AidUI: Toward Automated Recognition of Dark Patterns in User Interfaces

Anonymous Author(s)

***Abstract*— Past studies have illustrated the prevalence of *UI dark patterns*, or user interfaces that can lead end-users toward (unknowingly) taking actions that they may not have intended. Such deceptive UI designs can be either intentional (to benefit an online service) or unintentional (through complicit design practices) and can result in adverse effects on end users, such as oversharing personal information or financial loss. While significant research progress has been made toward the development of dark pattern taxonomies across different software domains, developers and users currently lack guidance to help recognize, avoid, and navigate these often subtle design motifs. However, automated recognition of dark patterns is a challenging task, as the instantiation of a single type of pattern can take many forms, leading to significant variability.**

In this paper, we take the first step toward understanding the extent to which common UI dark patterns can be *automatically* recognized in modern software applications. To do this, we introduce AIDUI, a novel automated approach that uses computer vision and natural language processing techniques to recognize a set of visual and textual *cues* in application screenshots that signify the presence of ten unique UI dark patterns, allowing for their detection, classification, and localization. To evaluate our approach, we have constructed CONTEXTDP, the current largest dataset of fully-localized UI dark patterns that spans 175 mobile and 83 web UI screenshots containing 301 dark pattern instances. The results of our evaluation illustrate that AIDUI achieves an overall precision of 0.66, recall of 0.67, F1-score of 0.65 in detecting dark pattern instances, reports few false positives, and is able to localize detected patterns with an IoU score of 0.84. Furthermore, a significant subset of our studied dark patterns can be detected quite reliably (F1 score of over 0.82), and future research directions may allow for improved detection of additional patterns. This work demonstrates the plausibility of developing tools to aid developers in recognizing and appropriately rectifying deceptive UI patterns.

I. Introduction

Modern user interfaces (UIs) have unprecedented influence on the daily lives of users due to the increasing digitization of common tasks – ranging from financial transactions to shopping. As such, an emphasis on ease of use in the design of these interfaces has never been more critical. However, while UI designers typically strive to create interfaces that facilitate seamless completion of computing tasks, they are also capable of creating interfaces that may subtly mislead users into performing tasks they did not intend, but which may benefit business stakeholders. This dichotomy is influenced by competing design pressures: (i) designing easy to use UIs may increase the reputation and overall market share of an application; and (ii) designing UIs that intentionally influence users into performing certain actions may benefit the goals of application stakeholders.

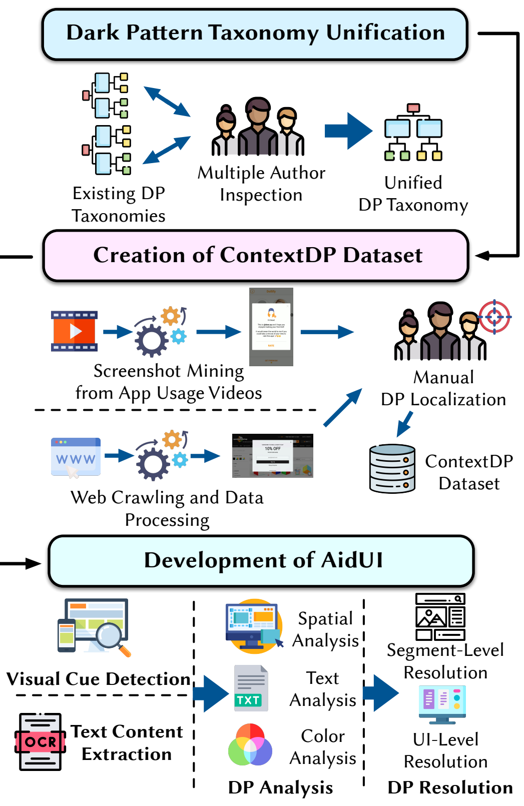


Fig. 1. This paper presents (i) a unified taxonomy of UI dark patterns, (ii) the CONTEXTDP dataset which contains 501 screenshots depicting 301 DP and 243 non-DP instances, and (iii) the AIDUI approach, which is able to identify and process visual and textual cues to detect and localize dark patterns.

This "dark" side of UI design has received increasing attention from various research communities in recent years, and has led to an increased shared understanding of a phenomenon referred to as *dark patterns (DPs)*. While the notion of DPs appeared in research literature as early as 2010 [11], Mathur *et al.* [35] recently defined dark patterns as:

"User interface design choices that benefit an online service by coercing, steering, or deceiving users into making decisions that, if fully informed and capable of selecting alternatives, they might not make".

There is ongoing work from the human-computer interaction (HCI) community into understanding the ethical considerations [22, 23], constructions [10, 22, 24, 27, 28, 49, 50, 54], user perspectives [14, 22, 34], and practitioner perspectives [23] of DPs. Additionally, the community has worked to construct evidence-based taxonomies of generally agreed upon DPs present in modern UIs [10, 22–24, 27, 28, 35, 49, 50, 54]. The prevalence of these identified categories of DPs is undeniable, as recent studies have illustrated that nearly 11% of top shopping websites [35] and nearly 95% of the top 240 apps on Google Play [16] contain one or more defined DPs. It is important to note that such DPs may not always be the result of ill intent from a designers vantage point, and past work has suggested that the current prevalence of such patterns may be the result of complicit, and perhaps unintentional, design practices on the part of software designers and developers [23].

However, regardless of the reason for their introduction and continued use, it is clear that deceptive UI patterns can have a negative effect on end-users. Such effects have been documented by prior work illustrating that, in many cases, DPs can result in financial loss [7, 25], or in oversharing personal information [17, 54]. Additionally, a prior human study that examined end-user perception of DPs in mobile apps found that most users cannot recognize DPs [16]. Given the prevalence and potential negative effects of deceptive UI patterns, and past evidence that designers and developers may unintentionally introduce such designs into their apps [23], developer-facing tools that can automatically recognize and signal the presence of such patterns could aid in avoiding these deceptive designs and improve overall software quality.

Despite the potential benefit of automated techniques for detecting DPs, there are two key challenges related to the design and implementation of such an approach. First, while the research community has made considerable progress in defining various types of DPs across a number of domains [11, 16, 23, 35], there can still be significant variability in how these patterns are *instantiated* in software applications. That is, while the very etymology of the phrase "dark pattern" suggests the presence of strong semantic signals that characterize different deceptive UI designs, the actual implementation of such designs can vary significantly. This makes designing an approach to detect such patterns difficult. Second, the research community currently lacks a large dataset of DPs with *fine-grained localization information* mapped to a *unified* taxonomy of DPs. While prior work has led to the creation of existing datasets, such as the one produced by Mathur *et al.* [35] related to online shopping, these datasets do not contain localized DPs (*e.g.,* bounding boxes that denote the location and spatial properties of the DPs) and such datasets are typically created with domain-specific categories of DPs.

To address the need for the automation of the detection of DPs in UIs, we introduce AIDUI (**Aid** for detecting **UI** Dark Patterns) - an approach that conducts *textual*, *icon*, *color*, and *spatial* analysis of a UI to automatically detect the presence of underlying DPs. The key idea underlying AidUI is that there exist several visual and textual *cues*, that when (co)-appearing, signify the presence of various dark patterns. By detecting these individual cues, and analyzing their (co)-occurrence, AIDUI is able to overcome challenges related to the variability of dark patterns as they appear "in-the-wild". AIDUI is the first approach to attempt to detect DPs using a *fully automated* process, and performs cue detection using a combination of computer vision and natural language template matching techniques. AIDUI operates solely on *visual* data, requiring *only a screenshot of a user interface as input*, making it easily extensible to multiple software domains.

In the process of building and evaluating AIDUI we have constructed CONTEXTDP, the current largest dataset of fully-localized DP instances to UI screenshots, containing 258 screenshots with 301 DP instances. The DP instances map to a *unified set of ten DP categories* derived through merging common DP categories described in previous taxonomies [11, 23, 35]. This dataset spans multiple application domains, including 175 mobile app screenshots and 83 Web UI screenshots, for a total of 258 screenshots depicting 301 DPs. Additionally, we augment our dataset with a set of 243 screenshots (164 mobile & 79 web) that do not contain DPs to investigate the likelihood of AidUI to detect false positives.

