Predicting and Explaining Mobile UI Tappability with Vision Modeling and Saliency Analysis

Eldon Schoop[*]
eschoop@berkeley.edu
UC Berkeley EECS
Berkeley, CA, USA

Xin Zhou
zhouxin@google.com
Google Research
Mountain View, CA, USA

Gang Li
leebird@google.com
Google Research
Mountain View, CA, USA

Zhourong Chen
czhrong@gmail.com
Google Research
Mountain View, CA, USA

Björn Hartmann
bjoern@eecs.berkeley.edu
UC Berkeley EECS
Berkeley, CA, USA

Yang Li
yangli@acm.org
Google Research
Mountain View, CA, USA

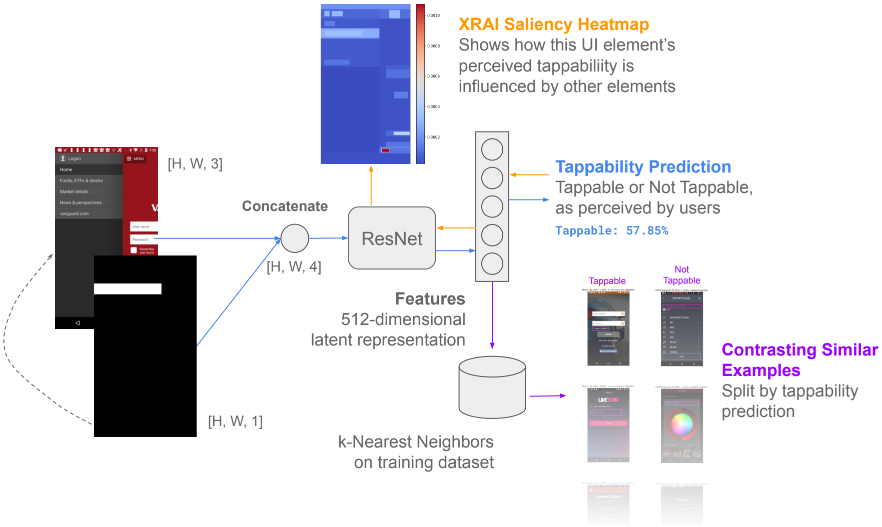

0.0010
0.0008
0.0006
0.0004
0.0002
XRAI Saliency Heatmap
Shows how this UI element's perceived tappabiliity is influenced by other elements
[H, W, 3]
Concatenate
ResNet
[H, W, 4]
[H, W, 1]
Tappability Prediction
Tappable or Not Tappable, as perceived by users
Tappable: 57.85%
Features
512-dimensional latent representation
k-Nearest Neighbors on training dataset
Tappable
Not Tappable
Contrasting Similar Examples
Split by tappability prediction

Figure 1: We use a deep learning based approach to predict whether a selected element in a mobile UI screenshot will be perceived by users as tappable, based on pixels only instead of view hierarchies required by previous work. To help designers better understand model predictions and to provide more actionable design feedback than predictions alone, we additionally use ML interpretability techniques to help explain the output of our model. We use XRAI to highlight areas in the input screenshot that most strongly influence the tappability prediction for the selected region, and use k-Nearest Neighbors to present the most similar mobile UIs from the dataset with opposing influences on tappability perception.

ABSTRACT

UI designers often correct false affordances and improve the discoverability of features when users have trouble determining if elements are tappable. We contribute a novel system that models the perceived tappability of mobile UI elements with a vision-based deep neural network and helps provide design insights with dataset-level and instance-level explanations of model predictions. Our system retrieves designs from similar mobile UI examples from our dataset using the latent space of our model. We also contribute a novel use of an interpretability algorithm, XRAI, to generate a heatmap of UI elements that contribute to a given tappability prediction. Through several examples, we show how our system can help automate elements of UI usability analysis and provide insights for designers to iterate their designs. In addition, we share findings from an exploratory evaluation with professional designers to learn how AI-based tools can aid UI design and evaluation for tappability issues.

[*] This work was completed while the author was an intern at Google.

CHI '22, April 29-May 5, 2022, New Orleans, LA, USA
© 2022 Copyright held by the owner/author(s).
ACM ISBN 978-1-4503-9157-3/22/04.
https://doi.org/10.1145/3491102.3517497

CCS CONCEPTS

• **Human-centered computing → Ubiquitous and mobile computing design and evaluation methods**; **HCI design and evaluation methods**.

KEYWORDS

Mobile UIs, Deep Learning, Explainable AI, Interpretability

ACM Reference Format:
Eldon Schoop, Xin Zhou, Gang Li, Zhourong Chen, Björn Hartmann, and Yang Li. 2022. Predicting and Explaining Mobile UI Tappability with Vision Modeling and Saliency Analysis. In *CHI Conference on Human Factors in Computing Systems (CHI '22), April 29-May 5, 2022, New Orleans, LA, USA.* ACM, New York, NY, USA, 21 pages. https://doi.org/10.1145/3491102.3517497

1 INTRODUCTION

Tapping is a fundamental gesture in mobile User Interfaces (UIs). However, because of the highly varied styles of mobile UIs, users can have difficulty telling if UI elements are tappable [37]. This harms the usability of applications, e.g., when false affordances suggest an item is tappable when it is not; or when the design of a new feature limits its discoverability.

UI designers and User Experience (UX) researchers traditionally run user studies to evaluate the usability of their designs. While these studies can provide actionable feedback and lead to significant design insights, they are often costly and time-consuming to conduct. Recent works have applied Deep Learning (DL) techniques to predict whether users will correctly estimate if mobile UI elements are tappable [37] and predict user engagement with mobile UI animations [42]. These automated approaches can help designers gain quick insights into the usability of their applications, but lack the design guidance and explanations that can be gained from controlled user studies. In addition, many automated tools rely on a functional mobile application or UIs with detailed specifications, such as view hierarchies, meaning that they may not be able to produce usable results on mockups. Yet, gaining feedback in the early stages of design is crucial.

The goal of this work is to produce a model that faithfully approximates the perception of real users for rapid, automated tappability evaluations, and a system which provides explanations of its predictions that offer insight for improving designs. To gain a basis for understanding tappability perception at scale, we create a new dataset of crowdworkers' estimates of the tappability of UI elements in thousands of mobile UI screenshots from the RICO dataset [11]. As shown in previous work [37], human perceptions of tappability can vary significantly. To account for this, our new dataset includes 5 crowdworkers' labels for each UI element, by which we can more reliably estimate user perceptions at scale. We use this dataset to train a purely vision-based deep neural network that, given a screenshot and a selected region of interest, predicts the perceived tappability of the selected UI element. This allows designers to rapidly assess how users may perceive elements of a mobile UI design, whether or not it is implemented in an application.

We take an important step further beyond tappability prediction by drawing upon techniques in Machine Learning (ML) interpretability and Explainable Artificial Intelligence (XAI) to explain our model's predictions in two ways [34]. We provide a *local* explanation which highlights regions in the input screenshot, indicating areas the model considers most important to the tappability prediction for a given element. We provide a *global* explanation which uses the latent space of our model to find contrasting nearest-neighbor examples in our source dataset, allowing users to discover patterns in visually similar UIs that have opposing influences on tappability perception. To evaluate our model and its explanation outputs, we share an in-depth analysis of the behavior of our model using random examples from our source dataset, and conduct an exploratory evaluation to seek feedback from professional UI/UX designers.

Specifically, this paper contributes:

- A new dataset collecting tappability labels from multiple crowdworkers per example on thousands of mobile application screenshots[1]. This extends previous work [37] to better address human uncertainty in tappability perception;
- A vision-based deep neural network that predicts the perceived tappability of selected UI element(s) in a mobile UI screenshot by only relying on pixels. Our model is capable of examining UI designs that are not fully specified (e.g., mockups). This significantly extends prior work since it enables a broader set of applications, e.g., to produce feedback for early-stage designs;
- A novel method for eliciting explanations of tappability predictions from our model by annotating the screen under inspection, and by surfacing similar examples from the dataset that have opposing influences on tappability perception;
- An in-depth analysis of model behavior on randomly selected examples from an evaluation dataset, and an exploratory evaluation with 13 professional UI/UX designers, from which we distill initial insights into how an AI-based tool can assist designers.

2 RELATED WORK

Our work builds on three primary areas: automated tools which assist UI designers in exploring and evaluating UIs; automated tools which assist in evaluating the usability of UIs; and algorithms and methods for interpreting the predictions of deep neural networks.

2.1 Data-Driven UI Design and Exploration

The HCI community has produced many research artifacts that help designers create UIs through the collection and use of large-scale UI datasets [27]. Datasets such as ERICA [13] and RICO [11] have enabled the creation of numerous data-driven systems in this domain. While the vast size of RICO has made it attractive for data-driven applications in research, it is known to have significant label noise [26]. Many works add annotations to RICO or take additional cleaning steps, e.g., ENRICO, which organizes RICO into design topics [24], and $\text{RICO}_{\text{clean}}$ which relabels icon elements in the original dataset [44]. Our work contributes a dataset that augments a cleaned subset of RICO with annotations from multiple crowdworkers predicting the tappability of various UI elements.

Designers benefit from viewing selections of varied UI design examples to serve as inspiration in the design process [38]. Gallery

[1] We release our dataset publicly at https://github.com/google-research/google-research/tree/master/taperception.

DC uses a neural network to tag elements in mobile UI screenshots, presenting them in a gallery to help designers explore a large set of UI element examples [7]. Other works help designers retrieve examples from datasets like RICO, e.g., from hand-drawn sketches [20], low-fidelity wireframes [9], and text-annotated layout information [2, 18, 25]. We also use the latent space of a deep neural network for UI retrieval. However, our model is trained on the perception of human raters, rather than to reconstruct UI layouts. This means that retrieved examples are similar in how they are perceived by humans to be tappable, rather than in visual similarity alone. In addition, our model uses the raw pixels of a mobile UI as input, allowing it to capture more detailed visual features than layouts.

2.2 Computationally Mediated UI Evaluation

Because of the cost and time involved in running controlled user studies, many systems have emerged which use heuristics, data-driven techniques, or crowdsourcing to evaluate UIs more rapidly. An early example is CogTool, which predicts task completion time for skilled users [3]. Other tools detect underlying usability hurdles by analyzing UI layouts to find rendering errors [8], or by using crowdsourcing to find issues in interaction traces [12].

Other approaches detect usability issues by modeling visual perception and highlighting mismatches with designers’ expectations [23]. Deep neural networks have been used to create attention maps of visual designs [4, 14]. Our work is most similar to TapShoe, which uses a deep neural network to model users’ tappability perceptions of mobile UI elements [37]. We extend this work by introducing a purely vision-based neural network, which enables several new applications due to its ability to run on mockups as well as functional applications. In addition, a key limitation of many automated evaluation tools is that designers must rely on their own judgment to decide how to modify their designs to improve evaluation results. Our work takes a significant step beyond prior work by using ML interpretability techniques to give designers more actionable information than predictions alone. Specifically, our system highlights the regions that influence our model’s tappability predictions, and it retrieves relevant, contrasting UI examples for design inspiration.

2.3 Interpreting and Explaining Deep Neural Network Predictions

Deep neural networks are considered “black box” models since they often have too many parameters to be easily understood, and are not considered to be inherently interpretable [28]. Emergent work in the ML community has produced several algorithms and techniques that can help highlight the particular inputs to a neural network that influence its predictions. Some methods use backpropagation to attribute pixels in an input image [36], use the convolutional features of vision models [35], or aggregate and merge highly salient pixels into regions [21]. Other methods approximate a more interpretable, linear model to annotate what input features are near decision boundaries [32], or combinatorially perturb the input to determine which of its features are most influential [30]. We modify the XRAI algorithm [21] to attribute input features which influence our model’s tappability predictions.

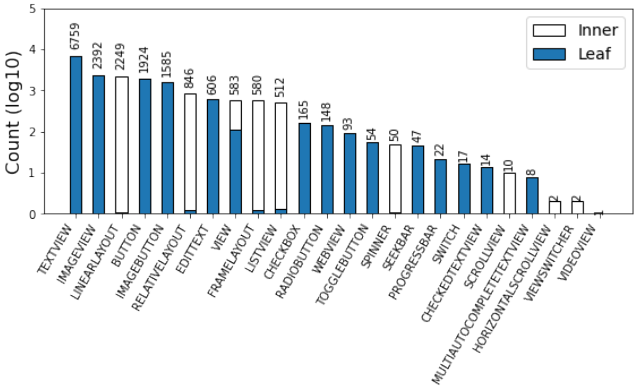


Figure 2: The type distribution for the 18,667 labeled UI elements. Blue and white splits show the proportion of leaf and inner elements in the view hierarchy.

Other methods use the training dataset to provide external context that can help explain model predictions. A well-known example is to use concept vectors, which can detect the presence of learned “concepts” (e.g., “stripes”, “wheels”, or “clouds” in images) in a model prediction [22], or identify important features across a dataset [16]. In our work, we use the latent space of our model to retrieve similar examples from our dataset, a known technique for describing model predictions by using other examples [31]. We split retrieved UIs into contrasting examples [5] by their tappability prediction. This exposes designers to similar UI elements with differing effects on perception, a technique based on the variation theory of learning [10].

3 CROWDSOURCING PERCEIVED TAPPABILITY FROM SCREENSHOTS

Similar to [37], we perform a tappability study on a large set of UI elements in Android mobile app screens. The raters are given a screenshot from the screen set with one of the elements highlighted, and indicate whether the UI element is tappable or not. Each UI element is labeled by 5 different raters. Each worker completed up to 90 UI elements, with a median of 30.

We collect 18667 unique UI elements from 3218 screens from the RICO dataset [11]. In the view hierarchy of each screen, we select up to five unique clickable and non-clickable elements for labeling. Similar to [44], we asked crowdworkers to discard examples whose bounding boxes were not aligned with underlying UI elements. The same filter rules are applied as [37]: we (1) choose top-level clickable elements starting from leaves and, (2) avoid choosing the children of already-chosen non-clickable elements.

There are 24 different types of collected elements, 77% of which are leaves in their corresponding view hierarchy trees (Figure 2). By analyzing the labels and the screens, we notice that some UI elements are labeled with high agreement, but others are not (Table 1). For 44.4% of UI elements, 5 raters agreed unanimously. However, 24.1% of UI elements were ambiguous to raters, i.e., at most 3 agreed on a label. Nonetheless, as each element is inspected by multiple raters, our dataset has more precise labels about human tappability

Table 1: Agreement of tappability for the 18,667 labeled UI elements.

