Magic Layouts: Structural Prior for Component Detection in User Interface Designs

Dipu Manandhar[1], Hailin Jin[2] , John Collomosse[1,2]
[1]CVSSP, University of Surrey, — Guildford, UK
[2]Adobe Research, Creative Intelligence Lab — San Jose, CA.
d.manandhar@surrey.ac.uk, {hljin, collomos}@adobe.com

Abstract

We present Magic Layouts; a method for parsing screenshots or hand-drawn sketches of user interface (UI) layouts. Our core contribution is to extend existing detectors to exploit a learned structural prior for UI designs, enabling robust detection of UI components; buttons, text boxes and similar. Specifically we learn a prior over mobile UI layouts, encoding common spatial co-occurrence relationships between different UI components. Conditioning region proposals using this prior leads to performance gains on UI layout parsing for both hand-drawn UIs and app screenshots, which we demonstrate within the context an interactive application for rapidly acquiring digital prototypes of user experience (UX) designs.

1. Introduction

User interface (UI) layout is a critical component in user experience (UX) design. UI Layouts are commonly ideated and developed through sketched ('wireframe') designs, or by mocking up screenshots. Digital prototypes are then built using sequences of such layouts, to evaluate the UX and rapidly iterate on layout design. The ability to quickly move from such prototypes (sketches or screenshots) to digital prototypes in which components may be modified or rearranged, is valuable in expediting the design process.

This paper presents Magic Layouts; a technique for parsing existing UI layouts (for example wireframe sketches, or UI screenshots) into their UI components. Our technical contribution is a deep learning method for detecting UI components within UI layouts that exploits common spatial relationships of components as a learned prior knowledge to improve detection accuracy.

For example, UI elements often occur together and have a meaning underpinning that co-occurrence relationship. A 'text input field' and a 'button' occuring side-by-side in a UI is often a query-text and a response-button. We propose to explore the use of such co-occurrence information as an external knowledge graph to learn these component relationships, and incorporate this learning knowledge to boost the performance of state of the art detection algorithms.

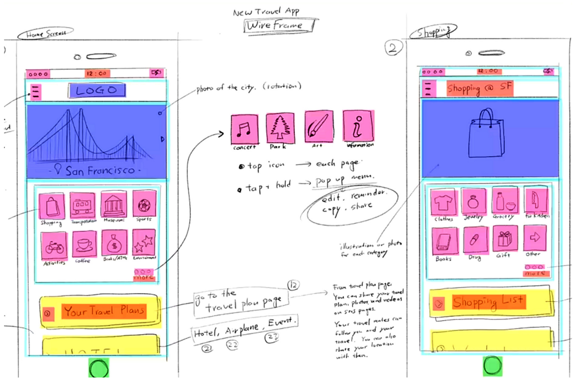


Figure 1. Magic Layouts parses UI layouts from sketched designs or app screenshots, exploiting learned prior knowledge of common component arrangements to improve recognition accuracy. In this parsed example, colour indicates different component classes.

We conduct experiments on two publicly available datasets of UI layouts; the RICO dataset of mobile app UX designs, and the DrawnUI dataset comprising hand-sketched UX wireframes. Our proposed approach yields improvements in detection for modalities, demonstrating that co-occurrences of UI components is a useful prior upon which to condition component detection and recognition when parsing UI layouts. We incorporate our detection model into an interactive tool dubbed 'Magic Layouts' capable of parsing UI layouts from mobile camera photographs of sketches (Fig. 1), or screenshots from mobile app stores. Additionally, Magic Layouts incorporates sketch based image search to replace sketched graphics with higher fidelity artwork.

2. Related Work

Detection and recognition of objects within images is a long-standing computer vision problem. Classical detectors include sliding-window approaches [39], super-pixel grouping [2, 32] and object proposal methods [1] often combined with sparse gradient features and dictionary learning for the recognition step. With the advent of deep learning, simultaneous object localization and detection was initially explored via semantic segmentation [23], and region-based convolutional neural networks (R-CNN [12, 30]) that classify a short-list of bounding boxes generated via selective search [32]. Region proposal networks (RPNs) were later fused with classifiers and trained end-to-end, to recognise candidates bounding boxes proposed with associated objectness scores in Faster-RCNN [30]. Improvements upon Faster-RCNN included RetinaNet mitigating foreground-background class imbalance [20], and Mask-RCNN to detect and classify arbitrary shaped object regions [15] (unlike UI components). All these approaches make decisions locally, without consideration of neighbouring regions or image structure. Recently, SGRN [36] aims to improve object proposal features by encoding spatially-related regions using Graph Neural Network (GNN). The SGRN graph encourages visual similarity and so spatially coherent labelling (*i.e.* biased towards connecting similar objects of the same class). This differs from our goal of modelling frequently co-occurring arrangements of objects from different classes; common in UX designs.

Layout has been studied from the perspective of synthesis, including automated reflow of banner adverts and graphic design [17], steered by gaze-tracking [35] or learned common design patterns [27, 28]. Aesthetic score prediction for document layout has been modelled [14] and used to drive automated layout decisions [9, 13]. UI Layouts specifically have been addressed through re-use of layouts via similarity search [22, 24] leveraging Rico; a crowd-annotated dataset [7] of mobile app screenshots. Most closely related are works that learn design heuristics to parse screenshots for layout re-use [38, 31] or for code generation [3]. All these techniques are bottom-up; driven by initial detection of individual UI components (*e.g.* via Faster-RCNN or edge-grouping heuristics [26]) which are post-processed and associated via learned (or designed) rules. Whilst we also parse UI layouts, our technical contribution is to enhance accuracy of that detection step by integrating a prior for component co-occurence at the initial step *i.e.* enhancing Faster-RCNN.

3. Methodology

We introduce *Magic Layouts* that exploits a learned structural prior for user interface (UI) layout parsing. Fig. 2 shows the architecture of the proposed framework. We propose to condition region proposals using a structural prior which essentially encodes common spatial co-occurrence among UI components distributed over various UX regions as knowledge graphs. To this end, we learn co-occurrence graphs from various UX regions and use high-level semantic representations that are readily available in the network to propagate them through the graphs. Representations from different regions are aggregated based on the proposal-graph associations. We show that such representations when integrated with original features offer more accurate UI parsing for both app screenshots as well as hand-drawn UI layouts (subsec. 4.4). The following sections describe our approach to learning the prior and how this information is embeded into the network.

