



저작자표시-비영리-변경금지 2.0 대한민국

이용자는 아래의 조건을 따르는 경우에 한하여 자유롭게

- 이 저작물을 복제, 배포, 전송, 전시, 공연 및 방송할 수 있습니다.

다음과 같은 조건을 따라야 합니다:



저작자표시. 귀하는 원저작자를 표시하여야 합니다.



비영리. 귀하는 이 저작물을 영리 목적으로 이용할 수 없습니다.



변경금지. 귀하는 이 저작물을 개작, 변형 또는 가공할 수 없습니다.

- 귀하는, 이 저작물의 재이용이나 배포의 경우, 이 저작물에 적용된 이용허락조건을 명확하게 나타내어야 합니다.
- 저작권자로부터 별도의 허가를 받으면 이러한 조건들은 적용되지 않습니다.

저작권법에 따른 이용자의 권리는 위의 내용에 의하여 영향을 받지 않습니다.

이것은 [이용허락규약\(Legal Code\)](#)을 이해하기 쉽게 요약한 것입니다.

[Disclaimer](#)

Master's Thesis

GUIComp: A GUI Design Assistant  
with Real-Time, Multi-Faceted Feedback

Chunggi Lee

Department of Computer Science and Engineering

Graduate School of UNIST

2020

GUIComp: A GUI Design Assistant  
with Real-Time, Multi-Faceted Feedback

Chunggi Lee

Department of Computer Science and Engineering

Graduate School of UNIST


**GUIComp: A GUI Design Assistant  
with Real-Time, Multi-Faceted Feedback**

A thesis  
submitted to the Graduate School of UNIST  
in partial fulfillment of the  
requirements for the degree of  
Master of Science

**Chunggi Lee**

12. 04. 2019 of submission

Approved by



---

Advisor

Sungahn Ko

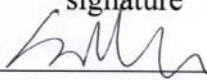
# GUIComp: A GUI Design Assistant with Real-Time, Multi-Faceted Feedback

Chunggi Lee

This certifies that the thesis of Chunggi Lee is approved.

12/04/2019

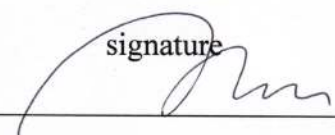
signature



---

Advisor: Sungahn Ko

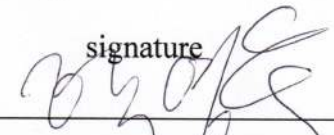
signature



---

Won-Ki Jeong: Thesis Committee Member #1

signature



---

Young-Woo Park: Thesis Committee Member #2

## Abstract

Maintaining the high quality of mobile Graphic User Interfaces (GUIs) is essential to make a mobile application more useful for its users. For this reason, there are many mobile GUI prototyping tools. However, users may face challenges while designing graphic user interfaces, due to a lack of relevant skills, experience, and guidance. In particular, the users can often be overwhelmed in the first prototyping stage and it is hard to recognize mistakes for users in advance. These challenges can be significantly alleviated by supporting immediately feedback (e.g., recommendation and evaluation) while designing GUI. Due to absence of feedback, users still rely on their own ideas and intuition with mobile GUI prototyping tool that requires time-consuming and error-prone design procedures.

In this thesis, we aim at investigating what causes users to become frustrated during the design process, and how to resolve the issues. To achieve this goal, first, we conducted semi-structured interviews with 16 users, to understand their challenges with an existing tool, and to identify features that can facilitate the design process. Second, based on the interview results, we built a GUI prototyping tool, called GUIComp that provides real-time multi-faceted feedback on a user's current design, such as visual complexity scores, viewer's attention heatmap, and recommended example designs. Third, we performed a between-user experiment, where 30 participants were asked to create mobile GUIs with either GUIComp or an existing design tool. Fourth, we asked 26 online workers to assess the designs produced in the experiment. The results indicate that GUIComp users produced more acceptable designs than the non-GUIComp users and designing with GUIComp results in a more enjoyable, satisfactory, and affordable user experience during the design process than that with the existing tool. We discuss how to design for multi-faceted feedback while designing GUI to effectively interact human and AI and limitations of the study.

The fundamental idea of this thesis is to go beyond traditional GUI prototyping to create more acceptable designs to general users using real-time multi-faceted feedback. The resulting systems establish a research framework where real-time multi-faceted feedback can alleviate the challenges while designing GUIs.



## Contents

I	Introduction . . . . .	1
II	Related work . . . . .	3
	2.1 GUI Prototyping Tools . . . . .	3
	2.2 Approaches for Assisting GUI Prototyping . . . . .	4
	2.3 Metrics for Measuring Visual Complexity . . . . .	4
	2.4 Computational Method for Graphic User Interface Design . . . . .	5
III	Semi-Structured Interviews To Identify Difficulties Encountered During GUI Prototyping . . . . .	14
	3.1 Participants . . . . .	14
	3.2 Procedure . . . . .	14
	3.3 Identifying Difficulties Encountered by Participants . . . . .	15
	3.4 Requirements of a Tool for Assisting GUI Prototyping . . . . .	16
IV	GUIComp: Add-on for GUI Design Guidance . . . . .	18
	4.1 Tool Overview . . . . .	18
	4.2 Capturing and Processing Users' Design on Canvas . . . . .	19
	4.3 Recommendation Panel . . . . .	19
	4.4 Attention Panel . . . . .	21
	4.5 Evaluation Panel . . . . .	21



4.6	Implementation Notes . . . . .	23
V	User Study Design . . . . .	24
5.1	Participants, Procedure, Apparatus, and Tasks . . . . .	24
VI	Experiment Results, Analysis, and Implications . . . . .	27
6.1	GUIComp Users Produce Designs Acceptable to General Users (RQ1) . .	27
6.2	Using GUIComp Is Enjoyable, Satisfactory, and Affordable (RQ2) . . . .	27
6.3	Users Employ Multi-faceted Feedback for Overcoming Difficulties in the Iterative Design Process (RQ3) . . . . .	28
VII	Lessons, Limitations, and Discussion . . . . .	31
7.1	Limitations . . . . .	32
VIII	Conclusion . . . . .	33
	Acknowledgements . . . . .	34
	References . . . . .	35

## List of Figures

1	Total number of active apps on the apps store by 2020 [1] (Left) and similar functionality weather apps with different GUI designs (Right). . . . .	1
2	Adobe Photoshop toolbox [2] with visual tools, select tools, enhance tools, draw tools, modify tools, and color (Top) and Introduction of InVision [3] (Bottom). . . . .	3
3	Structure of auto-encoder with three layers: input layer, hidden layer and output layer. . . . .	5
4	A biological neuron in human nervous system. An axon is connected with dendrites of other neurons. On the other hand, a dendrite is connected with axons of other neurons. A synapse is a connected point between synapse and dendrite. Here one neuron is signaled to another neurons. . . . .	6
5	A mathematical model of neuron. . . . .	7
6	Commonly used activation functions (a) sigmoid function, (b) tanh function, and (c) relu function . . . . .	7
7	High-level overview of KNN. . . . .	8
8	Overall architecture of convolutional neural network that consists convolution layer, pooling layer, and fully-connected layer. . . . .	9
9	Procedure of convolution operator with 4 x 4 image and 3 x 3 filter for getting 3 x 3 computed feature map. . . . .	10
10	Different filter type (a) Edge Detection, (b) Sharpen, and (C) Blur. . . . .	10
11	Procedure of max pooling. . . . .	11
12	(a) A conventional convolution operation, (b) A transpose convolution operation, and (c) A transpose deconvolution operation. . . . .	11

13	Fully convolution network for semantic segmentation and final output (heatmap) of tabby cat. . . . .	12
14	Fully convolution network and prediction result of FCN-32s which is 32x upsampled feature map from pool5 and FCN-16s which is 16x upsampled from pool4 and pool5. . . . .	13
15	Three feedback panels in the GUIComp (Add-on)–Evaluation Panel (B), Recommendation Panel (C), and Attention Panel (D). (A1)–(A3) are interfaces of an online GUI prototyping tool (the base tool). . . . .	18
16	. . . . .	19
17	Online workers’ ratings of the participants’ designs (left) and exit survey (right) results. The dot and the whisker represent the mean and the 95% confidence interval, respectively. . . . .	28
18	User gaze sequence data on the feedback panels. . . . .	29

## I Introduction

Today, graphic user interface is everywhere. The graphic user interfaces play a pivotal role in attracting potential customers as number of web or mobile applications (or “apps”) increase Figure 1. Many similar functionality apps have been published in different GUI designs. Therefore, user-friendly interfaces are important because the interfaces make interactions simple, effective and engaging as being connected customer loyalty and satisfaction. However, different customers have differing criteria when evaluating designs, which makes the design tasks challenging.

One suggested approach to effectively produce acceptable designs is to perform iterations among design stages [4]: produce many shareable design alternatives [5,6], compare the competitiveness of the alternatives [7], and evaluate how general users feel while employing the designs (e.g., attention [8]). Iterative design also helps people improve their design skills for quickly prototyping alternatives and allows training time with many design trials to achieve the desired goals (e.g., simplicity, theme expressions, visual aesthetics, and creativity).

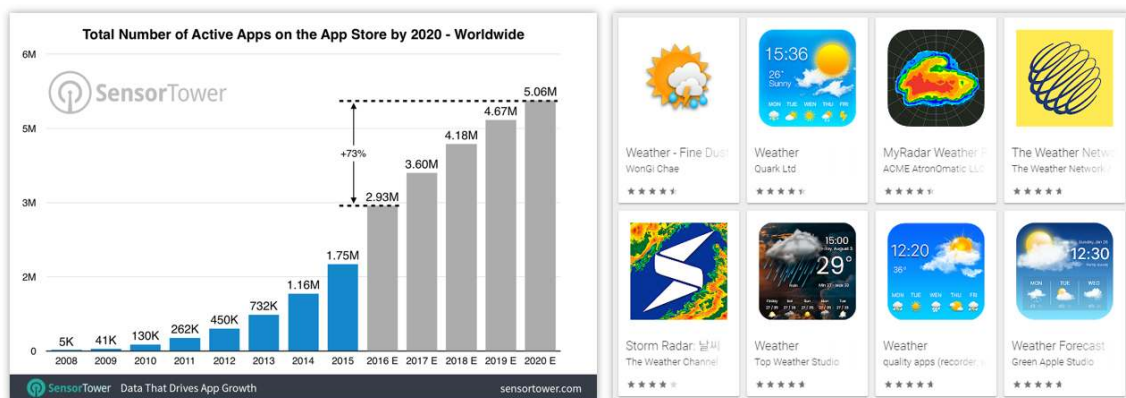


Figure 1: Total number of active apps on the apps store by 2020 [1] (Left) and similar functionality weather apps with different GUI designs (Right).

Although performing iterative design is effective, doing so may not be easy. In particular, people with little design experience seem to be more frustrated during the design process. Examples include students who need to produce designs for a class or developers who need to not only program, but also design Graphical User Interfaces (GUIs) for the program (e.g., freelance or independent app developers [9]). We conjecture that they encounter obstacles whenever they iterate the design stages. For example, beginning with a blank canvas in the first prototyping stage often can be overwhelming due to the difficulty in conceptualizing designs and harmonizing the concepts with the given design constraints (e.g., layouts or color themes) [10]. Non-professional users are likely to make mistakes through design iterations, a few of which may result in design failure in the end [4]. Generally speaking, it is hard for them to recognize mistakes in advance. Even if they recognize mistakes, they are still left without any guidance on how to fix the mistakes and how to evaluate their resulting designs. As such, there is a need for a tool that can help the people throughout the design stages.

In this work, we aim at designing a tool for assisting users with little design experience. To achieve this goal, we conducted semi-structured, in-depth interviews with 16 participants to understand difficulties of mobile GUI design. Based on the observed the difficulties from the interviews, we designed a tool, **GUICompanion (GUIComp)** by integrating three different types of feedback mechanisms: *recommendation*, *evaluation*, and *attention*. We designed GUIComp as an add-on to browsers, so that it can be easily linked to existing GUI prototyping tools. In an experiment, we linked GUIComp with a base tool, called Kakao Oven [11] and asked 30 participants to design GUIs for user profile input and a list of products with either GUIComp or the base tool. Then we asked 47 Amazon Mechanical Turk (AMT) workers to assess the resulting designs. The results indicate that the proposed tool helped users who lack experience in mobile GUI design to easily begin and develop at GUI design within a short period. With GUIComp, the users were able to efficiently start their designing with examples, check whether their design seemed acceptable, and produced it as they intended with visual complexity scores. The results also show that mobile GUIs produced with GUIComp were more acceptable designs to general users than those produced with the base tool. The participants reported that designing mobile GUIs with GUIComp was more enjoyable, satisfactory, and affordable than with the base tool. We think our findings and approach of providing real-time multi-faceted feedback can be applicable to non-mobile GUI design tasks (e.g., web design).

The contributions of this work include 1) characterization of the difficulties that users encounter while designing GUIs by conducting semi-structured interviews, 2) design and evaluation of GUIComp that provides users with real-time, multi-faceted feedback for mobile GUI design and facilitates an iterative design process, and 3) lessons learned from our experience and design guidelines for GUI prototyping assistance.

## II Related work

### 2.1 GUI Prototyping Tools

Many tools have been developed for GUI prototyping. Adobe Photoshop and Illustrator are popular among designers for providing many toolboxes with which designers can draw various simple and complex shapes Figure 2. Several tools have been proposed to allow easy prototyping by reducing the time for interaction and adding animation Figure 2. Adobe XD [12], InVision [3], Sketch [13], and Principle [14] are tools in this category. Other tools have also been proposed to allow more rapid prototyping, such as UXPin [15], Proto.io [16], and Axure [17], all of which are equipped with a large number of ready-made elements. Despite the tools' helpfulness in rapid prototyping, these tools may not be sufficient for users, because they do not provide timely feedback on the users' designs to improve the quality.