# of workers for agreement	# of UI elements	ratio
3-agreement	4508	24.1%
4-agreement	5872	31.5%
5-agreement	8287	44.4%

perception than prior work, which is desirable for machine learning tasks and data analysis. Our dataset also reveals UI elements that are indeed ambiguous, for future analysis. For model training in this work, we randomly split the dataset into 80% of the UI elements for training, 10% for validation to tune hyperparameters, and 10% for testing. We release our dataset publicly on github: https://github.com/google-research/google-research/tree/master/taperception.

4 MODELING PERCEIVED TAPPABILITY FROM IMAGES

Since the applications in our dataset use many UI frameworks and design styles, the patterns persistent in this data can be generalized to predict the tappability of elements in many kinds of mobile UIs. In this section, we describe how we use our dataset to train a Convolutional Neural Network (CNN) model for tappability prediction. The problem statement for our model is: given an input screenshot and region of interest (a rectangular area within the input screenshot), predict whether or not users will perceive the indicated UI element as tappable or not tappable.

Our CNN model is purely vision-based, which significantly differs from prior work in tappability prediction [37], and provides several advantages. While earlier tappability prediction models required multiple feature types as input (e.g., a screenshot and a selected element's Android View type, text content, and its intended tappability), our model only uses screenshot pixels as input. This significantly broadens the set of applications our model may be used for, such as UIs that are not fully-specified. For example, designers may be able to use our model to evaluate iterations in earlier design stages since it can operate on visually realistic mockups. However, since our model does not directly capture text, element type, or intended clickability information from input UIs, the model from Swearngin et al. [37] may have advantages in contexts where non-visual signifiers (e.g., text content) are used by designers to explicitly indicate tappability. Since our vision-based model does not rely on platform-specific inputs (i.e., element types), it can be fine-tuned for platform-agnostic applications. This also makes it easier to adapt our model to other domains in future work, such as emergent datasets of iOS applications [41], or other downstream tasks, e.g., predicting accessibility barriers [45].

Our model's inputs are specified as follows. Let $I \in \mathbb{R}^{h \times w \times 3}$ denote the pixel values of a UI screenshot, where h and w are the screen height and width, and 3 is the number of channels (i.e., RGB). Let (x_{min}, y_{min}) and (x_{max}, y_{max}) denote the top-left and bottom-right corner coordinates of a target UI element bounding box respectively.

A naive implementation of using CNNs for learning tappability is to crop the target element's pixels from I and feed them to a CNN. However, this discards important contextual information in the screen, making it difficult to learn an effective model. Instead, we feed the entire RGB screenshot to the model along an additional mask channel in the input. For a given element, we first create a binary mask $M \in \{0, 1\}^{h \times w}$, using i and j as row and column indices, respectively:

$$M_{ij} = \begin{cases} 1, & \text{if } y_{min} <= i < y_{max} \text{ and } x_{min} <= j < x_{max} \\ 0, & \text{otherwise} \end{cases}$$

In other words, the entries corresponding to the target element's pixels are 1's and the others are 0's in the binary mask. We then concatenate I and M along the channel dimension to form the input to the model: $I' = [I, M]$ of shape $[h, w, 4]$. To the model, I provides pixel information of the whole screen, while M indicates the screen area for which the model should predict tappability (Figure 1).

Specifically, our model is a Resnet-18 [17], modified to accept a larger input image with a dimension of 960 by 540 (to accommodate mobile UI screenshots) along with the corresponding binary mask. The model outputs softmax probabilities for two classes: tappable, or not tappable. We train our model on the training set by minimizing cross-entropy loss, using Stochastic Gradient Descent with Nesterov momentum, with a learning rate of 0.05 and a batch size of 1024, for 1500 epochs. Our learning rate decayed by an order of magnitude (dividing by 10), after epochs 100, 500, 1000, and 1300. We evaluated how well our model predicted user perceptions of the tappability of UI elements with our test set. Our model achieved a precision of 91.54% and recall of 80.23% with a decision threshold of 50%, and AUC of 0.9030.

To compare the performance of our model to previous work in tappability prediction, we replicated the model from Swearngin et al. [37] and benchmarked this model on our new dataset in two separate configurations: by using all of its input features (screenshot pixels, region pixels, component text, component type, and intended tappability), and by using pixels only (from the screenshot and region). Our model, which only uses pixels, clearly outperforms the replicated model [37] when it only runs on pixels. When the replicated model uses all input features, including those from the view hierarchy, on our dataset, our model achieves better AUC and similar precision, but has slightly lower recall when using a 0.5 decision boundary (Table 2). The slightly lower recall of our model is likely due to the distribution of tappable elements in our dataset, which can be addressed by fine-tuning the decision threshold.

5 EXPLAINING TAPPABILITY PREDICTIONS

Our neural network can be used to model users' perceptions of tappability for a broad variety of mobile UI elements. However, the predictions of models like ours are limited in the sense that designers must rely on their own judgment to determine what visual cues were responsible for the prediction, and, if needed, how the design must be modified to improve its perception (Figure 3). We draw upon techniques from XAI and ML interpretability to provide deeper explanations of our model's predictions, both in the context of the input itself, as well as examples the model has learned from. We implement two types of explanations: at the *local* level, to suggest which elements in the input screenshot were most influential, or "salient", to a given prediction, and at the *global* level,

Table 2: Tappability prediction model performance on our new dataset. Our model, which only uses pixels, clearly outperforms [37] when only run on pixels. It has slightly higher AUC, similar precision, and slightly lower recall compared to the all-features replicated model.

Model	AUC	Precision (%)	Recall (%)
Ours (pixels only)	0.9030	91.54	80.23
Swearngin et al. [37]; pixels + all other features	0.8437	91.65	84.53
Swearngin et al. [37]; pixels only	0.6521	76.79	80.79

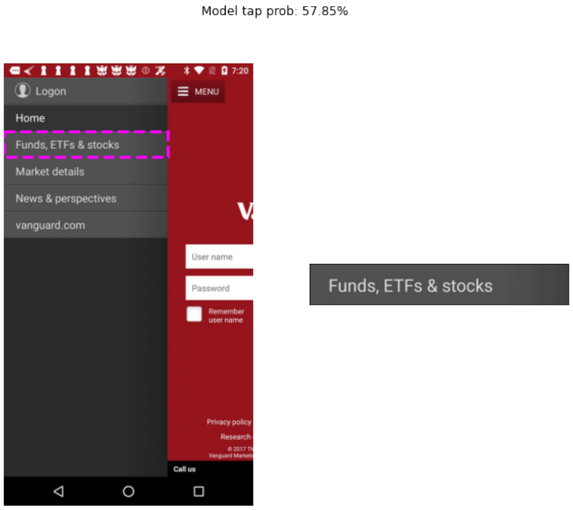


Figure 3: The input to our model as a running example to this section, a randomly selected screenshot from our dataset. The element of interest is indicated as a magenta dashed rectangle. Our model predicts the element is tappable with a probability of 57.85%.

to show how other applications with similar design patterns can influence tappability perception positively and negatively.

5.1 Attributing Tappable UI Elements with Saliency Techniques

To provide a *local* explanation of the model's predictions, we use the XRAI algorithm [21], a gradient-based algorithm which produces a heatmap highlighting what regions of an input image were the most influential to a given model output, also known as a saliency map (Figure 4). Importantly, while the output of saliency algorithms like XRAI are correlative, and cannot explain the causal reasons behind model predictions, they are often useful for gaining a better understanding of model behavior in many applications [1]. In our use case, we use XRAI to generate a heatmap of the UI components in a mobile app screenshot that most strongly influence the tappability prediction for a particular element. XRAI calculations and heatmaps are particular to the specified UI element in a tappability prediction, since predictions for different UI elements can depend on their particular context and relationship to other UI elements. Designers can use the XRAI heatmap to see when the perceived tappability of a particular element is heavily influenced by other regions on the screen, e.g., how introducing a new component changes the perception of surrounding elements.

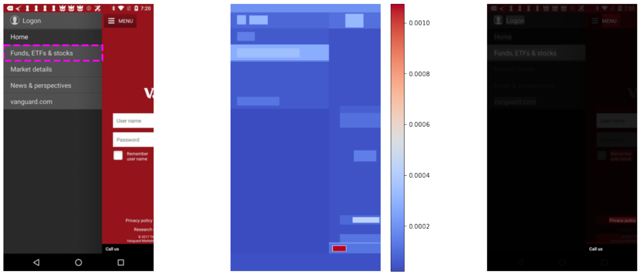


Figure 4: Left: The same input screenshot as in Figure 3 with a selected element in a magenta dashed rectangle. Center: the heatmap generated by XRAI, using regions from UI elements. The regions which most strongly influence the selected element's tappability prediction are rendered in red, while the least influential regions are rendered in blue. Some text is extremely highly attributed (an anomaly). Right: the input screenshot filtered by the values of the saliency heatmap. The elements most important to the tappability prediction are the brightest.

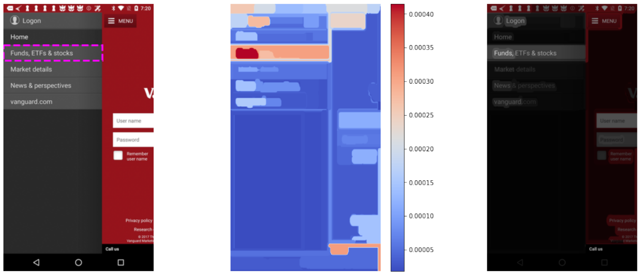


Figure 5: Center: the same XRAI calculation as in Figure 4, but without using provided regions from the UI element bounding boxes. Regions are generated using Felzenszwalb segmentation.

The XRAI algorithm works by first oversampling the input image into overlapping superpixels of different sizes. Next, Integrated Gradients, a pixel-based attribution method [36], is calculated on the input image from black and white baselines. These pixel-level attributions are then aggregated by summing over segments, ranking segments from most to least important, and merging them up to a selected threshold. We make one key modification to the

XRAI technique. Rather than oversample the image using Felzenszwalb segmentation, we use the native bounding boxes of mobile UI elements if they are available or can be specified. This means we can directly summarize model attributions for regions corresponding to mobile UI widgets, reducing the noise from automated segmentation methods (Figure 5).

5.2 Explaining Predictions with Similar Contrasting Examples

Our *global* explanation method situates the given prediction in the context of retrieved examples from our mobile UI dataset. We use nearest neighbors on embeddings from our model to find examples the model considers similar. The model's embeddings capture visual similarity, and the rough position and size of the input bounding box (see section 6). These nearest neighbors are then split by the model's tappability prediction, creating a contrasting explanation [5]—a visualization of a set of UIs that have similar designs to the input, but opposing influences on the perception of tappability. This acts as a set of curated examples for design inspiration to help designers make changes that affect users' tappability perceptions of UI elements (Figure 6).

To capture embeddings from our neural network, we take the output from its final convolutional layer and flatten it into a 512-dimensional vector. We precompute embeddings for every mobile UI example in our source dataset, and split them into two separate indexed arrays of predicted tappable and nontappable examples. To filter out potentially confusing or ambiguous examples, we limit these lists to examples which have >65% and <35% tappability probabilities, per our model's predictions. In practice, we found that splitting based on model predictions produces more consistent results than ground-truth human labels. We use the NearestNeighbors learner from the sklearn Python package to search for the 5 nearest neighbors from each list (showing 10 examples total), to embeddings from an input image.

6 ANALYSIS OF SELECTED EXAMPLES

In this section, we sample real-world screenshots from our dataset to show how our model performs and what our explanations capture. We randomly select four elements from our dataset that have associated regions corresponding to common Android UI leaf elements: ImageView, Button, TextView, and EditText. For each of these inputs, we show the output of our model and explanation methods, and describe what could be inferred about the behavior of our model. In section 8, we summarize trends apparent in our model across examples and discuss their implications and opportunities for future work.

6.1 ImageView: Food App Header Logo

This randomly-selected UI and element is a screenshot from a food application, presenting a complex login view with many clickable buttons and graphics. The selected region paired with this UI screenshot is a logo placed above the login form. The model predicts this element is not tappable, with a 10.01% tap probability.

The XRAI heatmap strongly attributes the input element as important to its tappability prediction, and does not factor other elements in the screen much. It is possible that, since the input element

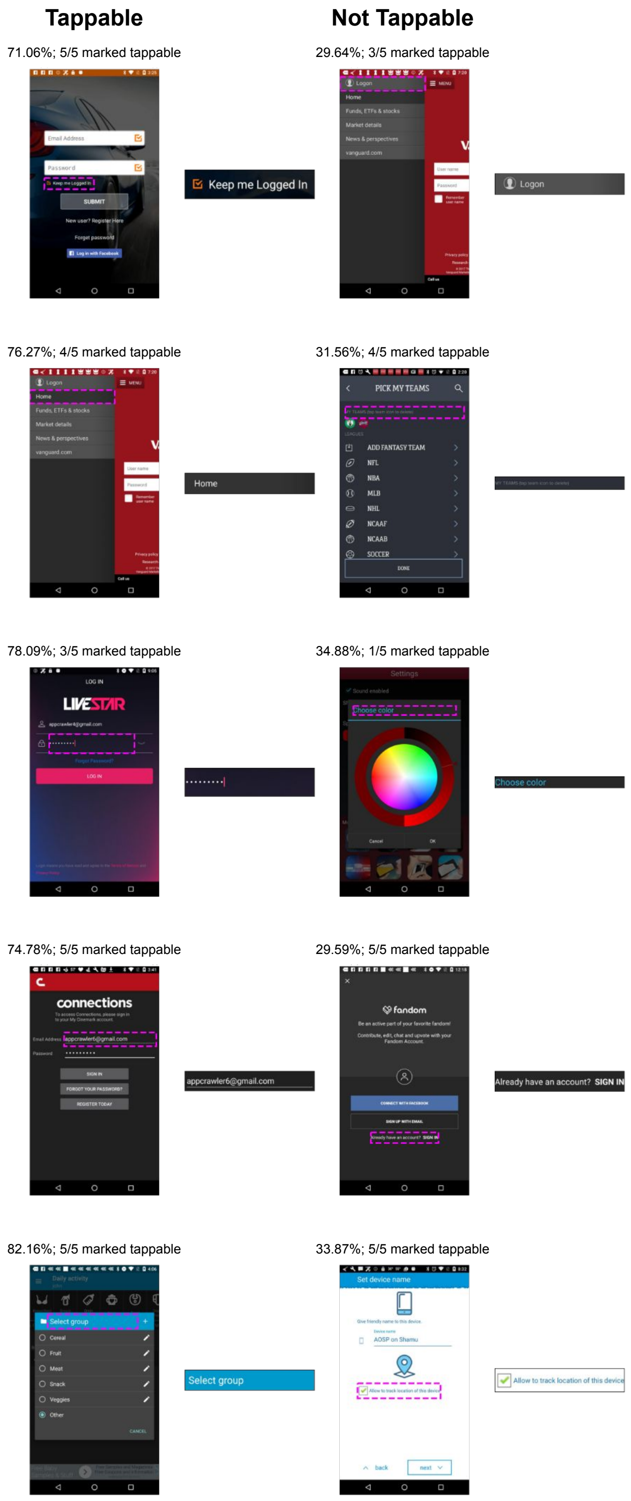

Tappable
Not Tappable
71.06%; 5/5 marked tappable
Keep me Logged In
29.64%; 3/5 marked tappable
Logon
76.27%; 4/5 marked tappable
Home
31.56%; 4/5 marked tappable
78.09%; 3/5 marked tappable
LIVESTAR
34.88%; 1/5 marked tappable
Choose color
74.78%; 5/5 marked tappable
connections
appcrawler6@gmail.com
29.59%; 5/5 marked tappable
fandom
Already have an account? SIGN IN
82.16%; 5/5 marked tappable
Select group
33.87%; 5/5 marked tappable
Set device name
Allow to track location of this device

Figure 6: Nearest neighbors to the input screenshot from Figure 3, split by tappability predictions. Examples the model predicts as tappable are on the left, with non-tappable examples on the right. Examples contain an entire screenshot with a specified region, and the region in a larger view. Columns are sorted by distance to the original input in the model's latent space (most similar on top).