3.1. Object Co-occurrence as Knowledge Graph

UX components usually co-exist together to form layout designs and have semantics underpinning the co-occurrence. We propose to explore such information and integrate this into detection frameworks to enhance UI detection. Formally, let $\{c_i\}_{i=1}^{C}$ be set of UI components C being the total number of component classes. We aim to obtain knowledge graphs $\boldsymbol{\mathcal{G}} = \{\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{E}}\}$ where the vertices $\boldsymbol{\mathcal{V}}$ represent UI classes and edges $\boldsymbol{\mathcal{E}}$ encode common spatial occurrence information from a training set, and use that prior for both training and inference.

The co-occurrence statistics among UI components vary across different regions of UX especially along the vertical direction - for example, the distribution statistics of components in the top of UX (often consisting of *Toolbar, Multi-tabs* etc.) may differ from that in the bottom which may consists of emphAdvertisementor *Exit* controls. Moreover, UXs are usually scrolled vertically and component relations are often within local regions. In view of this, we propose to estimate co-occurrence information in local regions of layouts. To achieve this, we divide the UI into several horizontal bands and observe frequency statistics of co-occurring classes. Note that the bands can be designed to be disjoint or overlapping in a sliding-window fashion. Let N_b be the number of bands that divide UX layout. We initialise $C \times C$ graph for each band with edge values $e_{mn} \in \boldsymbol{\mathcal{E}} = 0, \forall m, n \in \{1, \cdots, C\}$. A component is associated to a band if its center lies inside the upper and lower bounds of the band. We count a hit and increment the value of edge e_{mn} by 1 if two components from class m and class n both lie in the corresponding band. Algorithm 1 summarises co-occurrence graph computation. This process yields graphs corresponding to the various bands in the UX. The obtained graphs are row-column normalised $e_{mn} := \frac{e_{mn}}{\sqrt{\sum_n e_{mn} \sum_m e_{mn}}}$. Finally, we obtain $N_g (= N_b)$ graphs that carry component co-occurrence information distributed across various regions in UX layout. We leverage these knowledge graphs to condition the proposal features for better UI detection as described in following sections.

3.2. Semantic node representation

Our aim is to enrich the proposal representation with the learned co-occurrence knowledge graphs. We first need to define node features that would be propagated through the edges of the graphs. Regions and proposals are often represented using appearance features within an individual image [6, 4, 25]. However, such representations may not be robust when there are overlapping and nested objects that lead to heavy occlusions which is often observed in UI layouts. Moreover, visual ambiguities among various compo-

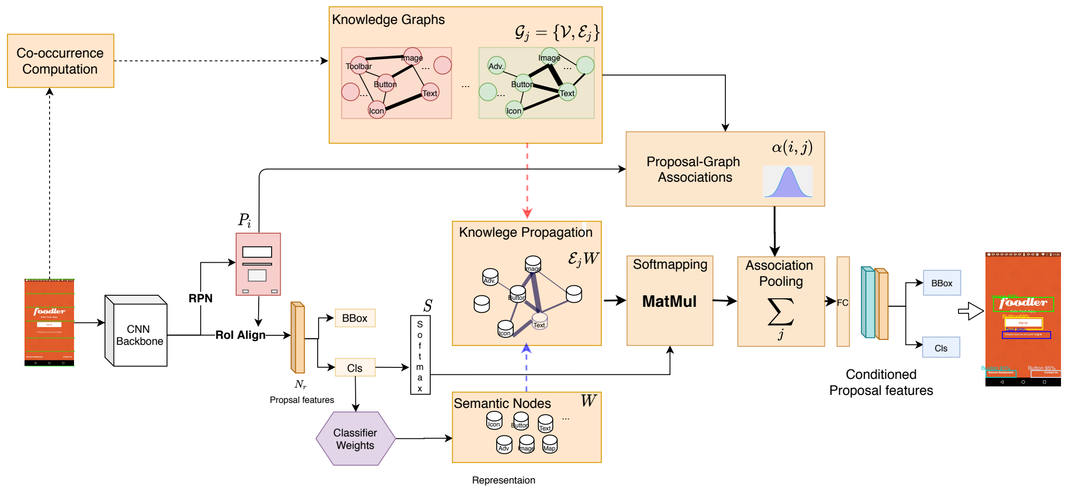


Figure 2. Magic Layout Architecture. Our framework exploits co-occurrence knowledge graphs (computed offline) as a prior to condition the proposal features. The semantic node representations obtained from the classifier are used to propagate features along the edges of knowledge graphs which are further soft-mapped to region features. Conditioned proposal features are then obtained by pooling the features based on proposal-graph associations eventually producing better UI detections.

Algorithm 1 Co-occurrence graph computation

```
    Input: $N$ Bounding boxes $bb = \{bb_i\}_{i=1}^N$ and their labels $L = \{l_i\}_{i=1}^N$ for all training UXs; Width $W_b$, number of bands $N_b = N_g$
    Output: Co-occurrence graphs $\{\mathcal{G}_j = (\mathcal{V}_j, \mathcal{E}_j)\}; j = \{1, \cdots, N_g\}$
 1: for all UX in training set do
 2:     Get UX height $H$
 3:     Compute Bands $\boldsymbol{S} = \{S_j\}_{j=1}^{N_b}$ with upper and lower bounds:
          (a) $Upper = \texttt{range}(0, H, H/N_b)$
          (b) $Lower = Upper + W_b$
 4:     Compute matrix $M$ s.t. $M[i,j] = 1$ if $bb_i$ in band $S_j$ else 0
 5:     $M := M[:, \texttt{sum}(M,0) > 1]$          ▷ Bands with co-occurrences
 6:     for all $bb_i$ in bb do
 7:         $S(bb_i) = \{S_j\} \in \boldsymbol{S}$ s.t. $M[i,j] \neq 0$   ▷ Band where $bb_i$ lies
 8:         $coind = \{i\}$ s.t. $M[i, S(bb_i)] \neq 0$     ▷ Co-occurrence index
 9:         $coind = \texttt{Unique}(coind)$               ▷ Remove duplicates
10:         Get class labels for $bb_i$ and $coind$: $L_i$ and $L_{coind}$
11:         $\mathcal{E}_j[L_i, L_{coind}] += 1$                 ▷ Update edges
12:     end for
13: end for
14: $e^j_{mn} := \frac{e^j_{mn}}{\sqrt{\sum_n e^j_{mn} \sum_m e^j_{mn}}}; \forall j$          ▷ Normalisation
15: $e^j_{mm} := 1 \; \forall m, j$
```

