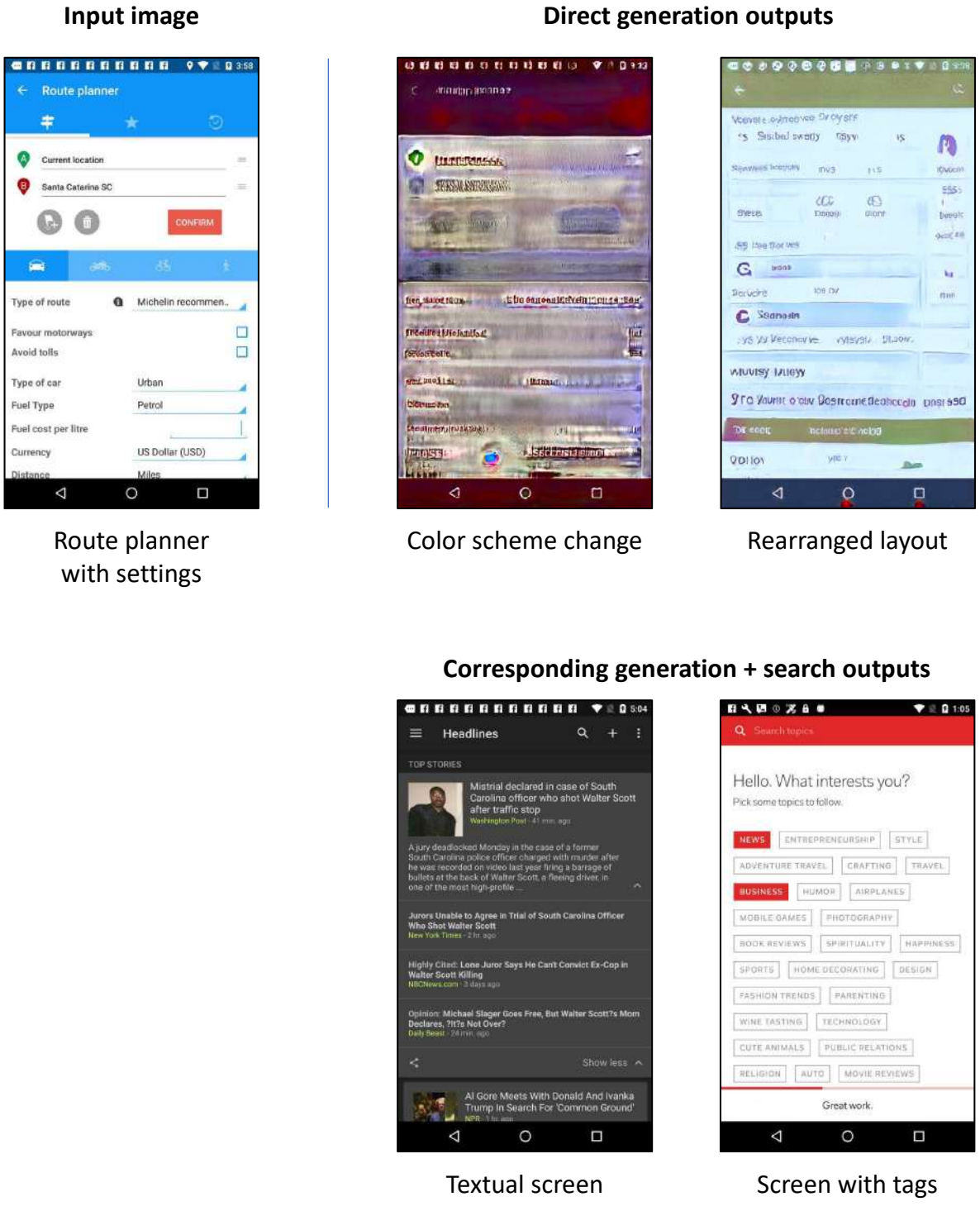


Figure 3.12 Search results sometimes do not reflect the intention of the style-based generation results.

Table 3.3 Characteristics of participants in the user study

ID	Job title	Organization description	Years of experience	Projects contributed to
P1	UX researcher	Freelancer	1 year	1 project
P2	UI/UX designer	A small-sized company developing software solutions that allow the creation of printable personalized products	8 years	>10 projects
P3	UI designer	A medium-sized company developing a cloud-based computer-aided design (CAD) software	3 years	10 projects
P4	UI designer	Freelancer	10 years	>10 projects
P5	UX researcher	A large multinational company developing business management software	2 years	2 projects

these six conditions was randomized among the participants to mitigate the order effects; the participants were also not aware of the condition numbers (i.e., the way in which the examples were generated) when examining the examples. After all six conditions were covered, we described the techniques we used and asked the participants about their general perception of the generated design examples. (See the appendices A and B for more detail)

Analysis. Two researchers watched recordings of all study sessions and took detailed notes regarding the participants' comments and reactions. These notes were then combined and qualitatively analyzed to identify the common themes. During this analysis, we focused on identifying the positive and negative aspects that the participants mentioned about the design examples resulting from the four categories of the experimental conditions: direct style-based generation of UI images (Conditions 1 and 2), style-based generation then search of real UI examples (Conditions 3 and 4), random selection (Condition 5), and similarity-based search (Condition 6).

3.4.2 Results

All of the participants indicated that they look for examples in their design projects. A common reason that the participants search and examine examples is to learn from the examples and borrow elements from them. For example, P3 mentioned "Every single time when I need to design an interface, I will go to all these different reference websites. I always

start from there to see what a possible solution is out there. Because it saves my time not to reinvent the wheel." P2 talked about the granularity in which the design examples may help, saying "Even for a specific project, every part of the project, I need to assess what exists right now and what I can learn from what is already existing." P4 also emphasized the importance of examples by mentioning the efforts she often spends on it, saying "Every project I have to search for examples for inspiration. I will spend lots of time on it."

Perceptions on the directly generated examples (Conditions 1 and 2).

The negative aspects of these generated examples mentioned by the participants were most concerned with the quality of the images and the noises included in the UI. For example, P3 mentioned: "It looks a bit strange to me because all the UI elements are not real UI elements. They just mimic the shapes. It doesn't provide a lot of realistic details for me to get a record for my design." P5 also declared, "If there is not any noise in the images, it is a good idea to have this technique."

Some participants discussed the potential of the generated examples. For example, P2 mentioned: "The idea is pretty close to what I am looking for, overall, conceptually... Since they are all pretty abstract, they are already helping me more, so that I can see the shapes and stuff ... to think out of the box." P3 said: "If it can generate a lot of good information online that we don't have time to look for, and it manages to make a merge of all the relevant interface, I can use it as a reference to see the trend on color pallets and layouts."

Perceptions on the examples created from generation and then search (Conditions 3 and 4).

All but one of the participants agreed the examples can be effective for inspiration. For example, P1 mentioned: "It is diverse, we have different designs... They are also relevant since I can see some of the same patterns.." P5 also said, "These are effective for inspiration for example images. Since they include both text and big image and different kinds of components inside." Participants also noticed some of the components in the images could be inspiring. P4 mentioned, "Image number 14 at the top right there is an alert icon which I can use in my design and Number 3 also has a search icon." P3 stated, "There are some images which have tabs, which I can use as references for the tab of the main image, and also the toggle switch, radio buttons, ..."

The participant (P2) who thought the examples are not effective for inspiration was mainly concerned about the overall design style. She mentioned: "It is all the same [Google] Material

Design style and nothing exciting – it is very similar to what I would work on already [in the input image]." This comment pointed out the limitations of using UI screenshots from the same platform for inspiration. Some participants also voiced other concerns related to our current StyleGAN-based technique. For example, P3 mentioned that she wished to have the examples contain more UI screenshots from the same page type and the same application domain: "I did not find any image that included extended forms or any transportation apps." P1 and P4 also would like to see more recent and modern designs in the examples; this comment, along with P2's concerns, highlighted the important role of the dataset used for generating the examples and searching for existing designs.

Perceptions on the randomly selected examples (Condition 5).

All participants were impressed by the obvious diversity of the examples. For example, P2 stated, "I like that the color schemes are getting different. It has different layouts from different stuff." However, they also noticed the randomness of the examples. P3 said, "There are a lot of things that are not relevant here. Some of the examples are definitely noise." P2 also said, "I do think that those layouts, that are different, are not related to the design I am looking for."

Perceptions on the examples selected from similarity-based search (Condition 6).

All participants agreed that the examples are similar to the source image, but not diverse enough for inspiration. P3 said, "The first thing that I noticed is the similar color pallets and elements." P2 also said, "All of them are lists and have the same colors, not very helpful but relevant." P4 also mentioned, "Although the information is clear and easy to use, [the examples are] too boring."

Overall, the participants considered our StyleGAN-based approach, particularly the ones that output full-fledged UI screenshots (i.e., conditions 3 and 4), as viable ways to gain inspiration in their practical design workflows. For example, P1 indicated "This tool would help in the competitive analysis part. If it has more trendy images it could be really helpful." And P2 stated that "I could use this technique... to think out of the box."