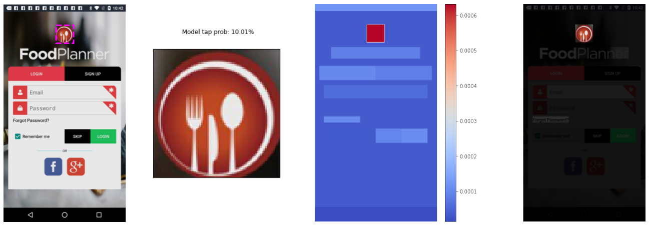


Figure 7: Example from subsection 6.1. Left: input screenshot with a ImageView element selected, annotated with a magenta dashed rectangle. Center Left: Close-up view of the selected UI element. The model predicts it is not tappable, with a 10.01% tappability probability. Center Right: The XRAI heatmap most strongly illuminates the selected element, and does factor in other elements significantly. Right: the input screenshot filtered by the values of the saliency heatmap.

is a graphic, the model does not consider surrounding elements a significant factor. Combined with the relatively high model confidence, we can assume that properties of the input element itself (its appearance or position) strongly signify non-tappability on their own. This means that making significant changes to other elements on the screen would likely not impact the perceived tappability of this element.

Most nearest neighbors of the food app also contain graphical elements and icons, with the exception of large text objects that have similar locations and sizes on the screen as the input (Figure 8). The non tappable elements are generally larger, and closer to the center of the screen, matching the style of the input. Tappable elements tend to be icons commonly associated with actions, e.g., a shopping cart and an "X" to close a dialog.

6.2 Button: Health App Card Button

This element is a screenshot from a health application, presenting a complex view with a card, image, and list. The selected region paired with this UI screenshot is a "Dismiss" button within a card. The model predicts this element is tappable, with a 99.07% tap probability (Figure 9).

Similar to the ImageView example, the XRAI heatmap most strongly illuminates the input element itself. This is likely because the model has learned to associate Material Design buttons with a strong perception of tappability, and does not need to reference much context to establish a confident prediction. The attributed text in the screen's title card ("Learn") may suggest the model's attention to a common Material UI standard.

Tappable neighbors are entirely buttons and tabs with overlaid text and high tappability scores. Non-tappable neighbors are more mixed, including descriptive text, icons, and even images. The predictions for several non-tappable examples disagree with the underlying raters' labels (Figure 10). The noise in the non-tappable examples could be a limitation of discretizing the neighbors by tappability prediction, an effect discussed further in subsection 8.3.

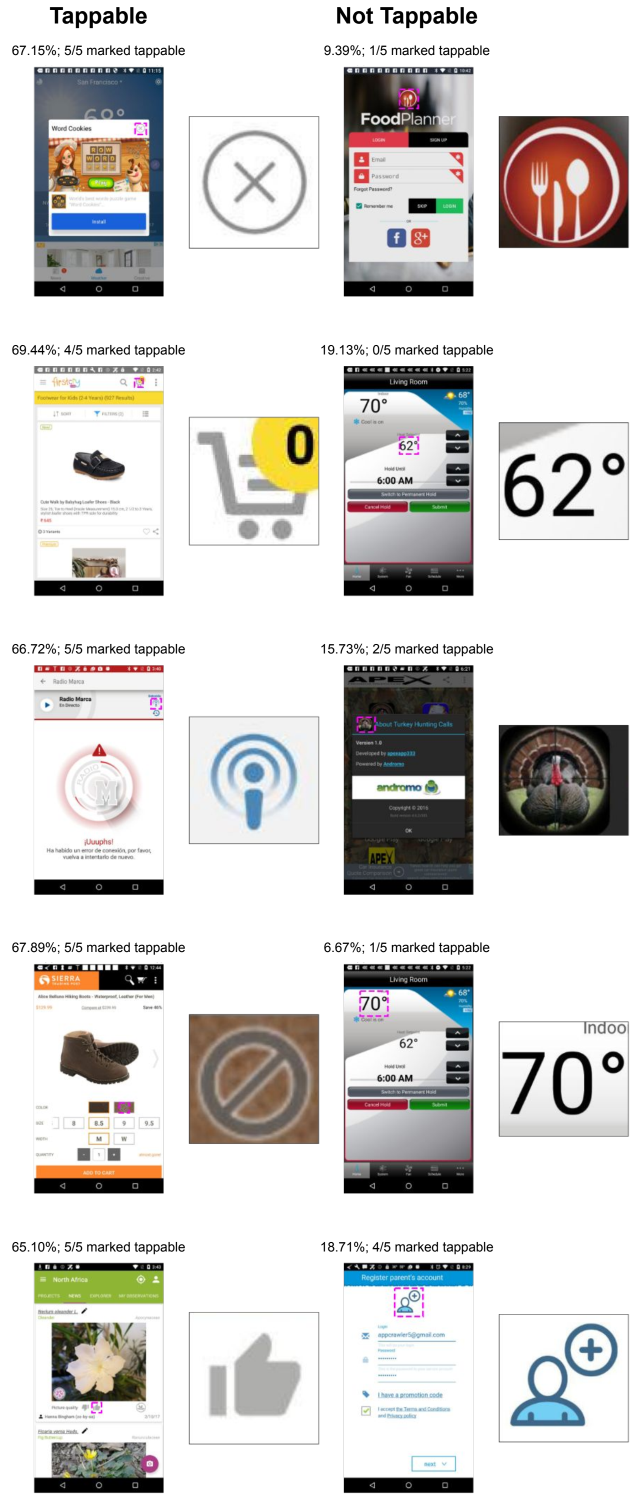

Tappable
Not Tappable
67.15%; 5/5 marked tappable
9.39%; 1/5 marked tappable
69.44%; 4/5 marked tappable
19.13%; 0/5 marked tappable
66.72%; 5/5 marked tappable
15.73%; 2/5 marked tappable
67.89%; 5/5 marked tappable
6.67%; 1/5 marked tappable
65.10%; 5/5 marked tappable
18.71%; 4/5 marked tappable

Figure 8: Nearest neighbors from subsection 6.1, split by thresholded model predictions (tappable neighbors on the left). Many neighbors (both tappable and not) are graphical (icons and drawings). Tappable elements tend to be smaller, and situated near the edges of other elements. Non tappable elements are generally larger, and closer to the center of the screen.

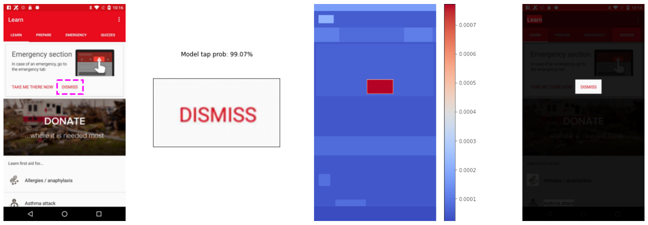


Figure 9: Example from subsection 6.2. Left: input screenshot with a Button element selected, annotated with a magenta dashed rectangle. Center Left: Close-up view of the selected UI element. The model predicts it is tappable, with a 99.07% tappability probability. Center Right: The XRAI heatmap most strongly illuminates the input element itself. This is likely because the model has learned to associate Material Design buttons with perceptions of tappability, and does not need to reference much context to establish a confident prediction. Right: the input screenshot filtered by the values of the saliency heatmap.

6.3 TextView: Finance App List Item

This example is a screenshot from a finance app, with a view presenting a chart and pricing details of a stock (Figure 11). The selected region paired with this UI screenshot is a text field displaying a bid price. The model predicts this element is not tappable, with a 36.38% tap probability.

The XRAI heatmap strongly illuminates the region itself, while also strongly highlighting text in tab navigation and an icon adjacent to a nearby text view. Although the input region is generally expected to be the most important element for its own prediction, one element of tab text is highly attributed, an anomaly. This may be due to variances in TextView tappability when below navigation tabs. It is also worth noting that surrounding text views are lightly attributed as well, suggesting the model has factored some surrounding context into the input element's prediction.

All nearest neighbors of this input screenshot share strong visual similarities with the input (text on a light background), but appear in different contexts (Figure 12). Many tappable elements have icons or graphics nearby, which possibly serve as signifiers of the tappabillity of the adjacent text. Non tappable elements have brighter text, and are often placed as descriptions next to tappable elements. It is worth noting that many tappable elements are also ListViews, 2 of which have similar color schemes, indicating the model is factoring multiple contextual elements within the input example besides the region itself.

6.4 EditText: Entertainment App Login Field

This element is a screenshot from an entertainment application, a simple login view. The selected region paired with this UI screenshot is a "Password" text field. The model predicts this element is tappable, with a 99.47% tap probability (Figure 9).

Like previous examples, the XRAI heatmap strongly attributes the selected EditText view, and does not attribute other elements on the screen. Similar to the Button example, it is possible that the

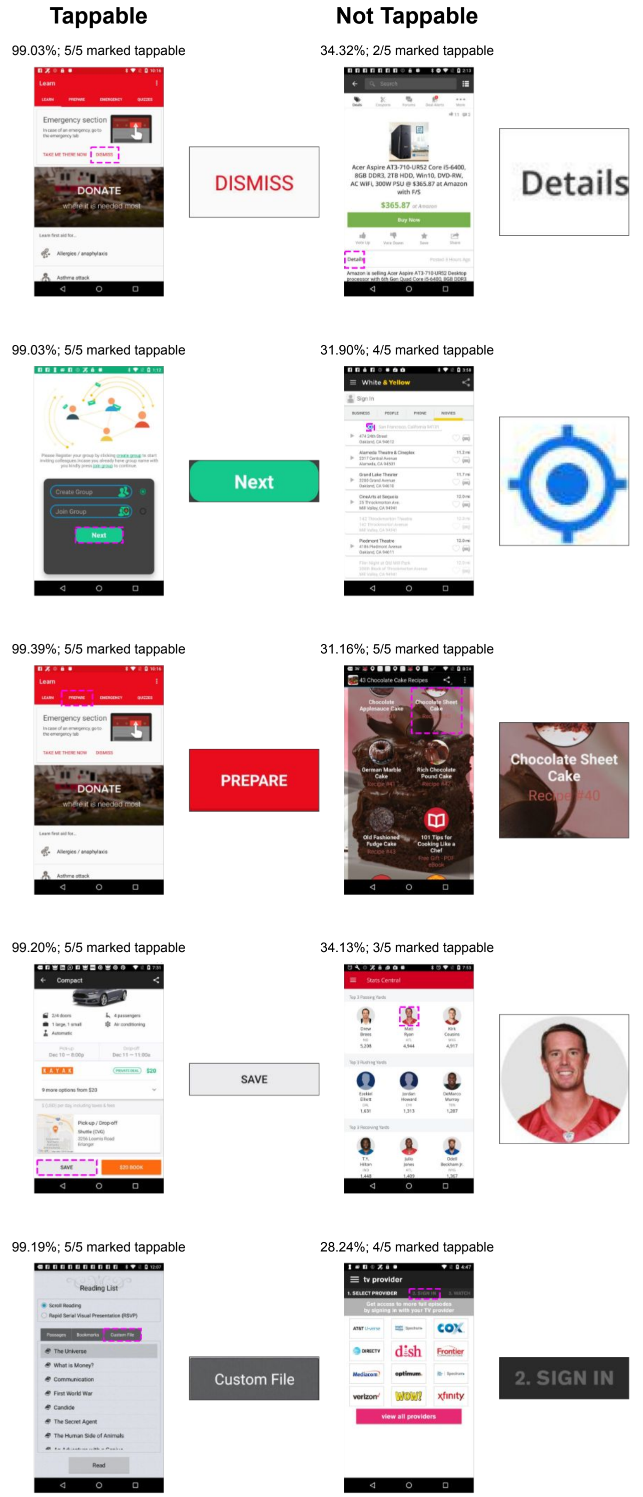

Tappable
Not Tappable
99.03%; 5/5 marked tappable
34.32%; 2/5 marked tappable
DISMISS
Details
99.03%; 5/5 marked tappable
31.90%; 4/5 marked tappable
Next
99.39%; 5/5 marked tappable
31.16%; 5/5 marked tappable
PREPARE
Chocolate Sheet Cake
99.20%; 5/5 marked tappable
34.13%; 3/5 marked tappable
SAVE
99.19%; 5/5 marked tappable
28.24%; 4/5 marked tappable
Custom File
2. SIGN IN

Figure 10: Nearest neighbors for subsection 6.2, split by thresholded model predictions (tappable neighbors on the left). While many tappable elements contain similarly-styled buttons from other apps, the non-tappable elements display highly varied elements, with and without text. A potential cause of this is that most elements near the input button in latent space are other buttons; and the nearest non-tappable examples are significantly further away, so they are not as visually similar.