We conducted a comprehensive evaluation to measure the effectiveness and robustness of AIDUI. Using CONTEXTDP, we measured the precision, recall and F1-score of AIDUI's DP detection capabilities, as well as its localization performance in different settings. Additionally, we performed an ablation study to examine which components of our technique contributed most to the overall detection performance. We found that, across all DP instances in our dataset, AIDUI achieved an overall average precision of 0.66, average recall of 0.67 and average F1-score of 0.65. However, for five DPs, AIDUI was able to achieve higher precision (0.87), recall (0.80) and F1-score (0.82) compared to the five other DP categories, illustrating the variance in difficulty of detecting different types of patterns. Our analysis of localization performance illustrates that AidUI can provide useful localization that is specific to the various salient patterns it detects. Finally, our ablation study illustrated that combining text analysis with the color and spatial analysis led to the highest performance. In summary, the contributions of our work are as follows:

- AIDUI, the first automated approach capable of detecting the presence of a diverse set of DPs. To accomplish this, the proposed approach adapts techniques from natural language processing and computer vision to detect various cues that signify dark patterns.
- CONTEXTDP, which is the current largest dataset of DP instances localized to UI screenshots. CONTEXTDP contains 162 web and 339 mobile screenshots depicting 301 DP and 243 Non-DP instances. We make this dataset fully open source to encourage future work on automated DP detection and localization.
- A comprehensive empirical evaluation of AIDUI that measures the precision, recall, F1-score, localization performance, and the robustness of the approach. Our evaluation illustrates that AIDUI can detect DPs from a subset of our studied DP categories with higher precision, recall and F1-score, with useful localization results.
- An online appendix [2] containing source code, experimental data, and trained models that can facilitate the replication of our results and encourage future work on automated detection of UI DPs.

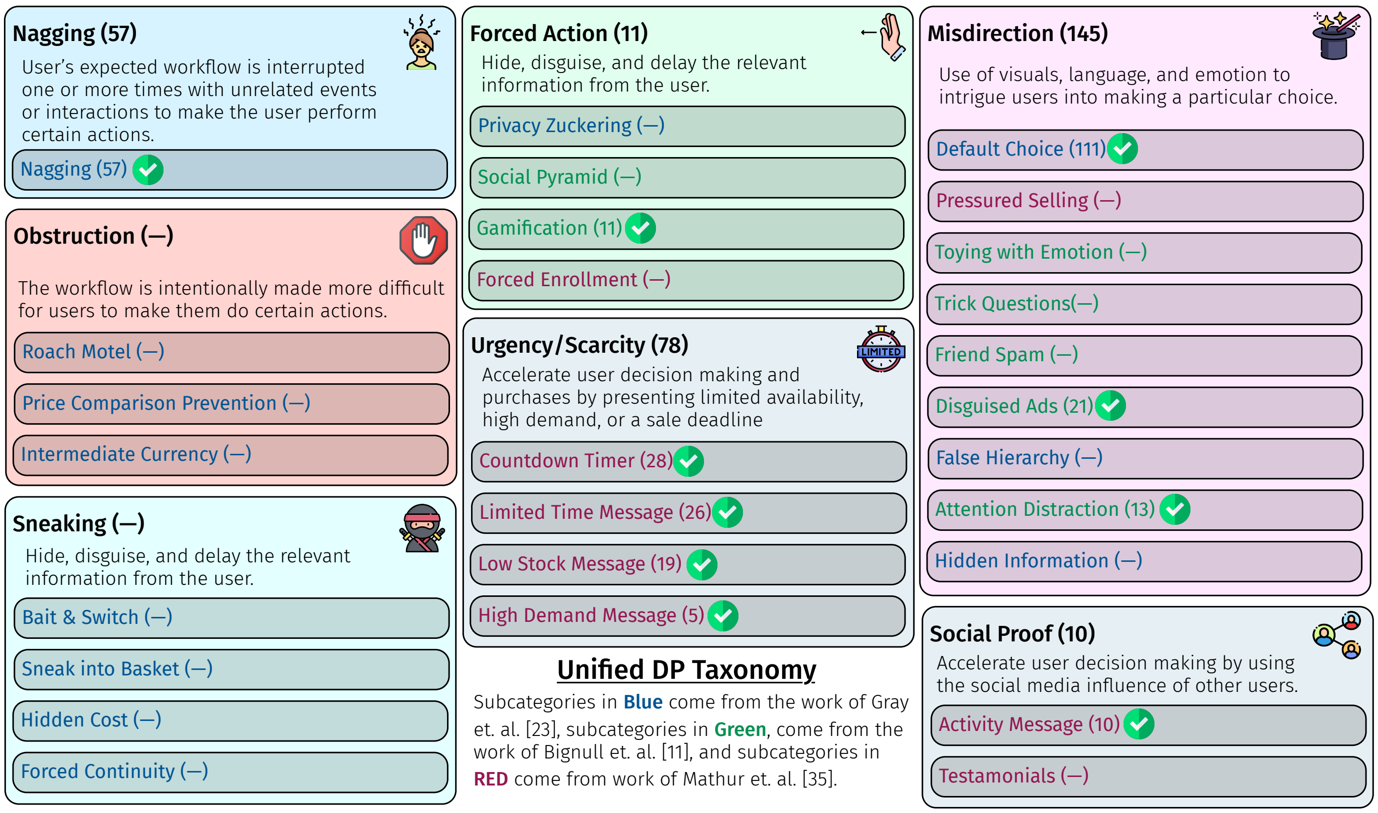

Nagging (57)
User's expected workflow is interrupted one or more times with unrelated events or interactions to make the user perform certain actions.
Nagging (57) ✓
Obstruction (—)
The workflow is intentionally made more difficult for users to make them do certain actions.
Roach Motel (—)
Price Comparison Prevention (—)
Intermediate Currency (—)
Sneaking (—)
Hide, disguise, and delay the relevant information from the user.
Bait & Switch (—)
Sneak into Basket (—)
Hidden Cost (—)
Forced Continuity (—)
Forced Action (11)
Hide, disguise, and delay the relevant information from the user.
Privacy Zuckering (—)
Social Pyramid (—)
Gamification (11) ✓
Forced Enrollment (—)
Urgency/Scarcity (78)
LIMITED
Accelerate user decision making and purchases by presenting limited availability, high demand, or a sale deadline
Countdown Timer (28) ✓
Limited Time Message (26) ✓
Low Stock Message (19) ✓
High Demand Message (5) ✓
Unified DP Taxonomy
Subcategories in Blue come from the work of Gray et. al. [23], subcategories in Green, come from the work of Bignull et. al. [11], and subcategories in RED come from work of Mathur et. al. [35].
Misdirection (145)
Use of visuals, language, and emotion to intrigue users into making a particular choice.
Default Choice (111) ✓
Pressured Selling (—)
Toying with Emotion (—)
Trick Questions(—)
Friend Spam (—)
Disguised Ads (21) ✓
False Hierarchy (—)
Attention Distraction (13) ✓
Hidden Information (—)
Social Proof (10)
Accelerate user decision making by using the social media influence of other users.
Activity Message (10) ✓
Testamonials (—)

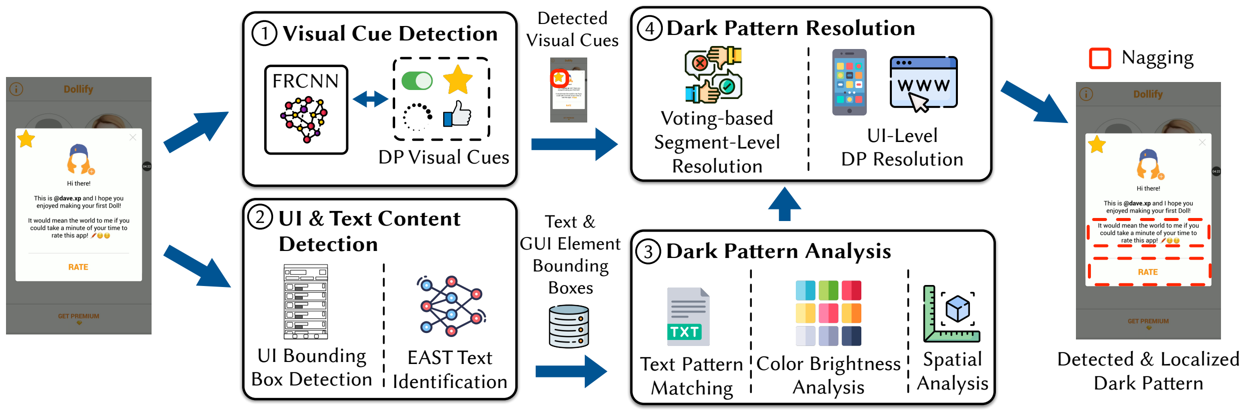
Fig. 2. Our Unified Dark Pattern Taxonomy – numbers in parentheses signify the number of examples of the DPs that are present in the CONTEXTDP dataset, and the ✓ signifies those patterns that AIDUI is designed to detect. Note that this unified taxonomy is a targeted combination of past works [11, 23, 35] with minor modifications.