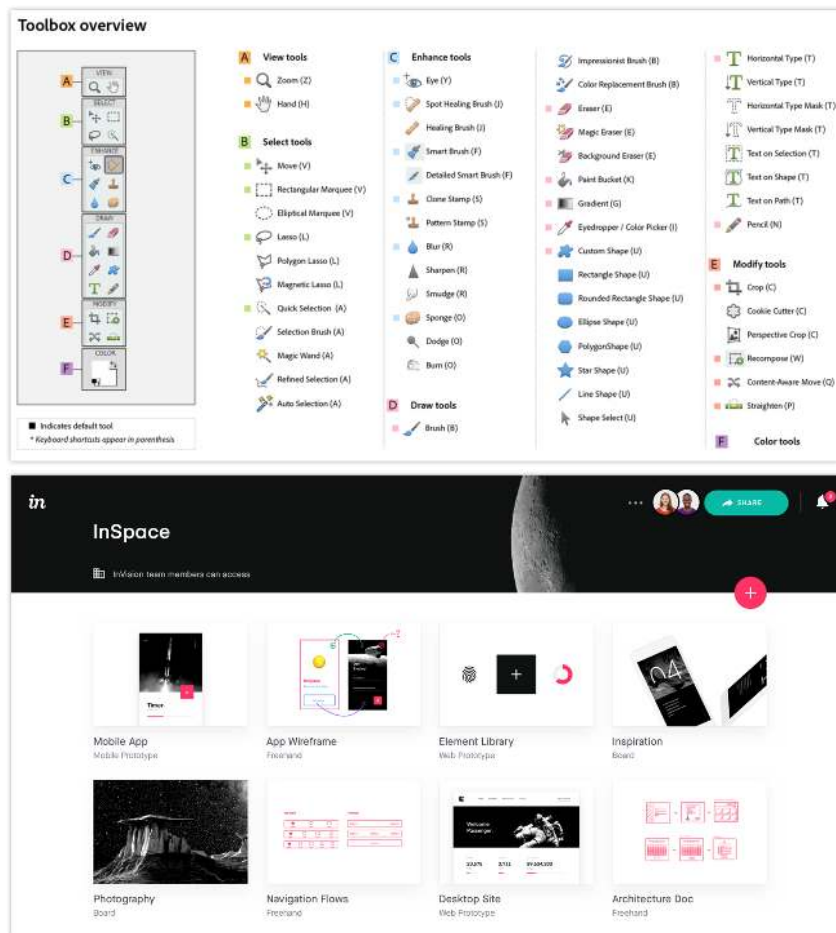


Figure 2: Adobe Photoshop toolbox [2] with visual tools, select tools, enhance tools, draw tools, modify tools, and color (Top) and Introduction of InVision [3] (Bottom).

## 2.2 Approaches for Assisting GUI Prototyping

Many approaches exist for assisting GUI prototyping, and they can be categorized into three groups. The first approach allows users to browse examples. Existing studies report that browsing examples inspires designers and leads to alternative designs [18–20]. For example, ‘d.tour’ proposed by Ritchie et al. [21] allows users to search for examples by using user-provided keywords. The second approach enables users to interactively explore, extract, and use specific design elements from the examples. The extracted elements can be directly used as a template or building block in prototyping. Tsai and Chen’s framework [22] is an early system for supporting template-based mobile user interface design. However, in this approach, users are limited in terms of which elements they can extract, such as structures, layouts, and styles. Rewire [23] automatically converts a UI screenshot to a vector representation of design elements, which users can import into their graphic editing tools and revise for their own purposes. The third approach aims to partially or fully automate the design stages using computational methods. O’Donovan et al. [24,25] develop DesignScope that provides users with various layout suggestions for refinement and brainstorming. Todi et al. [26] propose an approach that produces familiar designs by restructuring existing layouts. In contrast, ReDraw [27] and pix2code [28] are designed to produce a code, which can be executed to create an application, generated by a convolutional neural network model based on a sample UI image. Moran et al. [29] propose an approach that automatically checks whether the users’ design implementation follows a design guideline, provided as an image. Despite the usefulness, automated approaches have several weaknesses. First, users are often excluded from the automatic design recommendation process, except for the choice of its input (e.g., an image of example UI). In addition, the automated approach does not guarantee the high quality of its output; the quality is often delegated to users. Automated methods can also prevent users from freely envisioning creative designs constrained by their output examples, as reported in previous studies [24,30]. These weaknesses can be overcome by supporting users to lead the design processes, but rare studies exist on how to support. We investigate how humans can lead design efforts, with the machine acting as a smart companion during the process.

## 2.3 Metrics for Measuring Visual Complexity

There exist metrics for evaluating web pages and mobile GUIs [31–34]. Web page evaluation metrics focus on the visual complexity of websites with individual and numeric factors (e.g., word count or graphics count [31]), but those for mobile GUIs concern detailed elements that contribute to the overall visual complexity of designs, such as words, color, image counts and sizes, symmetry, balance [31,32], and layouts [33]. There are other approaches for evaluating mobile GUIs’ visual complexity [35–38]. Miniukovich and Angeli [37] propose metrics for measuring visual impressions of mobile GUIs and demonstrated that their model can explain 40% of the variation of subjective visual complexity scores. Their experiment also revealed that dominant

colors and the symmetry of UIs are correlated with aesthetics, while color depth information is more correlated with visual complexity. The model is further extended to measure the visual complexity of desktop GUIs [39]. Riegler and Holzmann [35] propose eight visual complexity metrics for evaluating mobile GUIs. Their metrics evaluate the quality of the number of UI elements, misalignment, imbalance, density, element smallness, inconsistency, color, and typographic complexity. The metrics were mathematically formulated and evaluated through a user study. We use Riegler and Holzmann’s metrics [35] in this work, because they can present visual complexity in numeric values in real-time.

## 2.4 Computational Method for Graphic User Interface Design

### Auto-Encoder

The **auto-encoder** is a neural network which is a kind of unsupervised learning structure with three layers: input layer, hidden layer, and output layer. It aims to learn a representation or encoding for a set of data like dimensionality reduction. Along with the reduction part, the reduced encoding representations are generated most similar to its original input. It trains two parts an encoder and a decoder in Figure 3 by simply coping inputs to outputs.

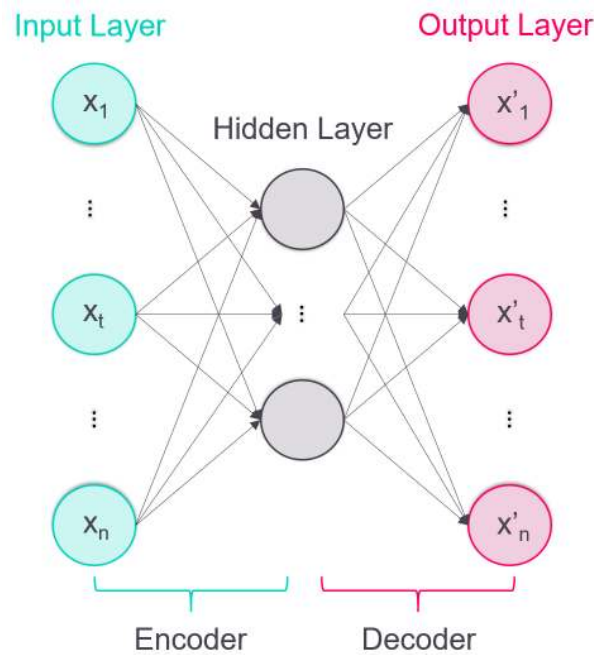


Figure 3: Structure of auto-encoder with three layers: input layer, hidden layer and output layer.

$$\begin{aligned}
 \text{Encoder} : y &= f_{\theta}(x) = \sigma(Wx + b) \\
 \text{Decoder} : x' &= g_{\theta'}(y) = \sigma'(W'y + b')
 \end{aligned}
 \tag{1}$$



where  $x, y$ , and  $x'$  are input vector, latent representation, and reconstructed vector, respectively from 0 to 1. Note  $\theta$  and  $\theta'$  are encoder parameter and decoder parameter, respectively. The parameter  $\theta$  contains  $W$  and  $b$  as encoder of weight matrix and bias vector. Also, the parameter  $\theta'$  contains  $W'$  and  $b'$  as decoder of weight matrix and bias vector. Here,  $\sigma$  is an element-wise activation function such as a sigmoid function or rectified linear unit.

The parameter sets of the auto-encoder which are  $\theta$  and  $\theta'$  are optimized to minimize the reconstruction error (e.g., mean squared error):

$$CostFunction : J(\Theta) = \underset{\theta, \theta'}{\operatorname{argmin}} \frac{1}{n} \sum_{i=1}^n L(x_i, x'_i) \quad (2)$$

where  $L$  presents a loss function  $L(x, x') = \|x - x'\|^2 = \|x - \sigma'(W'(\sigma(Wx + b)) + b')\|^2$ .

### Fully Connected Layer (Dense Layer)

A **fully-connected layer** is a neural network between two adjacent layers are fully pairwise connected, but there is no connection within a single layer. Also, this layer is a collection of links in an acyclic graph and the output of a neural can be the input of another neuron. This layer describes the nervous system of human. In the human nervous system, a neuron is connected to axons of many other neurons. The strength of connected synapses determines influence of connected neurons. A signal is generated and transmitted to another neurons through the axon if the strength is greater than a certain threshold.

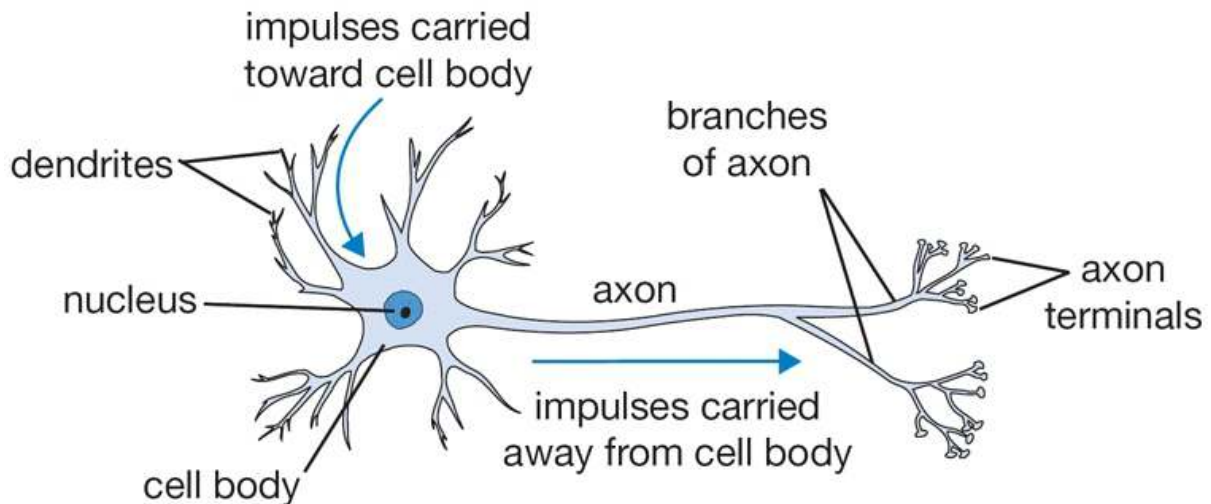


Figure 4: A biological neuron in human nervous system. An axon is connected with dendrites of other neurons. On the other hand, a dendrite is connected with axons of other neurons. A synapse is a connected point between synapse and dendrite. Here one neuron is signaled to another neurons.

This mathematical model corresponds to the human nervous system:

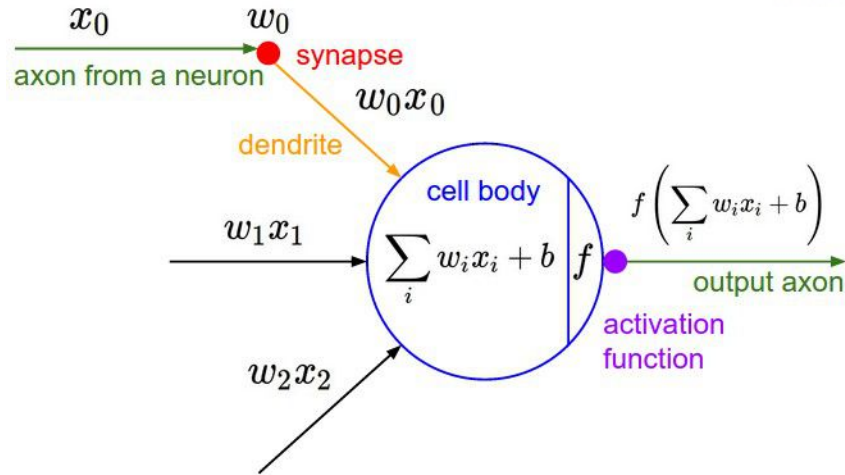


Figure 5: A mathematical model of neuron.

- $x_0, x_1, \text{ and } x_2$ : The amount of signal from axons of input neuron.
- $w_0, w_1, \text{ and } w_2$ : Strength of synapse (influence of input neuron).
- $f$ : Activation function which determines a certain threshold.

The  $\sum_i w_i x_i$  plays the same role in the mathematical model like several axons are connected to a dendrite in the human nervous system. It becomes input of activation function and a signal is generated and transmitted to next layer if the strength is greater than a certain threshold of activation function.