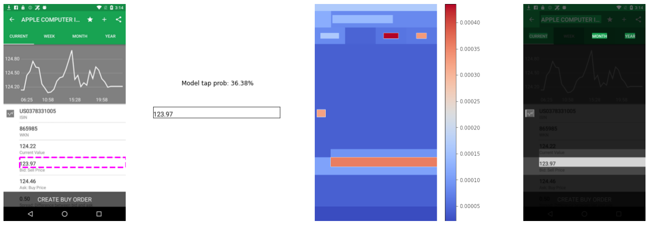


Figure 11: Example from subsection 6.3. Left: input screenshot with a TextView element selected, annotated with a magenta dashed rectangle. Center Left: Close-up view of the selected UI element. The model predicts it is not tappable, with a 36.38% tappability probability. Center Right: The XRAI heatmap strongly illuminates the text view region itself, while also strongly highlighting text in tab navigation and an icon adjacent to a nearby text view. Right: the input screenshot filtered by the values of the saliency heatmap.

model has learned an association between the Material UI EditText component and strong perceptions of tappability.

The image in this view does not appear to be attributed differently from the entire login card. While it is likely that the model determined the image is not a signifier of tappability, it could also be, in part, due to these two being the same actual element in the source UI view hierarchy. Our use of XRAI is limited by the bounding boxes provided from the source UI view structure—large objects in UIs may cause XRAI to aggregate too much detail from pixel attributions beneath. In practice, this may not be a significant limitation, since many large UI objects inherit a single tappability attribute.

Like the TextView example, the tappable neighbors are all visually similar, with text over a light-colored background, comprising buttons and text fields. Non tappable neighbors are, similarly, text elements in different contexts: descriptions of nearby objects, instructions, or hyperlinks. Of note, the third non-tappable neighbor is also a EditText element. A probable distinguishing feature of this element is that the text is dark (not grayed), and thus the model could be confusing this element for a text description (Figure 14).

7 EXPLORATORY EVALUATION WITH PROFESSIONAL DESIGNERS

To better understand how our model and its explanation outputs can be used in design practice, we conducted an exploratory evaluation with professional UI/UX designers, and analyzed the successes and drawbacks of our approach.

7.1 Participants & Study Design

We recruited 14 participants at a large technology company. We excluded one participant's results from analysis because they did not submit any written feedback. These participants were from multiple teams and had an average 11 years (standard dev. 7.5 years) of professional UI/UX design experience. To capture a variety of scenarios, we randomly selected six UI examples (mobile app screenshots with a preselected UI element) from our dataset for review. Input UIs and predictions are shown in Figure 15, outputs from explanation

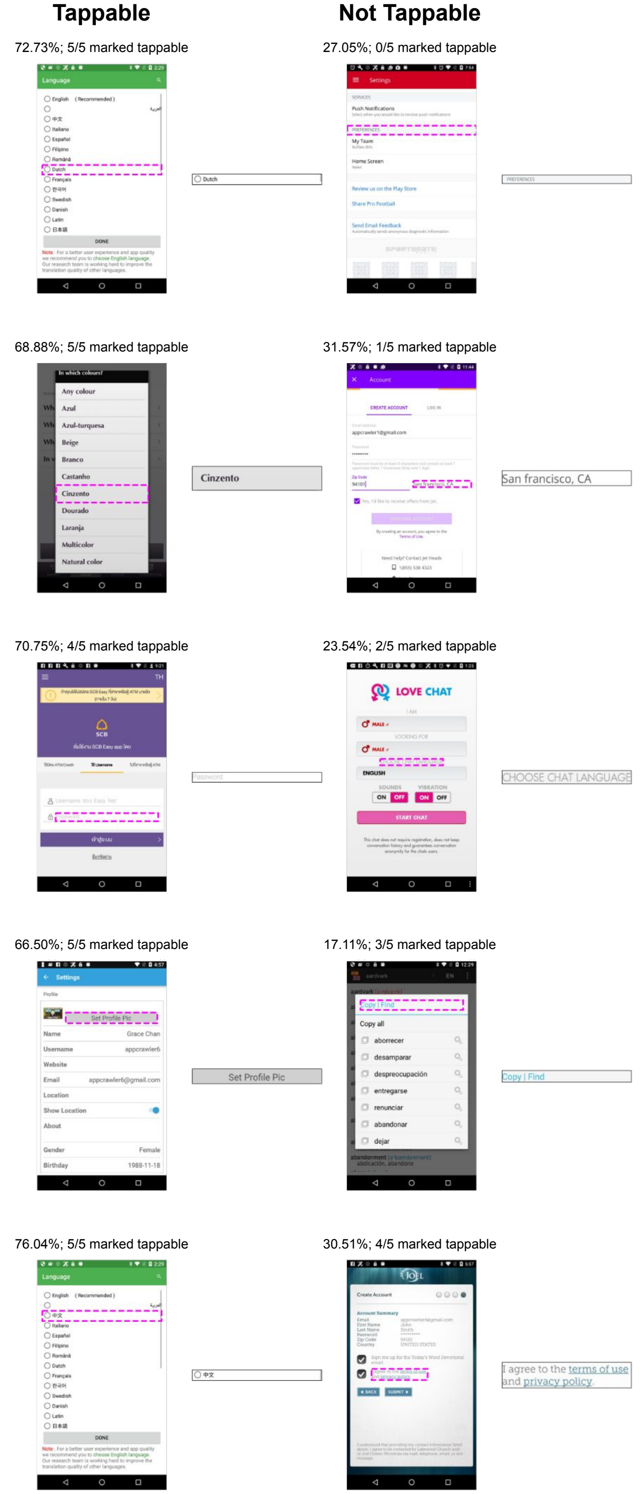

Tappable
Not Tappable
72.73%; 5/5 marked tappable
Dutch
27.05%; 0/5 marked tappable
PREFERENCES
68.88%; 5/5 marked tappable
Cinzento
31.57%; 1/5 marked tappable
San francisco, CA
70.75%; 4/5 marked tappable
Password
23.54%; 2/5 marked tappable
CHOOSE CHAT LANGUAGE
66.50%; 5/5 marked tappable
Set Profile Pic
17.11%; 3/5 marked tappable
Copy | Find
76.04%; 5/5 marked tappable
中文
30.51%; 4/5 marked tappable
I agree to the terms of use and privacy policy

Figure 12: Nearest neighbors from subsection 6.3, split by thresholded model predictions (tappable neighbors on the left). Tappable elements are similar to the input, in that most contain stacked text elements (two with a green header bar).

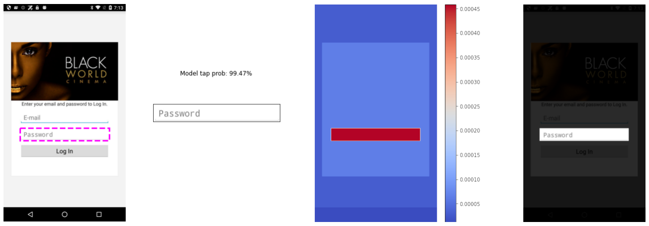


Figure 13: Example from subsection 6.4. Left: input screenshot with a `EditText` element selected, annotated with a magenta dashed rectangle. Center Left: Close-up view of the selected UI element. The model predicts it is tappable, with a 99.47% tappability probability. Center Right: The XRAI heatmap strongly attributes the `EditText` view, and does not attribute other elements on the screen. Right: the input screenshot filtered by the values of the saliency heatmap.

algorithms are shown in Appendix A. Examples counterbalanced prediction (tappable/non tappable), prediction confidence (high: > 0.85; low: < 0.15), and rater (worker) agreement (high: 5-rater agreement; low: split 2-3 in either direction). We selected the tappable/high confidence/high agreement example for use in onboarding. Tappability labels from raters associated with the examples were not shown to participants. Our study plan was reviewed by our company's legal and privacy boards, and participants were required to give informed consent before trials.

During each session, we first described our model and its explanation outputs using the onboarding example. Then, the five remaining examples were shown in a randomized order, together with the model's outputs. For each example, we asked our participants to think out loud; reflect on whether they understood or agreed with the model's outputs; and suggest how the selected element in the example could be altered to influence its perceived tappability. We explicitly informed participants that both positive and negative feedback would be useful to the design team for making improvements. A researcher took notes of verbal responses while examples were shown. After seeing all examples, participants filled out a short survey asking what they thought performed well, needed improvement, and could fit into their design practice. An entire session took approximately 45 minutes to complete. For analysis, feedback from written responses was processed in an open coding phase, and further grouped by one researcher into the related topics, which were agreed upon with the other researchers [39]. Quotes shared below are exclusively from survey responses.

7.2 Results

7.2.1 Tappability predictions can save significant time and effort compared to user studies. From survey responses, 11 participants perceived the system as accurate, and 7 remarked how the system would be valuable for evaluating designs as a time-saving alternative to running user studies: *"It's fairly accurate in predicting whether an element is tappable or not"* (P7); *"I think it's great to see a quantified results of tappability - it can reduce the time to conduct usability study."* (P6); *"UI designers could use the model to cross check and see if they match the anticipated results. If that happens, it will*

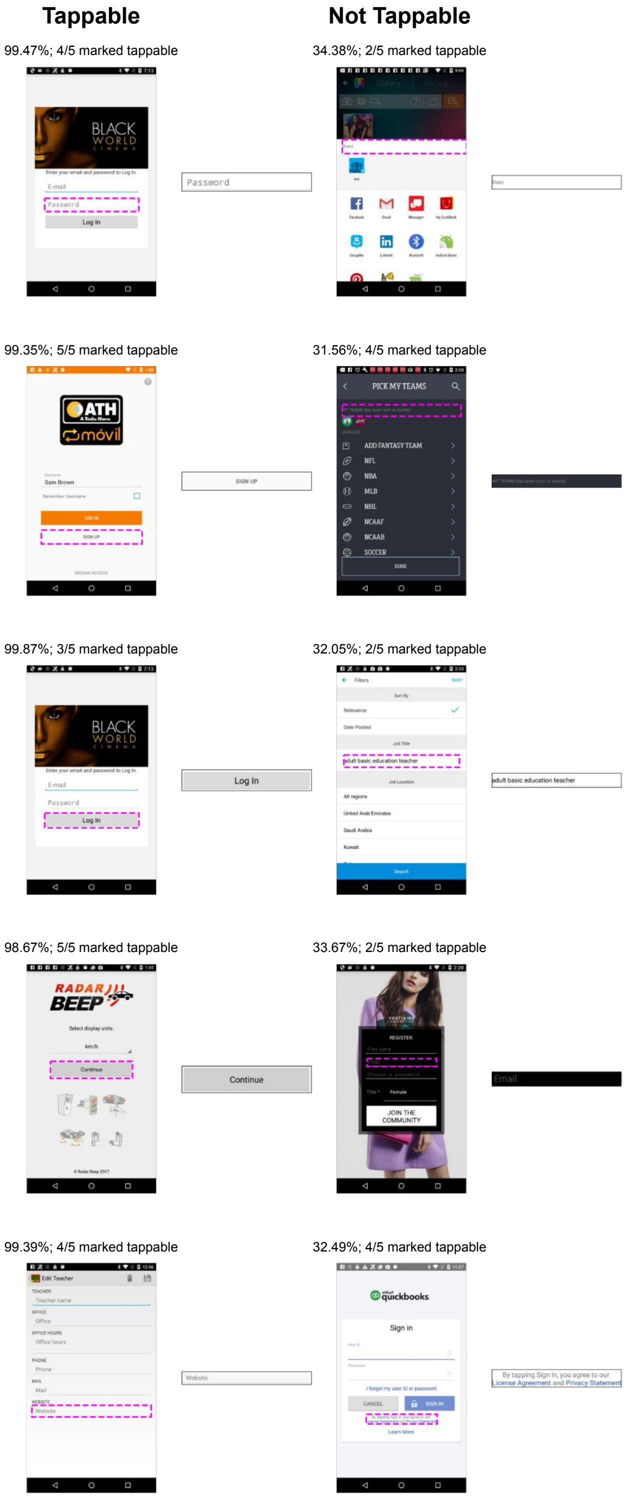

Tappable
Not Tappable
99.47%; 4/5 marked tappable
34.38%; 2/5 marked tappable
Password
99.35%; 5/5 marked tappable
31.56%; 4/5 marked tappable
SIGN UP
PICK MY TEAMS
99.87%; 3/5 marked tappable
32.05%; 2/5 marked tappable
Log In
adult basic education teacher
98.67%; 5/5 marked tappable
33.67%; 2/5 marked tappable
RADAR BEEP
Continue
Email
99.39%; 4/5 marked tappable
32.49%; 4/5 marked tappable
Website
quickbooks
By tapping Sign in, you agree to our License Agreement and Privacy Statement

Figure 14: Nearest neighbors from subsection 6.4, split by thresholded model predictions (tappable neighbors on the left). The tappable neighbors are highly visually similar, although they are not all the same input type as the input (many are buttons with a light background).

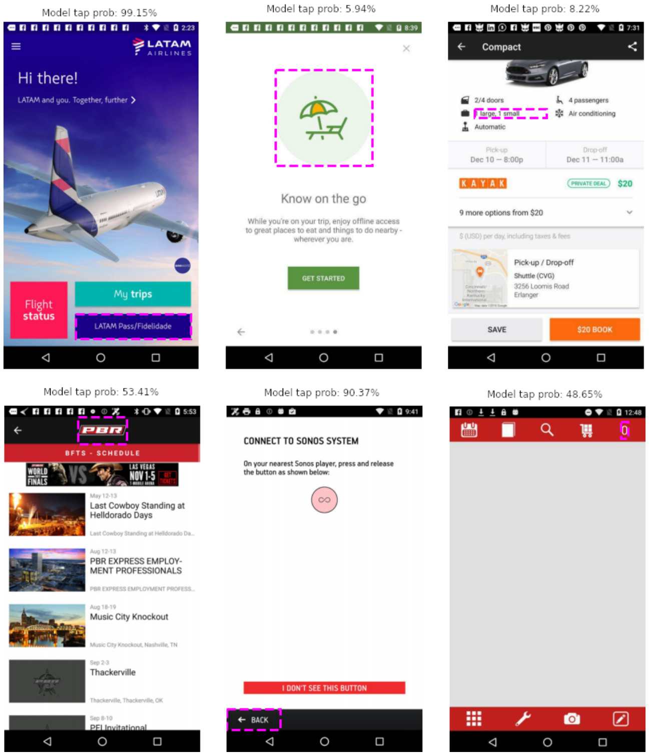


Figure 15: The six mobile UI screenshots with preselcted UI elements used in our user evaluation. Selected elements are circled with a dotted magenta line. The top left UI screenshot was used for onboarding participants.

save a lot of time running user studies." (P3). One noteworthy theme was the value of using our system for rapid evaluations at multiple stages of the design process: *"It might be useful during handoff to engineers as a final check on design quality, or assessing a built app during a usability audit."* (P4); *"I would use it in the evaluative stages of design as a gut check on what I've done."* (P9). Our system could directly enable this capability if implemented in an end-to-end application, since it only requires pixels as input and thus can operate on fully implemented UIs or visually realistic mockups.