II. UNIFYING WEB AND MOBILE DARK PATTERNS

There has been a wealth of work from the general HCI community that has constructed DP taxonomies. One of the earliest taxonomies comes from Brignull [11] who not only coined the term "Dark Pattern", but also proposed a classification of DPs into different categories. The DP categories originally derived by Brignull *et al.* [11] are available on the `darkpatterns.org` portal with examples from web and mobile applications. Gray *et al.* [23] redefined Brignull's taxonomy and proposed a taxonomy consisting of five main categories of DPs which includes (i) Nagging, (ii) Obstruction, (iii) Sneaking, (iv) Interface Interference and (v) Forced Action. Geronimo *et al.* [16] used Gray's taxonomy and extended the original meaning of Aesthetic Manipulation and Forced Action classes to include new DPs instances. The authors analyzed pervasiveness of DP in mobile applications by conducting an approach similar to cognitive walk through techniques [40] whereas previous works [23, 35, 38] analyzed screenshots to classify DPs. Finally, Mathur *et al.* [35] performed a semi-automated, large-scale collection of DPs in online shopping websites, and derived a taxonomy of 15 dark patterns grouped into 7 categories. In the course of their data collection, they found 1,818 instances of DPs on the top ≈11k shopping websites. More recent work has aimed to identify DPs that are specific to various different contexts or domains including (i) shopping web apps [35], (ii) computer games [54], (iii) privacy-centric software [10], (iii) robotics [27], and (iv) digital consent forms [49].

Given the somewhat complementary, yet disparate nature of existing taxonomies of DPs, we aimed to create a unified taxonomy that merges together similar categories and provides a larger landscape of patterns for mobile and web apps toward which we can design and evaluate our automated detection approach. Given that the scope of our work is primarily concerned with web and mobile UIs, we did not include many of the domain specific taxonomies mentioned above. Instead, our unified taxonomy is primarily a fusing of the various categories and subcategories derived by Gray *et al.* [23], Mathur *et al.* [35] and Brignull *et al.* [11]. To build our unified taxonomy, first one author gathered one-two existing examples of each type of DPs that exists in the categories described by Gray *et al.* [23], Mathur *et al.* [35], and Birgnull *et al.* [11]. Next, two authors met to review each DP example and re-group the various categories under unified headings. Our final unified taxonomy, illustrated in Figure 2, spans 7 parent categories which organize a total of 27 classes that describe different DPs. Note that many of the DPs in our taxonomy are self-documenting (*e.g.*, `countdown timer`), however, we provide full descriptions and examples of each DP in our online appendix [2].

Not all dark patterns are created equal from viewpoint of the underlying UI motifs that signal their presence. For example, for the `Sneak Into Basket` DP type, typically there would



Dollify
Hi there!
This is @dave.xp and I hope you enjoyed making your first Doll!
It would mean the world to me if you could take a minute of your time to rate this app!
RATE
GET PREMIUM
① Visual Cue Detection
FRCNN
DP Visual Cues
② UI & Text Content Detection
UI Bounding Box Detection
EAST Text Identification
Detected Visual Cues
Text & GUI Element Bounding Boxes
④ Dark Pattern Resolution
Voting-based Segment-Level Resolution
UI-Level DP Resolution
③ Dark Pattern Analysis
TXT
Text Pattern Matching
Color Brightness Analysis
Spatial Analysis
Nagging
Detected & Localized Dark Pattern

Fig. 3. Workflow of the AIDUI approach

be several actions and screens required to detect the presence of such a pattern, which introduces far more variability in its potential observed visual and textual cues. With this in mind we aimed to prioritize the detection strategy of AIDUI toward certain patterns that carry with them distinct visual and textual cues which both manifest on a *single screen*. We leave the detection of dynamic DPs involving multiple screens and actions for future work. To perform this prioritization, two authors met and further discussed the examples of each of the 27 classes of DPs and marked those that exhibited salient visual and textual cues, noting these cues for use in the later implementation of AIDUI. Thus, we identified a final set of 10 target DPs , toward which we oriented AIDUI's analysis. The targeted DP categories are marked with a ✓ in Figure 2.

III. THE AIDUI APPROACH

This section presents the AIDUI approach for automatically detecting DPs in UIs. The architecture of AIDUI, depicted in Figure 3, is designed around four main phases: ① the *Visual Cue Detection* phase, which leverages a deep learning based object detection model to identify UI objects representing visual cues for DPs, ② the *UI & Text Content Detection* phase, which extracts UI segments containing both text and non-text content, ③ the *DP Analysis* phase, which employs text pattern matching, as well as color and spatial analysis techniques to analyze the extracted UI segments and identifies a set of potential DPs, and ④ the *DP Resolution* phase, which uses results from both Visual Cue Detection and DP Analysis phases to predict a final set of underlying DPs in the given UI. It is important to note that AIDUI operates *purely on pixel data* from UI screenshots, making it extensible to different software domains. In the remainder of this section, we first discuss the motivation of the architecture of our approach and then discuss each of the four phases in detail.

A. Approach Motivation

While studying existing DP taxonomies, we observed that different DP categories tend to include certain types of icons and text as well as exhibit distinct patterns in color brightness and spatial organization of UI elements. For example, instances from `nagging` category are likely to have both visual (*i.e.,* rating related icons such as like buttons, stars, etc.) and textual cues (*i.e.,* keywords such as "rate", "rating" etc.). Based on these observations regarding the visual and textual cues across different DP categories, we have identified five tasks that are required for detecting DPs. These include two detection tasks (*i.e.,* visual cue detection and text content detection) as well as three analysis tasks related to properties of UI components (*i.e.,* text, color, and spatial analysis). The major phases of our approach are designed around these detection and analysis tasks.

B. Phase 1: Visual Cue Detection

The main goal of this phase is to identify the icons that can serve as important visual cues for detecting DPs. To accomplish this goal, our approach leverages deep learning based object detection techniques. In this case, we adapt an implementation of Faster R-CNN [45] to accurately identify and localize specific types of icons in a target UI.

The Visual Cue Detection phase receives a UI as an input, then uses Faster R-CNN model to detect the positions and bounding boxes of the target icons. Finally, through a mapping of detected icons to likely candidate DPs, it provides a list of potential DP categories. The output of this phase is a `json` file consisting of bounding boxes and DP categories for detected icons with the highest confidence scores.

1) Icon Detection: Typically, DL models such as the Faster R-CNN model are data hungry and need to be trained with manually labeled, large datasets. However, for the visual cues that we wish to detect (*e.g.,* rating bars, like buttons, etc.), we only have a small set of examples. Labeling such UI components in existing UI datasets, such as RICO [15], would be extremely time consuming, and hence would not scale. As such, we develop a fully-automated training data generation technique that allows for the creation of large sets of training data for our target visual cues. This training data generation technique has been successfully used in past work [9] to detect touch indicators on mobile UI screenshots with extremely high accuracy (*e.g.,* 98%). It should be noted that this data generation procedure is completely independent and separate from the CONTEXTDP dataset. That is, we automatically generate the dataset to train our neural icon detection model such that no screens or apps used in the training of this model appear in the CONTEXTDP dataset.