### Activation Function

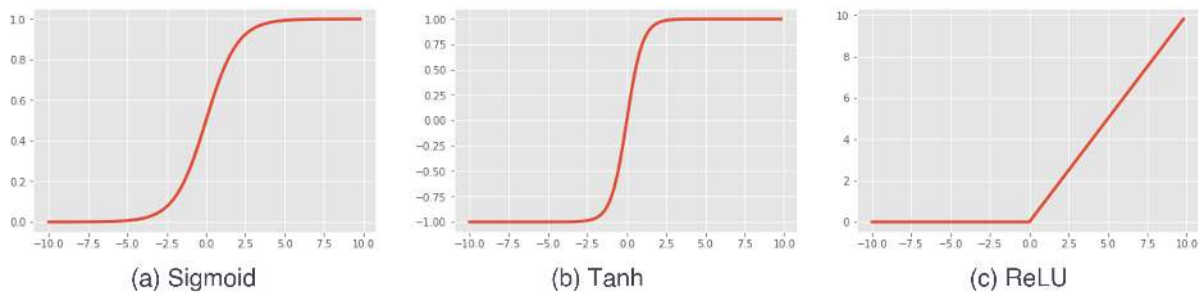


Figure 6: Commonly used activation functions (a) sigmoid function, (b) tanh function, and (c) relu function

The **activation function** defines the output of neuron given an input or set of inputs. A common activation functions can be sigmoid, hyperbolic tangent, and rectified linear unit:

$$\begin{aligned}
 \text{Sigmoid} : \sigma(x) &= \frac{1}{1+e^{-x}} \\
 \text{Tanh} : \tanh(x) &= 2\sigma(2x) - 1 \\
 \text{ReLU} : f(x) &= \max(0, x)
 \end{aligned}
 \tag{3}$$

The **sigmoid** has non-linearity between  $[0, 1]$  and "S"-shaped curve characteristic as shown in the Figure 6 (a). Large negative and positive numbers become 0 and 1, respectively.

The **tanh** is hyperbolic tangent and has non-linearity as shown in the Figure 6 (b). This function is squashing a real-valued number from range  $[-1, 1]$ .

The **ReLU** is simply thresholded at zero. It is enable to accelerate convergence of training compared to the sigmoid/tanh functions due to its linear and non-saturating form. However, it can "die" during training which means the neuron will never activate on any datapoint again.

### K-nearest Neighbor Algorithm

The **k-nearest neighbors algorithm (KNN)** can be used to solve both classification and regression tasks as a supervised machine learning algorithm. Output of classification and regression are a class membership and property value. This algorithm assumes that similar things are near to each other. The KNN algorithm captures similarity using some metrics (e.g., distance, proximity, or closeness).

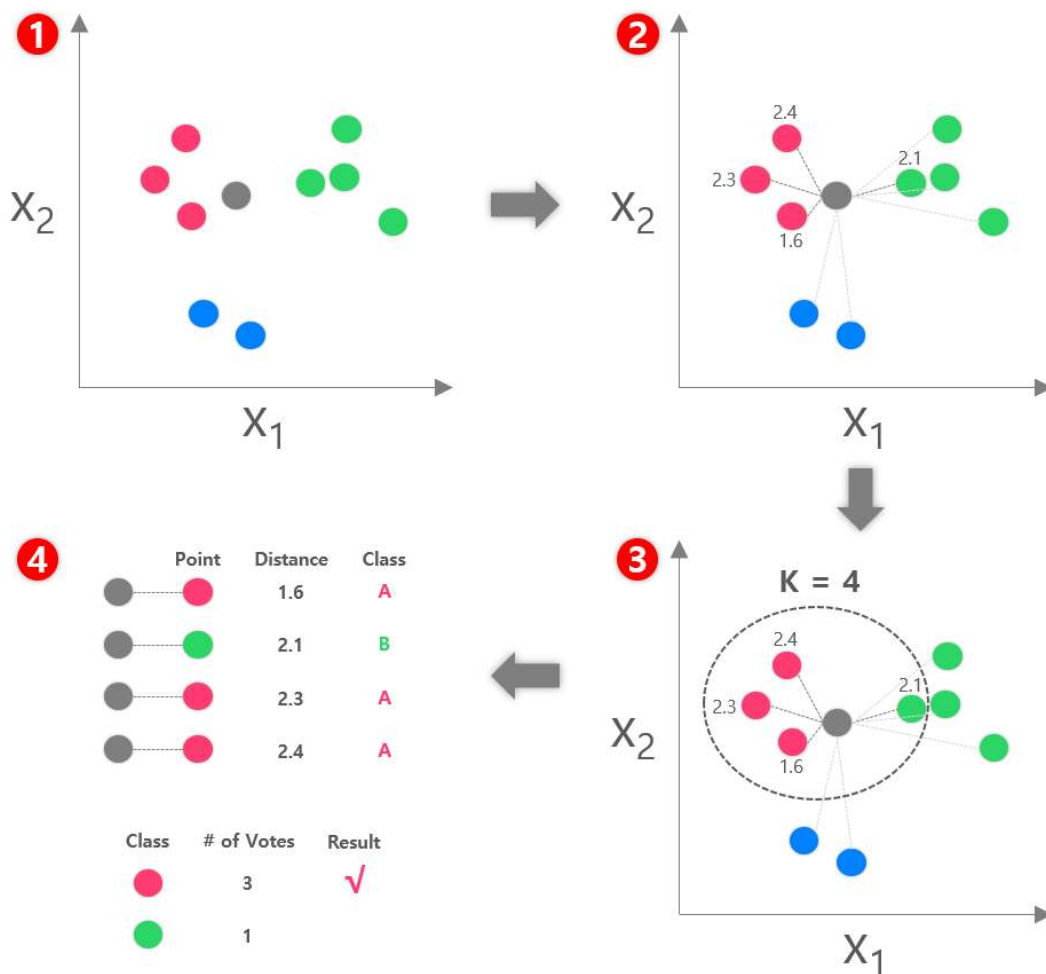


Figure 7: High-level overview of KNN.

There are nine data points which are red (class A), green (class B), and blue (class C) as

shown Figure 7 (1). The gray point is new data point what we should classify into a class. This algorithm calculates distance for the point to predict (grey point) and all other points as shown in Figure 7 (2). Selecting the number of  $k$  as 4, the algorithm find four nearest neighbors by increasing distance which have distance 1.6, 2.1, 2.3, and 2.4. The algorithm votes on the predicted class labels based on the classes of the  $k$  nearest neighbors. Here, the gray point was predicted based on the  $k=4$  nearest neighbors.

## Convolutional Neural Network

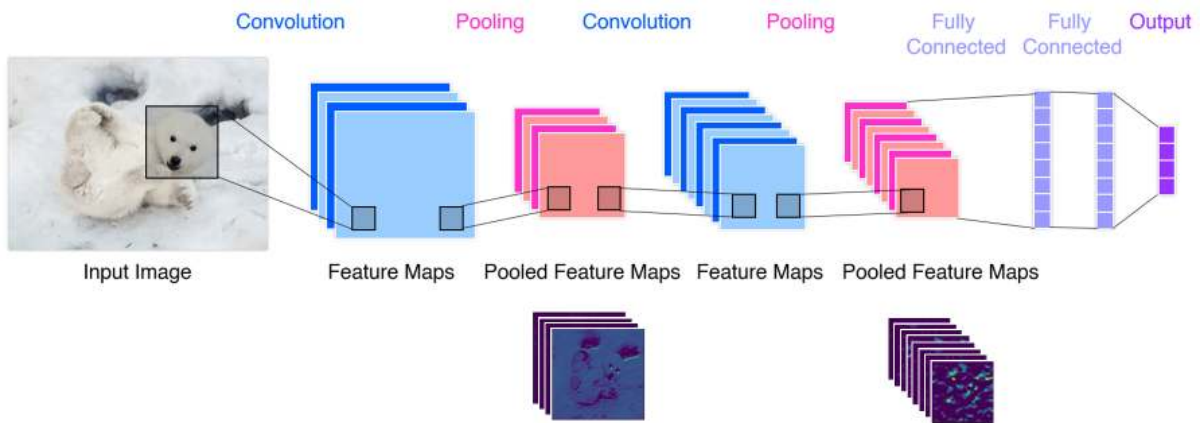


Figure 8: Overall architecture of convolutional neural network that consists convolution layer, pooling layer, and fully-connected layer.

A **Convolutional Neural Network (CNN)** is a deep learning algorithm which most commonly used for analyzing visual imagery. Compared to multi-layer perceptrons or fully connected layer, CNNs are regularized version because "fully-connectedness" of those networks makes them tend to overfitting data. However, it can take in an input image, assign importance (learnable weights and biases) to various aspects/objects in the image and be able to differentiate one from the other. The CNN can be divided into two parts that extract features of the image and classify classes as shown in the Figure 8. The part of extracting feature consists stacking form of convolution layer and pooling layer.

The **convolution** is a mathematical operator that multiplies one function by another, and then integrates over the interval to find a new function.

$$(f * g)(t) \stackrel{\text{def}}{=} \int_{-\infty}^{\infty} f(\tau)g(t - \tau)d\tau \quad (4)$$

The main idea is to look at the part which is filter, not the whole image. Since the filter cannot captures whole image at the same time, the filter shift through whole image several times as shown in Figure 9. Then, the convolution of  $4 \times 4$  image matrix multiplies with  $3 \times 3$  filter matrix which is called "Feature Map". In a general input tensor, the shape of tensor is (number of images)  $\times$  (image width)  $\times$  (image height)  $\times$  (image depth) unlike the example as shown in the Figure 9.

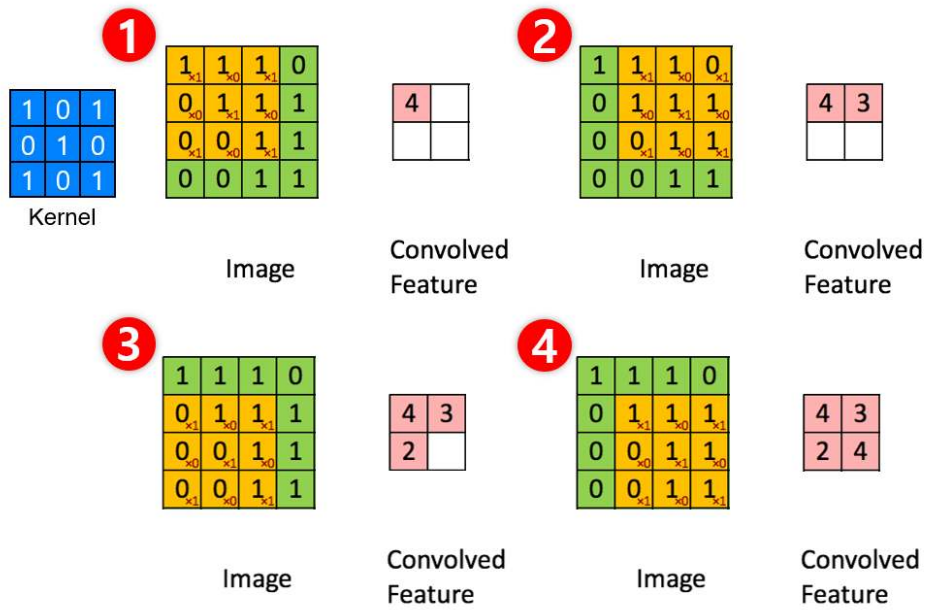


Figure 9: Procedure of convolution operator with 4 x 4 image and 3 x 3 filter for getting 3 x 3 computed feature map.

Applying different types of filters shows various convolution image such as edge detection, blur, and sharpen as shown in Figure 10. **Stride** is the number of pixels shifts over the input matrix. Giving stride to 1, the filter scans the image as shown in Figure 9. Giving stride to 2, the filter moves to 2 pixels at a time and so on.

$$\text{Edgedetection} = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad \text{Sharpen} = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \quad \text{Blur} = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$



Figure 10: Different filter type (a) Edge Detection, (b) Sharpen, and (C) Blur.

A **pooling layer** gets the output data from the convolutional layer as input and is used to reduce size of the output data or to highlight specific data. There are two methods for dealing with pooling layers: max pooling, average pooling, or min pooling. This layer works by gathering the maximum, average, or minimum value within a square. For example, data of square area

which is 7, 5, 2, and 1 become 7 as maximum value as shown in Figure 11 (a). The extracted features pass through the fully-connected layer to classify the images into classes.

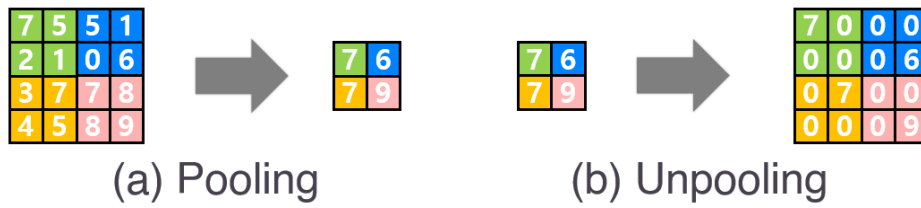


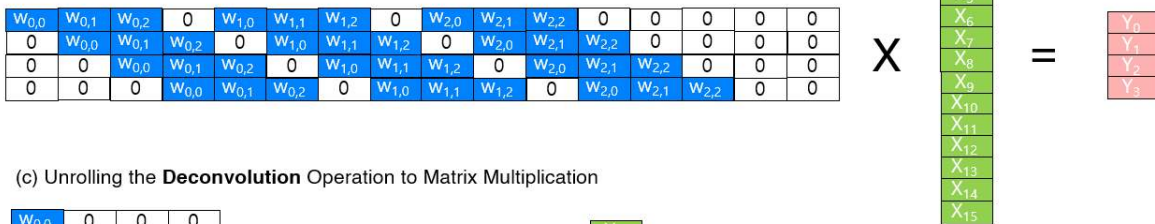
Figure 11: Procedure of max pooling.