P5 pointed out a trade-off of using our system for evaluation, reflecting on its static nature, versus the open-ended format of traditional usability studies: *"Users might have an advantage by being able to trial and error. It seems like the model gets a lot correct and points out possible design flaws, but users tend to explore openly anyway making choices still situational."* Overall, this feedback suggests strong potential uses cases in rapid, heuristic evaluations of UIs when user studies would be too time-consuming, both for the early stages of design (when prototyping alternatives) and for catching potential errors in a design as it nears production.

7.2.2 Analysis of a single screen offers limited notion of context in a UI flow. While many participants remarked on the model's generally good performance, 4 were more critical or skeptical when it came to UI elements that were sensitive to the context of other screens in a UI flow, e.g.: *"It doesn't seem as useful for navigation or text where the tappability is more contextual"* (P4); *"Considering context and looking at the whole page holistically are very important in UI design. The system tend to ignore the the context of the screen. E.g. Is it the home screen or interior page? The app logo can be tappable depending on the context"* (P13). The cases referred to in these quotes are examples with low model confidence, reflecting a potentially ambiguous perception of tappability that depends on how the input screenshot is situated in a flow of multiple UIs. One limitation of our model is that it only uses a *static* snapshot of a UI as input. In future work, *temporal* information could be used in our model's inputs to add additional context, as prior work has done for predicting user engagement with animations [42] and grounding UI action sequences [26]. In addition, this is a case where including additional input modalities (e.g., text) can provide additional cues to boost prediction confidence.

7.2.3 Contrasting similar examples provided design feedback for iteration. At least 5 participants wrote favorably of the contrasting similar examples in open-ended feedback, and remarked on their value for inspiring potential design changes: *"The initial "Model tap prob" metric is extremely useful as are the examples of similar UI elements that have both low and high [tappability] scores."* (P14); *"I might also use some of the comps [examples] to find inspirations on other ways to design a certain element"* (P9). Some participants liked the diversity of some sets of contrasting similar examples (*"the provided examples are useful to reference and compare to, even if the similar elements are not exactly the same."* (P10)), while others desired a greater degree of semantic similarity (*"Heading component compares to a CTA button in Settings page. It feels like comparing apples to oranges. I would suggest, using similar UI component proximity for similar examples"* (P1)). One direction for future work could be to allow users to filter and set thresholds for examples (e.g., by certain types or locations of UI elements), or reporting actual distances (*"Maybe for the nearest neighbors, provide some indications of how near or far the neighbor is, e.g. 90% vs 10%"* (P12)).

Overall, this feedback suggests that the contrasting similar examples, curated based on a specified UI element, have the potential to provide useful inspiration for designers. This may help "close the loop" beyond tappability prediction scores alone.

7.2.4 Participants desire more explicit explanations beyond the heatmap. While 2 participants remarked that the heatmap was useful, 4 participants noted the heatmap was confusing to use, or needed better instructions, e.g.: *"Heat map. Confused me and would need some guidance on how to process the info."* (P14); *"Saliency heatmap definitely needs some mental shift to understand."* (P6). While this could potentially be mitigated with improved onboarding or more experience [6, 43], the "black-box" nature of our model means, while it may be effective at predicting users' perceptions of tappability, the mechanisms which enable those predictions may not reflect the same reasoning as users [28]. The mismatch in mental models could explain this result: *"I found the heat maps and nearest neighbors less helpful because they didn't resemble my own mental model / instincts for evaluating the usability of these mockups"* (P4).

Some participants expressed a desire for deeper explanations of *why* certain elements in the heatmap contribute to a tappability prediction more than others, which could help improve its usability: *"On the heat map, add some explanation about why the other elements* *might or might not impact the probability score of an elemement"* (P14); and others wished the system could output design suggestions directly: *"It'd be amazing if the system can provide recommendation like boost the color contrast"* (P6). One promising direction for future work could draw from techniques in ML debugging research, by identifying common "heuristics" from patterns in the model's and XRAI heatmap's outputs and raising messages with concrete design suggestions (e.g., increasing contrast or changing colors) [19, 33].

8 THEMES IN MODEL BEHAVIOR: DISCUSSION AND LIMITATIONS

In this section, we describe patterns observed in our selected examples and discuss implications for the use of our model and explanation mechanisms.

8.1 Persistent Signals and Signifiers

Text as a feature indicating tappability. In examples in section 6 as well as examples in our user evaluation, bounding boxes surrounding text elements were highly attributed by XRAI. This does not necessarily mean the text itself is perceived as tappable, but rather that the existence of text serves as a signifier of tappability to nearby elements (see subsection 8.2). This is one potential drawback to our pixel-based model compared to multimodal models that use text as input to gain a deeper understanding of an element's context (e.g., a "submit" button or "click here to unsubscribe" text).

Icons next to text generally indicate tappable regions. To our model, small icons or graphics appearing next to text strongly signify tappability. This is demonstrated in Figure 12, where most tappable text elements are near radio buttons, icons, and other graphics. Using icons to signify the tappability of adjacent text elements is a well-known practice [29]. However, our model does not always produce reliable tappability predictions of checkbox elements (E.g., Figure 12, bottom right). This is likely due to ambiguity in the labeling task. Since the checkbox, accompanying description, and parent element containing both are each distinct UI elements with separate bounding boxes, any one of these elements within a given screenshot could be selected for labeling. Crowdworkers may have different perceptions of the tappability of the different elements, and this uncertainty is reflected in our model's prediction scores.

Image views in apps are not consistent predictors. Because the content of `ImageView` elements can be highly varied (e.g., containing icons, logos, thumbnails, previews, ...), they can sometimes confound our purely vision-based model (see Figure 10). While our model likely also uses the location and context of the image element, the content of the image can overwhelm predictions, possibly due to the texture sensitivity of CNNs [15]. One way to potentially mitigate this effect would be to replace images with placeholders, similar to wireframes [9, 11].

8.2 Challenges in Interpreting XRAI Attributions

XRAI attributions highlight influential regions; highly influential regions are not necessarily tappable themselves. As reflected in the results of our user evaluation, the XRAI heatmaps require practice to take full advantage of, and could benefit from the addition of heuristic-based explanations. A critical note for our use of XRAI is that the heatmap it produces is not a tappability heatmap, but a heatmap showing how regions in the UI screen influence the tappability prediction *for a particular element.* For example, if highly attributed text near a button was removed, that button would likely no longer be classified as tappable. As such, saliency methods like XRAI are often useful in practice for diagnosing the features that influence predictions, and the sensitivity of that prediction to contextual factors.

Summarizing attributions with regions may leave out important details. In contrast to XRAI, which uses regions, pixel-based saliency methods like Integrated Gradients [36] highlight inputs at a finer scale. While this may be useful for debugging features of small UI elements, pixel-based methods are known to be difficult to interpret by humans compared to region-based methods, and can be susceptible to errors [1, 21].

XRAI attribution values cannot be compared between examples. Like other gradient-based saliency methods, the raw values of XRAI attributions are specific to input examples [1]. Some other algorithms, such as DeepSHAP [30], sum to the probabilities of predictions, and may be compared between examples. These other methods could also enable new interactions, such as aggregated analyses, a promising direction for future work.

8.3 Browsing Nearest Neighbor Examples

Nearest Neighbors capture many dimensions of similarity. Across all of our examples, nearest neighbors appear to capture dimensions beyond visual similarity alone. In particular, bounding box locations, sizes, and aspect ratios are generally similar among neighbors. This indicates that our model has not only learned to use the appearance of an element to predict its tappability, but also contextual information such as its location, shape, and proximity to other elements. As participants in our user evaluation noted, adding interactivity to the nearest neighbor examples, such as the ability to filter and sort by component types and application properties, could help narrow down these contextual cues to provide more relevant feedback for iterating UI designs.

Splitting neighbors by binary tappability predictions discards some information. While using a discrete boundary can provide useful contrasting examples, the average tappability prediction probabilities and distances between splits can contain subtle yet important information about the landscape of UI design patterns related to the input. For example, the health app's non-tappable neighbors included many seemingly unrelated graphics. This may be because of skewed distances, i.e., most nearby examples are tappable, and the closest non-tappable neighors are significantly further away, and, thus, less similar. In other cases, the sets of neighbors may have skewed average probabilities (e.g., near 99% for tappable, and near 49% for non tappable). This is an additional, strong, indicator that similar UIs are generally either perceived as tappable or uncertain, rather than non-tappable. The contrary is also true: the neighbors of confidently *non-tappable* examples often score near 51% for tappable, and 0% for non-tappable. In future work, these details could be made explicit in more continuous, interactive visualizations, to help designers explore related UI designs.

Encouraging exploration of neighboring examples. While using UIs with similar designs but different effects on tappability perception are useful for contrasting explanations, designers often value seeing diverse examples of designs for inspiration [9, 38]. Future iterations of this work could sample more distant UI examples, filtered by UI element types, or even learned concepts, i.e., with concept vectors [22]. In addition, since nearest neighbor examples are split by model predictions, the similar examples do not have to be limited to our source dataset. In other words, our model can be used to retrieve nearest neighbors or similar examples from other datasets.

8.4 Additional Limitations

Concept drift. While UI design styles and trends change over time, our model is trained on a "static" snapshot of application UIs, and may give less reliable predictions over time. This phenomenon is known as concept drift [40], and may be mitigated by augmenting the dataset with new examples over time. Furthermore, since our dataset comprises only Android applications, our model may require fine-tuning to generalize well to UI screenshots from other platforms.

Perceived tappability predictions contain multiple signals. As we have found by analyzing the distribution of label agreement in our dataset, the tappability of many UI elements in the wild appear ambiguous to users. While this uncertainty explicitly limits the possible accuracy of our model, it also means that predictions near the decision boundary suggest user confusion. This signal, along with other usability metrics (e.g., engagement [42] or cognitive load) may be useful outputs from future models.

9 CONCLUSION

We presented a novel, automated system for predicting the human perceived tappability of mobile UI elements and explaining model predictions to users. Our work significantly advanced the art by developing a purely vision-based deep neural network, which only relies on pixels and does not require a UI to be fully specified; and by enabling mechanisms for explaining design insights to the user with contextual and instance-level interpretations of model predictions. We also create a new tappability dataset where each element is labeled by multiple crowdworkers for reliable tappability estimation. We provided an in-depth discussion of our model behavior and explanation mechanisms through extensive analysis of examples and collected feedback from experienced professional UI/UX designers in how they would use and improve our system. Together, our work not only advances tappability modeling research but also demonstrates how deep learning approaches can be used for automatic UI usability analysis.

REFERENCES

[1] Julius Adebayo, Justin Gilmer, Michael Muelly, Ian Goodfellow, Moritz Hardt, and Been Kim. 2018. Sanity Checks for Saliency Maps. In *Advances in Neural Information Processing Systems*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett (Eds.), Vol. 31. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2018/file/294a8ed24b1ad22ec2e7efea049b8737-Paper.pdf

[2] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Agüera y Arcas. 2021. UIBert: Learning Generic Multimodal Representations for UI Understanding. *CoRR* abs/2107.13731 (2021). arXiv:2107.13731 https://arxiv.org/abs/2107.13731

[3] Rachel Bellamy, Bonnie John, and Sandra Kogan. 2011. Deploying CogTool: integrating quantitative usability assessment into real-world software development. In *2011 33rd International Conference on Software Engineering (ICSE)*. 691–700. https://doi.org/10.1145/1985793.1985890

[4] Zoya Bylinskii, Nam Wook Kim, Peter O'Donovan, Sami Alsheikh, Spandan Madan, Hanspeter Pfister, Fredo Durand, Bryan Russell, and Aaron Hertzmann. 2017. Learning Visual Importance for Graphic Designs and Data Visualizations. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, Québec City QC Canada, 57–69. https://doi.org/10.1145/3126594.3126653

[5] Carrie J. Cai, Jonas Jongejan, and Jess Holbrook. 2019. The Effects of Example-Based Explanations in a Machine Learning Interface. In *Proceedings of the 24th International Conference on Intelligent User Interfaces* (Marina del Ray, California) *(IUI '19)*. Association for Computing Machinery, New York, NY, USA, 258–262. https://doi.org/10.1145/3301275.3302289

[6] Carrie J Cai, Samantha Winter, David Steiner, Lauren Wilcox, and Michael Terry. 2021. Onboarding Materials as Cross-functional Boundary Objects for Developing AI Assistants. In *Extended Abstracts of the 2021 CHI Conference on Human Factors in Computing Systems*. ACM, Yokohama Japan, 1–7. https://doi.org/10.1145/3411763.3443435

[7] Chunyang Chen, Sidong Feng, Zhenchang Xing, Linda Liu, Shengdong Zhao, and Jinshui Wang. 2019. Gallery D.C.: Design Search and Knowledge Discovery through Auto-created GUI Component Gallery. *Proceedings of the ACM on Human-Computer Interaction* 3, CSCW (Nov. 2019), 1–22. https://doi.org/10.1145/3359282

[8] Chun-Fu (Richard) Chen, Marco Pistoia, Conglei Shi, Paolo Girolami, Joseph W. Ligman, and Yong Wang. 2017. UI X-Ray: Interactive Mobile UI Testing Based on Computer Vision. In *Proceedings of the 22nd International Conference on Intelligent User Interfaces*. ACM, Limassol Cyprus, 245–255. https://doi.org/10.1145/3025171.3025190

[9] Jieshan Chen, Chunyang Chen, Zhenchang Xing, Xin Xia, Liming Zhu, John Grundy, and Jinshui Wang. 2020. Wireframe-based UI Design Search through Image Autoencoder. *ACM Transactions on Software Engineering and Methodology* 29, 3 (July 2020), 1–31. https://doi.org/10.1145/3391613

[10] W. L. Cheng. 2016. Learning through the variation theory: A case study. *The International Journal of Teaching and Learning in Higher Education* 28 (2016), 283–292.