TABLE I
MAPPING OF ICONS (VISUAL CUES) TO LIKELY DPS

Icons	**Likely DPs**
Like	Nagging
Dislike	Nagging
Star	Nagging
Toggle Switch (on)	Default Choice
Ad	Nagging, Disgiused Ads
Ad Loader	Nagging, Gamification

We use the existing large scale RICO [15] UI dataset to aid in our data generation process. RICO, by far the largest repository of mobile app designs, contains 66k unique UI screens from 9.3k free Android apps spanning over 27 categories. First, 1020 unique UIs of different apps are randomly sampled from the RICO dataset. Then we randomly sample different versions of the icons that we target to detect as visual cues which are freely available on icon or UI design websites. In total, we collect total 16 different versions of 6 icon types (*i.e.,* 2 Google Ad icons, 3 loading icons, 3 like icons, 3 dislike icons, 2 star icons, and 3 toggle switch icons). Next, we programmatically overlay the icons on the UIs to create a dataset consisting of total 16320 images (*i.e.,* 16 icons x 1020 UIs) and corresponding annotations in MSCOCO [29] format. While creating synthesized data, we place randomly sized icons (*i.e.,* foreground image) at random locations on the UIs (*i.e.,* background image). To create training and validation sets, we split the dataset into 80%/20% respectively. We then use the open source machine learning framework PyTorch [4] and Torchvision [6] library to adapt an implementation of the Faster R-CNN [45] object detection model. When performing inference, the model outputs bounding box and label predictions with confidence scores for identified icons.

2) Mapping Detected Icons to DPs: As stated earlier in this section, some icons tend to relate to different types of DPs. Based on this observation, our approach defines rules to map the detected icons to a set of possible DPs (illustrated in Table I). Once the trained Faster R-CNN model detects icons in the given UI, our approach first selects the label and bounding box having the highest confidence score and then uses the predefined rules for mapping the detected icon to potential DP(s). Thus, as a final output of *Visual Cue Detection* phase, a `JSON` file is produced that includes labels, bounding boxes and confidence scores of the detected icon(s) as well as a set of related DPs that are likely to be present in the UI.

C. Phase 2: UI & Text Content Detection

To identify the general UI components and textual cues related to DPs, we first need to extract the UI segments that have text contents. To accomplish this, we adapted the implementation of the approach introduced by Xie *et al.* [52], called UIED, that consists of two parts for detecting graphical and textual UI elements. For text detection and extraction, it leverages the EAST text detection model [57]. For graphical UI elements, it uses an unsupervised edge detection algorithm and CNN to locate and classify elements. As we are interested in extracting UI segments with text contents, our approach collects the OCR'd output in `JSON` format from *UIED*. The `JSON` file contains the text contents along with the corresponding bounding box information and serves as an input for the next phase (*i.e., DP Analysis*).

TABLE II
MAPPING OF SAMPLE LEXICAL PATTERNS TO DPS

Dark Pattern	**Sample Lexical Patterns**
Nagging	watch + <ad/session>
Gamification	<ask/invite/refer>+ friends + to + [subject] signup + for + <credits/points/tokens>
Default Choice	I + <agree/consent>+ to + [predicate]
Attention Distraction	I +<decline/don't>+ <want/opt out/refuse to>+ [predicate]
Countdown Timer/ Limited Time Message	sale + <ends/countdown/now>; shop + <now/within>
Low Stock/ Limited Time/ High Demand	<apply/order>+ by; only + [number] + in stock; [number] + available
Activity Message	[number] + items sold + [number] + time

D. Phase 3: DP Analysis

In *DP Analysis* phase, the main goal is to apply different analysis techniques on the UI text segments to predict the potential underlying DPs associated with those segments. As stated earlier in section III-A, different DP categories tend to exhibit certain textual, visual and spatial patterns. Hence, *DP Analysis* phase incorporates different techniques for analyzing text, color, and spatial information extracted from the identified UI segments.

First, the UI text segments are provided as inputs. Then, text analysis is used to select the segments that have text contents showing patterns related to different DP categories. Next, color analysis is performed to categorize these selected segments based on their brightness level. Finally, in the spatial analysis, relative size information of the neighbor segments is computed. The final output of the *DP Analysis* phase is a `JSON` file that combines the results from textual, color, and spatial analysis of the UI segments.

1) Text Analysis: To select the UI segments exhibiting specific textual cues, our approach leverages a pattern matching technique. Based on the observed textual cues for different DP categories, we define heuristic pattern matching rules. We defined corresponding lexical patterns for each of the targeted 10 DP categories that match keywords as well as sentence patterns. We provide a subset of these lexical patterns in Table II and provide a detailed list of these keywords and patterns in our online appendix [2]. When a match occurs, our approach returns a `JSON` object containing the segment information (*i.e.,* UI coordinates, width, height etc.) of the matched text content and the corresponding DP category related to the pattern rule used. In the case that there are multiple matched contents under the same DP category, the content with the longest sequence is selected. We implement our linguistic pattern matching using spaCy [5], python library.

2) Color Analysis: Next, in the *Color Analysis* step, we categorize the detected segments from *Text Analysis* step based on their brightness. To accomplish this, our approach uses a color histogram analysis technique. In this analysis process, we first calculate the grayscale histogram of the segment where we use two bins, one for the intensity values in the range of

0-127 and another one for the intensity values ranging from 128-255. If 65% or more of the pixels have values in the range of 0-127, the segment is categorized as *"Darker"* segment, whereas if 65% or more of the pixels have values in the range of 128-255, the segment is considered as a *"Brighter"* one. In all other cases, the segment is categorized as a *"Normal"* one. The reason for performing this classification of color intensity is that there are some DP types (such as `Attention Distraction`, `Default Choice`) wherein orthogonal brightness/contrast levels are likely to signal the presence of a DP. To implement color analysis, we use the Python API of OpenCV [3]. For each segment, the *Color Analysis* step outputs a `JSON` object containing the histogram results and the associated brightness category (*i.e.,* darker/brighter/normal).

3) Spatial Analysis: It has been observed that some DP categories (such as `Attention Distraction`, `Default Choice`) are signified by size differences between their nearest components. Hence, in the *Spatial Analysis* step, our main goal is to find the neighbor segments around a given segment and compute the relative size of a segment in comparison to the size of the neighbors. Here, our approach conducts a two step process for *Spatial Analysis*. First, for a given segment, we calculate the neighborhood area around it by adding a proximity factor to the segment boundaries. The proximity factor is calculated as a very small percentage (*e.g.,* $\approx 5\%$) of the segment size, which is derived empirically. If any other segment's boundary intersects with the boundary of the neighborhood, that segment is then considered as a neighbor of the current segment being analyzed. Once the neighbors are found, the second step is to compute the relative width and height of a segment with regard to the width and height of its neighbors. To do this, each segment's width and height are divided by the maximum width and height found in the neighborhood. Finally, for each segment, the *Spatial Analysis* step outputs a `JSON` object containing the information regarding neighborhood coordinates, neighbor segments and relative size (width and height).

E. Phase 4: DP Resolution

The final phase of our approach is the *DP Resolution* phase where our main goal is to identify the underlying DPs that are most likely to be present in the given UI. *DP Resolution* is a two step process, consisting of *Segment Level Resolution* and *UI Level Resolution*. In *Segment Level Resolution*, our approach identifies potential DPs in UI segments by considering the textual, color, and spatial analysis results (section III-D). In the *UI Level Resolution*, results from both *Visual Cue Detection* (section III-B) and *Segment Level Resolution* (section III-E1) are used to to predict a final set of underlying DPs in the UI. Localization is performed using the bounding boxes of the identified UI elements from the screen.

1) Segment Level Resolution: To identify potential DPs in a segment, we take into consideration the results from textual, color, and spatial analyses (section III-D). We employ a voting mechanism among the neighbor segments to resolve the most likely DPs at the segment level. In this process, a UI segment gets votes for a DP category from its neighbor segments if some specific textual, color, or spatial criteria are met. For instance, in case of text based resolution, a segment gets a vote from a neighbor segment if both of them exhibit textual patterns related to a similar DP. In the visual resolution, a segment gets a vote from its neighbor if the color brightness of the segment and the neighbor is opposite (*i.e.,* brighter/darker). In the spatial resolution, a segment gets a vote from a neighbor, if the difference of the relative height or width between the segment and the neighbor is more than a predefined threshold value. In the voting process, we not only calculate the number of votes but also compute a score for the earned votes. The score of the votes depends on the number of factors or criteria satisfied in textual, color, and spatial resolution processes. The final output of the *Segment Level Resolution* is a `JSON` object where for each segment a set of DPs with corresponding votes and scores are reported.