### Deconvolution and Unpooling

(a) **Convolution** Operation



(b) Unrolling the **Convolution** Operation to Matrix Multiplication



(c) Unrolling the **Deconvolution** Operation to Matrix Multiplication

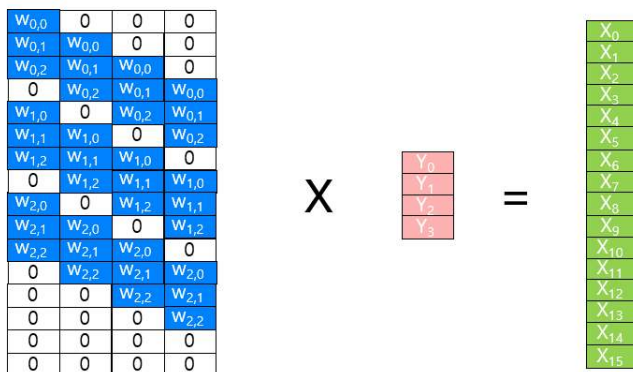


Figure 12: (a) A conventional convolution operation, (b) A transpose convolution operation, and (c) A transpose deconvolution operation.

The **deconvolution** and **unpooling** are commonly used in the context of CNN to denote reverse convolution and pooling. The deconvolution is an algorithm which is widely used in the signal processing and image processing. It is a method to identify which which part of the input made the part of output activated. In the mathematical notation, the deconvolution is to calculate  $g$  using given  $f$  and  $h$  where  $f$ ,  $g$ , and right side equation are filter, feature map (or layer



input), and output, respectively as shown in the Equation 4. For example, we track back last layer of some value and get eyes of human when the input was human image. The fact is that this filter activates human eyes and play roles for capturing human eyes. Here are transpose convolution operation and deconvolution operation examples as shown in the Figure 12. The transpose convolution is a method which use a matrix multiplication. As shown in the Figure 12 (b), padding and stride are 0 and 1, respectively. The examples shows a matrix multiplication with 3 x 3 kernel and 4 x 4 input. Likewise, the transpose deconvolution can be represented as shown in the Figure 12 (c) as transposing the 3 x 3 kernel to 4 x 16 matrix and decomposing input to 16 x 1 matrix and output to 4 x 1 matrix. From this operation, we can reconstruct the input feature map from the transposed kernel and output. All reconstructed values performs classification to predict class.

For performing **unpooling**, we need to record position of maximum values when doing max pooling as shown Figure 11 (b). This approach need less memory compared to other method, but it cannot avoid loss of information. The loss of information is very small percentage of CNN, so it does not matter to reconstruct.

### Fully Convolutional Network (FCN)

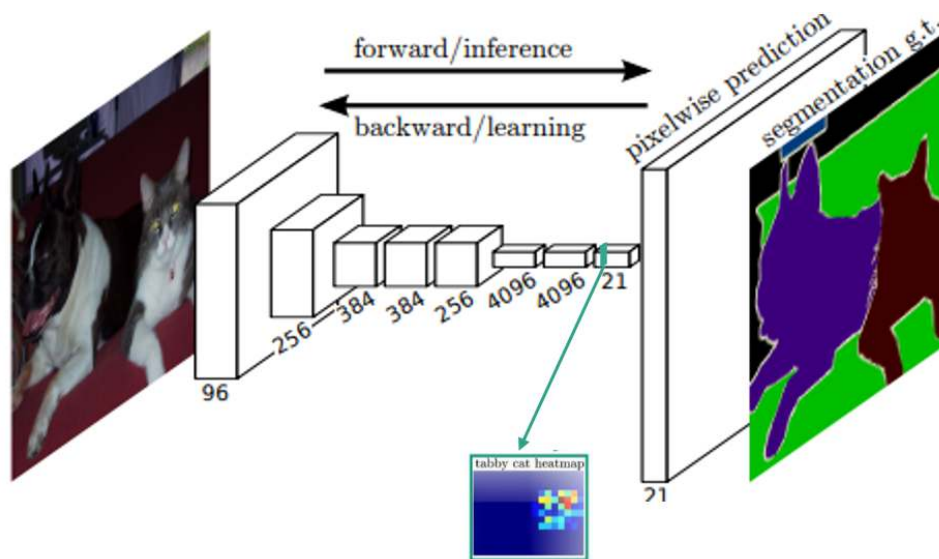


Figure 13: Fully convolution network for semantic segmentation and final output (heatmap) of tabby cat.

The **fully convolutional network** is most representative model in semantic segmentation. The CNN has convolution layer and fully-connected layer and it can classify an object into specific class. However, it cannot predict where the object is due to lost of location information from fully-connected layer. To address this issue, fully-connected layer replaces convolution layer for gaining the location information as shown in Figure 13. The main difference between original CNN and FCN is that the final output is determined by the number of classes. The final output

is heatmap which represents each class. For example, the heatmap in the Figure 13, represents tabby cat and it has high value in location of tabby cat compared to dog.

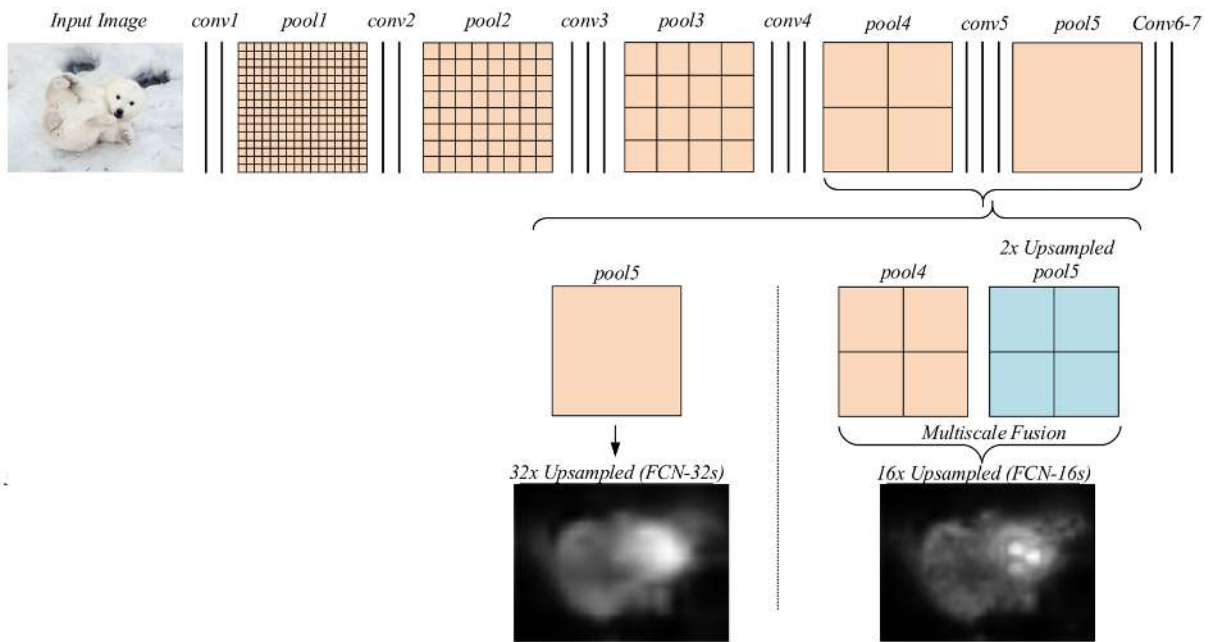


Figure 14: Fully convolution network and prediction result of FCN-32s which is 32x upsampled feature map from pool5 and FCN-16s which is 16x upsampled from pool4 and pool5.

Using fully convolution network, each class of corresponding coarse heatmap size increase up to original image size. The upsampled feature forms the final segmentation map. In other words, the class having the highest probability per pixel is selected to determine which class the pixel belongs to. By simply upsampling, the size of the feature map is restored to its original image size, but we get a coarse segmentation map like Figure 14 32x Upsampled (FCN-32s). This is result of 32x upsampling from downsampled 32x feature maps. However, we can get more detailed segmentation maps using skip combining. Referring to previous result of convolution layer, we get a more detailed segmentation maps. The FCN-16s upsampled combination of pool4 (16x downsampled) feature map and 2x upsampled pool5 for getting more detailed segmentation maps.



### III Semi-Structured Interviews To Identify Difficulties Encountered During GUI Prototyping

We conducted semi-structured interviews to understand the difficulties users face during GUI prototyping. We chose the mobile GUI domain, because mobile GUI design is a difficulty design task due to 1) the innate characteristics in the domain, such as small screens with touch-based input [39–43], 2) the increasing popularity mobile GUIs [1, 44] and demands [45, 46], 3) lack of appropriate guidance with existing tools [13], UXPin [15], Proto.io [16].

#### 3.1 Participants

We recruited 16 participants (3 women, avg. age: 23.56) at a university in South Korea, who were interested in designing GUIs for mobile applications. When advertising the interviews, we noted that individuals with previous GUI design experience were not eligible to participate. The recruited participants were all undergraduate students with various engineering majors, namely, electrical, computer, mechanical, and biomedical engineering. No participants reported experience with designing mobile GUIs.

#### 3.2 Procedure

As the participants entered the experimental room located in a university building, they were first asked to fill out a demographic questionnaire that sought information about their GUI design experience. Each participant was then given a computer (Intel i7 and 16GB RAM with a 2560 × 1440 resolution, 27-inch monitor) that had a GUI design tool (Kakao Oven [11]). We chose Kakao Oven as the GUI prototype software for three reasons. First, we looked for design tools for users with little design experience. We also checked whether the design tools provide pre-defined icons (e.g., pins) and design components (e.g., buttons) for use, as using the provided icons and components would prevent the additional difficulty of making the components from scratch during the design. Second, we wanted a tool that provides a description and instructions in Korean so that participants could easily learn how to use it. Third, we sought a tool that was similar to other popular tools used outside South Korea so that the study’s findings could be generalized to other tools (e.g., Sketch [13], UXPin [15]). As Kakao Oven fit the criteria well among the many alternatives available, we chose to use it for our study. Figure 16 A is a screenshot of Kakao Oven, presenting an example design for shoe sales. The Canvas panel (A2) indicates the area where the users create GUI designs. The Element panel (A3) includes various pre-built design elements (e.g., text, button, and tooltip) users can easily customize and apply to their design on the Canvas panel. We used an eye-tracker (Eyelink 1000+, sampling rate up to 2kHz) to analyze users’ behavior.

The participants were given as much time as they needed to become familiar with the tool during the practice session. The researcher also answered any questions they had about the

tool during this session. After the practice session, we conducted the eye-tracker calibration. Then, we introduced the task that the participants had to perform, which was to create a GUI for an online shopping mall application that lists product information along with product images. Note that we did not impose detailed or strict design requirements, such as the color theme or product domain. Accordingly, we set an open-ended task to investigate the obstacles encountered by the users, while planning a design and building a GUI from scratch using the given tool. The resulting list of obstacles identified would serve as a guide in deriving the design requirements for the proposed design assistant tool. The task session for each participant lasted about an hour, and the participants were filmed during the session. After completing the task, each participant watched the recorded video along with the researcher, and they discussed the challenges that the participant encountered during the GUI design process. We transcribed the audio conversation from these discussions.

### 3.3 Identifying Difficulties Encountered by Participants

We coded the transcribed audio conversations using the grounded theory approach [47], to identify roadblocks encountered by participants [48]. First, two of the authors of this paper individually coded the reports on the difficulties encountered. Then, the coders reviewed the collected codes, and discussed them to identify common themes. Following this, we were able to group the participants' difficulties into three distinct categories: (1) choosing the design direction, (2) measuring the design quality, and (3) determining where the viewer's attention would fall. After identifying these categories, we coded each participant's response relating to the difficulties as either "0" or "1" for each category. Only one category was assigned to each difficulty (for example, a difficulty could be coded as [0,0,1]). Finally, we collected and compared the results. When there were disagreements, the coders resolved them through a discussion. If the coders did not reach agreement, another author of this work got involved to resolve the disagreement. The inter-coder agreement level was 85% computed by the Pearson correlation coefficient. Based on the observations, we characterized and illustrated the three main difficulties that users may face during GUI prototyping.

#### **D1: OK, how do I start?**

One common observation was that the participants initially struggled to determine how to go about the process. They often began the task by creating a few GUI elements (e.g., text, buttons, and images) by dragging and dropping from the element panel to the canvas area. However, soon after, the participants deleted their created elements and began looking for examples online. This series of actions demonstrated that the participants encountered difficulties in the initial stages. Twelve participants reported that they could not easily conceptualize their prototyping direction [49] and sought examples for inspiration [18, 23, 50, 51] and quality comparisons [7]. Eight participants reported that they found it difficult to start designing GUIs without concrete

examples. Twelve participants commented that they needed templates that provide a skeleton layout indicating areas where elements could be replaced based on the user's judgment.

### **D2: Is my design OK?**

The second difficulty was that they were uncertain whether they were making good or bad design choices. One participant reported facing difficulty in deciding the optimal size of the elements (e.g., icons, images, and buttons) for the application users. Eight participants mentioned that it was difficult to choose colors that would be aesthetically pleasing to potential customers. During the task session, one participant looked for GUI evaluation methods or guidelines online, but the results returned were not helpful. The participant reported two reasons for this: The searched guidelines were often too abstract (e.g., *"use conventional elements"*) or difficult to apply to her current design context (e.g., regarding the instruction to *"use about 44 squared pixels for a touch interface,"* she did not have 44 squared pixels to allocate).