[11] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. 2017. Rico: A Mobile App Dataset for Building Data-Driven Design Applications. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology*. ACM, Québec City QC Canada, 845–854. https://doi.org/10.1145/3126594.3126651

[12] Biplab Deka, Zifeng Huang, Chad Franzen, Jeffrey Nichols, Yang Li, and Ranjitha Kumar. 2017. ZIPT: Zero-Integration Performance Testing of Mobile App Designs. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software and Technology* (Québec City, QC, Canada) *(UIST '17)*. Association for Computing Machinery, New York, NY, USA, 727–736. https://doi.org/10.1145/3126594.3126647

[13] Biplab Deka, Zifeng Huang, and Ranjitha Kumar. 2016. ERICA: Interaction Mining Mobile Apps. In *Proceedings of the 29th Annual Symposium on User Interface Software and Technology* (Tokyo, Japan) *(UIST '16)*. Association for Computing Machinery, New York, NY, USA, 767–776. https://doi.org/10.1145/2984511.2984581

[14] Camilo Fosco, Vincent Casser, Amish Kumar Bedi, Peter O'Donovan, Aaron Hertzmann, and Zoya Bylinskii. 2020. Predicting Visual Importance Across Graphic Design Types. In *Proceedings of the 33rd Annual ACM Symposium on User Interface Software and Technology* (Virtual Event, USA) *(UIST '20)*. Association for Computing Machinery, New York, NY, USA, 249–260. https://doi.org/10.1145/3379337.3415825

[15] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A. Wichmann, and Wieland Brendel. 2018. ImageNet-trained CNNs are biased towards texture; increasing shape bias improves accuracy and robustness. *CoRR* abs/1811.12231 (2018). arXiv:1811.12231 http://arxiv.org/abs/1811.12231

[16] Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. 2019. Towards Automatic Concept-based Explanations. In *Advances in Neural Information Processing Systems*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.), Vol. 32. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2019/file/77d2afcb31f6493e350fca61764efb9a-Paper.pdf

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep Residual Learning for Image Recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 770–778. https://doi.org/10.1109/CVPR.2016.90

[18] Zecheng He, Srinivas Sunkara, Xiaoxue Zang, Ying Xu, Lijuan Liu, Nevan Wichers, Gabriel Schubiner, Ruby B. Lee, and Jindong Chen. 2020. ActionBert: Leveraging User Actions for Semantic Understanding of User Interfaces. *CoRR* abs/2012.12350 (2020). arXiv:2012.12350 https://arxiv.org/abs/2012.12350

[19] Andrew Head, Codanda Appachu, Marti A. Hearst, and Björn Hartmann. 2015. Tutorons: Generating context-relevant, on-demand explanations and demonstrations of online code. In *2015 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. 3–12. https://doi.org/10.1109/VLHCC.2015.7356972

[20] Forrest Huang, John F. Canny, and Jeffrey Nichols. 2019. *Swire: Sketch-Based User Interface Retrieval.* Association for Computing Machinery, New York, NY, USA,

1–10. https://doi.org/10.1145/3290605.3300334

[21] Andrei Kapishnikov, Tolga Bolukbasi, Fernanda Viegas, and Michael Terry. 2019. XRAI: Better Attributions Through Regions. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 4947–4956. https://doi.org/10.1109/ICCV.2019.00505

[22] Been Kim, Martin Wattenberg, Justin Gilmer, Carrie J. Cai, James Wexler, Fernanda B. Viégas, and Rory Sayres. 2018. Interpretability Beyond Feature Attribution: Quantitative Testing with Concept Activation Vectors (TCAV). In *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018 (Proceedings of Machine Learning Research, Vol. 80)*, Jennifer G. Dy and Andreas Krause (Eds.). PMLR, 2673–2682. http://proceedings.mlr.press/v80/kim18d.html

[23] Chunggi Lee, Sanghoon Kim, Dongyun Han, Hongjun Yang, Young-Woo Park, Bum Chul Kwon, and Sungahn Ko. 2020. *GUIComp: A GUI Design Assistant with Real-Time, Multi-Faceted Feedback.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376327

[24] Luis A. Leiva, Asutosh Hota, and Antti Oulasvirta. 2020. Enrico: A Dataset for Topic Modeling of Mobile UI Designs. In *22nd International Conference on Human-Computer Interaction with Mobile Devices and Services* (Oldenburg, Germany) *(MobileHCI '20)*. Association for Computing Machinery, New York, NY, USA, Article 9, 4 pages. https://doi.org/10.1145/3406324.3410710

[25] Toby Jia-Jun Li, Lindsay Popowski, Tom Mitchell, and Brad A Myers. 2021. *Screen2Vec: Semantic Embedding of GUI Screens and GUI Components.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445049

[26] Yang Li, Jiacong He, Xin Zhou, Yuan Zhang, and Jason Baldridge. 2020. Mapping Natural Language Instructions to Mobile UI Action Sequences. In *Annual Conference of the Association for Computational Linguistics (ACL 2020)*. https://www.aclweb.org/anthology/2020.acl-main.729.pdf

[27] Yang Li, Ranjitha Kumar, Walter S. Lasecki, and Otmar Hilliges. 2020. Artificial Intelligence for HCI: A Modern Approach. In *Extended Abstracts of the 2020 CHI Conference on Human Factors in Computing Systems* (Honolulu, HI, USA) *(CHI EA '20)*. Association for Computing Machinery, New York, NY, USA, 1–8. https://doi.org/10.1145/3334480.3375147

[28] Zachary C. Lipton. 2017. The Mythos of Model Interpretability. arXiv:1606.03490 [cs.LG]

[29] Hoa Loranger. 2015. *Beyond Blue Links: Making Clickable Elements Recognizable.* Nielsen Norman Group. https://www.nngroup.com/articles/clickable-elements/

[30] Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 4768–4777.

[31] Nicolas Papernot and Patrick D. McDaniel. 2018. Deep k-Nearest Neighbors: Towards Confident, Interpretable and Robust Deep Learning. *CoRR* abs/1803.04765 (2018). arXiv:1803.04765 http://arxiv.org/abs/1803.04765

[32] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) *(KDD '16)*. Association for Computing Machinery, New York, NY, USA, 1135–1144. https://doi.org/10.1145/2939672.2939778

[33] Eldon Schoop, Forrest Huang, and Bjoern Hartmann. 2021. *UMLAUT: Debugging Deep Learning Programs Using Program Structure and Model Behavior.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445538

[34] Jessica Schrouff, Sebastien Baur, Shaobo Hou, Diana Mincu, Eric Loreaux, Ralph Blanes, James Wexler, Alan Karthikesalingam, and Been Kim. 2021. Best of both worlds: local and global explanations with human-understandable concepts. *CoRR* abs/2106.08641 (2021). arXiv:2106.08641 https://arxiv.org/abs/2106.08641

[35] Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. 2017. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*. 618–626. https://doi.org/10.1109/ICCV.2017.74

[36] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic Attribution for Deep Networks. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70* (Sydney, NSW, Australia) *(ICML'17)*. JMLR.org, 3319–3328.

[37] Amanda Swearngin and Yang Li. 2019. Modeling Mobile Interface Tappability Using Crowdsourcing and Deep Learning. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*. ACM, Glasgow Scotland Uk, 1–11. https://doi.org/10.1145/3290605.3300305

[38] Amanda Swearngin, Chenglong Wang, Alannah Oleson, James Fogarty, and Amy J. Ko. 2020. *Scout: Rapid Exploration of Interface Layout Alternatives through High-Level Design Constraints.* Association for Computing Machinery, New York, NY, USA, 1–13. https://doi.org/10.1145/3313831.3376593

[39] Robert Stuart Weiss. 1994. *Learning from strangers: The art and method of qualitative interview studies.* Free Press. ix, 246 pages.

[40] Gerhard Widmer and Miroslav Kubat. 1996. Learning in the Presence of Concept Drift and Hidden Contexts. *Mach. Learn.* 23, 1 (April 1996), 69–101. https://doi.org/10.1023/A:1018046501280

[41] Jason Wu, Xiaoyi Zhang, Jeffrey Nichols, and Jeffrey P. Bigham. 2021. Screen Parsing: Towards Reverse Engineering of UI Models from Screenshots. *CoRR* abs/2109.08763 (2021). arXiv:2109.08763 https://arxiv.org/abs/2109.08763

[42] Ziming Wu, Yulun Jiang, Yiding Liu, and Xiaojuan Ma. 2020. Predicting and Diagnosing User Engagement with Mobile UI Animation via a Data-Driven Approach. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*. ACM, Honolulu HI USA, 1–13. https://doi.org/10.1145/3313831.3376324

[43] Qian Yang, Alex Scuito, John Zimmerman, Jodi Forlizzi, and Aaron Steinfeld. 2018. Investigating How Experienced UX Designers Effectively Work with Machine Learning. In *Proceedings of the 2018 Designing Interactive Systems Conference* (Hong Kong, China) *(DIS '18)*. Association for Computing Machinery, New York, NY, USA, 585–596. https://doi.org/10.1145/3196709.3196730

[44] Xiaoxue Zang, Ying Xu, and Jindong Chen. 2021. *Multimodal Icon Annotation For Mobile Applications.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3447526.3472064

[45] Xiaoyi Zhang, Lilian de Greef, Amanda Swearngin, Samuel White, Kyle Murray, Lisa Yu, Qi Shan, Jeffrey Nichols, Jason Wu, Chris Fleizach, Aaron Everitt, and Jeffrey P Bigham. 2021. *Screen Recognition: Creating Accessibility Metadata for Mobile Applications from Pixels.* Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3411764.3445186

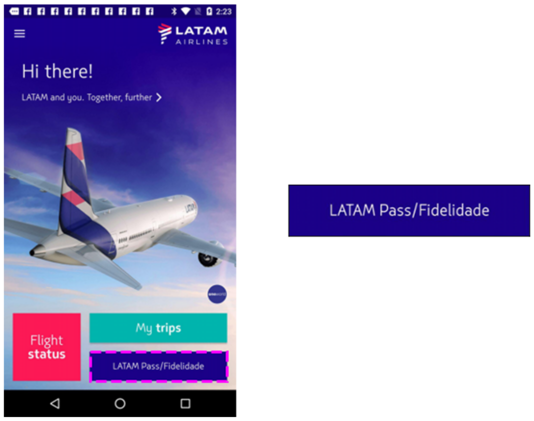
A USER STUDY FIGURES

Following are the UIs and outputs of our explanation algorithms shown to participants in the user evaluation in section 7.