3.5 Discussion

In this study, we found that our StyleGAN-based approach are able to generate design examples that are both diverse and relevant to the input image. Particularly, our approach

can help resolve the design fixation issue, which is common when similarity-based approaches were applied for retrieving design examples in practice. The user evaluation also indicated that our StyleGAN-based approach is able to help designers broaden their horizons and get inspired. Participants generally preferred the StyleGAN-based outputs, particularly the ones that suggests full-fledged UI screenshots based on the generated examples, over random examples and similarity-based examples. In this section, we discuss the implications of our study results for designing tools that leverage generative techniques such as StyleGAN for design inspiration.

3.5.1 Style-based generation provides design inspiration in different granularity levels

Our user study results indicated that the style-based generation techniques is able to provide design inspiration on three levels: (1) the coarse level that provides ideas for layout or structural changes, (2) the middle level that suggests component design alternatives, and (3) the fine level that proposes different aesthetics such as color schemes. This is made possible by the ability of StyleGAN to alter the input image based on different granularity levels (i.e., different spatial resolutions) of the target 'style images'. In other words, the style images can be used to alter either the structure or the details of the input image; and this is controllable by the users of the system. The three aspects of inspiration were all appreciated by the participants during the user study. The tool design that leverages the StyleGAN-based approaches can incorporate these three levels of design inspiration. Particularly, a design inspiration tool can indicate and explain the intention of the suggested examples by checking the granularity level of the style image used for style merge. Based on this information, a descriptive label of inspiration granularity can be assigned to each example. This way, if the designers have a particular concern when searching for inspiration (e.g., need to find a different layout but using the same color scheme), they would be able to better focus on such examples. Additionally, it is also possible to give the designers control over the granularity level of style merge.

3.5.2 The visual quality of the generated image is an important factor for inspiration

Our results revealed that the participants preferred full-fledged UI screenshots over directly generated images that typically include noises, although the direct generation and the generation-then-search conditions achieved the same level in the diversity and relevance metrics. While some participants were impressed that the directly generated images look

like a UI, our results indicated that the visual quality of the generated images does affect how well the designers perceive the examples. In this study, we retrieved the most visually similar UI screenshots to the directly generated images to address this issue. However, our manual inspection revealed that the search results based on the generated images do not always match the intention of the directly generated results. To further resolve these issues, techniques for identifying components on generated images could be investigated. With such techniques, the quality of the individual components can be improved to make the generated images look more similar to real ones. Further, if components were identified, the directly generated images can be converted to wireframes in order to provide layout or structural suggestions. Color schemes of the components can also be matched to the input image to provide direct alternatives.

3.5.3 A diverse and relevant training dataset would help generate more insightful examples

Our study relies on the Rico dataset, which is created in 2017. Some of our participants voiced that the examples retrieved do not reflect the most recent design trends, thus limit the inspirational power of the examples. This result indicates that the dataset used for training the generative model, as well as the dataset used for retrieving real UI screenshots, are important factors to consider. Potential solutions to this problem include using only the newest apps in the dataset and collecting datasets from the most recent design sharing platforms (e.g., dribbble.com). Additionally, using a merged dataset including UI screenshots from different platforms (e.g., Android, iOS, desktop app, web app, etc.) with different design frameworks/systems [48] would help avoid platform or framework-specific design stereotypes.

3.5.4 Combine generative models with other techniques

In our experimental design, conditions 3 and 4 (i.e., generation + search) are our first attempts to combine the pure generative approach with other techniques to provide examples for effective inspiration. These attempts can be further enriched and expanded. Particularly, our participants seemed to desire examples from the same application domain (e.g., route planning) and/or focusing on the same type of page (e.g., configuration page) as the input UI page. Thus, it could be useful to perform the style-merging generation using the style images from the same application domain and/or page type as the input image. Moreover, combining the generative technique with textual thematic specifications that describe characteristics of the desired examples (e.g., the ones similar to [28]) would give designers more control over the returned examples and support a more effective inspiration. Finally, techniques that

can highlight interesting areas in the examples related to the suggested alternatives to the input image would also facilitate a more efficient exploration for serendipitous and targeted inspiration.

CHAPTER 4 COMPONENT DETECTION

After developing the StyleGAN-based approach, we extended our work by enabling the detection of the UI components in UI images, particularly generated UI images. The detection of the components could be the first step of enhancing the quality of the generated UI images as well as some full-fledged UI images that need to be redesigned. In addition, by having a good component detector, we would be able to develop a tool that uses our StyleGAN-based approach to support UI editing and design automation.

Regarding the results of our generative model, we found that the network is able to generate most of the frequent UI components such as Texts, Icons, Drawers, Modals, and EditTexts (see Figure 4.1). The weakness of the generator is generating components that included an image. Since there are lots of different images in the dataset which are not even similar in terms of size, shape, color, and content, it makes learning difficult for the model. Precisely, the network is not trained on enough similar images to learn and generate them effectively. Our preliminary investigation on a few pre-trained UI component detection models show that the previous studies are not able to detect all the components of the generated UI images. We trained, fine-tuned, and evaluated Yolov5 to overcome this problem.

Figure 4.1 There are several frequent components in UIs that our StyleGAN-based approach is able generate, such as: Text, Drawer, Model, Button, Icon and Edit Text

4.1 Methodology

We aimed to retrain the pre-trained Yolov5 to detect the components of the UI images, including images generated by our StyleGAN-based approach. Since our generation approach can generate most of the components recognizable as full-fledged UI's components (see Figure 4.1), we decided to divide our work into two parts. First, we trained the Yolov5 model on a dataset including full-fledged UI images to detect those recognizable components of the generated images. Second, we fine-tuned the model of the first step to detect unclear components of the generated images like Images and Icons. .

4.1.1 Dataset

VINS dataset

We used VINS dataset[1] to train our network. The VINS dataset contains 4800 UI designs. Table 4.1 shows the structure of the VINS dataset. The VINS dataset also includes information about the location and label of the UI components in each image. In total, the dataset comprises 20 types of labels, including EditText, Image, Modal, Toolbar, Switch, Icon, Bottom_Navigation, Drawer, TextButton, Map, BackgroundImage, Card, CheckBox, Spinner, Text, UpperTaskBar, Multi_Tab, CheckedTextView, PageIndicator, and Advertisement.

Table 4.1 The VINS dataset comprises UI images from six different sources

VINS dataset	
Source	Number of images
Android screen(from Rico dataset)	2000
New Android screens	740
Iphone screens	1200
UI designs(from websites)	603
Wireframes UIs	257

Generated images

We generated our images using StyleGAN2 which is trained on the same dataset as we used to train the StyleGAN-based approach. We used the second version of StyleGAN since our preliminary experiments shown that its performance evaluated by FID was better than the first version. The FID score of the final model was 28.69. The Frechet Inception Distance

[1]https://github.com/sbunian/VINS

(FID) score proposed by [49] shows how good the generated images are; a lower FID score indicates a smaller difference between the generated images and real ones. After obtaining the trained model, we randomly generated 2000 images using the model. Then we manually chose 200 images from the 2000 images to annotate and used them for the fine-tuning part of our work; manually choosing the images allowed us to obtain a sample that contained a diverse set of UI components and layouts. We used the labels that exist in the VINS dataset for annotating the generated images. The annotation was done with the Makesense[2] online tool.

4.1.2 Network architecture

Based on our objectives, we selected Yolov5, which is one of the most recent and accurate object detectors, for detecting the UI elements [50]. It has a much smaller weight file than other models and thus runs faster. The biggest improvement of Yolov5 compared to Yolov4 is related to mosaic data augmentation and auto-learning bounding box anchors. Like other object detectors, Yolov5 comprises three main parts: Backbone, Neck, and Head. Figure 4.2 summarizes the architecture of Yolov5. In the Yolov5 network architecture, three important modules are used, we describe them in detail below.

Modules

There are three different modules that are used in the Yolov5 architecture. In this section, we are going to explain each module.