### **D3: Will users see what I want them to see?**

The third difficulty was related to determining where the viewers' attention would fall in the process of modifying the design (D3). Four participants stated that they had no clue how to determine which areas would be mostly viewed by users. Based on our observation, this difficulty seemed to lead to frequent changes in the position and size of the images. A participant reported, *"It was hard to guess which part the customer will look for first, as all of the text and images are important, I think."* Another participant stated that *"I want to emphasize selling points in my design, but I do not know which part is most proper to be the point."* When we asked the participant what made her think about this emphasis, she answered that she had experienced an online shopping mall, where important information, such as coupons or product prices were not sufficiently evident on a mobile screen to shoppers.

## **3.4 Requirements of a Tool for Assisting GUI Prototyping**

Based on the difficulties identified, we derived the requirements for a tool that can solve these problems while assisting GUI prototyping:

R1: Provide examples to guide users in how to begin and modify the design.

R2: Evaluate the current state of the design.

R3: Indicate the projected areas that users will see.

R4: Ensure R1–R3 on the most up-to-date design in progress.

R5: Ensure that R1–R3 are met non-intrusively.

R6: Meet R1–R3 using an add-on that can be attached to a web-based prototyping tool.

We found that providing timely feedback is essential to solve problems. Feedback can be considered one of the most powerful pieces of equipment for helping users achieve a desired goal [52], in particular for creative work [53]. Feedback given during an early design stage can allow users to iteratively improve the quality of the design [6, 19]. Thus, we derived three requirements (R1, R2, R3) that aim to provide relevant examples (R1), evaluation of the design (R2), and areas of interest for target users (R3). In addition to the three requirements derived directly from the three difficulties (D1, D2, and D3) identified, we included three requirements. We added R4, because it is critical for users to receive prompt feedback on their current designs, as early and timely feedback could improve creative work [6, 54]. R4 also implies the design process should not involve an offline computation so that the iterative processes of designing, reviewing, and revisiting can be efficiently supported [55]. In addition, we included R5, because we did not want users to be interrupted by such feedback while making their design choices. R5 implies the need for a dedicated space where feedback can be visually displayed. R6 intends easy installation and compatibility of GUIComp. Following these requirements, we designed a tool for assisting GUI designers. We discuss this tool design in the next section.

## IV GUIComp: Add-on for GUI Design Guidance

We introduce a web-based add-on tool, called GUICompanion (GUIComp), which provides prompt feedback on GUI designs. GUIComp is designed as an add-on for mobile GUI design applications so that it can be easily adapted to any other tool with proper configuration of client-server communication.

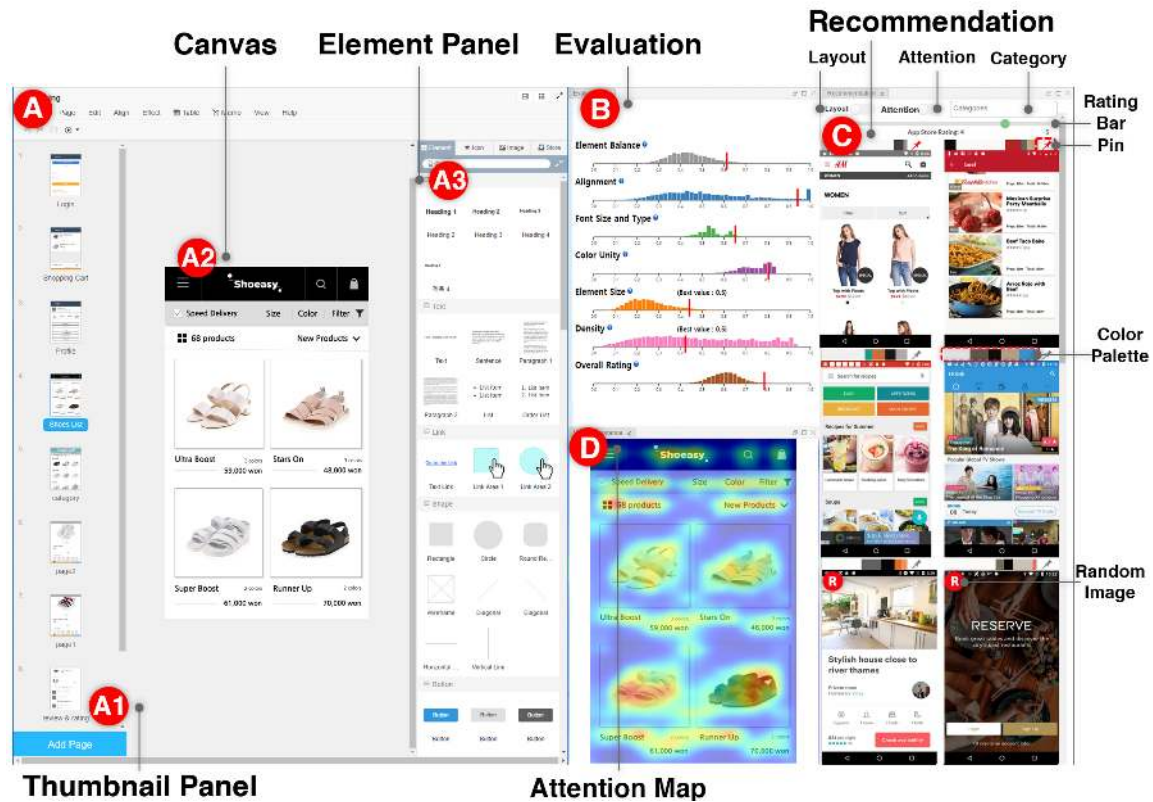


Figure 15: Three feedback panels in the GUIComp (Add-on)–Evaluation Panel (B), Recommendation Panel (C), and Attention Panel (D). (A1)–(A3) are interfaces of an online GUI prototyping tool (the base tool).

### 4.1 Tool Overview

GUIComp consists of three panels (the evaluation, recommendation, and attention panels), where each panel fulfills the three requirements (i.e., R1-R3) described in the previous section. We implemented the tool as an extension to Chrome Browser [56], which uses responsive web technologies to capture and process information in real time (R4), and provides feedback in a separate window outside the user’s design canvas (R5). As an extension, GUIComp can be linked to any online GUI prototyping tool (R6). For demonstration and evaluation purposes in this work, we linked GUIComp to Kakao Oven [11], a base GUI prototype tool. GUIComp captures the most up-to-date design states (Figure 15 A) by using open source libraries (i.e., MutationObserver), generates feedback (i.e., a visual complexity evaluation), recommendation templates, and attention heatmaps based on the states, and presents them on the three panels

(Figure 16 B–D). In the following sections, we describe how we captured the users’ design states, parsed the web elements, and extracted the design elements (e.g., element types, dimensions, and color maps), and then introduce the panels, metrics, and datasets used for GUIComp.

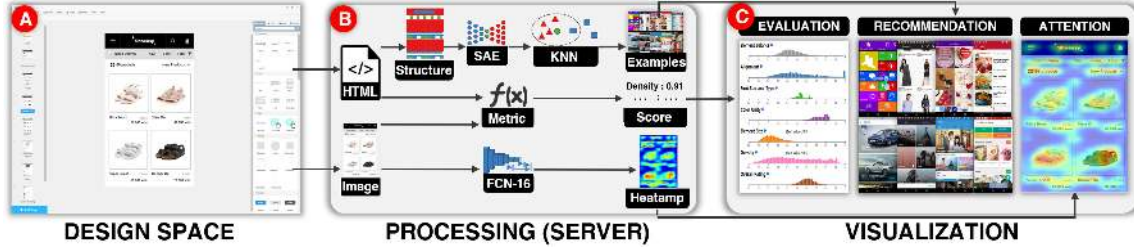


Figure 16

## 4.2 Capturing and Processing Users’ Design on Canvas

GUIComp captures users’ design states and returns two different outputs, HTML elements and images as shown Figure 15 A and B. Then, using the two outputs as inputs, GUIComp generates three different types of feedback (examples, evaluation scores, and attention heatmap) as shown Figure 15 B. When there is an interaction on the user canvas (Figure 16 A2), an internal HTML file for describing the user canvas is also updated. Once an update is detected by using MutationObserver, a JavaScript API, GUIComp converts the HTML file into a json file using `html2json` [57] and `html2canvas` [58] libraries. GUIComp saves the json file in the Chrome extension’s local storage temporarily to prevent Cross-site Request Forgery (CSRF) vulnerabilities. Then, GUIComp sends the json file to the processing server (Figure 15 B) along with an image of the GUI on the canvas.

As shown in Figure 15 B, the processing server extracts GUI components’ ID, type (e.g., button), and attributes (e.g., color, width, and height) from the json files and computes i) the dominant colors using OpenCV [59] and ii) the visual complexity scores using equations proposed by Riegler and Holzmann [35], which are described further in the Evaluation Panel section. GUIComp also captures and stores the GUI image from canvas; see Figure 16 (A2). The captured GUI image is used as an input for the FCN-16 model, and the model provides an attention map score as an output as shown in Figure 15 B (bottom).

## 4.3 Recommendation Panel

### Description of In-the-Wild Mobile GUI Data (RICO)

To create a pool for recommendation templates, we considered both the ERICA [60] and RICO datasets [61], but chose to use RICO, as it includes one of the most extensive lists of mobile application screenshots. The RICO dataset [61] contains 72,219 mobile application GUI screenshots, meta-data, and element hierarchy information from 9,700 Google Play Store popular apps. The dataset also contains meta-data of the applications, including the average rating, the number of



installs and downloads, and the category assigned for the app marketplace. The average ratings and categories of the apps are utilized for filtering examples in Figure 16 (C). We reviewed each screenshot manually and excluded those that were inappropriate for use for GUIComp. Specifically, we excluded screenshots of playing games, commercial advertisements, pop-up menus, Android basic screens (e.g., the home screen), data loading, password typing, black images, web views, and maps. In the end, we used 6683 screenshots from 2448 applications for this work.

### Generating Recommendations from HTML



From the HTML file as shown in Figure 15 B, the GUI structure information (e.g., TextView, EditText, Button, ImageView, and ImageButton) is also extracted and used as an input for the autoencoder and the k-nearest neighbor algorithm, as suggested by Deka et al. [61] for recommending examples that are similar to the current GUI on the canvas. In our implementation, the encoder has 13,500 ( $90 \times 50 \times 3$ ) input (the image size is the same ratio as the RICO dataset) and 64 output dimensions with two hidden layers of 2048 dimensions and 256 dimensions with the ReLU activation function [62]. The decoder has the same but reversed architecture of the encoder. For training and recommendation, we used the RICO dataset. We used 90% of the data (6154 images) for training and hold the rest for validation. We used the Adadelta optimizer and mean squared error as the loss function. During the training, the validation loss was stabilized at 4.96 after about 1900 epochs for 1.5h with 512 batches. Using the trained model, we generated a 64-dimensional representation per GUI screenshot for the RICO dataset. The k-nearest neighbor algorithm was implemented with the brute search algorithm and the cosine distance metric.

### Presenting Similar Examples and Random Examples

The recommendation panel provides users with example templates that are relevant to and inspirational for their design goals. It is not feasible to present all examples from the RICO dataset to users and ask them to explore the list without any guidance. Furthermore, as users make progress in their designs, they might desire different examples for different purposes. Thus, our goal was to provide a list of the most relevant examples at the moment based on the current state of the design. To achieve this goal, we computed similarity scores between the user’s design and examples so that we could rank the examples in the order of similarity.

In addition to similar examples, we decided to present ‘random (four images)’ examples for two reasons. First, giving random examples increases diversity in design choices, as the standard affinity algorithm can generate recommendation lists that are similar [63]. Second, users can rethink the choices they made (e.g., the layout). By viewing intriguing alternatives, users may have a chance to think outside the box. To distinguish random examples from recommended examples, we show a visual mark (R), as shown in Figure 16 C (bottom). When there is no component on the user canvas (e.g., at the beginning), random examples are populated in the

panel.

We speculated that users might want to ‘keep’ some templates while changing others as they make progress. Thus, we allowed users to pin examples so that they maintain the selected examples in the list; they can also unpin the examples (Figure 16 PIN ). The panel also lists dominant colors with their color palettes (e.g., Figure 16 Color Palette ) generated from the OpenCV library. When a color in the palette is hovered over, the RGB information of the color is shown to help users refer and learn combinations of colors in the examples. When a user clicks on an example template, the user canvas is cleared and then elements of the clicked template populate the user canvas in the same layout and alignment as shown in the template. To allow this automated element-populating function, we first mapped the RICO leaf-level nodes’ information (e.g., x, y, width, height, and type) to that of the Kakao Oven elements (Figure 16 A3), and computed the ratio of RICO’s nodes to screen width and height and Oven’s elements to the canvas width and height. Then the drag-mock library [64] updates the HTML code to indicate that new elements were added. Note that users can recover previous designs by using a revert function provided by the base tool.

#### 4.4 Attention Panel

To visually show how much attention will be paid to elements on a GUI design (R3), we incorporated Bylinskii et al.’s attention model [65]. We feed the captured image from the users’ design on the canvas into the attention model (FCN-16) [65] that had been pre-trained with a graphics design importance (GDI) dataset [24] to produce a heatmap whose pixel values ranged from 0 to 255. Figure 16 D shows an attention heatmap with the given GUI on the canvas in Figure 16 A2. The color map used in the heatmap linearly ranges over blue-green-yellow-red, where the red indicates the areas that will receive the most attention. The more intense the red on an element on the heatmap, the higher the importance of the element in the attention map. Guided by the heatmap, users can expect where viewers’ attention will be directed. Based on the information, users can revise the design for optimal guidance. With the attention map, users can balance the expected audience’s attention by revising the original design. For example, when a text field unintentionally receives too much attention, a user may change the position or size of the text to redirect the audience’s attention to more important areas.