nents can lead to ineffective or even wrong propagation. Recently, few/zero-shot methods [33, 10] and object recognition [36, 37] have used the classifier weights as a visual embedding for unseen classes and the proposal's latent representation to guide recognition. Motivated by this, we use classifier weights as semantic node feature of the graphs. In particular, to obtain this representation, we copy the weights of the previous classifier head of the base network including the bias *i.e.* $\mathbf{W} \in \mathbb{R}^{C\times(D+1)}$ where D is input dimension to the classifier head and C is the total number of UI classes. The use of this representation comes with three main advantages: (i) the representation captures high-level semantics which acts as class embedding for each category, (ii) they are readily available without requiring computationally expensive feature averaging or clustering over large samples [18], and (iii) the representations are dynamically updated during training thus they improve over time.

3.3. Knowledge graph-based proposal conditioning

The co-occurrence knowledge graphs contain component relationship information across different regions of UX layouts. We associate each proposal to the learned knowledge graphs in order to propagate their representations through their respective edges. A natural rule of proposal-to-graph assignment can be associating each proposal to its nearest band (and hence the corresponding graph) or to the band that encloses the proposal. However, this single hard-assignment may be too strict and can be noisy as proposal boxes are only initial estimates of objects which are essentially regressed for the final predictions. Thus, we propose to assign proposals to multiple graphs in a weighted manner; as a Gaussian function of their spatial proximities.

Formally let $\{P_i\}_{i=1}^{N_r}$ be N_r region proposals with features f and bounding boxes $\{x_{i1}, y_{i1}, x_{i2}, y_{i2}\}_{i=1}^{N_r}$. Similarly, we have $\{\boldsymbol{\mathcal{G}}_j\}_{j=1}^{N_g}$ knowledge graphs related to $N_b (= N_g)$ bands from different regions of UX (sec. 3.1). We compute the association $\alpha(i,j)$ between proposal P_i to graph $\boldsymbol{\mathcal{G}}_j$ using the following equations

$$\alpha(i,j) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp^{-\frac{1}{2}\left(\frac{\delta^j_i - \mu}{\sigma}\right)^2}, \tag{1}$$

$$\delta^j_i = \frac{yc_i - yb_j}{H}; \tag{2}$$

where δ^j_i is the vertical displacement between the proposal P_i and band j; and σ and μ are the parameters of Gaussian

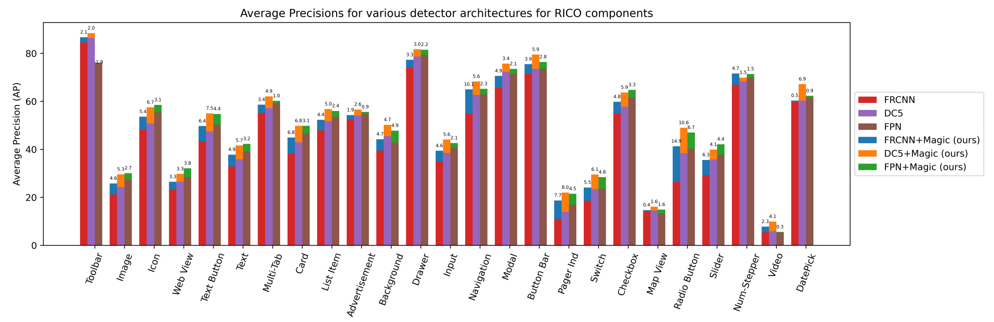


Figure 3. Average precision (AP) for RICO UI categories for baselines and the proposed method using various architectures: Faster-RCNN (FRCNN) [30], DC5 [19] and FPN [19]. For each network, the AP obtained using our proposed method are shown as stacked bars over their corresponding baselines where the figures on tops show the absolute improvements for each category (zoom-in for best view).

Table 1. **Performance comparison on RICO dataset**

Method	AP	AP50	AP75	APs	APm	APl	AR	AR	AR	ARs	ARm	ARl
@IoU	0.5:95	0.5	0.75	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95
maxDets	100	100	100	100	100	100	1	10	100	100	100	100
Faster-RCNN [30]	43.0	52.8	46.0	2.3	16.5	43.2	40.5	58.1	60.1	5.7	29.2	60.0
RetinaNet [20]	41.7	52.8	45.8	**3.6**	20.5	42.0	38.8	57.0	59.1	7.2	34.7	58.8
FRCNN+SGRN [36]	45.2	54.2	47.8	2.4	19.0	45.8	41.4	60.0	62.1	6.4	32.3	62.0
FRCNN+Magic	**47.8**	**57.4**	**51.0**	3.1	**21.7**	**48.4**	**42.6**	**61.9**	**63.9**	**6.9**	**34.9**	**63.9**
DC5 [19]	46.7	56.2	49.8	2.9	20.5	47.2	41.9	60.7	62.7	6.7	33.0	62.7
DC5+SGRN [36]	49.0	58.0	51.8	**4.9**	27.5	49.6	43.0	61.8	63.9	**8.4**	40.5	63.9
DC5+Magic	**51.8**	**61.1**	**54.9**	4.4	**29.9**	**52.2**	**44.8**	**64.8**	**66.7**	8.1	**41.6**	**66.9**
FPN [19]	47.6	57.1	50.4	4.6	30.6	47.7	41.6	61.0	63.1	9.5	44.6	62.6
FPN+SGRN [36]	49.9	59.6	52.7	7.6	33.5	50.0	42.6	62.4	64.5	**13.5**	45.8	64.0
FPN+Magic	**50.3**	**60.1**	**53.4**	**8.4**	**34.7**	**50.2**	**43.0**	**63.0**	**65.0**	13.2	**46.4**	**64.5**

distribution. Similarly, yc_i and yb_j are y-components of centriods of the proposal and the band given by $yc_i = (y_{i1} + y_{i2})/2$ and $yb_j = (y_{j,uppper} + y_{j,lower})/2$ respectively; and H is the height of the UX which normalises the displacement taking care of varying UX dimensions. We further normalise the association values $\alpha(i,j) := \frac{\alpha(i,j)}{\sum_j \alpha(i,j)}$.