Conv : Conv is a 2d convolutional layer that will be applied over the input. Four parameters for the input of the Conv layer are included: the number of input channels, the number of output channels, filter or kernel size, and step size of the stride. After applying convolution a 2d BatachNormalization with Sigmoid liner Unit (SiLU) activation function will be applied to get the final output of the Conv layer. See Figure 4.3

C3 : C3 layer is a CSP net with three convolutional layers and a Bottleneck module. Figure 4.4 shows the C3's architecture. The CSP Net was proposed by Wang et al. [51] to reduce inference computations compared to the previous work. The results show that CSP Net is able to reduce 20% of computations on the ImageNet dataset. CSP Net is one of the well-performed backbones which could be used in networks for classification or object detection. CSP Net split the input into two parts. The first part goes through

[2]https://www.makesense.ai/

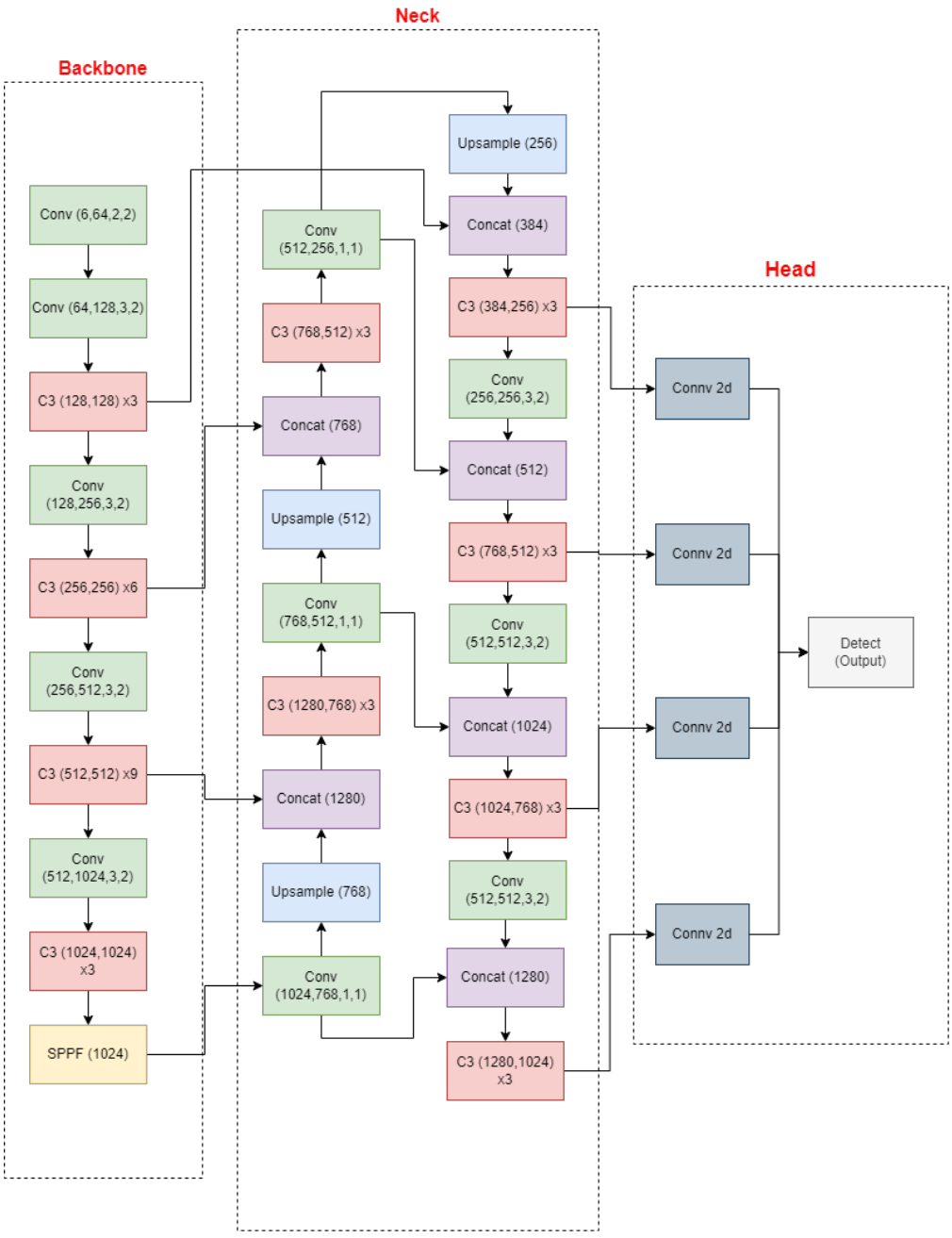


Figure 4.2 The architecture of Yolov5

the network and the next layers to concatenate with the feature maps. And the second part goes to the Partial transition layer to be concatenated with the output of the previous block. On the other hand, the bottleneck layers have fewer neurons compared to their previous layers. So their role in a network is to have a good representation of the input while compressing feature maps and reducing the dimensionality.

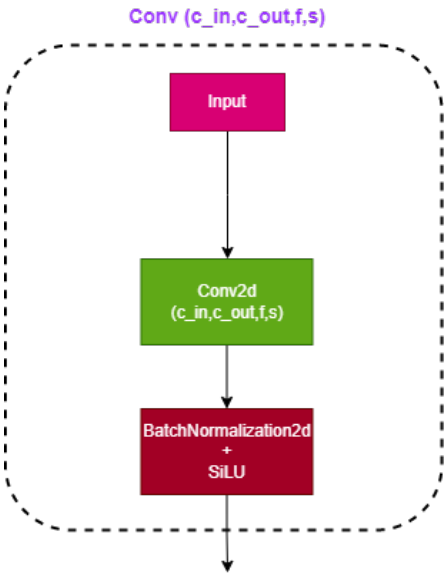


Figure 4.3 The network architecture of the convolutional layer

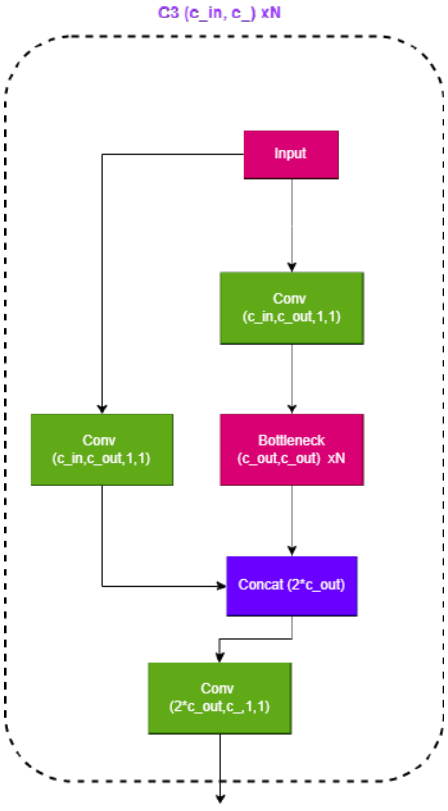


Figure 4.4 The network architecture of the C3 layer

SPPF (Spatial Pyramid Pooling-Fast) : SPPnet enables the CNN to take input with arbitrary size. Traditional CNNs require a fixed input size. However, by using the SPP pooling layer we can eliminate this restriction. Results show that besides this benefit, it can speed up the detection and classification tasks [52]. SPPF is a customization of the SPPnet; its structure is shown in Figure 4.5.

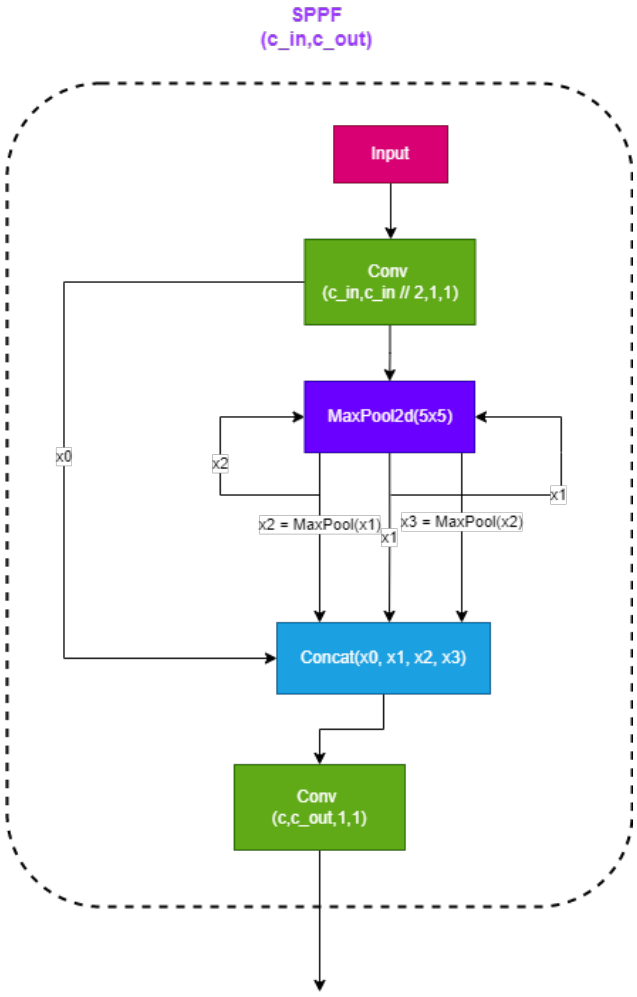


Figure 4.5 The network architecture of the SPPF layer

4.1.3 Metrics

In this section, we describe several principal metrics that are used in model evaluation. One of the most important metrics in object detection is mean average precision (mAP), which is calculated by using precision and recall. As all the metrics that are being used in this study are the ones that are used in the field of object detection, we are going to define them in this

context for further illustration, since they might be interpreted or used differently in other domains.

TP (True Positive): A correct detection of one particular ground truth. A **ground truth** in our context includes two elements: (1) the particular location and dimension of a component and (2) the label of the component.

FP (False Positive): An incorrect detection of one particular ground truth. It could be a detection of an object that does not exist or wrongly detect a ground truth.

FN (False Negative): A failure to detect an existing ground truth.

TN (True Negative): It is not being used in the object detection context. Since there are unlimited areas and bounding boxes exist in the image which are correctly not detected.