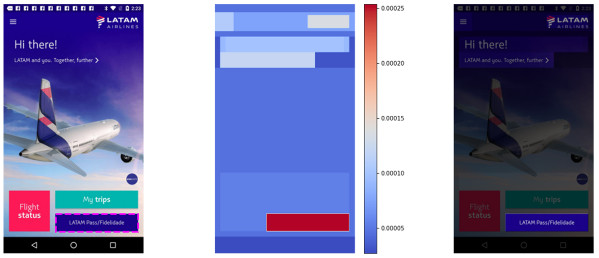
A.1 Example 1: Onboarding Example

Input Image and Region

Model tap prob: 99.15%

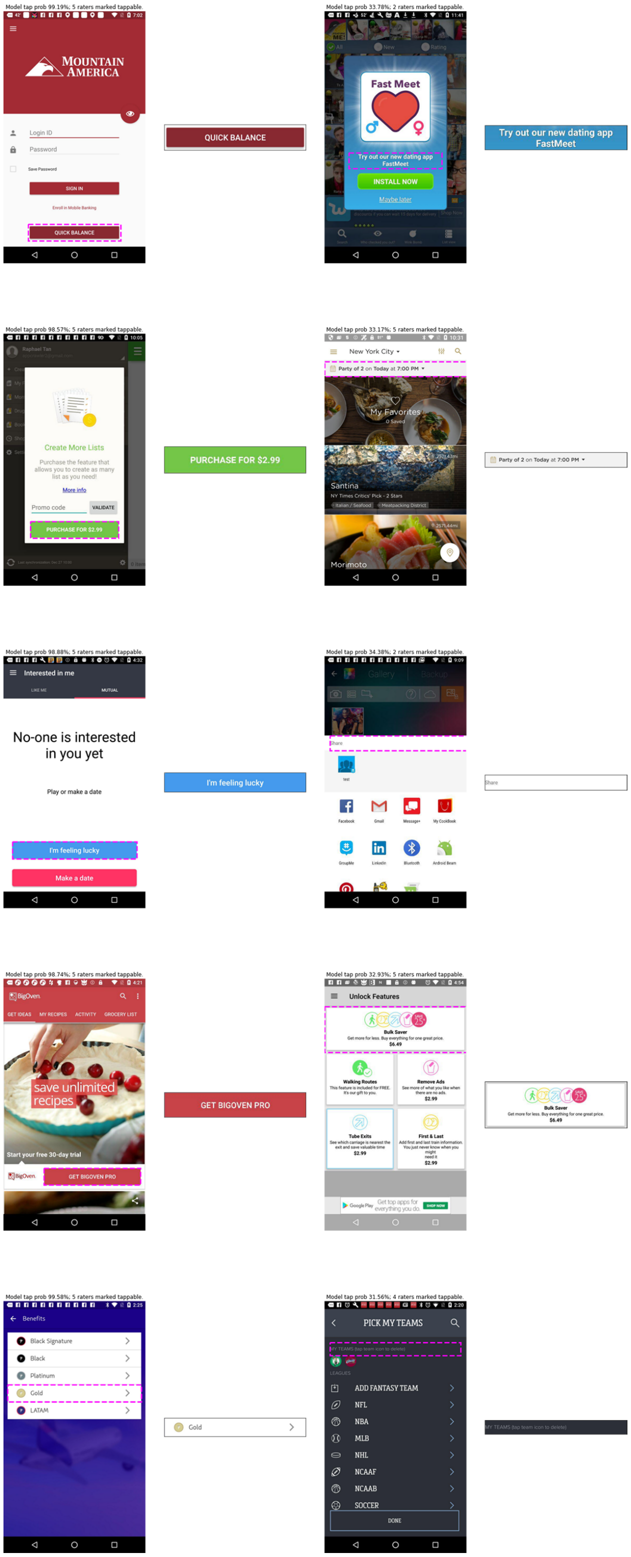


Saliency Heatmap

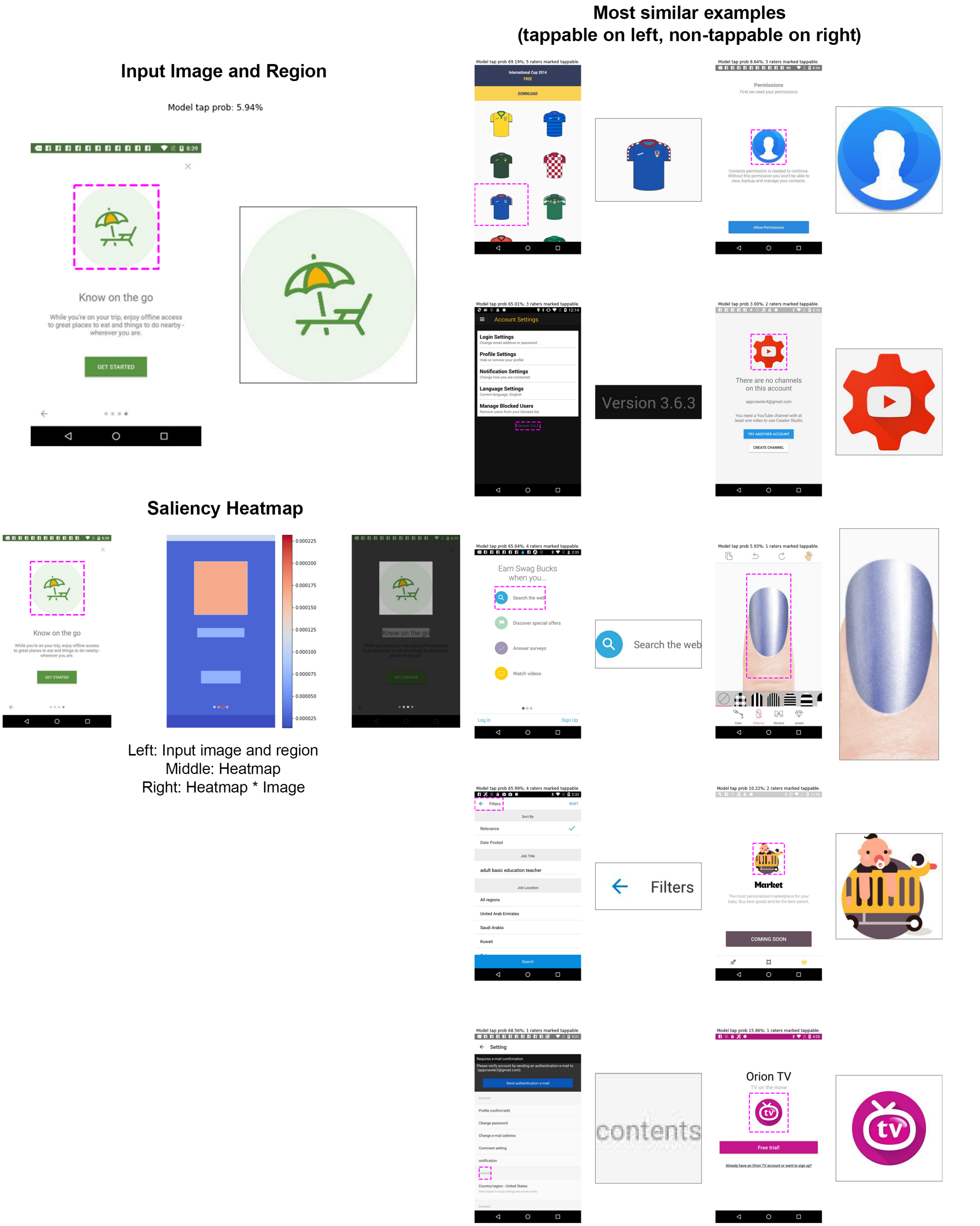


Left: Input image and region
Middle: Heatmap
Right: Heatmap * Image

Most similar examples (tappable on left, non-tappable on right)

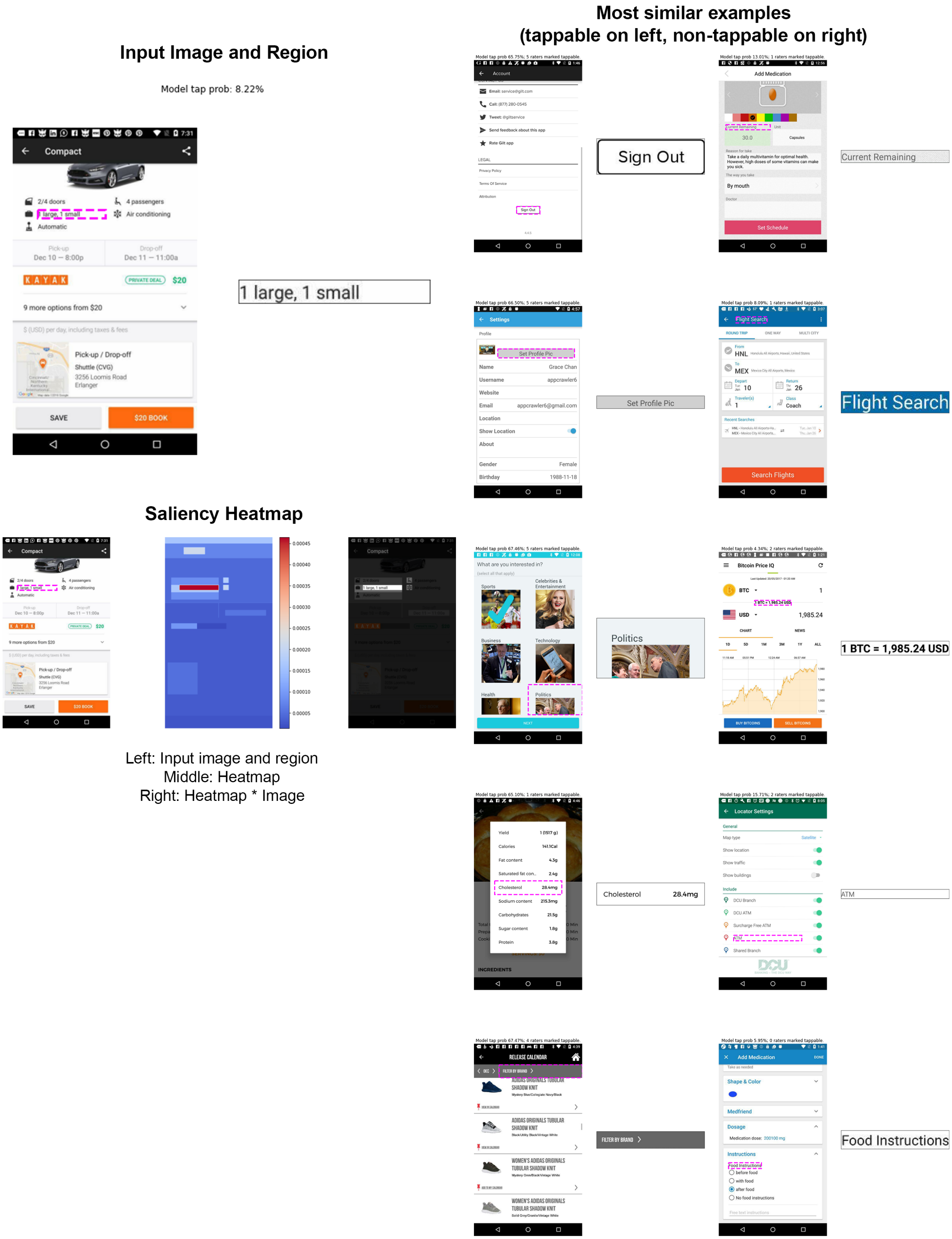

Model tap prob 99.19%; 5 raters marked tappable.
QUICK BALANCE
Model tap prob 33.78%; 2 raters marked tappable.
Try out our new dating app FastMeet
Model tap prob 98.57%; 5 raters marked tappable.
PURCHASE FOR $2.99
Model tap prob 33.17%; 5 raters marked tappable.
Party of 2 on Today at 7:00 PM
Model tap prob 98.88%; 5 raters marked tappable.
I'm feeling lucky
Model tap prob 34.38%; 2 raters marked tappable.
Share
Model tap prob 98.74%; 5 raters marked tappable.
GET BIGOVEN PRO
Model tap prob 32.93%; 5 raters marked tappable.
Bulk Saver
Model tap prob 99.58%; 5 raters marked tappable.
Gold
Model tap prob 31.56%; 4 raters marked tappable.
MY TEAMS (tap team icon to delete)

A.2 Example 2

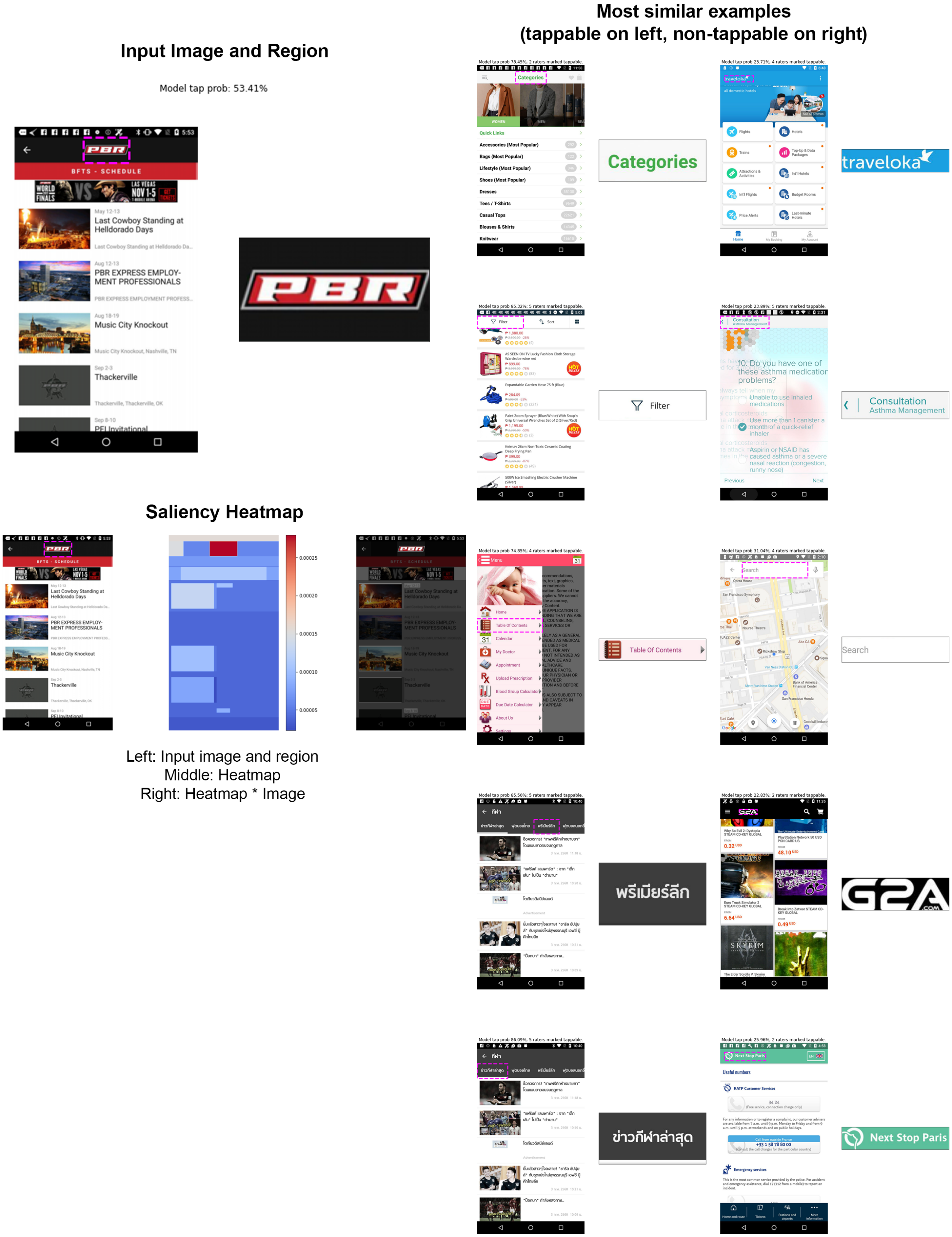

Input Image and Region
Model tap prob: 5.94%
Know on the go
While you're on your trip, enjoy offline access to great places to eat and things to do nearby - wherever you are.
GET STARTED
Saliency Heatmap
0.000225
0.000200
0.000175
0.000150
0.000125
0.000100
0.000075
0.000050
0.000025
Most similar examples
(tappable on left, non-tappable on right)
Model tap prob 69.19%; 5 raters marked tappable.
International Cup 2014
FREE
DOWNLOAD
Model tap prob 8.64%; 3 raters marked tappable.
Permissions
First we need your permissions
Contacts permission is needed to continue. Without this permission you won't be able to view, backup and manage your contacts.
Allow Permissions
Model tap prob 65.01%; 3 raters marked tappable.
Account Settings
Login Settings
Change email address or password
Profile Settings
Hide or remove your profile
Notification Settings
Change how you are contacted
Language Settings
Current language: English
Manage Blocked Users
Remove users from your blocked list
Version 3.6.3
Model tap prob 3.00%; 2 raters marked tappable.
There are no channels on this account
appcrawler4@gmail.com
You need a YouTube channel with at least one video to use Creator Studio
TRY ANOTHER ACCOUNT
CREATE CHANNEL
Model tap prob 65.04%; 4 raters marked tappable.
Earn Swag Bucks when you...
Search the web
Discover special offers
Answer surveys
Watch videos
Log in
Sign Up
Search the web
Model tap prob 5.93%; 1 raters marked tappable.
Color
Patterns
Stickers
Jewels
Model tap prob 65.99%; 4 raters marked tappable.
Filters
RESET
Sort By
Relevance
Date Posted
Job Title
adult basic education teacher
Job Location
All regions
United Arab Emirates
Saudi Arabia
Kuwait
Search
Filters
Model tap prob 10.22%; 2 raters marked tappable.
Market
The most personalized marketplace for your baby. Buy best goods and be the best parent.
COMING SOON
Model tap prob 68.56%; 1 raters marked tappable.
Setting
Requires e-mail confirmation
Please verify account by sending an authentication e-mail to (appcrawler2@gmail.com).
Send authentication e-mail
Account
Profile confirm/edit
Change password
Change e-mail address
Comment setting
notification
Country/region - United States
contents
Model tap prob 15.86%; 1 raters marked tappable.
Orion TV
TV on the move
Free trial!
Already have an Orion TV account or want to sign up?