We enhance proposal features using priors from different layout regions taking their associations into account. We propagate the node representations $\mathbf{W} \in \mathbb{R}^{C\times(D+1)}$ via the graphs $\mathcal{G}_j$ using prior knowledge in edges $\boldsymbol{\mathcal{E}}_j$ given by $\boldsymbol{\mathcal{E}}_j\mathbf{W}$. This allows to share high-level semantics between related UI categories according to the graph knowledge. Next we map this semantic level information into individual proposal by a category-to-region mapping to condition them. A direct mapping for each region to a class can be obtained based on prediction of the previous classifier. However, such one-to-one mapping can be harsh as it is prone to noise due to false predictions. In this paper, we instead propose to use a soft-mapping strategy which operates on probability distributions of proposals over all classes. Concretely we compute a mapping matrix $\mathbf{S} \in \mathbb{R}^{Nr\times C}$ given by $s_{ij} = \frac{\exp p_{ij}}{\sum_j \exp p_{ij}}$ where s_{ij} is the probability that proposal P_i belong to class c_j. Using these definitions, we obtain conditioned features as follows.

$$f_i' = \sum_j^{Nb} \alpha(i,j) \odot \mathbf{S}\boldsymbol{\mathcal{E}}_j\mathbf{W}\mathbf{Z}_e, \tag{3}$$

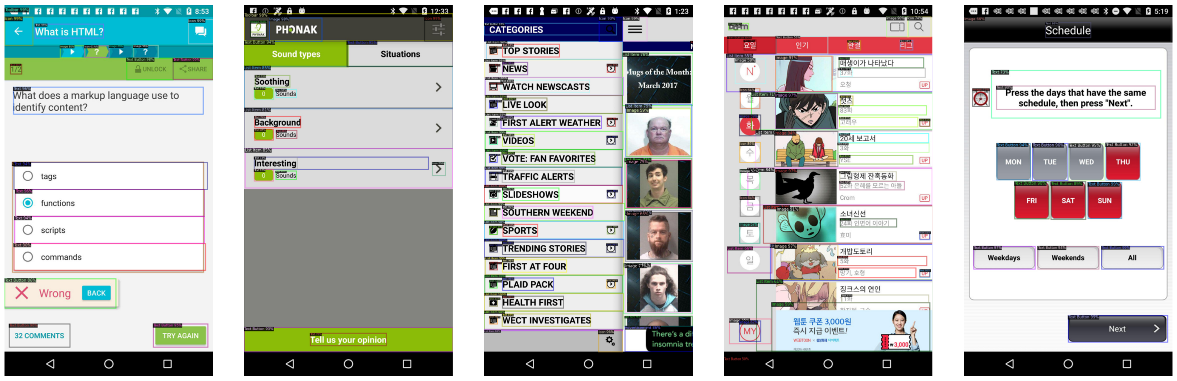
where $\odot$ is scalar element-wise multiplication, and $\mathbf{Z}_e \in \mathbb{R}^{(D+1)\times D'}$ is weight of the final embedding layer that $D'-$dimension proposal features conditioned on the structural priors. Note these proposal features are computed by aggregating all common spatial co-occurrence information from various regions from UX layout. We concatenate this conditioned representation with original feature $f = [f, f']$ and pass them through a final classifier head and a bounding-box regression head to obtain better detection results based on the conditioned proposal features.

4. Experiments and Discussion

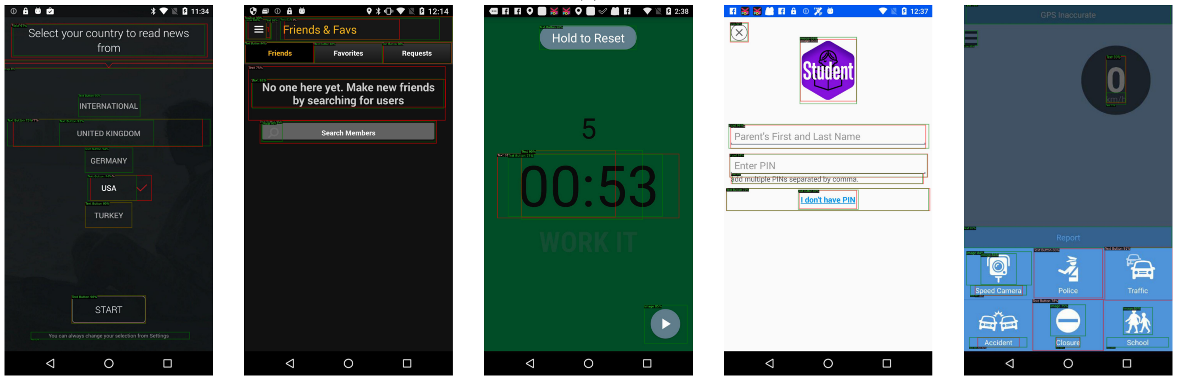
We evaluate the performance of Magic Layouts and contrast state-of-the-art baselines such as (Faster-RCNN [30] and popular variants [19], and RetinaNet [20]) as well as the recent spatial-aware graph network (SGRN) [36].

4.1. Datasets

We evaluate on images from two input modalities: (i) mobile app screenshots and (ii) hand-sketched UI designs.
RICO dataset [7] is the largest publicly available dataset of UX designs containing 66K screenshots of mobile apps curated by crowd-sourcing and mining 9.3K free Android apps. The screenshots are annotated using bounding boxes to create semantic view hierarchies which are each assigned to one of $C = 25$ classes of user interface (UI) component. We partition the dataset into 53K training/validation samples $\mathcal{T}$ and a test set of 13K layouts for inference.
DrawnUI dataset [8] contains 2,363 images of hand-drawn sketches released as development set of ImageCLEF 2020



(a)



(b)

Figure 4. UX parsing on RICO mobile screenshots [7]. (a) Examples of UX parsing using MagicLayouts (Magic+FPN) (b) Comparison: Standard FPN (shown in red boxes) vs. MagicLayouts (green boxes). Our method is able to recognise components with higher confidence with lower false detections. Zoom-in for better view.

drawnUI recognition task. The main motivation of this dataset is to enable designers to build UX layout by drawing them on whiteboard or on paper. The idea is to develop automatic UI parsing algorithm that can be further leveraged to convert them into UX codes. Each image is annotated for UI components with their bounding boxes and class labels from a set of $C = 21$ predefined UI classes. We partition the dataset into a training set of 2000 images and perform evaluation on the remaining 263 images.