IoU (Intersection over Union): The identification of a correct or incorrect detection of ground truth for calculating TP, FP and FN in our context rely on the concept of Intersection over Union (IoU). IoU is determined by dividing the overlapping area over the union areas of the detected object and the ground truth. Figure 4.6 explains the IoU calculation. Generally, by specifying threshold t, a correct detection against a ground truth is achieved if $IoU \geq t$ and a correct label is classified. In this study, we set the threshold t to 0.6 according to literature.

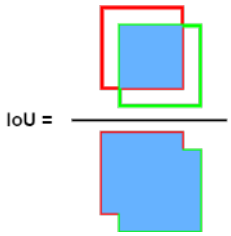


Figure 4.6 Intersection over Union. The red bounded box represents the ground truth and the green bounded box represents the detected object.

Precision: Precision is defined as the number of relevant objects which the model can detect. More accurately, it is equal to the number of TP divided by the number of TP + FP which is all the detections. So it measures the percentage of the correct detections among all detections.

Recall: Recall shows us, how is the model in terms of correctly detecting the objects among all the existing ground truths. It is calculated using the number of TP divided by TP + FN.

$$Precision = \frac{TP}{TP + FP} \qquad Recall = \frac{TP}{TP + FN}$$

To have a good object detector we have to try to increase the Precision and Recall along with each other. A higher Precision means the model detects all objects correctly; the highest number of Precision is equal to 1 when FP = 0 which means there is no wrong detection among all the detected objects. While the model tries to increase Precision, the Recall has to increase as well. If this does not happen, it means the model probably can detect some particular objects and is not generalizable enough. So the higher Recall shows the model is able to truly detect more ground truths of the image. The highest value of Recall is equal to 1 when FN = 0, which means there is no existing object that is not detected by the model, and also all the detections are correct. Since precision and recall are basic and also important for evaluating an object detector, the Precision x Recall curve is one of the informative plots to see the performance of a detector; it shows the trade-off between the two metrics.

mAP (Mean Average Precision): Generally, the area under the Precision x Recall curve (AUC) shows how well a detector can achieve a balanced performance on these two metrics.Since the Precision x Recall curve is usually a zigzag, measuring the AUC can be complicated. So we use another metric to estimate AUC, called Average Precision. We use n-point interpolation to measure the AP. The N-point interpolation approach to calculate the AP is described below :

$$AP_n = \frac{1}{n} \sum_{R \in \{0, \frac{1}{n-1}, \frac{2}{n-1}, \ldots, \frac{n-1}{n-1}\}} P_{interp}(R)$$

where :

$$P_{interp}(R) = \max_{\tilde{R}:\tilde{R} \geq R} P(\tilde{R})$$

$P_{interp}(R)$ returns the maximum precision in which the corresponding recall value is greater than the R. After we calculate the AP for each class of the dataset, we can measure the mean of Average Precision (mAP) using :

$$mAP = \frac{1}{N} \sum_{i}^{N} AP_i$$

in which AP_i is the average precision of the i_{th} class and N is the total number of classes

of the ground truths in the dataset.

4.2 Results

As we discussed before, the component detection part of our research has two steps. We are going to explain our results in the three following sections, respectively:

4.2.1 Training Yolov5 using the VINS dataset

The training set of our dataset contained 4339 images and the test set was comprised of 468 images.The pre-trained Yolov5 is retrained for 150 epochs. It took about 2 hours and 50 minutes to be completed. For the hyperparameters, the optimizer was Adam with a learning rate of 0.01. As the author of the Yolov5 recommended [50], we keep the default values of the other parameters.

During the training and testing of the model, we calculated the metrics to evaluate the model. Figure 4.7 shows that the precision and recall of the model were high throughout the training. As Figure 4.8 shows, the mAP of the network in the confidence level of 50% is about 0.803. Additionally, as we can see in the Precision x Recall curve (Figure 4.8), the precision and recall of each class are high at the same time, which shows the model not only predicts the components correctly but also detects most of the ground truths for each class.