2) UI Level Resolution: In this step, AIDUI makes the final prediction about any underlying DPs that are likely to be present in the UI. To accomplish this, results from both *Segment Level Resolution* (section III-E1) and *Visual Cue Detection* (section III-B) are taken as input. For each identified DP category, we take the votes with the highest score found in *Segment Level Resolution*. Later, if applicable, we also take into account the vote and confidence score from *Visual Cue Detection*. In the case that both types of information are present, we combine the confidence scores from the *Segment Level Resolution* and the *Visual Cue Detection* using an 80/20 ratio (which we derived empirically). The output of *UI Level Resolution* is a JSON object containing identified DP categories along with the number of votes, scores, and associated segment information for localization. Finally, the top (up to) two DP categories having the highest number of votes along with a certain confidence level are selected as the most likely underlying DP in the given UI.

IV. EVALUATION METHODOLOGY

In this section, we present the design of the study we employed for the evaluation of AIDUI. Our empirical study is aimed at assessing the *performance* and *robustness* of the approach as well as the contribution of different analysis modules to the overall effectiveness. To accomplish these study goals we formulated the following research questions:

- **RQ$_1$:** *What is the precision, recall and F1-score of* AIDUI *in detecting DP in UIs?*
- **RQ$_2$:** *How robust is* AIDUI *in detecting DPs in UIs from different domains (mobile/web)?*
- **RQ$_3$:** *How often does* AIDUI *detect false positives in screens that known to not contain DPs?*
- **RQ$_4$:** *How well is* AIDUI *able to localize DPs in UI screens?*
- **RQ$_5$:** *What is the contribution of different analysis modules in detecting DPs in UIs ?*

A. *Derivation of the* ContextDP *Dataset*

To evaluate AidUI, we have derived ContextDP, the current largest dataset of labeled UIs containing both DP and Non-DP instances. ContextDP includes total 501 mobile and web UI screenshots that represent 301 DP instances as well as 243 Non-DP instances. To collect the mobile UIs, we made use of the comprehensive video dataset by Geronimo *et al.* [16] which includes mobile app usage screen recordings and classifications of identified DPs at various timestamps in the videos. The publicly available dataset by Geronimo *et al.* [16] includes 15 videos of mobile apps and information about the timestamps of observed DPs in those videos. Based on the available videos and the timestamps data, we extract the frames from the videos according to given timestamps. As the public dataset only contains 15 videos, we contacted and worked with the authors of the paper to extract all relevant DPs from their entire corpus of user videos, while carefully excluding video segments that may contain personal information. During this process, we collected 2994 UI screenshots in total spanning over 68 apps from 3 categories (communication, entertainment, and music). Similarly, based on the data provided by Mathur *et al.* [35], we collected over 1500 web screenshots from popular shopping websites with known DPs by leveraging their publicly available dataset. To further bolster the number of DP examples and the generalizability of our dataset, we also randomly collected a combined 5000 mobile UI screenshots from the RICO dataset [15] (not used to the train the icon detection model) and web UI screenshots from popular Alexa 100 shopping websites respectively. After the curation process, we randomly selected a total of 501 (339 mobile and 162 web) screenshots for labeling. This sampling represents a 95% confidence level and 8% confidence interval for mobile apps, and 12% confidence interval for web UIs.

Next, three authors of this paper participated in a rigorous labeling process for both categorizing and creating bounding boxes for each observed DP in our sampled dataset. In order to ensure a consistent and agreed upon labeling strategy, prior to the labeling process, we conducted a comprehensive discussion among the authors to review the rationales behind the labeling decisions of our 501 UI screenshots. This process included examining and discussing two-three examples of each of the 10 targeted DPs from our taxonomy, and deriving a set of labeling and bounding box guidelines (these guidelines are available in our online appendix [2]). Then, two of the authors independently labeled each of the 501 UI screens using the Label Studio [1] application, resolving necessary conflicts.

B. *Evaluation Metrics*

We evaluate AidUI with respect to all RQs by using following metrics: *precision*, *recall*, *F1-score*, and *IoU*. The metrics are defined as follows:

- **Precision** describes the ability of the classifier to properly distinguish between true positives and false positives. *Precision* is computed as $Precision = TP/TP + FP$, where TP is the number of true positives and FP is the number of false positives. In our context a TP (true positive) is a correct prediction that a given DP type exists on a given UI screen when it does, and an FP (false positive) is when the approach predicts a DP type that is not present on a given screen.
- **Recall** is intuitively the ability of the classifier to find all the samples that are positive. *Recall* is computed as $Recall = TP/TP + FN$, where TP is the number of true positives and FN is the number of false negatives. In this context a false negative is a case when a DP is present on a screen, but AidUI could not detect it.
- **F1-score** is the harmonic mean of the precision and recall. *F1-score* is defined as $F1\text{-}score = 2 \times Precision \times Recall/Precision + Recall$
- **IoU** computes the amount of overlap between the predicted and ground truth bounding boxes. *IoU* is defined as: $IoU = Area\ of\ Overlap/Area\ of\ Union$. Note that we define two versions of *IoU* for our evaluation. First, *strict IoU* operates according to the formula above, wherein predictions are directly compared to the ground truth. Second we define a *contained IoU* wherein we set the *IoU* to 100% if the predicted bounding box falls *within* the ground truth bounding box. This is because during our investigation of the localization results, we found that AidUI often predicts more specific bounding boxes for a given DP that could still be helpful to end users. Thus, this measure may offer a more realistic characterization of AidUI's performance by rewarding segments identified within the ground truth bounding box.

C. *RQ_1 & RQ_2: DP Detection Performance Across Domains*

To answer RQ_1, we assess the ability of AidUI to accurately detect DP instances. This experiment aims to measure AidUI's performance both as a whole, and for individual DP categories in order to determine AidUI's effectiveness on individual DP categories. In this process, AidUI is evaluated on ContextDP (section IV-A) using the aforementioned evaluation metrics (*precision, recall and F1-score*) described in section IV-B. The evaluation metrics are computed using the predictions and derived ground truth from ContextDP. To answer RQ_2 we compare the detection results across both the web and mobile portions of the ContextDP dataset.

D. *RQ_3: Potential False Positive DP Detections*

AidUI's utility as a potential developer tool is predicated on the intention that it perform reasonably well at both detecting dark patterns when they do exist, and not triggering false alarms for screens that do not contain DP instances. Thus, to evaluate the potential of AidUI to trigger false positives, we applied our technique to the 243 screenshots that do not contain DPs from ContextDP.

E. *RQ_4: Localization Performance*

We answer RQ_4 using both *contained* and *strict* IoU which are calculated by measuring the difference between the predicted bounding boxes from AidUI and the ground truth bounding boxes from ContextDP.

TABLE III
AidUI DP Category-wise Detection/Classification Performance

	All				Mobile				Web			
DP Category	**Instances**	**Precision**	**Recall**	**F1-score**	**Instances**	**Precision**	**Recall**	**F1-score**	**Instances**	**Precision**	**Recall**	**F1-score**
Activity Message	10	1.0	0.80	0.89	–	–	–	–	10	1.0	0.80	0.89
High Demand Message	5	1.0	0.80	0.89	–	–	–	–	5	1.0	0.80	0.89
Low Stock Message	19	0.75	0.79	0.77	–	–	–	–	19	0.75	0.79	0.77
Limited Time Message	26	0.95	0.73	0.83	–	–	–	–	26	0.95	0.73	0.83
Countdown Timer	28	0.63	0.86	0.73	–	–	–	–	28	0.83	0.86	0.84
Attention Distraction	13	0.55	0.46	0.50	9	0.29	0.22	0.25	4	1.0	1.0	1.0
Default Choice	111	0.67	0.59	0.62	99	0.73	0.60	0.66	12	0.38	0.50	0.43
Disguised Ads	21	0.32	0.57	0.41	21	0.33	0.57	0.42	–	–	–	–
Nagging	57	0.52	0.77	0.62	57	0.54	0.77	0.64	–	–	–	–
Gamification	11	0.80	0.36	0.50	11	1.0	0.36	0.53	–	–	–	–
Total Instances	**301**				**197**				**104**			
Avg. Precision	**0.66**				**0.63**				**0.82**			
Avg. Recall	**0.67**				**0.61**				**0.77**			
Avg. F1-score	**0.65**				**0.60**				**0.79**			

TABLE IV
AidUI Overall Detection/Classification Performance

	Instances	**Precision**	**Recall**	**F1-score**
Non-DP	243	0.85	0.86	0.85
DP	301	0.66	0.67	0.65

F. RQ$_5$: Ablation Study of DP Analyses

To answer this RQ, we conduct an experiment that is aimed at exploring the contributions of the textual, color, and spatial analysis modules in detecting DPs. To accomplish this, we choose different combinations of modules while conducting the evaluation process. As text analysis acts as the base module, it is selected in all the combinations. The combinations that we use in the evaluation are: *Text*, *Text + Color Analysis*, *Text + Spatial Analysis*, and *Text + Color + Spatial Analysis*. For each combination, we calculate the same evaluation metrics that we use in RQ1.