4.2. Evaluation Metrics

For both datasets, we report performance metrics used in COCO detection evaluation criterion [21] and provide mean Average Precision (AP) across various IoU thresholds *i.e.* IoU = $\{0.50 : 0.95, 0.5, 0.75\}$ and various scales: $\{$small, medium and large$\}$. We also report Average Recall (AR) with different number of detection - $\{1, 10, 100\}$ and scales: $\{$small, medium and large$\}$. Unless specified, we refer mAP@[0.50:0.95] to as mAP (primary metric) and AR@[0.50:0.95] as AR for conciseness.

4.3. Experimental Settings

4.3.1 Architectures

We conduct experiments with widely adopted backbone network (ResNet [16]) and best-performing detectors to demonstrate the effectiveness and generality of the proposed method. In particular, we build our method using three popular variants of the Faster-RCNN architecture: **(i) Faster-RCNN** [30], **(ii) Dilated Convolutional Network (DC5)** [19], and **(iii) Feature Pyramid Network (FPN)** [19]. For Faster-RCNN [30], following the standard practise [16], we compute region proposals on top of *conv4*, and all layers of *conv5* are adopted as predictor head with two sibling layers for classification and regression. The DC5 architecture uses dilated convolution in *conv5* layers and compute region proposals and perform RoI pooling over *conv5* features. As the prediction head, DC5 uses 2-*fc* MLP followed by the two siblings layers which is lighter weight and faster than the *conv5* head [19]. FPN [19] has an alternate backbone where top-down and lateral connections are used to build a pyramid of features. Proposals are computed from all the pyramid scales and RoI pooling is performed on the most appropriate scale based on size of each proposal. FPN achieves the best speed and accuracy trade-off when compared to Faster-RCNN and DC5 [19]. We use regional proposal network (RPN) to generate proposals and RoIAlign [15] is used for pooling the region features from feature maps. We show that all three architectures benefit from our spatial prior for object co-occurrence (subsec. 4.4).

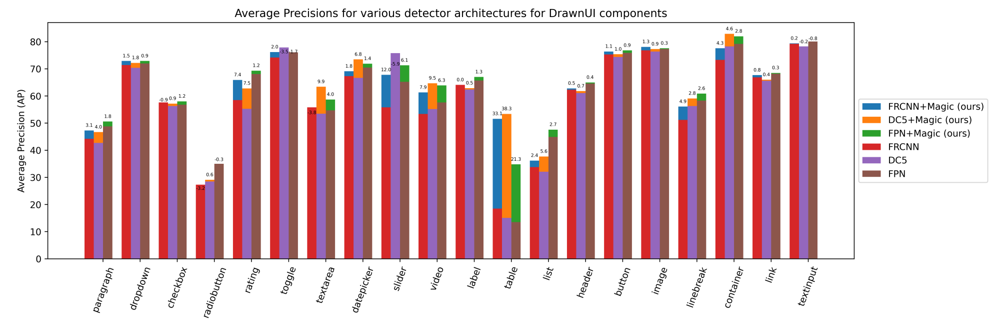


Figure 5. Average precision (AP) for DrawnUI categories for baselines and the proposed method using Faster-RCNN (FRCNN) [30], DC5 [19] and FPN [19]. See Fig. 3 for details. Zoom-in for best view

Table 2. **Performance comparison on DrawnUI dataset.**

Method	AP	AP50	AP75	APs	APm	APl	AR	AR	AR	ARs	ARm	ARl
@IoU	0.5:95	0.5	0.75	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95	0.5:95
maxDets	100	100	100	100	100	100	1	10	100	100	100	100
Faster-RCNN [30]	58.6	86.3	65.7	21.7	51.5	60.8	27.7	61.7	65.5	35.2	58.3	67.0
RetinaNet [20]	58.6	85.5	66.1	19.0	52.9	60.6	29.9	63.9	68.1	23.6	60.1	69.0
FRCNN+SGRN [36]	61.4	87.9	71.5	27.0	**56.9**	63.5	29.0	64.7	68.4	34.7	**63.6**	69.6
FRCNN+Magic	**62.2**	**88.5**	**72.8**	**27.2**	55.5	**64.9**	**30.4**	**66.4**	**70.1**	**33.6**	62.2	**71.5**
DC5 [19]	59.1	85.4	69.3	23.6	51.3	61.2	28.1	62.6	66.3	29.9	58.9	67.1
DC5+SGRN [36]	62.5	**90.2**	70.7	26.4	55.1	65.0	30.0	65.8	69.5	**31.6**	61.4	70.5
DC5+Magic	**63.4**	89.9	**72.9**	**26.9**	**56.9**	**65.9**	**30.7**	**66.8**	**70.6**	31.1	**63.1**	**71.6**
FPN [19]	61.6	87.3	70.6	32.1	57.3	63.5	28.9	64.5	68.6	36.9	64.3	69.3
FPN+SGRN [36]	63.3	88.6	73.7	**34.6**	**58.1**	65.8	30.1	66.3	70.5	**39.1**	**64.6**	71.4
FPN+Magic	**64.3**	**89.5**	**74.4**	32.2	54.4	**66.5**	**30.3**	**66.7**	**71.0**	37.1	64.0	**71.7**

4.3.2 Implementation Details

We implement our framework using Pytorch [29] with detectron2 [34] codebase. We use ResNet50 [16] pretrained on ImageNet as our backbone network. Images are resized such that shorter side has maximum of 800 pixels and larger side has 1333 pixels. For all settings, we sample $Nr = 256$ proposals from each image after non-maximal suppression (NMS) which are assigned as positive if the proposal and a ground-truth box has IoU > 0.7 or as negative if the IoU < 0.3. We follow other standard settings as in [19].

We use SGD optimizer with a momentum update of 0.9 and a weight decay of 0.0001; and set the initial learning of 0.02 and decay it by a factor of 0.1 twice during training. We use 3 images per GPU and a mini-batch of 9 for training. We train all three network architectures for 21 epochs and 45 epochs for RICO and DrawnUI dataset respectively. We use more epochs for DrawnUI as it has fewer UXs compared to RICO. We observe that the performances saturates on validation data after 16 and 31 epoch for RICO and DrawnUI respectively; further training does not improve the performance. To obtain the conditioned proposal features in our framework, we initialise our model using pre-trained networks from their respective architectures.