Left: Input image and region
Middle: Heatmap
Right: Heatmap * Image

A.3 Example 3


Input Image and Region
Model tap prob: 8.22%
1 large, 1 small
Saliency Heatmap
0.00045
0.00040
0.00035
0.00030
0.00025
0.00020
0.00015
0.00010
0.00005
Left: Input image and region
Middle: Heatmap
Right: Heatmap * Image
Most similar examples
(tappable on left, non-tappable on right)
Model tap prob 65.75%; 5 raters marked tappable.
Sign Out
Model tap prob 13.01%; 1 raters marked tappable.
Current Remaining
Model tap prob 66.50%; 5 raters marked tappable.
Set Profile Pic
Model tap prob 8.09%; 1 raters marked tappable.
Flight Search
Model tap prob 67.46%; 5 raters marked tappable.
Politics
Model tap prob 4.34%; 2 raters marked tappable.
1 BTC = 1,985.24 USD
Model tap prob 65.10%; 1 raters marked tappable.
Cholesterol 28.4mg
Model tap prob 15.71%; 2 raters marked tappable.
ATM
Model tap prob 67.47%; 4 raters marked tappable.
FILTER BY BRAND
Model tap prob 5.95%; 0 raters marked tappable.
Food Instructions

A.4 Example 4



Input Image and Region

Model tap prob: 53.41%

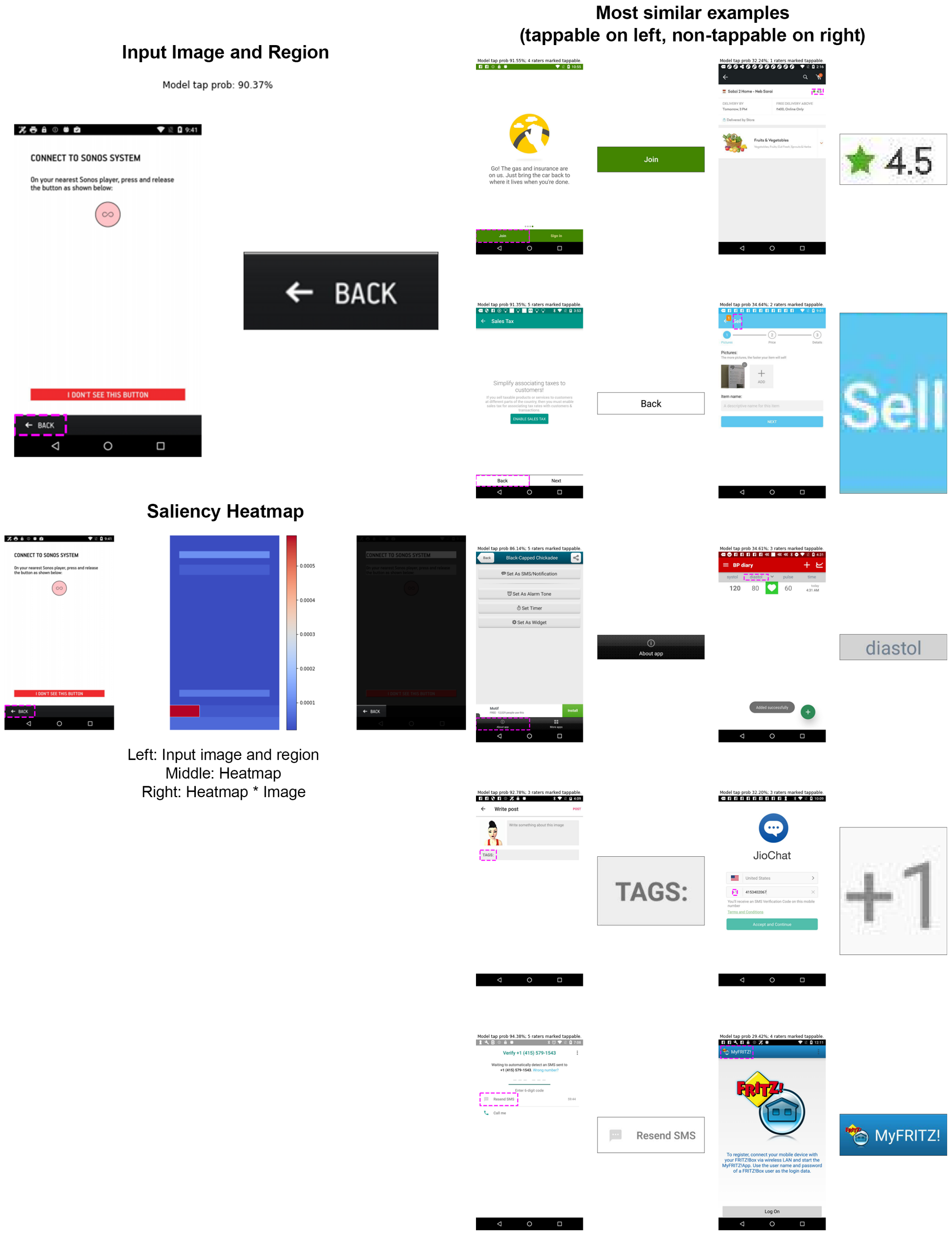
Saliency Heatmap

Left: Input image and region
Middle: Heatmap
Right: Heatmap * Image

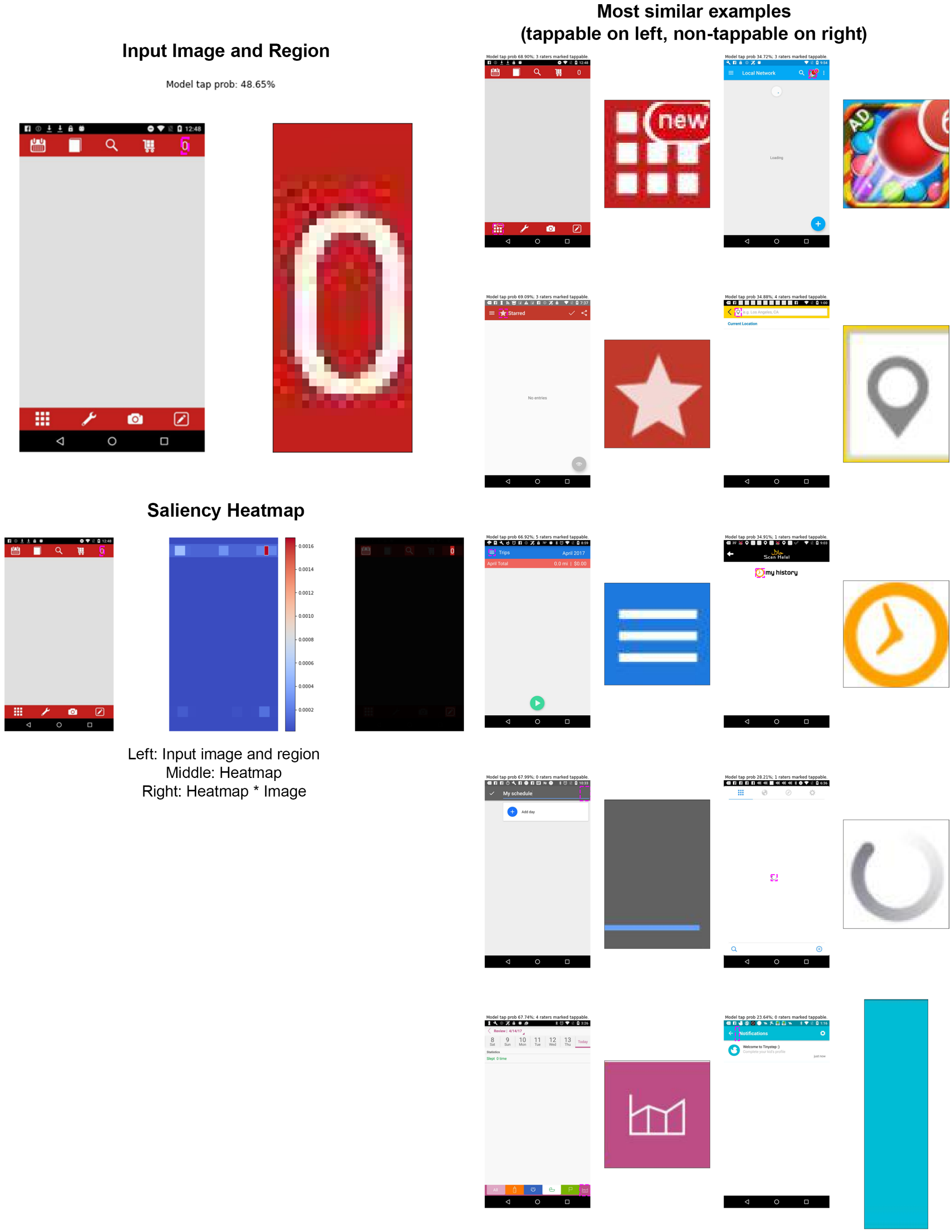
Most similar examples
(tappable on left, non-tappable on right)

PBR
BFTS - SCHEDULE
Last Cowboy Standing at Helldorado Days
PBR EXPRESS EMPLOYMENT PROFESSIONALS
Music City Knockout
Thackerville
PFI Invitational
0.00025
0.00020
0.00015
0.00010
0.00005
Model tap prob 78.45%; 2 raters marked tappable.
Categories
Model tap prob 23.71%; 4 raters marked tappable.
traveloka
Model tap prob 85.32%; 5 raters marked tappable.
Filter
Model tap prob 23.89%; 5 raters marked tappable.
Consultation Asthma Management
Model tap prob 74.85%; 4 raters marked tappable.
Table Of Contents
Model tap prob 31.04%; 4 raters marked tappable.
Search
Model tap prob 85.50%; 5 raters marked tappable.
พรีเมียร์ลีก
Model tap prob 22.83%; 2 raters marked tappable.
G2A
Model tap prob 86.09%; 5 raters marked tappable.
ข่าวกีฬาล่าสุด
Model tap prob 25.96%; 2 raters marked tappable.
Next Stop Paris

A.5 Example 5


Input Image and Region
Model tap prob: 90.37%
CONNECT TO SONOS SYSTEM
On your nearest Sonos player, press and release the button as shown below:
I DON'T SEE THIS BUTTON
BACK
Saliency Heatmap
0.0005
0.0004
0.0003
0.0002
0.0001
Left: Input image and region
Middle: Heatmap
Right: Heatmap * Image
Most similar examples
(tappable on left, non-tappable on right)
Model tap prob 91.55%; 4 raters marked tappable.
Go! The gas and insurance are on us. Just bring the car back to where it lives when you're done.
Join
Sign in
Model tap prob 32.24%; 1 raters marked tappable.
Sabzi 2 Home - Neb Sarai
4.5
DELIVERY BY
Tomorrow, 3 PM
FREE DELIVERY ABOVE
₹400, Online Only
Delivered by Store
Fruits & Vegetables
Model tap prob 91.35%; 5 raters marked tappable.
Sales Tax
Simplify associating taxes to customers!
ENABLE SALES TAX
Back
Next
Model tap prob 34.64%; 2 raters marked tappable.
Sell
Pictures
Price
Details
Pictures:
The more pictures, the faster your item will sell!
ADD
Item name:
NEXT
Model tap prob 86.14%; 5 raters marked tappable.
Back
Black-Capped Chickadee
Set As SMS/Notification
Set As Alarm Tone
Set Timer
Set As Widget
Motif
Install
About app
More apps
Model tap prob 34.61%; 3 raters marked tappable.
BP diary
systol
diastol
pulse
time
120
80
60
today 4:31 AM
Added successfully
Model tap prob 92.78%; 3 raters marked tappable.
Write post
POST
Write something about this image
TAGS:
Model tap prob 32.20%; 3 raters marked tappable.
JioChat
United States
Accept and Continue
+1
Model tap prob 94.38%; 5 raters marked tappable.
Verify +1 (415) 579-1543
Waiting to automatically detect an SMS sent to +1 (415) 579-1543. Wrong number?
Enter 6-digit code
Resend SMS
59:44
Call me
Model tap prob 29.42%; 4 raters marked tappable.
MyFRITZ!
FRITZ!
To register, connect your mobile device with your FRITZ!Box via wireless LAN and start the MyFRITZ!App. Use the user name and password of a FRITZ!Box user as the login data.
Log On

A.6 Example 6


Input Image and Region
Model tap prob: 48.65%
12:48
0
Saliency Heatmap
0.0016
0.0014
0.0012
0.0010
0.0008
0.0006
0.0004
0.0002
Left: Input image and region
Middle: Heatmap
Right: Heatmap * Image
Most similar examples
(tappable on left, non-tappable on right)
Model tap prob 68.90%; 3 raters marked tappable.
new
Model tap prob 34.72%; 3 raters marked tappable.
Local Network
Loading
Model tap prob 69.09%; 3 raters marked tappable.
Starred
No entries
Model tap prob 34.88%; 4 raters marked tappable.
e.g. Los Angeles, CA
Current Location
Model tap prob 66.92%; 5 raters marked tappable.
Trips
April 2017
April Total
0.0 mi | $0.00
Model tap prob 34.91%; 1 raters marked tappable.
Scan Halal
my history
Model tap prob 67.99%; 0 raters marked tappable.
My schedule
Add day
Model tap prob 28.21%; 1 raters marked tappable.
Model tap prob 67.74%; 4 raters marked tappable.
Review | 4/14/17
8 Sat
9 Sun
10 Mon
11 Tue
12 Wed
13 Thu
Today
Statistics
Slept 0 time
Model tap prob 23.64%; 0 raters marked tappable.
Notifications
Welcome to Tinystep :)
Complete your kid's profile
just now