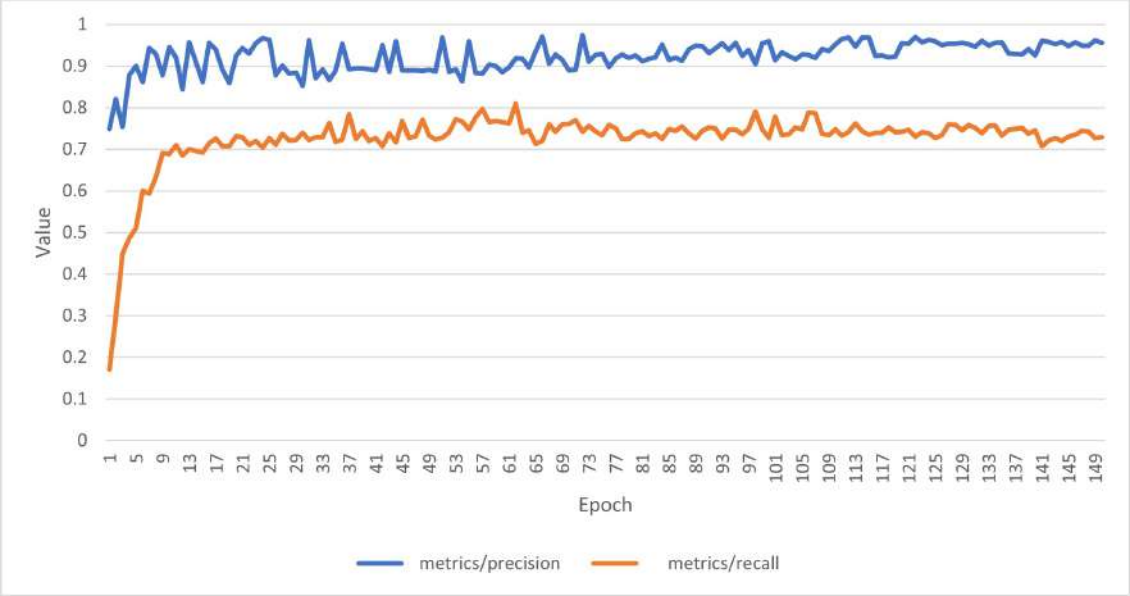


Figure 4.7 Precision and recall during training Yolov5 with the VINS dataset

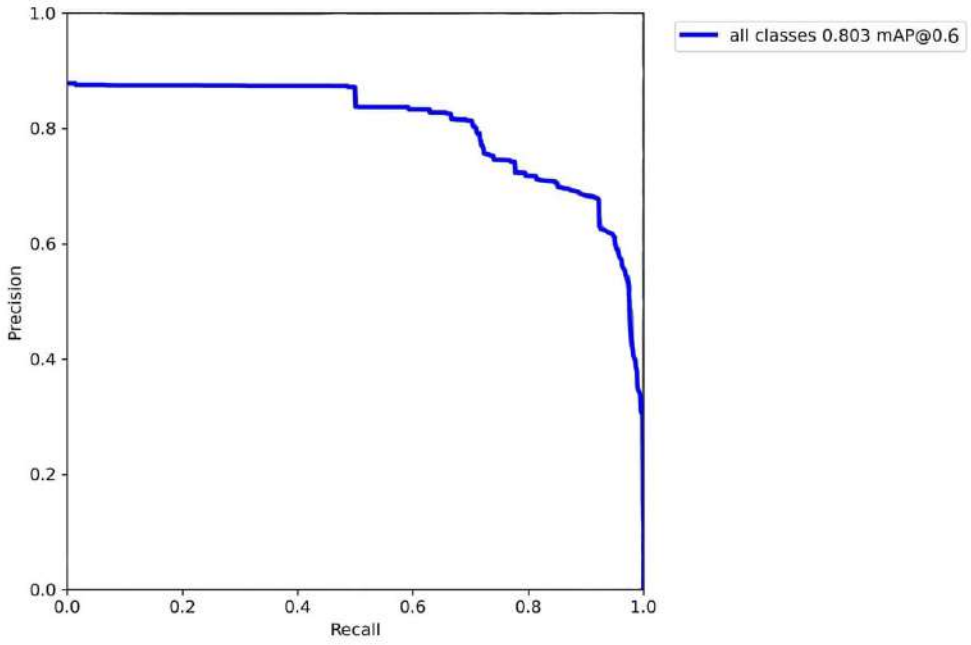


Figure 4.8 Precision x Recall curve on the trained Yolov5 with the VINS dataset

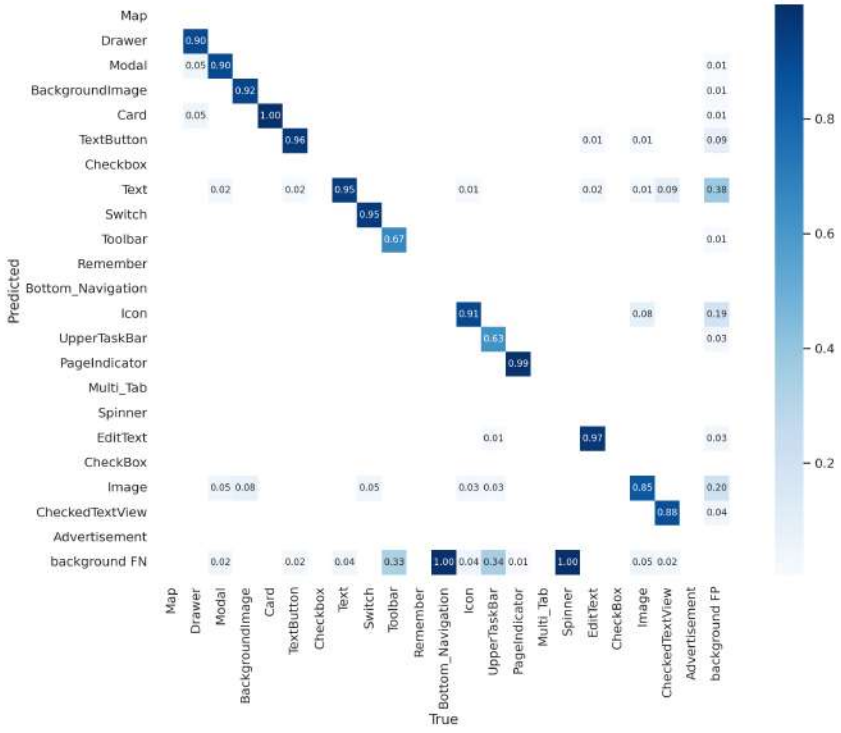


Figure 4.9 Confusion Matrix of the trained Yolov5 with the VINS dataset

We also plotted the confusion matrix (Figure 4.9) to explore which components that were better detected than others. As the matrix shows, five components (i.e., Checkbox, Remember,Multi_Tab, Advertisement, and Map) are not included in the test data set. Apart from those, about 70% of the components are predicted with an accuracy higher than 90%; these components included Card, TextButton, PageIndicator, Text, etc. Several components (e.g., Bottom_Navigation and Spinner) were not predicted at all. And components like Toolbar and UpperTaskBar are predicted but with lower accuracy. The reason for it is that there is a limited amount of these components in the dataset, so the model could not learn the structure of these elements well. In the confusion matrix, the background FN class represents those objects which Yolov5 has failed to detect and the background FP class represents those objects that were detected incorrectly.

In Figure 4.10 we bring some samples of the images in the test set. The ground truths of the images with their corresponding labels are shown on the left side. And the right-side image shows the predicted components with their corresponding confidence level.

Figure 4.10 Prediction samples of the trained Yolov5 using the VINS dataset

4.2.2 Fine-tuning of the preliminary network using generated images

After training the Yolov5 model with the VINS dataset, we used the Transfer learning technique to fine-tune the model in order for it to better detect the UI images generated with our StyleGAN-based approach. Transfer learning [53] addresses our problem of having a

smaller annotated dataset of generated UI images. Particularly, following the recommendation of [50], we froze the backbone of the Yolov5 pre-trained from the previous step and retrain the neck and head of the network for 30 epochs. We randomly split the 200 generated images that we annotated into 149 training images and 51 test images. The training took about 2.5 minutes to be completed. As Figure 4.11 shows, the precision and the recall both increased during training. It means the network can detect more ground truths over the epochs. The highest precision achieved was about 0.87 and the highest recall was 0.44. The mAP of all the classes is about 0.57 (see Figure 4.12), which is acceptable according to our limitation of preparing a more accurate dataset.

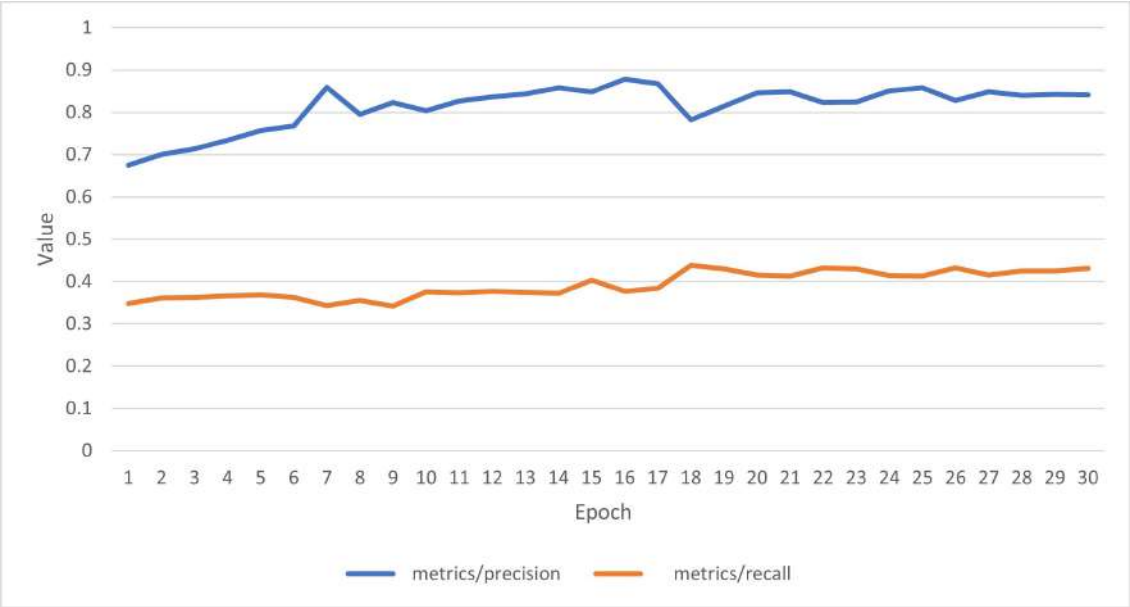


Figure 4.11 Precision and recall during fine-tuning of the preliminary network using the generated images

One of the main reasons for applying transfer learning is learning to detect the less clear component. Based on Figure 4.13, the confusion matrix shows the detector can detect the Image and the Icon components with more than 85% accuracy. However, there is a high false negative in many classes. This may be due to the limited number of training and testing data points in the annotated dataset.

Figure 4.14 illustrates the generated images with their ground truths on the right side, and the corresponding predictions of those ground truths are shown on the left-side image. As we can see in some of the predictions, the Text Button components are detected as Text + Icon. These errors are counted as FPs but semantically they are not wrong completely and are the result of the limited number of training images.

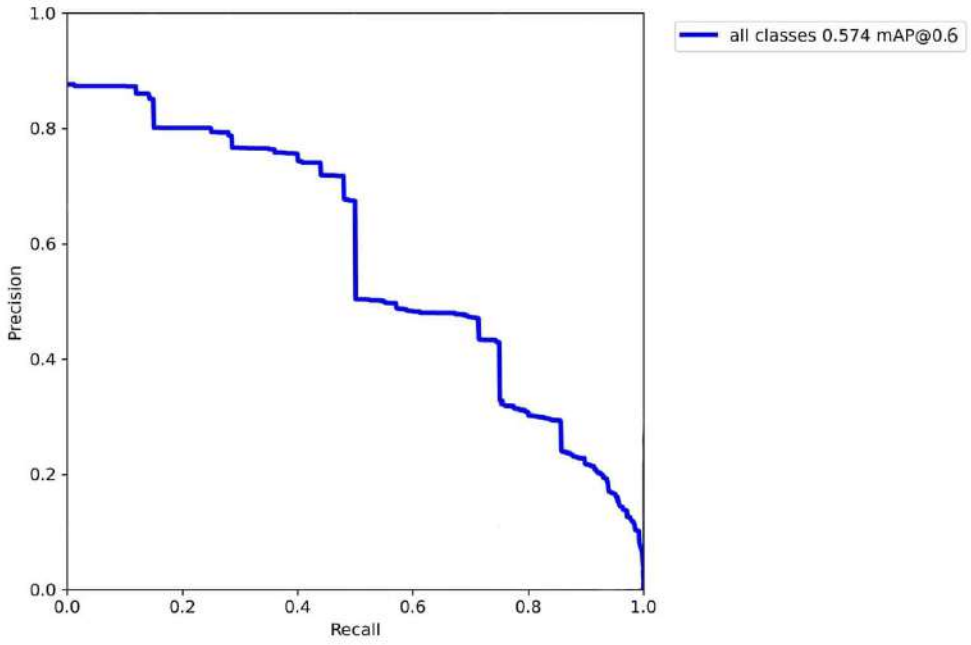


Figure 4.12 Precision x Recall curve of the fine-tuned Yolov5 using the generated images