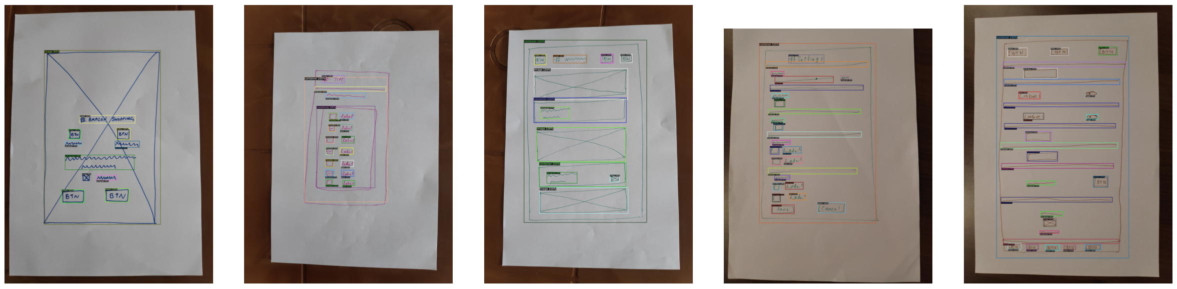
We conducted experiments using both non-overlapping as well as overlapping bands and observed that structural priors estimated from both achieve similar performances. Hence we opt for simplicity and divide UXs into $N_b (= N_g) = 10$ non-overlapping bands for all experiment unless stated. We also study impact of Ng on performance (subsec. 4.4.3). We used a zero-mean Gaussian distribution with a standard deviation empirically set to $\sigma = 0.3$ in order to compute the associations of proposals to the bands. The dimension of conditioned feature D' is set to 512.

4.4. Results

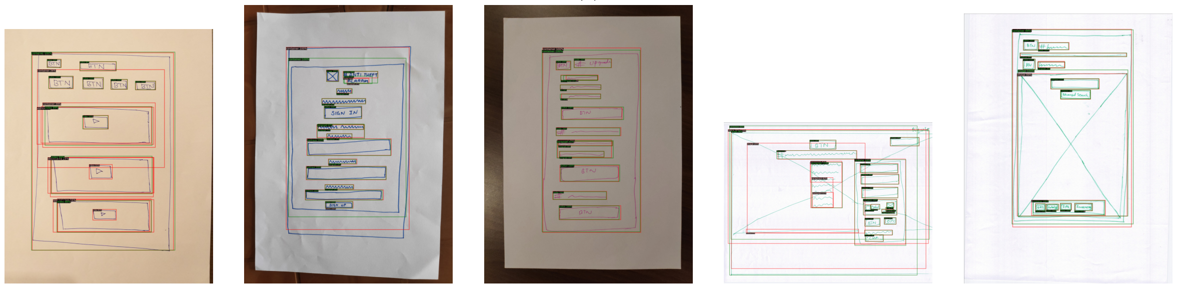
4.4.1 Parsing UX screenshots on RICO

Fig. 3 shows average precision (AP) using the proposed method and compares with baselines with various network architectures illustrating the improvements brought by Magic Layouts over various RICO UI categories. From the figure it can be observed that for all the architectures, our proposed method is able to boost the performance (numbers on top of each stacked bar, Fig. 3) of nearly all UI categories. For example, our method is able to deliver improvements of +7.7%, +8.0% and +4.5% AP for *Page Indicator* class over FasterRCNN, DC5 and FPN method respectively. This clearly demonstrates the advantage of incorporating structural priors of layout in UI recognition.

Table 1 summarizes recognition performance in terms of mean Average Precision (mAP) at different IoU thresholds and scales and Average Recall (AR) at different number of detections and scales. Faster-RCNN [30] achieves an mAP of 43.0% whilst our proposed framework achieves an mAP of 47.8% - an absolute increase of +4.8% indicating the effectiveness of the proposed method and the use of structural prior. Our method also outperforms competing methods such as RetinaNet [20] and SGRN-FRCNN [36] by mAPs of 6.1% and 2.6% respectively. The recent SGRN method aims to enhance proposal representation, however, it assumes homogeneous object relations across various regions of image and creates edges based on visual similarity of proposals which can potentially limit the exploration



(a)



(b)

Figure 6. UX parsing on hand-sketched DrawnUI layouts [8]. (a) Examples of UX parsing with MagicLayouts (Magic+FPN). (b) Comparison: Standard FPN (shown in red boxes) vs. MagicLayouts (green boxes). See caption of Fig 4

of inter-class occurrences. Compared to this, our method aggregates structural priors from various regions explicitly considering the variability in UI distributions and conditions the proposals at a semantic level, improving accuracy.

Magic Layouts with DC5 architecture achieves an mAP of 51.8% outperforming its counterpart by a margin of 5.1% and SGRN-DC5 by 2.7%. Similarly, Magic-Layout with FPN architecture achieves 50.3% AP /ie +2.7% over the standard FPN [19] and +0.4% over SGRN [36]. From Table 1, we can observe similar improvements for Average Recall and related metrics; for example MagiLayouts with Faster-RCNN achieves 63.9% AR@100 outperforming its counterpart by +3.8% and SGRN by +1.8%. In a nutshell, the proposed method offers benefits for various detector architectures and outperforms existing approaches that also incorporate relations among components.

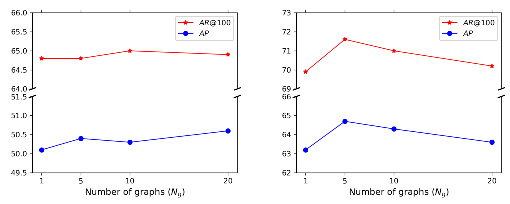
Fig. 4 shows sample qualitative UX parsed on RICO using our Magic-Layout (with FPN arch). In Fig. 4(a) we observe that Magic-Layout is able to detect and recognize various UI components at different scales; in (b) we compare MagicLayouts with baseline and show our method is more effective when compared to baselines.

4.4.2 Parsing hand-sketched UXs

We present evaluations on the DrawnUI dataset [8] and show that Magic-Layouts can effectively detect components on hand-drawn wire-frames while outperforming all baselines. Fig. 5 shows improvements in average precision (AP) obtained by the proposed method for DrawnUI components using the three architectures. Our method provides consistent improvements for almost all component classes providing an average boost of 3.5% AP over the three architectures which clearly shows the advantage of incorporating co-occurrence prior into the framework. We also observed that categories which are comparatively rarer are largely benefited by the structural prior; *e.g.* table component has less than 50 instances in DrawnUI dataset; and hence detectors may perform poorly for such rare classes. Faster-RCNN, DC5 and FPN achieve APs of 18.5%, 15.1% and 13.5% respectively for this class. Magic-Layouts boost these performance to 51.6%, 53.4% and 34.8% respectively for the three detectors demonstrating its effectiveness.