#### 4.5 Evaluation Panel

##### Generating Visual Complexity Metrics

To provide feedback on GUI quality (R2), we incorporated visual complexity metrics. We also expected that by using the metrics as evaluation rubrics, users could easily discern possible issues in their current design [66]. A review of previous studies that involved experimental evaluation enable us to find two candidate metrics [35,39]. Of the two metrics, we chose Riegler et al.’s metrics [35], because it enables real-time evaluation (R4) and allows easy-to-understand



numerical result presentation. From the authors' original seven metrics, we excluded two: the number of UI elements and inconsistency. We omitted the former because the absolute number of UI elements does not reflect the quality of a design, given that the number varies depending on the design goal. We eliminated inconsistency because it applies only when a design has more than one GUI page. We also modified the original metric names into more intuitive ones to help users understand the terms. Specifically, we changed imbalance to **element balance**, misalignment to **alignment**, color complexity to **color unity**, typographic complexity to **font size and type unity**, and element smallness to **element size**. We did not change the term **density**. For metrics that feature 0 as the best score (i.e., element balance, alignment, color utility, and font size and type unity), we reversed the score range from 1 to 0 to 0 to 1 so that 1 became the best score. We did not convert the scores for element size and density because the midpoint (i.e., 0.5) is considered the best score for the metrics [35]. These metrics were also used as rubrics for the evaluation by the online workers. We describe the metrics in detail as follows:

- **Element Balance (best score: 1.0)**: refers to the overall symmetry, balanced element distribution (e.g., consistent space between elements), and skewness of the elements.
- **Alignment (best score: 1.0)** pertains to the checking of alignment among elements. During computation, three vertical (left, middle, and right) and three horizontal (top, middle, and horizon) imaginary lines are drawn for each element to measure the score.
- **Color Unity (best score: 1.0)** shows the color use based on the ratio of dominant to non-dominant colors.
- **Font Size and Type Unity (best score: 1.0)** investigates the consistency of font sizes and types that are in the text.
- **Element size (best score: 0.5)** is intended to verify whether elements are excessively small or large for mobile interfaces. Scores lower than 0.5 mean the elements are small, while scores higher than 0.5 imply the elements are large, on average.
- **Density (best score: 0.5)** computes how much space is occupied. Scores of less than 0.5 translate into simplicity in design, whereas higher scores imply over-populated designs.

### Presenting Visual Complexity Scores with User Ratings.

To help users understand the strengths and weaknesses of their current design compared to those in the RICO dataset (R2) [67], the evaluation panel (Figure 16 B) presents six visual complexity scores and one overall rating score for the user's current design. This panel uses six histograms, each of which shows the distribution of examples in the RICO dataset with its corresponding complexity score on a horizontal scale. The vertical red bar on each scale shows how high or low the corresponding visual complexity score of the user's design is compared to that of the

examples. For instance, the example design in the user canvas (Figure 16 A2) can be evaluated as a high-quality design based on the positions of the red bars over the distributions in each evaluation dimension (Figure 16 B).

To support R4, GUIComp computes new scores whenever a user interaction occurs over the design canvas (A2). For example, when a button element is dragged from the element panel (A3) and dropped to the canvas (A2), the evaluation panel updates with newly computed scores to present the effects of the dropping of the button. Note that the current design’s scores are marked with a thick, vertical, red bar in each histogram (Figure 16 B). The scores of the recommended examples are depicted with black bars when a user hovers over an example in the recommendation panel.

#### 4.6 Implementation Notes

We used two servers: one (Intel Xeon E5-2630, 2.40GHz, 128GB RAM) for processing the captured user interaction and one (Intel Xeon E5-2630, 2.20GHz, 128GB RAM, 2 × Tesla P100-PCIe-12GB) for model training and prediction. GUIComp uses several web development libraries, including Django [68], Keras [69] D3.js [70], and Bootstrap [71]. Note that in the performance experiment, when a user interaction happens on the base tool’s canvas, it takes 725, 885, and 750 ms for each panel to produce feedback using the data processing server. We also logged all user interactions, GUI states, and attributes for qualitative analysis.

## V User Study Design

To evaluate GUIComp, we conducted a user study, in which 30 participants were asked to create two GUI designs: one with restrictions and one without. Then we presented the designs to online workers to evaluate the quality of the designs. This evaluation was guided by three primary research questions:

RQ1: Does GUIComp help users make the design closer to an acceptable design to general users than Kakao Oven?

RQ2: What UI/UX benefits does GUIComp provide to users?

RQ3: How do users use GUIComp’s feedback to overcome the difficulties during prototyping?

### 5.1 Participants, Procedure, Apparatus, and Tasks

#### Participants

We recruited 30 participants by posting an advertisement at a university. As participants entered the experiment room, they were asked to fill out a form on demographic and background information, including name, age, gender, major, prototyping, and development experience and time. Then, the participants were randomly assigned to either the Control Group (CG) or the Experiment Group (EG). The CG participants were allowed to use only a base GUI prototyping tool, called Kakao Oven (Figure 16 A only), whereas the EG participants were given the proposed tool and the base tool (Figure 16 A–D). Note again that we used Kakao Oven for the same reason as in the previous participatory study. The experiment was a **between-subject** study to minimize learning effects and was carried out using an Intel i7 (3.40GHz, 16GB RAM) computer and a 2560x1440 27-inch monitor.

#### Procedure and Apparatus

At the beginning, the experimenters explained the purposes and goals of the experiment to both groups for 5 minutes. Then the participants in both groups watched the explanatory video for 5 minutes each (using Kakao Oven) or 15 minutes (Kakao Oven and GUIComp, including the visual metrics explanation). We also provided concise descriptions of the visual metrics in a popup window, which the participants could mouse over to read during the study. The participants were allowed as much time as possible to familiarize themselves with the tool assigned to them, and explore the tool’s features. In addition, we told participants that 10% of them would receive an incentive (US\$10), in addition to the base payment (US\$20) based on the perceived quality scores rated by online workers afterward. Finally, the study began after the experimenters calibrated the eye-tracker (Eyelink 1000+, sampling rate up to 2 kHz) for the gaze transition analysis.

During the experiment, we gave the participants two tasks to evaluate whether GUIComp is effective, in situations where design restrictions are given by clients (e.g., embedding brands [43]) and no restrictions exist. The two tasks we used were a user profile interface with restrictions (T1) and an item-listing interface without any restrictions (T2). We chose the two interfaces as the tasks for users with little design experience, because we thought that 1) the difficulty level was appropriate, as the basic design components are provided by Kakao Oven, such as icons and buttons, and 2) the interfaces are commonly requested, given that many apps require input of user information and show items in a listing interface, regardless of the app categories. Inspired by Lee et al.'s persona-based web page design [20], we used a persona-based GUI design in T1 with detailed restrictions. There were no other restrictions, which means the participants were allowed to customize their UIs as much as they desired for the incentive.

## Tasks

Next we describe the personal-based task used in the experiment as follows: *A user's name is Elaine. Elaine's email is elaine@gmail.com. Elaine has 12 notifications. There is one shipping item and two items in the shopping basket. Elaine left 31 reviews.* Goal: Design a profile user interface for Elaine. Tasks and restrictions:

- Your GUI should include basic information about Elaine.
- You can add detailed information (e.g., order list, delivery tracking, cancellation returns, discount coupon, Service center, profile management).
- Use the headings, text, shape, button, and pagination components of the given tool in your basic design layout.
- Enter textual information about Elaine.
- Choose colors for each component. (optional)
- Choose the font and font size for each text component. (optional)

After completing the tasks, the participants responded to an exit survey about usability and user experience with GUIComp (RQ2), with the survey comprising items on efficiency, fulfillment, effectiveness, ease of use, ease of learning, user-friendliness, consistency, fun, and satisfaction [72, 73]. These items were rated by the participants on a 1–10 scale (1: "highly disagree," 10: "highly agree") for satisfaction scores, and a seven-point Likert scale (1: "highly disagree," 7: "highly agree") for the other scores. The users also provided preference scores for each feedback panel (1: "least preferred," 7: "most preferred").

Overall, we recruited 32 participants, but two could not complete the experiment due to sudden system failures. Thus, 30 participants remained (18 men and 12 women, avg. age: 22.4, all from engineering schools, 15 users/group) and they had little or no experience in

GUI design (less than a month: 6 participants, 1-3 months: 1, 3-6 months: 1, no experience: 22). 4 participants experienced a prototyping tool as their personal hobby (2: Kakao Oven, 2: Adobe Photoshop). Other than these four, no one had previous prototyping tool experience. We excluded the eye tracking data for 6 participants in the EG, as the data did not contain full prototyping processes due to a malfunction. Thus, we used 18 sets of eye tracker data (9 participants  $\times$  2 tasks) in our eye tracking-based analysis. We analyzed EG groups' gaze data only, because we are interested in how GUIComp helps users (RQ1–RQ3).

## VI Experiment Results, Analysis, and Implications

In this section, we report our evaluation approach with online workers. We chose online workers for the evaluation, because the goal of GUIComp is to help users produce designs that are acceptable to general users, and such relative acceptance levels can be measured by the scores given by online workers as general mobile GUI users.

We evaluated the designs in two sessions. In Session 1, we asked 26 MTurk workers to evaluate the perceived design quality at 0-10 scale. No rubric was provided in Session 1 to collect general users' subjective assessment scores. In Session 2, we asked 21 other MTurk workers to evaluate the designs with a rubric, which were the evaluation metrics (Figure 16 B-element balance, alignment, color unity, font and element size, and density), as the assessment criteria. We gave the workers the rubric to provide minimum guidance for preventing inconsistent and subjective evaluation [66]. We also provided detailed descriptions for each metric to help the workers understand and use the metrics for evaluation. We think the evaluation with the rubric was appropriate as evaluation guidance, because it includes indexes for visual complexity (e.g., color, size, density, and alignment) which affects the overall quality of the GUI designs [39,42,43]. Next we present the evaluation results using the Welch's t-test and the Kruskal-Wallis test, due to unequal variance in the data.

### 6.1 GUIComp Users Produce Designs Acceptable to General Users (RQ1)

In this section, we present statistical test results for the evaluation scores. In doing so, we merged the assessment scores of both tasks in each session, and ran Welch's t-tests to see whether GUIComp was effective regardless of the task type. Note that we report the results with 95% confidence interval plots, as shown in Figure 17, where the dot and the whisker represent the mean and the 95% confidence intervals, respectively. Figure 17 (left, top row, "Perceived Qual.") presents that the GUIs produced with GUIComp were regarded as more acceptable designs (i.e., higher scores) by general users than those produced with Kakao Oven ( $t(54.71)=3.24, p=0.002$ , Cohen's  $d=0.84$ ). We observe the same result for Session 2 results (Figure 17, left, second row, "Eval. w/ Rubric"), where GUIs were evaluated with the rubric ( $t(47.11)=4.24, p<0.001$ , Cohen's  $d=1.09$ ). But there is no statistically significant difference in the time spent performing the design tasks in the two sessions ( $t(50.00)=1.39, p=0.17$ ). Overall, the results indicate that using GUIComp helps people, especially those who are in the beginner stage of designing GUIs or have never designed GUIs, produce designs acceptable to the general users, without spending additional time on the designs.

### 6.2 Using GUIComp Is Enjoyable, Satisfactory, and Affordable (RQ2)

Figure 17 (Right) presents the exit survey results: The participants who used GUIComp felt that using the tool was more **efficient** [ $t(26.56)=3.19, p=0.003$ , Cohen's  $d=1.16$ ] and **effective**

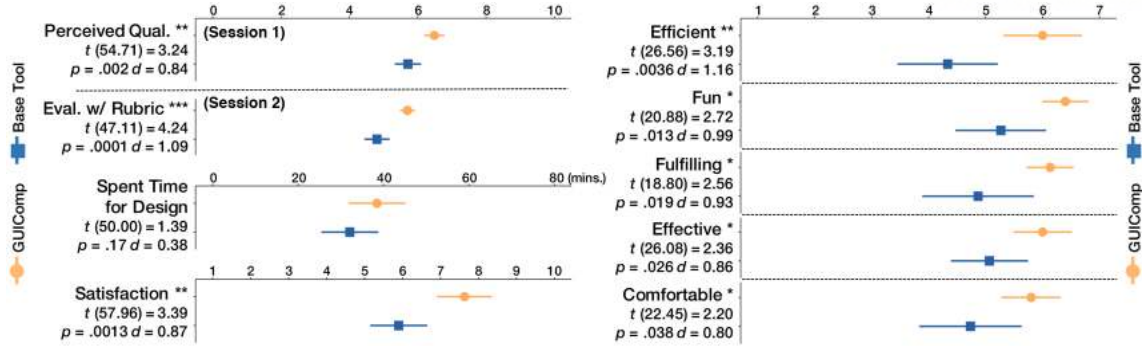


Figure 17: Online workers' ratings of the participants' designs (left) and exit survey (right) results. The dot and the whisker represent the mean and the 95% confidence interval, respectively. [t(26.08)=2.36,  $p=0.026$ , Cohen's  $d=0.86$ ] in GUI prototyping than those who did not use it. Participants who used GUIComp while prototyping also had more **fun** [t(20.88)=2.72,  $p=0.013$ , Cohen's  $d=0.99$ ], and felt more **comfortable** [t(22.45)=2.2,  $p=0.038$ , Cohen's  $d=0.8$ ] and **fulfilled** [t(18.8)=2.56,  $p=0.019$ , Cohen's  $d=0.93$ ], than those who did not use the tool. Finally, participants who used GUIComp were also statistically significantly more **satisfied** with the tool [t(57.96)=3.39,  $p=0.0013$ , Cohen's  $d=0.87$ ] than those who did not use GUIComp.