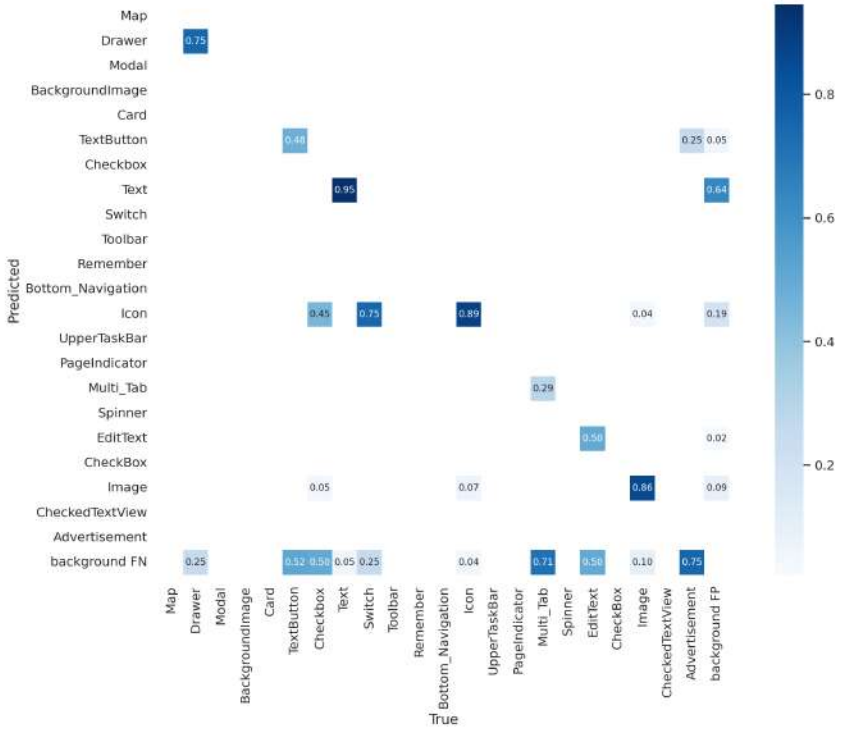


Figure 4.13 Confusion Matrix of the fine-tuned Yolov5 using the generated images

Figure 4.14 Prediction samples of the fine-tuned Yolov5

4.2.3 Testing the preliminary network on generated images

To better understand the effects of fine-tuning, we tested the preliminary model, which is trained only on the VINS dataset, on the same test set of section 4.2.2. The results show that the mAP is improved by almost 35% after fine-tuning the network. Figure 4.15 shows, the mAP of all classes is about 23%.

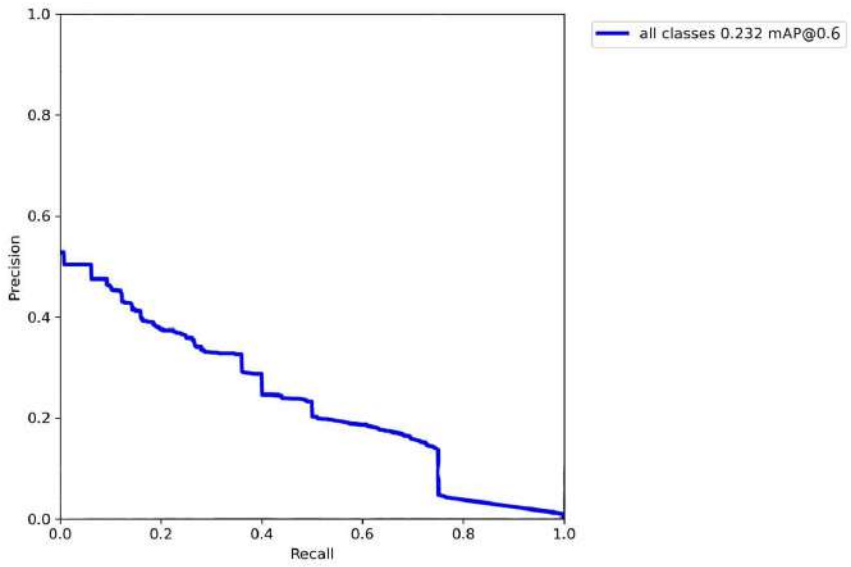


Figure 4.15 Precision x Recall curve of testing the preliminary network (the trained Yolov5 using the VINS dataset) on the generated images

As the confusion matrix (Figure 4.16) shows, Images and Icons are predicted with 32% and 53% of accuracy before fine-tuning. After fine-tuning, these values are increased to 89% and 86%, respectively. There are also improvements in the detection of Texts and EditText after fine-tuning (accuracy improved from 0.70 to 0.95 and from 0.0 to 0.5, respectively).

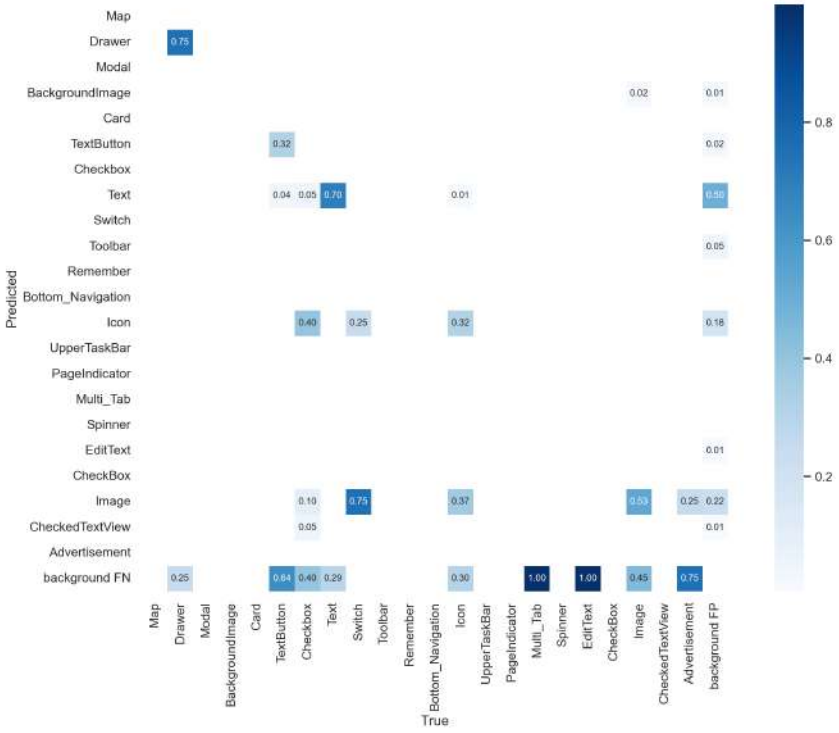


Figure 4.16 Confusion Matrix of testing the preliminary network (the trained Yolov5 using the VINS dataset) on the generated images

4.3 Discussion

Our results indicate that the Yolov5 model is able to achieve satisfactory performance on detecting UI components in both full-fledged images and generated images. For detecting the images which are not generated by the computer, the model achieved 0.80 mAP and overall precision of 98%. In comparison to the other works, Narayanan et al. [38] achieved 0.74 mAP by Cascade RCNN and 0.79 mAP by Yolov4 in detecting hand-drawn sketches; while focusing on a different detection task, our approach achieved a similar performance. Additionally, to compare with Chen et al.'s model [36] in detecting UI elements, we set the IoU threshold value to 0.9 which was used by Chen et al. and tested on our data; our model achieved 82% of precision and 53% of recall, which is 33% higher than Chen et al. model in precision and 2% lower in recall. To detect the components of the generated images, to our

knowledge, we are the first researchers who try to generate UI images and then detect their components. We targeted some components of generated UIs which have higher priority to be detected such as Images, Icons, and Texts. Based on our results, the Yolov5 model, fine-tuned with generated images, can detect those components with high accuracy and precision.

CHAPTER 5 CONCLUSION

5.1 Summary

We initiated our work concerning the inspiration of the UI/UX designers. Based on previous works and our experience, searching for examples before starting the design process is an ordinary but challenging step for every designer. Being inspired by the related works while avoiding to stuck in design drifts and design fixation is one of the most intricate parts of UI design.

To address this issue, we first proposed the StyleGAN-based approach which aims to balance both targeted and serendipitous inspiration in the user interface design practice. The evaluation studies highlighted the capacity of the StyleGAN-based approach in generating examples that are both relevant to the designers' work at hand and diverse for avoiding design fixation. Professional UI/UX practitioners appreciated such techniques as viable support in their day-to-day design practice. Our results also revealed possible improvements and design implications when a generative technique is used for supporting design inspiration. This part of our work demonstrates the potential of applying style-based generative machine learning techniques in the challenging context of design inspiration and creativity support. It opens a new direction and paves the road for future efforts in using advanced intelligent technology for supporting the creative, but at the same time constraint, design practice.

Based on the results of our StyleGAN-based approach, we extended our study to better support the inspiration part of the designers' work. Particularly, by detecting the UIs elements, we can give designers more ways to manipulate the UIs, either full-fledged ones or generated ones. As a result, we trained and evaluated the Yolov5 model to detect the UI elements. Our results show that the model can detect the UI elements on both full-fledged UI design images and generated images (with fine-tuning), with acceptable accuracy and precision.

5.2 Limitations

First, as mentioned before, the dataset we used for the StyleGAN-based approach (i.e., the Rico dataset) was published about five years ago and only included screenshots of Android applications. Although it is a large dataset that is frequently used in studies involving UI design artifacts, we do not know the inspirational power of our StyleGAN-based approach if a newer dataset or a dataset on another platform was used. We recognize that building a large-scale UI dataset is a non-trivial task. Even with the old dataset, our study demonstrated the

potential of our approach in supporting both serendipitous and targeted inspirations.

Second, our user study only included five participants. This is partially due to the challenges we experienced in recruiting participants during the pandemic. Although an in-person study might better facilitate the exploration of design examples, we were also only able to conduct remote studies with the participants. Additionally, while the evaluation is based on a realistic scenario, it only included one input UI and may not be able to incorporate all the real-world inspirational challenges related to UI design. However, our participants were all professional practitioners and represented diverse UI/UX-related experiences. With their professional experiences, they also reflected on their practice when examining the examples during the user study, providing a real-world perspective.