Table 2 presents mAPs and Average Recall for DrawnUI dataset obtained using the proposed method and compares with different architectures and existing methods. Magic-Layout with Faster-RCNN network achieves an mAP of 62.2% which outperforms its counterpart [30] by 3.6%. It also outperforms RetinaNet and spatially-aware SGRN by 3.6% and 0.8% respectively. MagicLayouts with FPN achieves the best mAP of 64.3% on this dataset.

In terms of recall, our method achieves the best AR@100 of 63.9%, 66.7% and 65% for FRCNN, DC5 and FPN respectively. Overall, the proposed Magic Layouts provides improvements over all baseline Faster-RCNN variants [30, 19] and also outperforms other baseline methods using spatial relations [36, 20]. Fig. 6 shows sample example of parsed UXs from DrawnUI dataset.Our method is able to better detect and recognize UIs at different scales despite potential variations in hand-sketches and illuminations.



(a) RICO (b) DrawnUI

Figure 7. Performance of FPN+Magic on (a) RICO [7] and (b) DrawnUI [8] at various values of Ng = $\{1, 5, 10, 20\}$.

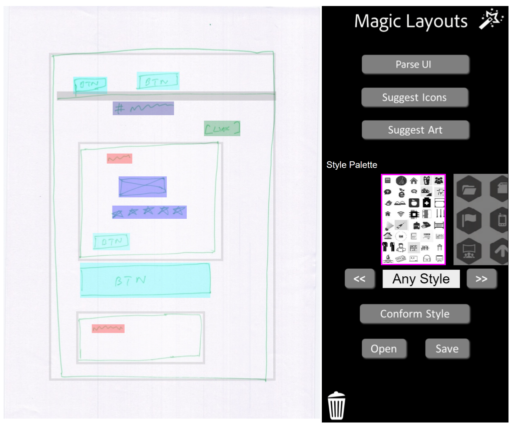


Figure 8. The Magic Layouts web app parsing a UI wireframe sketch from the DrawnUI dataset [8]; coloured regions indicate different classes of recognised UI component. UX sketches are rapidly converted into higher digital prototypes. The tool also incorporates a sketch based visual search to replace sketched artwork with higher fidelity graphics.

4.4.3 Ablation Studies

Impact of N_g: We conduct experiments using different number of graphs (N_g) during co-occurrence computations. Fig 7 shows the performance of the proposed method in term of AR@100 and AP[0.5:0.95] for different values of N_g. Our method achieves similar performances for different N_g values with deviations about 0.5% and 1% for RICO and DrawnUI respectively indicating that our method is fairly insensitive to the parameter. $N_g = 10$ provides good trade-off between AP and AR for both datasets.

Design choice study: We conduct experiments with various strategies for graph-proposal association and category-to-region mapping. In particular, we run experiment with following setups: A. Baseline, B. graph-proposal association: Single-Assignment vs. Equal vs. Gaussian Weighting, C. Category-to-region mapping: Soft vs. Hard/1-to-1, D. Graph node: classifier's weight vs. proposals features). Tables 3 and 4 summarise the performances for DrawnUI and RICO respectively; both datasets follow the same trend. For graph-proposal association, multiple assignment using Gaussian weighting performs the best outperforming single graph as well equal weighting scheme. Assigning proposals appropriately to their corresponding bands also provides on-par performance for some metrics. For category-to-region mapping, the proposed soft-mapping outperforms one-to-one hard mapping for both single and Gaussian-weighted assignments. We further conduct an extra study, substituting $\mathbf{W}$ with representations computed from proposal features ($\mathbf{P} = \mathbf{FS} \in \mathbb{R}^{C \times D}$); we achieve on-par performances (Table 3, 4-D & E) indicating that our method works well with alternative choices of node features.

4.5. Practical Use Case: Magic Layouts

We deployed our proposed detection model into an interactive 'Magic Layouts' web app capable of parsing UI layouts from mobile camera photographs of sketches (Fig. 8), or screenshots from mobile app stores. Magic Layouts incorporates sketch based image search ([5]) to replace sketched graphics and icons with higher fidelity artwork. Please see supplementary video demo.

Table 3. **Ablation studies DrawnUI**

Config/ Method		Nb	Assig-nment	Map-ping	Node Feat	**AP**	**AP50**	**AR @1**	**AR @10**
A	FPN	-	-	-	-	61.6	87.3	28.9	64.5
B	Magic	1	Single	Soft	**W**	63.2	88.8	29.0	65.6
	Magic	10	Equal	Soft	**W**	62.7	87.6	28.2	66.0
	Magic	10	Single	Soft	**W**	64.2	89.4	30.5	**67.2**
C	Magic	10	Single	Hard	**W**	63.5	89.8	29.4	66.6
	Magic	10	Gauss	Hard	**W**	63.9	88.7	30.0	66.6
D	Magic	10	Gauss	Soft	**P**	63.6	**90.4**	**30.5**	66.9
E	Magic	10	Gauss	Soft	**W**	**64.3**	89.5	30.3	66.7

Table 4. **Ablation studies on RICO**

Config/ Method		Nb	**Assig-nment**	**Map-ping**	**Node Feat**	**AP**	**AP50**	**AR @1**	**AR @10**
A	FPN	-	-	-	-	47.6	57.1	41.6	61.0
B	Magic	1	Single	Soft	**W**	50.1	**60.1**	42.8	62.8
	Magic	10	Equal	Soft	**W**	50.1	59.8	42.6	62.5
	Magic	10	Single	Soft	**W**	50.2	60.0	42.7	62.5
C	Magic	10	Single	Hard	**W**	50.3	**60.1**	**43.0**	**63.0**
	Magic	10	Gauss	Hard	**W**	49.9	59.9	42.8	62.6
D	Magic	10	Gauss	Soft	**P**	**50.4**	**60.1**	42.9	62.9
E	Magic	10	Gauss	Soft	**W**	50.3	**60.1**	**43.0**	**63.0**

5. Conclusion

We reported Magic Layouts; a technique for incorporating structural layout (common spatial object co-occurrences) as a prior to guide object detection and localization, investigating this in the context of UI layout parsing. We extended the Faster-RCNN backbone to incorporate a learned prior based on spatial distribution of UI components, showing performance improvements over several Faster-RCNN variants [30, 11], RetinaNet [20] and including an existing graph based approach to learning spatial prior for detection [36]. We demonstrated the utility of our model within 'Magic Layouts' – a UX parsing tool capable of automatically parsing UI layouts in two domains: app screenshots and free-hand sketched prototypes. Future work could explore not only object co-proximities but also hierarchical relationships which particularly for UI layouts could offer further structural cues.