In the post-experiment interviews, we observed multiple reasons for the positive results, including the feedback provided in real-time, which enabled efficient iteration of their design processes with fun and high satisfaction. Participants who used GUIComp liked the feedback feature and enjoyed the way their multi-faceted feedback (R1) was dynamically updated in real-time (R4) throughout the design process. Participant 2(P2) said, *"It was nice to design with the data (i.e., feedback provided by GUIComp), as up to now, my intuition has been the only option that I can rely on during design."*

No participant reported any perceived clutter or distraction caused by the real-time feedback from GUIComp during the design process. Instead, the real-time feedback (R4) was a feature of interest to some participants. P1 stated that *"I think it was fun seeing that feedback was changed in real-time based on my interactions."* We do not see statistically significant differences between the two conditions regarding how user-friendly and easy to use and learn the tools were. The results show that GUIComp could be added to existing design technologies without sacrificing their usability.

### 6.3 Users Employ Multi-faceted Feedback for Overcoming Difficulties in the Iterative Design Process (RQ3)

In this section, we present the results of the eye-tracking data analysis, where we find how GUIComp facilitated iterative designs, and how the users responded to the feedback during their design iterations. Figure 18 presents the panel gaze sequences of six participants who used GUIComp (three sequences per task, which received the highest quality scores, were chosen) based on participants' eye movements between panels. We also present example user designs to describe how the designs evolved, based on the feedback from GUIComp.



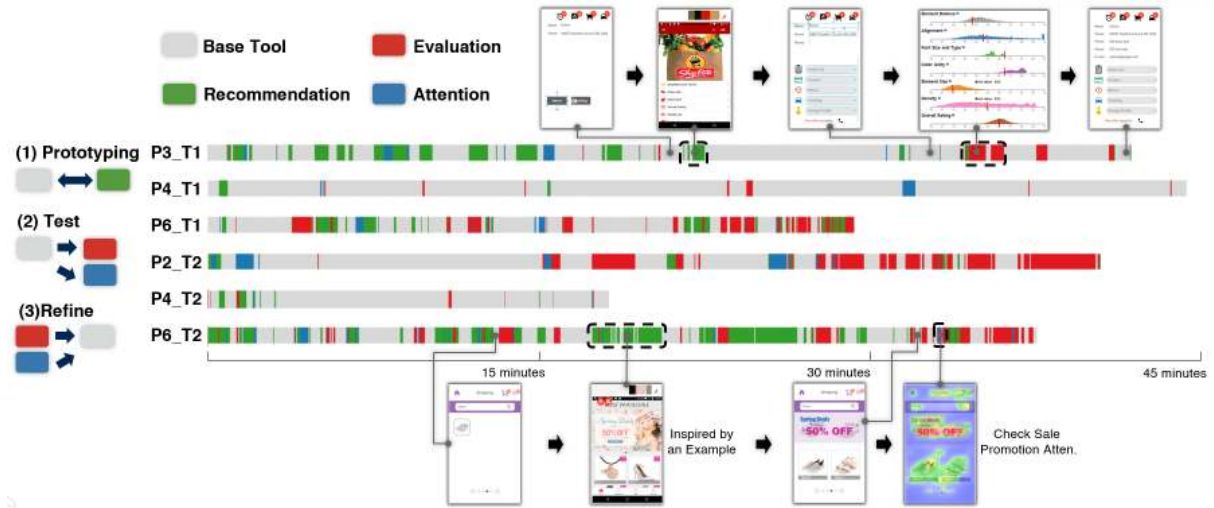


Figure 18: User gaze sequence data on the feedback panels.

Note that we denote eye gaze locations on the panels in four different colors—gray for the base tool (Figure 16 A), red for the evaluation panel (Figure 16 B), green for the recommendation panel (Figure 16 C), and blue for the attention panel (Figure 16 D). Note also that we use the terms ‘prototyping’, ‘testing’, and ‘refining’ borrowed from Nielsen’s iterative design [4] to capture distinctive design stages with GUIComp.

We also consider the intentions of each panel at a high level. By prototyping, we mean the transitions between the base tool and the recommendation panel (  $\text{gray} \rightarrow \text{green}$  ). By testing, we mean the transitions from the base tool to the evaluation or attention panel (  $\text{gray} \rightarrow \text{red}$  /  $\text{gray} \rightarrow \text{blue}$  ). Lastly, we define refining as the transitions from the evaluation or recommendation panel to the base tool (  $\text{red} \rightarrow \text{gray}$  /  $\text{blue} \rightarrow \text{gray}$  ).

First, we observe frequent color changes in each user’s gaze visualization in Figure 18, which mean that the participants performed an iterative design process, forming many different transition patterns. All six participants used the recommendation panel (green) frequently in the early stage, and some participants used the recommendation panel until the middle stages during their design, which is consistent with what Kulkarni et al. [6] found in their study. Based on our reviews on the interview transcripts, we conclude that participants viewed examples of the recommendation panel for inspiration at the beginning of the prototyping, and mainly for solving D1. P9 stated how useful GUIComp was during the early design stage for solving D1: *“this tool is useful, in particular in the early design procedure with the abundant feedback and references.”* P2 described her design strategy for solving D1 during prototyping as follows: *“I started with the recommendation panel in the early stage to search as many templates as possible so that I could work with many good options.”* Other participants, such as P9, also shared their experiences of using examples for solving D1: *“The recommendation panel was useful early on... With the many references, I could decide which layout to use and start with.”* We observe actual patterns of this strategy in Figure 18 (e.g., P3\_T1 and P6\_T2 middle green parts). When



examples are used, users tend to gradually build their design in parts, by referring to different parts of examples. This behavior is interesting, because we had expected users to employ a design strategy of taking all elements of the best example that they think and submitting their resulting design by slightly modifying the example.

We observe that participants tended to make more use of the evaluation panel in the later design processes (e.g., Figure 18 P3\_T1, red parts at the end). Participants' gaze often moved to the canvas to find an element to enhance the design quality, after they viewed the evaluation panel. We speculate that participants wanted to confirm their intermediate or final design quality using visual complexity scores (testing and refining), an effort for solving D2. This effort was illustrated by P13, who said, *"I consistently checked if the alignment is correct, colors are properly used, and the element balance is good. In this way, I have created a GUI that I like."* Other participants, such as P14, expressed a similar opinion as P13: *"Mostly, I paid attention to increase the evaluation scores during my prototyping and had a high quality design."*

During the prototyping, the participants monitored the attention information, to balance viewers' focus to solve D3. P6 directly expressed her intention of using this view—*"I can see which parts will attract the viewers on which I could edit my design."* It is of interest that P6 checked whether viewers' attention would lie on the most important sales words in the design (Figure 18 P6-T2 '50% OFF'), producing a high-quality design. Although most participants enjoyed the attention view, *"I like the (attention) panel, because it is very intuitive and easy to understand,"* as P2 stated, one participant indicated a concern that using this view may not always be easy, as it often requires interpretations of the results.

To understand which type of feedback played a critical role in improving the participants' designs, we reviewed the user preference scores (avg. scores—recommendation: 5.0, evaluation: 4.93, attention: 5.53) and ran Kruskal-Wallis ANOVA tests. The test results indicate that there is no significant difference in user preference among the three types of feedback ( $p=0.60$ ). In terms of the time spent for each panel for reviewing feedback (avg. duration—recommendation: 168s, evaluation: 126s, and attention: 27s), we find significant differences among the panels ( $p=0.001$ ,  $F=7.40$ ), according to the ANOVA test. The Tukey post-hoc analysis results indicate that users spent more time reviewing the feedback from the recommendation and evaluation panels than that from attention panel. We do not find any statistically significant difference in the eye-gazing duration between the recommendation and evaluation panels. To sum up, we think that all of the feedback in the panels contributed to improving user satisfaction in using GUIComp. In terms of contribution to improving designs, we conjecture that evaluation and recommendation feedback could have played more important roles, according to the time that users spent reviewing and reflecting on the feedback for their designs.

## VII Lessons, Limitations, and Discussion

In this section, we provide the lessons, guidelines, and research directions for design assistance tools.

**Provide Explanation for Feedback or Allow Users to Intervene in the Feedback Generation Process:** We believe design assistance tools can further support users by providing additional feedback for “why” (justified feedback [74]), to help users better understand the reason for the given feedback. In this study, participants requested justified feedback in GUIComp when the feedback was not clear. For example, few participants often wanted to know exactly which areas or elements contributed the most negatively to the evaluation score. P6 struggled to find areas for improvement, and stated, *“I designed in a way to increase my alignment score, but the score did not go up as I intended.”* The same issue occurred with the recommendations. When we asked how we could improve GUIComp, P5 commented, *“There were many recommended examples, but I could not understand why some examples were recommended to me.”* Users wanted to understand the reasons for the scores and the recommendations.

One solution is to provide a relevant explanation for the feedback. For example, when a user does not understand a low score in alignment, an assistance tool can point out the misaligned elements, so users can take appropriate actions. Another solution is to include users in the automated process. For example, users can be invited to intervene in the example recommendation process. After viewing the initial suggestions, users may provide feedback by setting constraints on desired designs (e.g., background color) or by directly labeling selected examples as “good” or “bad.” We can consider providing more granular feedback using semantics of UIs [75], referring to what each visual element means and how it is supposed to be used, to help users interpret the results better. Future research may investigate how to make the system’s feedback more interpretable and how to incorporate users’ feedback on the system’s feedback.

**Conflict between Users’ thought and the Feedback:** In this study, we provided participants with three types of feedback. The participants often struggled with feedback that was the opposed to what they believed. In particular, some participants were reluctant to accept numeric scores in the evaluation panel. In the experiment, P14 intentionally changed the layout so that some elements were overlapping. This action decreased the evaluation scores. After the experiment, P14 stated, *“Though I saw my scores dropped as I made the element overlap, [...] I still think my (overlapping elements) design looks better than others with high scores.”* The strength of numeric scores is the clear interpretability. However, when the scores do not match what users believe, the scores may not be appreciated. In addition, the evaluation scores, which do not reflect subjective elements, layouts, or themes, may be disregarded. Future research should investigate how to reconcile the potential conflicts either by augmenting the numeric score with more convincing, descriptive, textual feedback or by improving the feedback using users’ input.

**Automation and Feedback:** As we discussed above, the participants often demanded

more explanation and more explicit guidance. In this study, we provided feedback in a non-intrusive manner so that the users could be able to contemplate the feedback for themselves. In this way, we believe that users can lead their design efforts using their creativity [30]. However, the participants revealed that they would have appreciated some automated corrections of details, such as misalignment. Future research should investigate the effects of such automated designs or corrections on users' experiences, and the quality of their work. For instance, an automated GUI and code generation technology from a GUI screenshot is one of the emerging technologies in GUI prototyping, including pix2code [28], ReDraw [27], and DesignScape [24,25]. However, other studies report that auto-generation can take the imagination of developers out of the design process and remove opportunities for learning how to improve designs [24,30]. In addition, there is a wide design space between automation and manual design. Future studies should investigate how to integrate automated support, while facilitating user-driven, creative design processes for GUIs.

## 7.1 Limitations

In the study, the users extensively searched for examples at the beginning for use as a template design for the given task. This behavior during design may imply that users can achieve high-quality designs by starting with high-quality examples that are more relevant to their design goals. However, our study does not report how the example quality affected the users' experiences, and the quality of their designs. Future research could investigate the effects of recommended example quality. Individual or subjective differences are an influencing factor in the design domain, but our experiments did not fully consider such differences, as we recruited participants at a university. We think users who perform design tasks other than the task in this work may call for other types of feedback. As GUIComp provides real-time feedback, some users may feel distraction, while designing, although we do not have a report on distraction in this study. A simple function for taking user preference could be incorporated for the distraction issue, such as an "on and off" switch interface for real-time feedback or a "New Feedback" button that updates feedback views, only when users click the button.

## VIII Conclusion

We propose and evaluate GUIComp, a feedback-based GUI prototyping tool to assist users, based on semi-structured interviews to identify users' roadblocks users face. GUIComp provides three types of feedback (evaluation, attention, and recommendation), which are dynamically updated as users continuously make changes in their design. The experimental results indicate that GUIComp allows users to create more acceptable mobile GUI designs with more fun, fulfilling, and satisfactory experiences.

## Acknowledgements

This thesis would not have been possible without support from many indispensable individuals:

- my advisor Prof. Sungahn Ko who leads visual analytics and human computer interaction fields. I would like to thank you for encouraging my thesis and for allowing me to grow as a researcher.
- my thesis committee Prof. Won-Ki Jeong and Young-Woo Park who give their insightful comments and encouragement, but also for the hard question which incited me to widen my research from various perspectives.
- my co-author Dr. Bum Chul Kwon for his precise and compact comments, along with a burst of insightful research ideas in every meeting we had.
- my co-authors Sanghoon Kim, Dongyun Han, and Hongjun Yang who participate my thesis project and discuss struggle points.
- my other paper co-authors Yeonjum Kim, Seungmin Jin, and Dongmin Kim who make valuable paper for publishing journal.
- iVADER labmates for the sleepless nights we were working together, including Kihwan Kim, Juyoung Oh, Yousang Oh, Yunha Han, Hwiyeon Kim, Gorakh Parsad, Kuartbek Mukabak
- friends from UNIST, especially members of the electrical and computer engineering.
- study participants for my research, including workers from Amazon's Mechanical Turk and student from UNIST.
- my family Yeongil Lee, Geumja Moon, and Seunghye Lee for their love and encouraging me whenever I was struggling with the hard graduate work.
- my best friend Junheon Min who is always on my side.

thanks.