V. Empirical Results

A. RQ$_1$: Detection Performance

Our main goal in RQ1 is to measure the performance of AidUI in terms of detection/classification of both Non-DP and DP instances. In answering this RQ, we are interested in assessing AidUI's overall performance as well as performance specific to individual DP categories. We conduct the performance evaluation based on the metrics stated in section IV-B, *i.e., precision*, *recall* and *F1-score*. Table IV and III illustrate the aggregate and category-wise classification performance respectively.

From the classification results in table IV, we observe that AidUI achieves an overall *average precision* of 66%, *average recall* of 67% and *average F1-score* of 65% in detecting DP instances. Moreover, category-wise results in Table III show that a subset of DPs (*e.g.,* `Activity Message`, `High Demand Message`, `Low Stock Message`, `Limited Time Message`, `Countdown Timer` etc.) can reliably be detected with moderate to high *precision*, *recall* and *F1-score* values.

Instances where AidUI fails to identify DPs are mostly due to deviations in textual or visual patterns exhibited by some of the DP categories. For instance, 32% of the `Disguised Ads` instances are correctly predicted whereas 26% of them are wrongly predicted as *Non-DP*. To better understand, we present one such representative instance of *Disguised Ads* in Figure 4. Here, at a first glance, the appearance of the advertisement looks very similar to regular content. Though on the top right corner there is an *ad* icon, this particular advertisement has a subtle difference compared to other typical advertisements. To detect such subtle visual cues, we may need to develop separate deep learning solution based on a comprehensive study of diverse types of UI advertisement contents. Similarly, for some DP categories our approach fails due to the limitation of the vocabulary or textual patterns that AidUI considers, as illustrated in Figure 4b for the `Default Choice` DP. Future work could focus toward resolving these limitations through the use of *language modeling* wherein embeddings from the models are used to measure textual similarity, and hence would be able to better account for semantically similar, yet lexically varied text.

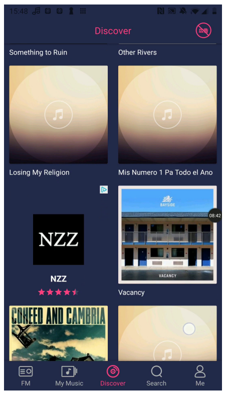


Figure 4a: Example of difficult to detect disguised ad.

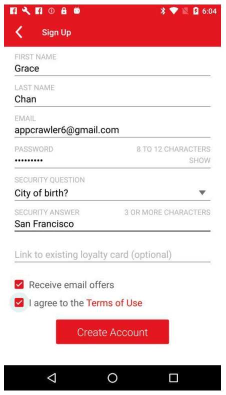

Sign Up
FIRST NAME
Grace
LAST NAME
Chan
EMAIL
appcrawler6@gmail.com
PASSWORD 8 TO 12 CHARACTERS
•••••••••
SHOW
SECURITY QUESTION
City of birth?
SECURITY ANSWER 3 OR MORE CHARACTERS
San Francisco
Link to existing loyalty card (optional)
☑ Receive email offers
☑ I agree to the Terms of Use
Create Account

Figure 4b: Example of difficult to detect Default Choice

Fig. 4. Example of difficult to detect Dark Patterns

Answer to RQ$_1$: AidUI achieves an overall average *precision*, *recall* and *F1-score* of 0.66, 0.67 and 0.65 respectively. However, for a subset of DP classes, including `Activity Message`, `High Demand Message`, `Low Stock Message`, `Limited Time Message` and `Countdown Timer`, AidUI performs well with high average *precision* (≈0.87), *recall* (≈0.80) and *F1-score* (≈0.82).

TABLE V
AIDUI LOCALIZATION PERFORMANCE

	All		Mobile		Web	
DP Category	**Avg. Strict IoU**	**Avg. Contained IoU**	**Avg. Strict IoU**	**Avg. Contained IoU**	**Avg. Strict IoU**	**Avg. Contained IoU**
Activity Message	0.351397	0.740450	–	–	0.351397	0.740450
High Demand Message	0.340726	0.730416	–	–	0.340726	0.730416
Low Stock Message	0.223910	1.000000	–	–	0.223910	1.000000
Limited Time Message	0.262617	0.808963	–	–	0.262617	0.808963
Countdown Timer	0.231990	0.823345	–	–	0.231990	0.823345
Attention Distraction	0.040522	1.000000	0.081790	1.000000	0.019887	1.000000
Default Choice	0.023972	0.569231	0.024030	0.542373	0.023409	0.833333
Disguised Ads	0.006635	0.916667	0.006635	0.916667	–	–
Nagging	0.015434	0.957007	0.015434	0.957007	–	–
Gamification	0.125986	0.777633	0.125986	0.777633	–	–
Overall	**0.162319**	**0.832371**	**0.050775**	**0.838736**	**0.207705**	**0.848072**

B. RQ_2: Domain-specific Performance

Table III illustrates that AIDUI performs significantly better on web UIs as compared to the UIs from mobile applications. In fact, we observe an increase of 30% for *avg. precision*, 26% for *avg. recall* and 31% for *avg. F1-score* between the two portions of our dataset. Our insight regarding the significant performance difference in identifying DPs across mobile and web domain is related to the prevalence of different types of DPs in those domains. While developing our unified taxonomy, we observed that there are particular groups of DPs that have dominant presence in a particular domain, *i.e.,* web or mobile. Moreover, we also observed that the patterns frequently found in web UIs typically skew toward textual cues whereas several patterns that are dominant in mobile UIs often contain *both* textual and visual cues. This phenomenon makes the DP identification task in mobile UIs more complex as compared to web UIs. As our current approach equally treats the outputs from text and color analysis, this could be one reason for the observed performance gap across the two software domains. Future work should aim to empirically examine the textual and visual differences in DPs across software domains in an effort to better inform future automated techniques.

Answer to RQ_2: AIDUI shows significant difference in performance across web and mobile domains. This is likely due to the fact that the observed DPs in our mobile dataset contain more varied patterns that are more difficult to detect overall.

C. RQ_3: False Positive DP Detection

To measure the rate of false positives, we applied AIDUI to a set of screens that, during the labeling process for ContextDP, were confirmed to not exhibit any dark patterns. Of the 243 to which we applied AIDUI, only 36 screens ($\approx$15%) were identified as having false positives, and a vast majority of these fell into the `Default Choice` DP category. It should be noted that given the scoring procedure among the various components of AIDUI's approach, it is possible to calculate a confidence threshold that a given screen contains a DP, which could be used to help indicate the potential severity or confidence of predictions for future developer tools.

Answer to RQ_3: AIDUI exhibits a false positive rate of 36/243 when applied to screens confirmed to not exhibit DPs. However, most of these misclassifications are into a single class `Default Choice` DP class, and could be further mitigated both by adjusting the sensitivity of cues for this class, and by displaying confidence scores for given predictions for future developer-facing tools.

D. RQ_4: Localization Performance

The results in Table V illustrate that our approach achieves a fairly low *strict IoU* both overall and on a category-by-category basis. However, this measurement approach presents a skewed view of the practical performance of AIDUI. This is because after examining both predicted and ground truth bounding boxes we noticed that this is largely due to precise bounding box selection by our text analysis module. Though our approach is actually able to localize a precise bounding box for various UIs, it is getting penalized because of having smaller intersections as compared to the ground truth, which tended to encompass areas between text, for example. Based on our observation, we defined the *contained IoU* to provide a more complete picture of AIDUI'S localization performance. The overall *avg. contained IoU* value 0.83 illustrates that AIDUI is properly localizing elements that exist within the ground truth bounding boxes, although the lablers of the CONTEXTDP dataset often felt that a *larger* portion of the screen should be considered to contain given DPs, usually due to negative space between text.