References

[1] B. Alexe, T. Deselaers, and V. Ferrari. Measuring the objectness of image windows. *IEEE Trans. Pattern Analysis and Machine Intelligence TPAMI)*, 2012. 1

[2] P. Arbelaez, J. Pont-Tuset, J. T. Barron, F. Marques, and J. Malik. Multiscale combinatorial grouping. In *Proc. CVPR*, 2014. 1

[3] T. Beltramelli. pix2code: Generating code from a graphical user interface screenshot. *arXiV 1705.07962v2*, 2017. 2

[4] Xinlei Chen, Li-Jia Li, Li Fei-Fei, and Abhinav Gupta. Iterative visual reasoning beyond convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7239–7248, 2018. 2

[5] J. Collomosse, T. Bui, M. Wilber, C. Fang, and H. Jin. Sketching with style: Visual search with sketches and aesthetic context. In *Proc. ICCV*, 2017. 8

[6] Bo Dai, Yuqi Zhang, and Dahua Lin. Detecting visual relationships with deep relational networks. In *Proceedings of the IEEE conference on computer vision and Pattern recognition*, pages 3076–3086, 2017. 2

[7] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha 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. 2, 4, 5, 8

[8] Dimitri Fichou, Raul Berari, Paul Brie, Mihai Dogariu, Liviu Daniel Ştefan, Mihai Gabriel Constantin, and Bogdan Ionescu. Overview of ImageCLEFdrawnUI 2020: The detection and recognition of hand drawn website uis task. In *CLEF2020 Working Notes*, CEUR Workshop Proceedings, Thessaloniki, Greece, September 22-25 2020. CEUR-WS.org <http://ceur-ws.org>. 4, 7, 8

[9] J. Geigel and A. Loui. Automatic page layout using genetic algorithms for electronic albuming. In *Proc. Electronic Imaging*, 2001. 2

[10] Spyros Gidaris and Nikos Komodakis. Dynamic few-shot visual learning without forgetting. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4367–4375, 2018. 3

[11] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2015. 8

[12] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proc. CVPR*, 2014. 2

[13] E. Goldenbert. Automatic layout of variable-content print data. Master's thesis, School of Cognitive & Computing Sciences, University of Sussex, UK, 2000. 2

[14] S. Harrington, J. Naveda, R. Jones, P. Roetling, and N. Thakkar. Aesthetic measures for automated document layout. In *Proc. ACM Document Eng.*, 2004. 2

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 2, 5

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 5, 6

[17] N. Hurst, W. Li, and K. Marriott. Review of automatic document formatting. In *Proc. ACM Document Eng.*, 2009. 2

[18] Kuang-Huei Lee, Xiaodong He, Lei Zhang, and Linjun Yang. Cleannet: Transfer learning for scalable image classifier training with label noise. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5447–5456, 2018. 3

[19] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125, 2017. 4, 5, 6, 7

[20] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017. 2, 4, 6, 7, 8

[21] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. 5

[22] Thomas F. Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. Learning design semantics for mobile apps. In *The 31st Annual ACM Symposium on User Interface Software and Technology*, UIST '18, pages 569–579, New York, NY, USA, 2018. ACM. 2

[23] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 2015. 2

[24] D. Manandhar, D. Ruta, and J. Collomosse. Learning structural similarity of user interface layouts using graph networks. In *Proc. ECCV*, 2020. 2

[25] Kenneth Marino, Ruslan Salakhutdinov, and Abhinav Gupta. The more you know: Using knowledge graphs for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2673–2681, 2017. 2

[26] K. Moran, C. Bernal-Cardenas, M. Curcio, R. Bonett, and D. Poshyvanyk. Machine learning-based prototyping of graphical user interfaces for mobile apps. *IEEE Trans. Soft. Eng.*, 2018. 2

[27] P. O'Donovan, A. Agarwala, and A. Hertzmann. Learning layouts for single-page graphic designs. *IEEE Transactions on Visualization and Computer Graphics*, 2014. 2

[28] P. O'Donovan, A. Agarwala, and A. Hertzmann. Designscape: Design with interactive layout suggestions. In *Proc. ACM Conf. Human Factors in Comp. Sys.*, pages 1221–1224, 2015. 2

[29] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in pytorch. 2017. 6

[30] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99, 2015. 2, 4, 5, 6, 7, 8

[31] A. Swearngin, M. Dontcheva, W. Li, J. Brandt, M. Dixon, and A. Ko. Rewire: Interface design assistance from examples. In *Proc. ACM CHI*, 2018. 2

[32] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *Intl. Journal Computer Vision (IJCV)*, 2013. 1, 2

[33] Xiaolong Wang, Yufei Ye, and Abhinav Gupta. Zero-shot recognition via semantic embeddings and knowledge graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6857–6866, 2018. 3

[34] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 6

[35] R. Lau X. Pang, Y. Cao and A. Chan. Directing user attention via visual flow on web designs. In *Proc. ACM SIGGRAPH*, 2016. 2

[36] Hang Xu, Chenhan Jiang, Xiaodan Liang, and Zhenguo Li. Spatial-aware graph relation network for large-scale object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9298–9307, 2019. 2, 3, 4, 6, 7, 8

[37] Hang Xu, ChenHan Jiang, Xiaodan Liang, Liang Lin, and Zhenguo Li. Reasoning-rcnn: Unifying adaptive global reasoning into large-scale object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6419–6428, 2019. 3

[38] X. Yang, E. Yumer, P. Asente, M. Kraley, D. Kifer, and C. Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. In *Proc. CVPR*, pages 5315–5324, 2017. 2

[39] C. L. Zitnick and P. Dollar. Edge boxes: Locating object proposals from edges. In *Proc. ECCV*, 2014. 1