Third, since our StyleGAN-based approach is a novel approach to generate UIs using deep learning, and to the best of our knowledge, there is no generative model to generate a UI from scratch without using other pre-designed works, we needed to create an annotated dataset by ourselves. Csreating a dataset needs lots of effort and time to be accurate. Since we had limited time, we could create an annotated dataset with only 200 generated images. If we were able to annotate more images, we may be able to achieve better results for the model.

5.3 Future Directions

There are a couple of future directions for our study. First, during developing our StyleGAN-based approach, StyleGan2 was released [34], which has some improvements when compared to StyleGan1. One of the future works could be building the StyleGAN-based approach on StyleGan2 to see whether the results of the generator would be better.

Second, since designs are changing constantly, and designers always try to bring new ideas and use them into their designs, the other direction could be providing a large-scale new dataset of mobile (Android/IOS) or desktop applications' UIs. As we mentioned, the current datasets are out of date, so it could directly affect the results of our models. So having an updated dataset can potentially improve the results and usability of our research.

Finally, since our results demonstrate that our proposed models are able to detect the components with high accuracy, the next direction can be to make an end-to-end automated design-support system. Using the proposed generative model, we can generate the UIs, detect their components, and by rearranging the components or using the existing pre-designed components. In other words, we would have an automated toolkit that can support the entire design process of a new UI regarding the users' needs.

REFERENCES

[1] S. R. Herring *et al.*, "TweetSpiration," in *Proceedings of the 2011 annual conference extended abstracts on Human factors in computing systems - CHI EA '11*. New York, New York, USA: ACM Press, 2011, p. 2311. [Online]. Available: http://portal.acm.org/citation.cfm?doid=1979742.1979923

[2] D. G. Jansson and S. M. Smith, "Design fixation," *Design Studies*, vol. 12, no. 1, pp. 3–11, 1991. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0142694X9190003F

[3] R. L. Marsh, J. D. Landau, and J. L. Hicks, "How examples may (and may not) constrain creativity," *Memory & Cognition*, vol. 24, no. 5, pp. 669–680, sep 1996. [Online]. Available: http://www.springerlink.com/index/10.3758/BF03201091

[4] S. R. Herring *et al.*, "Getting Inspired! Understanding How and Why Examples are Used in Creative Design Practice," in *Proceedings of the 27th international conference on Human factors in computing systems - CHI 09*. New York, New York, USA: ACM Press, 2009, p. 87. [Online]. Available: http://dl.acm.org/citation.cfm?id=1518701.1518717http://dl.acm.org/citation.cfm?doid=1518701.1518717

[5] M. Gonçalves, C. Cardoso, and P. Badke-Schaub, "What inspires designers? Preferences on inspirational approaches during idea generation," *Design Studies*, vol. 35, no. 1, pp. 29–53, 2014. [Online]. Available: http://linkinghub.elsevier.com/retrieve/pii/S0142694X13000744

[6] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*. Cambridge, MA, USA: MIT Press, 2014, pp. 2672–2680.

[7] T. Karras, S. Laine, and T. Aila, "A style-based generator architecture for generative adversarial networks," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE Computer Society, 2019, pp. 4396–4405.

[8] T. Yeh, T.-H. Chang, and R. C. Miller, "Sikuli: Using gui screenshots for search and automation," in *Proceedings of the 22nd Annual ACM Symposium on User Interface Software and Technology*, ser. UIST '09. New York, NY, USA: Association for Computing Machinery, 2009, p. 183–192. [Online]. Available: https://doi.org/10.1145/1622176.1622213

[9] T. D. White, G. Fraser, and G. J. Brown, "Improving random gui testing with image-based widget detection," ser. ISSTA 2019. New York, NY, USA: Association for Computing Machinery, 2019, p. 307–317. [Online]. Available: https://doi.org/10.1145/3293882.3330551

[10] D. de Souza Baulé *et al.*, "Automatic code generation from sketches of mobile applications in end-user development using deep learning," *CoRR*, vol. abs/2103.05704, 2021. [Online]. Available: https://arxiv.org/abs/2103.05704

[11] T. Beltramelli, "pix2code: Generating code from a graphical user interface screenshot," *CoRR*, vol. abs/1705.07962, 2017. [Online]. Available: http://arxiv.org/abs/1705.07962

[12] M. Xie *et al.*, "Uied: A hybrid tool for gui element detection," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: Association for Computing Machinery, 2020, p. 1655–1659. [Online]. Available: https://doi.org/10.1145/3368089.3417940

[13] T. M. Thrash and A. J. Elliot, "Inspiration as a Psychological Construct," *Journal of Personality and Social Psychology*, vol. 84, no. 4, pp. 871–889, 2003.

[14] T. M. Thrash *et al.*, "The psychology of inspiration," *Social and Personality Psychology Compass*, vol. 8, no. 9, pp. 495–510, 2014.

[15] C. Eckert and M. Stacey, "Sources of inspiration: a language of design," *Design Studies*, vol. 21, no. 5, pp. 523–538, 2000.

[16] N. Bonnardel, "Creativity in design activities," in *Proceedings of the third conference on Creativity & cognition - C&C '99*. New York, New York, USA: ACM Press, 1999, pp. 158–165. [Online]. Available: http://portal.acm.org/citation.cfm?doid=317561.317589

[17] D. G. Jansson and S. M. Smith, "Design fixation," *Design Studies*, vol. 12, no. 1, pp. 3–11, 1991.

[18] P. Siangliulue *et al.*, "Providing Timely Examples Improves the Quantity and Quality of Generated Ideas," in *Proceedings of the 2015 ACM SIGCHI Conference on Creativity and Cognition - C&C '15*. New York, New York, USA: ACM Press, 2015, pp. 83–92.

[19] C. Chen *et al.*, "From ui design image to gui skeleton: A neural machine translator to bootstrap mobile gui implementation," in *Proceedings of the 40th International*

Conference on Software Engineering, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 665–676. [Online]. Available: http://doi.acm.org/10.1145/3180155.3180240

[20] T. Beltramelli, "Pix2code: Generating code from a graphical user interface screenshot," in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, ser. EICS '18. New York, NY, USA: ACM, 2018, pp. 3:1–3:6. [Online]. Available: http://doi.acm.org/10.1145/3220134.3220135

[21] A. Swearngin *et al.*, "Rewire," in *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems - CHI '18*. New York, New York, USA: ACM Press, 2018, pp. 1–12.

[22] K. Moran *et al.*, "Machine learning-based prototyping of graphical user interfaces for mobile apps," *IEEE Transactions on Software Engineering*, vol. 46, no. 2, pp. 196–221, 2020.

[23] M. Xie *et al.*, "Uied: A hybrid tool for gui element detection," in *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ser. ESEC/FSE 2020. New York, NY, USA: ACM, 2020, pp. 1655–1659. [Online]. Available: https://doi.org/10.1145/3368089.3417940

[24] C. Chen *et al.*, "From ui design image to gui skeleton: A neural machine translator to bootstrap mobile gui implementation," in *Proceedings of the 40th International Conference on Software Engineering*, ser. ICSE '18. New York, NY, USA: ACM, 2018, pp. 665–676. [Online]. Available: https://doi.org/10.1145/3180155.3180240

[25] B. Lee *et al.*, "Designing with interactive example galleries," in *Proceedings of the 28th international conference on Human factors in computing systems - CHI '10*. New York, New York, USA: ACM Press, 2010, p. 2257.

[26] F. Behrang, S. P. Reiss, and A. Orso, "GUIfetch," in *Proceedings of the 5th International Conference on Mobile Software Engineering and Systems - MOBILESoft '18*. New York, New York, USA: ACM Press, 2018, pp. 236–246. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3197231.3197244

[27] Y. Hashimoto and T. Igarashi, "Retrieving web page layouts using sketches to support example-based web design," in *Proceedings of 2nd Eurographics Workshop on Sketch-Based Interfaces and Modeling*, 2005.

[28] D. Ritchie, A. A. Kejriwal, and S. 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*. New York, New York, USA: ACM Press, 2011, p. 165.

[29] F. Huang, J. F. Canny, and J. Nichols, "Swire: Sketch-based user interface retrieval," in *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, ser. CHI '19. New York, NY, USA: ACM, 2019, pp. 104:1–104:10. [Online]. Available: http://doi.acm.org/10.1145/3290605.3300334

[30] Y. LeCun and Y. Bengio, *Convolutional Networks for Images, Speech, and Time Series*. Cambridge, MA, USA: MIT Press, 1998, pp. 255–258.

[31] S. Bunian *et al.*, "Vins: Visual search for mobile user interface design," in *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, ser. CHI '21. New York, NY, USA: ACM, 2021. [Online]. Available: https://doi.org/10.1145/3411764.3445762

[32] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[33] X. Huang and S. Belongie, "Arbitrary style transfer in real-time with adaptive instance normalization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1501–1510.

[34] T. Karras *et al.*, "Analyzing and improving the image quality of stylegan," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, 2020, pp. 8107–8116.