Answer to RQ_4: When examining *strict IoU*, AIDUI performs poorly as it tends to localize smaller areas of the screen. However, when considering *contained IoU*, we find that the areas of the screen that AIDUI localizes consistently fall within the ground truth for the DPs in CONTEXTDP.

E. RQ_5: Ablation Study of DP Analyses

Finally, we conduct an ablation study to examine the contribution of AIDUI'S textual, color and spatial analysis modules in detecting DPs. In this experiment, we remove one module at a time to understand the contribution of other modules. As stated earlier in section IV-F, *text analysis* serves as the foundation of our implemented approach. Hence, we include

TABLE VI
CONTRIBUTION OF DIFFERENT MODULES

	All			Mobile			Web		
	DP Instances: 301			DP Instances: 197			DP Instances: 104		
Modules	**Avg. Precision**	**Avg. Recall**	**Avg. F1-score**	**Avg. Precision**	**Avg. Recall**	**Avg. F1-score**	**Avg. Precision**	**Avg. Recall**	**Avg. F1-score**
Text	0.75	0.44	0.40	0.75	0.32	0.27	0.74	0.67	0.70
Text + Color	0.78	0.45	0.42	0.78	0.33	0.29	0.74	0.67	0.70
Text + Spatial	0.65	0.65	0.63	0.63	0.61	0.60	0.78	0.73	0.75
Text + Color + Spatial	0.66	0.67	0.65	0.63	0.61	0.60	0.82	0.77	0.79

the *text analysis* module in every combination of modules we use in our ablation study. We first start with using all three modules (*i.e., text* + *color* + *spatial*). In later steps, we removed *color* and *spatial* modules respectively. Finally, we end up with using *text analysis* only.

Answer to RQ$_5$: AIDUI achieves best results when all three, *i.e., text*, *color* and *spatial*, analysis modules are present. This suggests that there are orthogonal features that exist across data modalities that are useful for automated DP detection.

VI. RELATED WORK

A. Automated Detection of Dark Patterns

Raju *et al.* [44] proposed an intended design aimed to analyze the source code of a loaded web page to detect the advertisements that are of dark pattern types. Another work by Liu *et al.* [30] proposed a framework that leverages automated app testing with ad traffic identification approach to detect devious ad contents. Unlike these works, which are solely focused on detecting suspicious advertisements, our approach is aimed to detect a wide range of different types of dark patterns coming from both mobile and web applications.

B. Automated UI Understanding & Detection of Design Issues

Wu *et al.* introduced Screen Parsing [51], and Chen *et al.* introduced UIED [13], both of which use computer vision and deep learning tehcniques to segment and classify UI elements from screenshot pixels. Zhang *et al.* and Chen *et al.* developed approaches for screen content recognition [55] and icon labeling [12] that can automatically infer accessibility metadata from screen pixels. Moran *et al.* introduced REDRAW [36], which uses a combination of unsupervised computer vision and deep learning techniques to automatically prototype UI code for mobile apps from mock-ups. AIDUI differs from these above techniques in two major ways. First, the screen properties which are identified by AIDUI (DPs) are novel compared to properties inferred by past work (*e.g.,* accessibility data, screen elements). Second, due to the nature of these properties, AIDUI employs different analysis techniques. While a majority of the techniques above used end-to-end deep learning, this is difficult given that dark pattern examples need to manually sourced, making large scale data collection challenging. Thus, AIDUI processes visual and textual cues, as opposed to training a purely deep learning-based classification solution – making it largely *complementary* to existing work.

Additionally, the software engineering research community has been working towards identifying UI display issues across multiple types of software including web [32, 33], and mobile apps [31, 37, 53, 56]. These techniques tend to use a combination of deep learning and unsupervised computer vision techniques to detect varying types of display issues such as design guideline violations, internationalization issues, and deviations from design specifications. However, none of these techniques is capable of detecting dark patterns.

C. Ethics and Dark Patterns in UI/UX Design

A longstanding goal of the broader HCI research community has been to develop effective frameworks and guidelines to improve the UI/UX of applications [26, 39, 41–43]. Hence, investigating the ethical aspects of UI/UX from designer's perspectives is a growing area of interest in the HCI community and researchers have already explored a number of framings regarding ethics and values [8, 18–21, 46–48]. In this paper, we aim to build upon past research done in the topics of DPs and ethical UI/UX design by investigating how automated approaches for DP detection may equip developers with the tools necessary to better understand/avoid deceptive designs.

VII. THREATS TO VALIDITY

Internal & Construct Validity: One potential threat to validity is incorrect labels or bounding boxes in ContextDP. However, we mitigate this threat by following a rigorous labeling methodology with multiple author agreement. An additional threat is related to the completeness of AIDUI'S textual and visual cues. However, these cues were derived empirically through examining a small set of DP examples, and our experiments illustrate that they function reasonably well.
External Validity: We developed the current largest and most diverse set of fully localized DPs across two application domains (web + mobile). Despite this, there is a potential threat to validity that AIDUI may not generalize beyond the CONTEXTDP dataset.

VIII. CONCLUSION

In this paper, we have taken the first steps toward investigating the feasibility of automated detection of UI dark patterns in mobile and web UIs. We created unified taxonomy of DP categories and derived CONTEXTDP, the largest current fully labeled and localized DP instances. Furthermore, we implemented AIDUI, a fully-automated DP detection and localization technique, which performs well on ContextDP.

References

[1] Label studio https://labelstud.io/, 2022.

[2] Online replication package. https://anonymous.4open.science/r/AidUI-ICSE2023/, 2022.

[3] Opencv https://opencv.org/, 2022.

[4] Pytorch https://pytorch.org/, 2022.

[5] spacy https://spacy.io/, 2022.

[6] Torchvision https://pytorch.org/vision/stable/index.html, 2022.

[7] K. Asbury. Affinion group faces class action after paying out claims to ags, Feb 2014. Accessed on November 16, 2020.

[8] J. Bardzell and S. Bardzell. What is" critical" about critical design? In *Proceedings of the SIGCHI conference on human factors in computing systems*, pages 3297–3306, 2013.

[9] C. Bernal-Cárdenas, N. Cooper, K. Moran, O. Chaparro, A. Marcus, and D. Poshyvanyk. Translating video recordings of mobile app usages into replayable scenarios. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 309–321, New York, NY, USA, 2020. Association for Computing Machinery.

[10] C. Bösch, B. Erb, F. Kargl, H. Kopp, and S. Pfattheicher. Tales from the dark side: Privacy dark strategies and privacy dark patterns. *Proceedings on Privacy Enhancing Technologies*, 2016(4):237–254, 2016.

[11] H. Brignull, M. Miquel, J. Rosenberg, and J. Offer. Dark patterns - user interfaces designed to trick people. 2010.

[12] J. Chen, C. Chen, Z. Xing, X. Xu, L. Zhu, G. Li, and J. Wang. Unblind your apps: Predicting natural-language labels for mobile gui components by deep learning. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 322–334, New York, NY, USA, 2020. Association for Computing Machinery.

[13] J. Chen, M. Xie, Z. Xing, C. Chen, X. Xu, L. Zhu, and G. Li. Object detection for graphical user interface: old fashioned or deep learning or a combination? In *proceedings of the 28th ACM joint meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 1202–1214, 2020.

[14] G. Conti and E. Sobiesk. Malicious interface design: exploiting the user. In *Proceedings of the 19th international conference on World wide web*, pages 271–280, 2010.

[15] B. Deka, Z. Huang, C. Franzen, J. Hibschman, D. Afergan, Y. Li, J. Nichols, and R. Kumar. Rico: A mobile app dataset for building data-driven design applications. In *Proceedings of the 30th Annual Symposium on User Interface Software and Technology*, UIST '17, 2017.

[16] L. Di Geronimo, L. Braz, E. Fregnan, F. Palomba, and A. Bacchelli. Ui dark patterns and where to find them: a study on mobile applications and user perception. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2020.