## References

- [1] R. Nelson, “Apple app store will hit 5 million apps by 2020,” August 2018. [Online]. Available: <https://sensortower.com/blog/app-store-growth-forecast-2020>
- [2] A. Photoshop, August 2019. [Online]. Available: <https://www.adobe.com/products/photoshopfamily.html>
- [3] InVision, August 2019. [Online]. Available: <https://www.invisionapp.com/>
- [4] J. Nielsen, “Iterative user-interface design,” *Computer*, vol. 26, no. 11, pp. 32–41, 1993.
- [5] S. Dow, J. Fortuna, D. Schwartz, B. Altringer, D. L. Schwartz, and S. R. Klemmer, “Prototyping dynamics: sharing multiple designs improves exploration, group rapport, and results,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2011, pp. 2807–2816.
- [6] C. Kulkarni, S. P. Dow, and S. R. Klemmer, “Early and repeated exposure to examples improves creative work,” in *Design thinking research*, 2014, pp. 49–62.
- [7] M. Tohidi, W. Buxton, R. Baecker, and A. Sellen, “Getting the right design and the design right,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2006, pp. 1243–1252.
- [8] S. Weinschenk, *100 things every designer needs to know about people*. Pearson Education, 2011.
- [9] A. S. S. from Beginner App Developers!, August 2019. [Online]. Available: <https://codewithchris.com/success-stories/>
- [10] M. Deininger, S. R. Daly, K. H. Sienko, and J. C. Lee, “Novice designers’ use of prototypes in engineering design,” *Design studies*, vol. 51, pp. 25–65, 2017.
- [11] KakaoOven, August 2019. [Online]. Available: <https://ovenapp.io/>
- [12] AdobeXD, August 2019. [Online]. Available: <https://www.adobe.com/creativecloud.html>
- [13] Sketch, August 2019. [Online]. Available: <https://www.sketchapp.com/>
- [14] Principle, August 2019. [Online]. Available: <http://principleformac.com/>

- [15] UXPin, August 2019. [Online]. Available: <https://www.uxpin.com/>
- [16] Proto.io, August 2019. [Online]. Available: <https://proto.io/>
- [17] Axure, August 2019. [Online]. Available: <https://www.axure.com/>
- [18] S. R. Herring, C.-C. Chang, J. Krantzler, and B. P. Bailey, “Getting inspired!: understanding how and why examples are used in creative design practice,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2009, pp. 87–96.
- [19] S. P. Dow, A. Glassco, J. Kass, M. Schwarz, D. L. Schwartz, and S. R. Klemmer, “Parallel prototyping leads to better design results, more divergence, and increased self-efficacy,” *ACM Transactions on Computer-Human Interaction*, vol. 17, no. 4, pp. 1–24, 2010.
- [20] B. Lee, S. Srivastava, R. Kumar, R. Brafman, and S. R. Klemmer, “Designing with interactive example galleries,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2010, pp. 2257–2266.
- [21] D. Ritchie, A. A. Kejriwal, and S. R. Klemmer, “d. tour: Style-based exploration of design example galleries,” in *Proceedings of ACM Symposium on User Interface Software and Technology*, 2011, pp. 165–174.
- [22] M.-J. Tsai and D.-J. Chen, “Generating user interface for mobile phone devices using template-based approach and generic software framework,” *Journal of Information Science and Engineering*, vol. 23, no. 4, pp. 1189–1211, 2007.
- [23] A. Swearngin, M. Dontcheva, W. Li, J. Brandt, M. Dixon, and A. J. Ko, “Rewire: Interface design assistance from examples,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–12.
- [24] P. O’Donovan, A. Agarwala, and A. Hertzmann, “Learning layouts for single-page graphic designs,” *IEEE Transaction on Visualization and Computer Graphics*, vol. 20, no. 8, pp. 1200–1213, 2014.
- [25] P. O’Donovan, A. Agarwala, and A. Hertzmann, “Designscape: Design with interactive layout suggestions,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2015, pp. 1221–1224.
- [26] K. Todi, J. Jokinen, K. Luyten, and A. Oulasvirta, “Familiarisation: Restructuring layouts with visual learning models,” in *23rd International Conference on Intelligent User Interfaces*. ACM, 2018, pp. 547–558.
- [27] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk, “Machine learning-based prototyping of graphical user interfaces for mobile apps,” *arXiv preprint arXiv:1802.02312*, 2018.

- [28] T. Beltramelli, “pix2code: Generating code from a graphical user interface screenshot,” in *Proceedings of the ACM SIGCHI Symposium on Engineering Interactive Computing Systems*, 2017, pp. 1–6.
- [29] K. Moran, B. Li, C. Bernal-Cárdenas, D. Jelf, and D. Poshyvanyk, “Automated reporting of gui design violations for mobile apps,” in *Proceedings of International Conference on Software Engineering*, 2018, pp. 165–175.
- [30] C. Oh, J. Song, J. Choi, S. Kim, S. Lee, and B. Suh, “I lead, you help but only with enough details: Understanding user experience of co-creation with artificial intelligence,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2018, pp. 1–13.
- [31] M. Y. Ivory, R. R. Sinha, and M. A. Hearst, “Empirically validated web page design metrics,” in *Proceedings of ACM CHI Conference on Human factors in Computing Systems*, 2001, pp. 53–60.
- [32] K. Reinecke, T. Yeh, L. Miratrix, R. Mardiko, Y. Zhao, J. Liu, and K. Z. Gajos, “Predicting users’ first impressions of website aesthetics with a quantification of perceived visual complexity and colorfulness,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2013, pp. 2049–2058.
- [33] J. Koch and A. Oulasvirta, “Computational layout perception using gestalt laws,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems Extended Abstracts*, 2016, pp. 1423–1429.
- [34] A. Oulasvirta, S. De Pascale, J. Koch, T. Langerak, J. Jokinen, K. Todi, M. Laine, M. Kristhombuge, Y. Zhu, A. Miniukovich *et al.*, “Aalto interface metrics (aim): a service and codebase for computational gui evaluation,” in *The 31st Annual ACM Symposium on User Interface Software and Technology Adjunct Proceedings*. ACM, 2018, pp. 16–19.
- [35] A. Riegler and C. Holzmann, “Measuring visual user interface complexity of mobile applications with metrics,” *Interacting with Computers*, vol. 30, no. 3, pp. 207–223, 2018.
- [36] —, “Ui-cat: Calculating user interface complexity metrics for mobile applications,” in *Proceedings of the International Conference on Mobile and Ubiquitous Multimedia*, 2015, pp. 390–394.
- [37] A. Miniukovich and A. De Angeli, “Visual impressions of mobile app interfaces,” in *Proceedings of Nordic Conference on Human-Computer Interaction: Fun, Fast, Foundational*, 2014, pp. 31–40.
- [38] G. Ines, S. Makram, C. Mabrouka, and A. Mourad, “Evaluation of mobile interfaces as an optimization problem,” *Procedia Computer Science*, vol. 112, pp. 235 – 248, 2017.



- [39] A. Miniukovich and A. De Angeli, “Computation of interface aesthetics,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2015, pp. 1163–1172.
- [40] H. Khalid, E. Shihab, M. Nagappan, and A. E. Hassan, “What do mobile app users complain about?” *IEEE Software*, vol. 32, no. 3, pp. 70–77, 2014.
- [41] D. Zhang and B. Adipat, “Challenges, methodologies, and issues in the usability testing of mobile applications,” *International journal of human-computer interaction*, vol. 18, no. 3, pp. 293–308, 2005.
- [42] B. Doosti, T. Dong, B. Deka, and J. Nichols, “A computational method for evaluating ui patterns,” *arXiv preprint arXiv:1807.04191*, 2018.
- [43] Z. Wu, T. Kim, Q. Li, and X. Ma, “Understanding and modeling user-perceived brand personality from mobile application uis,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2019, p. 213.
- [44] Statista, “Mobile app usage,” August 2019. [Online]. Available: <https://www.statista.com/topics/1002/mobile-app-usage/>
- [45] B. Castro, “The significance of mobile app ui/ux design in app development,” January 2019. [Online]. Available: <https://yourstory.com/mystory/the-significance-of-mobile-app-ui-ux-design-in-app-fwq0lkhtvk>
- [46] J. Katariya, “Why ui/ux design is important for mobile apps?” December 2015. [Online]. Available: <https://www.moontechnolabs.com/why-uiux-design-is-important-for-mobile-apps/>
- [47] B. Glaser and A. Strauss, *The Discovery of Grounded Theory: Strategies for Qualitative Research*. Routledge, 2000.
- [48] B. C. Kwon, B. Fisher, and J. S. Yi, “Visual Analytic Roadblocks for Novice Investigators,” in *IEEE Conference on Visual Analytics Science and Technology (VAST)*, 2011, pp. 3–11.
- [49] J. Nielsen, Bush, R. M., T. Dayton, Mond, N. E., Muller, M. J., Root, and R. W., “Teaching experienced developers to design graphical user interfaces,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 1992, pp. 557–564.
- [50] K. S.-P. Chang and B. A. Myers, “Webcrystal: understanding and reusing examples in web authoring,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2012, pp. 3205–3214.
- [51] R. Kumar, J. O. Talton, S. Ahmad, and S. R. Klemmer, “Bricolage: example-based re-targeting for web design,” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2011, pp. 2197–2206.

- [52] J. Hattie and H. Timperley, “The power of feedback,” *Review of Educational Research*, vol. 77, no. 1, pp. 81–112, 2007.
- [53] D. R. Sadler, “Formative assessment and the design of instructional systems,” *Instructional Science*, vol. 18, pp. 119–144, 1989.
- [54] C. E. Kulkarni, M. S. Bernstein, and S. R. Klemmer, “Peerstudio: rapid peer feedback emphasizes revision and improves performance,” in *Proceedings of the ACM conference on learning@ scale*, 2015, pp. 75–84.
- [55] B. Buxton, *Sketching user experiences: getting the design right and the right design*. Morgan Kaufmann, 2010.
- [56] C. Extension, August 2019. [Online]. Available: <https://developer.chrome.com/extensions>
- [57] html2json, August 2019. [Online]. Available: <https://github.com/Jxck/html2json>
- [58] html2canvas, August 2019. [Online]. Available: <https://html2canvas.hertzen.com/>
- [59] OpenCV, August 2019. [Online]. Available: <https://opencv.org/>
- [60] B. Deka, Z. Huang, and R. Kumar, “Erica: Interaction mining mobile apps,” in *Proceedings of the Symposium on User Interface Software and Technology*, 2016, pp. 767–776.
- [61] B. Deka, Z. Huang, C. Franzen, J. Hibschan, D. Afegan, Y. Li, J. Nichols, and R. Kumar, “Rico: A mobile app dataset for building data-driven design applications,” in *Proceedings of ACM Symposium on User Interface Software and Technology*, 2017, pp. 845–854.
- [62] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines,” in *Proceedings of International Conference on Machine Learning*, 2010, pp. 807–814.
- [63] N. Hurley and M. Zhang, “Novelty and diversity in top-n recommendation–analysis and evaluation,” *ACM Transactions On Internet Technology (TOIT)*, vol. 10, no. 4, p. 14, 2011.
- [64] Drag-Mock, August 2019. [Online]. Available: <https://github.com/andywer/drag-mock>
- [65] Z. Bylinskii, N. W. Kim, P. O’Donovan, S. Alsheikh, S. Madan, H. Pfister, F. Durand, B. Russell, and A. Hertzmann, “Learning visual importance for graphic designs and data visualizations,” in *Proceedings of ACM Symposium on User Interface Software and Technology*, 2017, pp. 57–69.
- [66] A. Yuan, K. Luther, M. Krause, S. I. Vennix, S. P. Dow, and B. Hartmann, “Almost an expert: The effects of rubrics and expertise on perceived value of crowdsourced design critiques,” in *Proceedings of ACM Conference on Computer-Supported Cooperative Work & Social Computing*, 2016, pp. 1003–1015.

- [67] B. Hartmann, D. MacDougall, J. Brandt, and S. R. Klemmer, “What would other programmers do: suggesting solutions to error messages.” in *Proceedings of ACM CHI Conference on Human Factors in Computing Systems*, 2010, pp. 1019–1028.
- [68] Django, “Django,” August 2019. [Online]. Available: <https://www.djangoproject.com/>
- [69] Keras, August 2019. [Online]. Available: <https://keras.io/>
- [70] D3.js, August 2019. [Online]. Available: <https://d3js.org/>
- [71] Bootstrap, August 2019. [Online]. Available: <https://getbootstrap.com/>
- [72] A. Lund, “Measuring usability with the use questionnaire,” *Usability Interface*, vol. 8, no. 2, pp. 3–6, 2001.
- [73] W. Albert and T. Tullis, *Measuring the user experience: collecting, analyzing, and presenting usability metrics*. Newnes, 2013.
- [74] T. J. Ngoon, C. A. Fraser, A. S. Weingarten, M. Dontcheva, and S. Klemmer, “Interactive guidance techniques for improving creative feedback,” in *Proceedings of the CHI Conference on Human Factors in Computing Systems*, 2018, p. 55.
- [75] T. F. Liu, M. Craft, J. Situ, E. Yumer, R. Mech, and R. Kumar, “Learning design semantics for mobile apps,” in *Proceedings of ACM Symposium on User Interface Software and Technology*, 2018, pp. 569–579.