[35] T. Zhao *et al.*, "Guigan: Learning to generate gui designs using generative adversarial networks," in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, pp. 748–760.

[36] J. Chen *et al.*, "Object detection for graphical user interface: Old fashioned or deep learning or a combination?" *CoRR*, vol. abs/2008.05132, 2020. [Online]. Available: https://arxiv.org/abs/2008.05132

[37] X. Zhou *et al.*, "East: An efficient and accurate scene text detector," 04 2017.

[38] N. Narayanan, N. N. A. Balaji, and K. Jaganathan, "Deep learning for ui element detection: Drawnui 2020."

[39] D. Nikitko, "Stylegan-encoder," https://github.com/Puzer/stylegan-encoder, 2019.

[40] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *3rd International Conference on Learning Representations, ICLR*, 2015.

[41] J. Deng *et al.*, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[42] R. Zhang *et al.*, "The unreasonable effectiveness of deep features as a perceptual metric," *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Jun 2018. [Online]. Available: http://dx.doi.org/10.1109/cvpr.2018.00068

[43] D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatial–temporal data," *Data & knowledge engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[44] A. Krizhevsky, I. Sutskever, and G. Hinton, "Imagenet classification with deep convolutional neural networks," *Neural Information Processing Systems*, vol. 25, 01 2012.

[45] B. Deka *et al.*, "Rico: A mobile app dataset for building data-driven design applications," in *Proceedings of the 30th Annual Symposium on User Interface Software and Technology*, 2017.

[46] M. Heusel *et al.*, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6629–6640.

[47] S. R. Herring, B. R. Jones, and B. P. Bailey, "Idea generation techniques among creative professionals," *Proceedings of the 42nd Annual Hawaii International Conference on System Sciences, HICSS*, pp. 1–10, 2009.

[48] D. Mounter *et al.*, *Design Systems Handbook*. InVision, 2019.

[49] M. Heusel *et al.*, "Gans trained by a two time-scale update rule converge to a nash equilibrium," *CoRR*, vol. abs/1706.08500, 2017. [Online]. Available: http://arxiv.org/abs/1706.08500

[50] G. Jocher *et al.*, "ultralytics/yolov5: v3.1 - Bug Fixes and Performance Improvements," Oct. 2020. [Online]. Available: https://doi.org/10.5281/zenodo.4154370

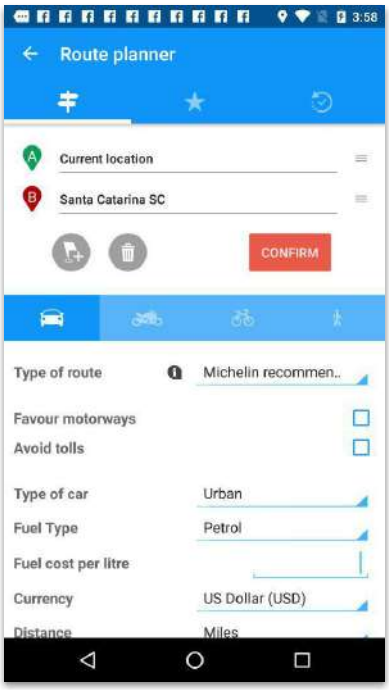
[51] C.-Y. Wang *et al.*, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 390–391.

[52] K. He *et al.*, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *CoRR*, vol. abs/1406.4729, 2014. [Online]. Available: http://arxiv.org/abs/1406.4729

[53] F. Zhuang *et al.*, "A comprehensive survey on transfer learning," *Proceedings of the IEEE*, vol. 109, no. 1, pp. 43–76, 2021.

APPENDIX A SAMPLES OF THE USER EVALUATION SLIDES

Source Image



Exp 1-0


1


2


3


4


5


6


7


8


9


10


11

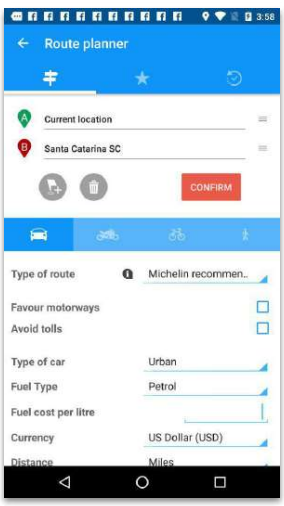

12


13


14


15


16




17


18


19


20


21


22


23


24


25

Exp 1-1


1


2


3


4


5


6


7


8


9


10


11

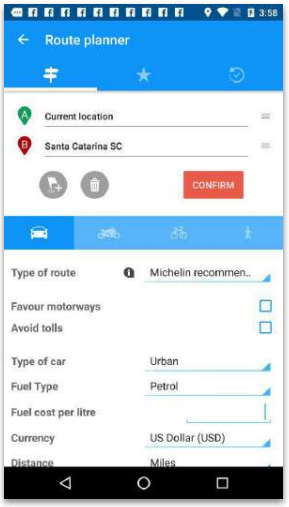

12


13


14


15


16




17


18


19


20


21


22


23


24


25

Exp 2-1.1


1


2

3

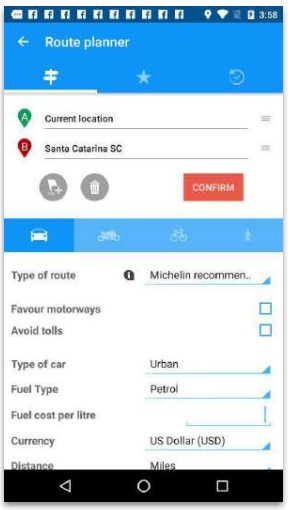

4

5


6


7

8




9

10


11


12


13


14


15


16


17


18


19


20


21


22

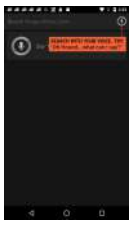

23


24

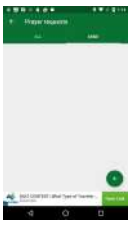

25

Exp 2-2


1


2

3


4

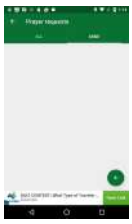

5


6


7


8


9


10


11


12

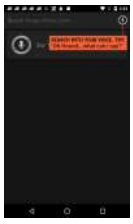

13


14


15


16


17


18


19


20


21


22


23


24

25

Exp 2-3


1


2


3

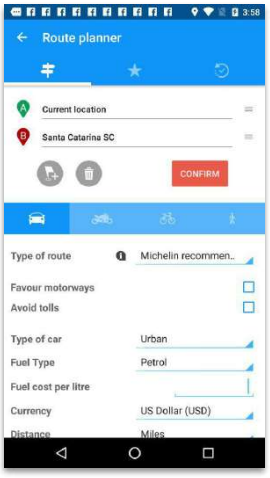

4


5

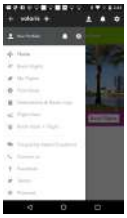

6


7

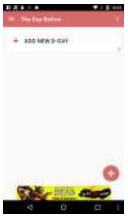

8




9


10


11


12


13


14


15


16


17


18


19


20


21


22


23


24


25

Exp 2-4


1


2


3


4


5


6


7


8

9


10


11


12


13


14


15


16


17


18


19

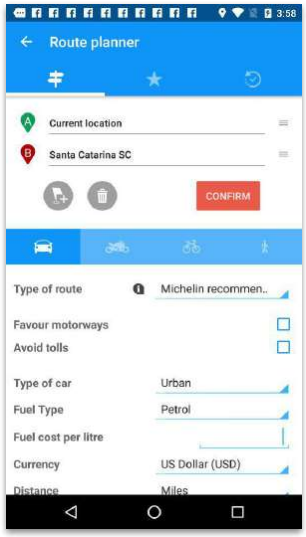

20


21


22


23


24




25

EXP 3-1.2


1
2
3
4
5
6
7
8
Route planner
Current location
Santa Catarina SC
CONFIRM
Type of route
Michelin recommen..
Favour motorways
Avoid tolls
Type of car
Urban
Fuel Type
Petrol
Fuel cost per litre
Currency
US Dollar (USD)
Distance
Miles
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26

APPENDIX B USER STUDY QUESTIONS

These questions were asked before the beginning of the study :

- How long have you been working as a user experience designer?
- How many design projects have you contributed to?
- What are the biggest challenges do you consider during your design practice?
- How often do you search for examples before starting your design process (for inspiration purposes)?

For each set of images which is related to each condition of the study, we asked practitioners the following questions :

- What are the two things you like about the examples generated for the preliminary design?
- What are the two things you dislike about the examples generated for the preliminary design?
- How relevant are the design examples with regard to the preliminary design?
- How diverse are the design examples with regard to the preliminary design?
- How effective do you think these design examples can inspire the designer to improve the preliminary design?

At the end of the session, we asked designers more general questions about their feeling and comments about our work :

- In general, how do you think about the generated design examples?
- Supposed that someone has developed a design tool that uses this technique, how well will this tool be integrated in your design workflow?
- Based on all designs which I showed you, which one is more effective for inspiration? wireframes or images(real UIs and generated ones)?
- Do you have any final comments or questions?