[17] M. Fansher, S. S. Chivukula, and C. M. Gray. # darkpatterns: Ux practitioner conversations about ethical design. In *Extended Abstracts of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–6, 2018.

[18] M. Flanagan and H. Nissenbaum. *Values at play in digital games*. MIT Press, 2014.

[19] B. J. Fogg. A behavior model for persuasive design. In *Proceedings of the 4th international Conference on Persuasive Technology*, pages 1–7, 2009.

[20] C. Frauenberger, M. Rauhala, and G. Fitzpatrick. In-action ethics. *Interacting with Computers*, 29(2):220–236, 2017.

[21] B. Friedman and P. H. Kahn Jr. Human values, ethics, and design. *The human-computer interaction handbook*, pages 1177–1201, 2003.

[22] C. M. Gray, S. S. Chivukula, and A. Lee. *What Kind of Work Do "Asshole Designers" Create? Describing Properties of Ethical Concern on Reddit*, page 61–73. Association for Computing Machinery, New York, NY, USA, 2020.

[23] C. M. Gray, Y. Kou, B. Battles, J. Hoggatt, and A. L. Toombs. The dark (patterns) side of ux design. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–14, 2018.

[24] S. Greenberg, S. Boring, J. Vermeulen, and J. Dostal. Dark patterns in proxemic interactions: a critical perspective. In *Proceedings of the 2014 conference on Designing interactive systems*, pages 523–532, 2014.

[25] A. R. Johnson. Marketing firm agrees to $30 million settlement, Oct 2013. Accessed on November 16, 2020.

[26] R. Kimball and B. V. E. Harslem. Designing the star user interface. *Byte*, 7(1982):242–282, 1982.

[27] C. Lacey and C. Caudwell. Cuteness as a 'dark pattern'in home robots. In *2019 14th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 374–381. IEEE, 2019.

[28] C. Lewis. *Irresistible Apps: Motivational Design Patterns for Apps, Games, and Web-Based Communities*. Apress, USA, 1st edition, 2014.

[29] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[30] T. Liu, H. Wang, L. Li, X. Luo, F. Dong, Y. Guo, L. Wang, T. Bissyandé, and J. Klein. *MadDroid: Characterizing and Detecting Devious Ad Contents for Android Apps*, page 1715–1726. Association for Computing Machinery, New York, NY, USA, 2020.

[31] Z. Liu, C. Chen, J. Wang, Y. Huang, J. Hu, and Q. Wang. Owl eyes: Spotting ui display issues via visual understanding. In *Proceedings of the 35th IEEE/ACM International Conference on Automated Software Engineering*, ASE '20, page 398–409, New York, NY, USA, 2020. Association for Computing Machinery.

[32] S. Mahajan, A. Alameer, P. McMinn, and W. G. J. Halfond. Automated repair of internationalization presentation failures in web pages using style similarity clustering and search-based techniques. In *2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST)*, pages 215–226, 2018.

[33] S. Mahajan, B. Li, P. Behnamghader, and W. G. J. Halfond. Using visual symptoms for debugging presentation failures in web applications. In *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, pages 191–201, 2016.

[34] M. Maier and R. Harr. Dark design patterns: An end-user perspective. *Human Technology*, 16(2), 2020.

[35] A. Mathur, G. Acar, M. J. Friedman, E. Lucherini, J. Mayer, M. Chetty, and A. Narayanan. Dark patterns at scale: Findings from a crawl of 11k shopping websites. *Proceedings of the ACM on Human-Computer Interaction*, 3(CSCW):1–32, 2019.

[36] K. Moran, C. Bernal-Cárdenas, M. Curcio, R. Bonett, and D. Poshyvanyk. Machine learning-based prototyping of graphical user interfaces for mobile apps. *IEEE Transactions on Software Engineering*, 46(2):196–221, 2020.

[37] K. Moran, B. Li, C. Bernal-Cárdenas, D. Jelf, and D. Poshyvanyk. Automated reporting of gui design violations for mobile apps. In *Proceedings of the 40th International Conference on Software Engineering*, ICSE '18, page 165–175, New York, NY, USA, 2018. Association for Computing Machinery.

[38] C. Moser, S. Y. Schoenebeck, and P. Resnick. Impulse buying: Design practices and consumer needs. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–15, 2019.

[39] J. Nielsen. Enhancing the explanatory power of usability heuristics. In *Proceedings of the SIGCHI conference on Human Factors in Computing Systems*, pages 152–158, 1994.

[40] J. Nielsen. Usability inspection methods. In *Conference companion on Human factors in computing systems*, pages 413–414, 1994.

[41] J. Nielsen. Site design. *Designing Web Usability: The Practice of Simplicity. New Riders Publishing, Indianapolis*, pages 166–167, 1999.

[42] J. Nielsen and R. Molich. Heuristic evaluation of user interfaces. In *Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 249–256, 1990.

[43] J. Preece, Y. Rogers, H. Sharp, D. Benyon, S. Holland, and T. Carey. *Human-computer interaction*. Addison-Wesley Longman Ltd., 1994.

[44] S. H. Raju, S. F. Waris, S. Adinarayna, V. C. Jadala, and G. S. Rao. Smart dark pattern detection: Making aware of misleading patterns through the intended app. In S. Shakya, V. E. Balas, S. Kamolphiwong, and K.-L. Du, editors, *Sentimental Analysis and Deep Learning*, pages 933–947, Singapore, 2022. Springer Singapore.

[45] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015.

[46] P. Sengers, K. Boehner, S. David, and J. Kaye. Reflective design. In *Proceedings of the 4th decennial conference on Critical computing: between sense and sensibility*, pages 49–58, 2005.

[47] K. Shilton. Values levers: Building ethics into design. *Science, Technology, & Human Values*, 38(3):374–397, 2013.

[48] K. Shilton, J. A. Koepfler, and K. R. Fleischmann. How to see values in social computing: methods for studying values dimensions. In *Proceedings of the 17th ACM conference on Computer supported cooperative work & social computing*, pages 426–435, 2014.

[49] C. Utz, M. Degeling, S. Fahl, F. Schaub, and T. Holz. (un) informed consent: Studying gdpr consent notices in the field. In *Proceedings of the 2019 acm sigsac conference on computer and communications security*, pages 973–990, 2019.

[50] F. Westin and S. Chiasson. Opt out of privacy or "go home": Understanding reluctant privacy behaviours through the fomo-centric design paradigm. In *Proceedings of the New Security Paradigms Workshop*, NSPW '19, page 57–67, New York, NY, USA, 2019. Association for Computing Machinery.

[51] J. Wu, X. Zhang, J. Nichols, and J. P. Bigham. Screen parsing: Towards reverse engineering of ui models from screenshots. In *The 34th Annual ACM Symposium on User Interface Software and Technology*, UIST '21, page 470–483, New York, NY, USA, 2021. Association for Computing Machinery.

[52] M. Xie, S. Feng, Z. Xing, J. Chen, and C. Chen. *UIED: A Hybrid Tool for GUI Element Detection*, page 1655–1659. Association for Computing Machinery, New York, NY, USA, 2020.

[53] B. Yang, Z. Xing, X. Xia, C. Chen, D. Ye, and S. Li. Don't do that! hunting down visual design smells in complex uis against design guidelines. In *Proceedings of the 43rd International Conference on Software Engineering*, ICSE '21, page 761–772. IEEE Press, 2021.

[54] J. P. Zagal, S. Björk, and C. Lewis. Dark patterns in the design of games. 2013.

[55] X. Zhang, L. de Greef, A. Swearngin, S. White, K. Murray, L. Yu, Q. Shan, J. Nichols, J. Wu, C. Fleizach, A. Everitt, and J. P. Bigham. Screen recognition: Creating accessibility metadata for mobile applications from pixels. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, CHI '21, New York, NY, USA, 2021. Association for Computing Machinery.

[56] D. Zhao, Z. Xing, C. Chen, X. Xu, L. Zhu, G. Li, and J. Wang. Seenomaly: Vision-based linting of gui animation effects against design-don't guidelines. In *Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering*, ICSE '20, page 1286–1297, New York, NY, USA, 2020. Association for Computing Machinery.

[57] X. Zhou, C. Yao, H. Wen, Y. Wang, S. Zhou, W. He, and J. Liang. EAST: an efficient and accurate scene text detector. *CoRR*, abs/1704.03155, 2017.