LayoutGMN: Neural Graph Matching for Structural Layout Similarity

Akshay Gadi Patil[1] Manyi Li[1] Matthew Fisher[2] Manolis Savva[1] Hao Zhang[1]
[1]Simon Fraser University [2]Adobe Research

Abstract

We present a deep neural network to predict structural similarity *between 2D layouts by leveraging Graph Matching Networks (GMN). Our network, coined LayoutGMN, learns the layout metric via neural graph matching, using an attention-based GMN designed under a triplet network setting. To train our network, we utilize weak labels obtained by pixel-wise Intersection-over-Union (IoUs) to define the triplet loss. Importantly, LayoutGMN is built with a structural bias which can effectively compensate for the lack of structure awareness in IoUs. We demonstrate this on two prominent forms of layouts, viz., floorplans and UI designs, via retrieval experiments on large-scale datasets. In particular, retrieval results by our network better match human judgement of structural layout similarity compared to both IoUs and other baselines including a state-of-the-art method based on graph neural networks and image convolution. In addition, LayoutGMN is the first deep model to offer both metric learning of structural layout similarity* and *structural matching between layout elements.*

1. Introduction

Two-dimensional layouts are ubiquitous visual abstractions in graphic and architectural designs. They typically represent blueprints or conceptual sketches for such data as floorplans, documents, scene arrangements, and UI designs. Recent advances in pattern analysis and synthesis have propelled the development of generative models for layouts [11, 25, 47, 15, 26] and led to a steady accumulation of relevant datasets [48, 42, 10, 46]. Despite these developments however, there have been few attempts at employing a *deeply learned metric* to reason about layout data, e.g., for retrieval, data embedding, and evaluation. For example, current evaluation protocols for layout generation still rely heavily on segmentation metrics such as intersection-over-union (IoU) [15, 30] and human judgement [15, 26].

The ability to compare data effectively and efficiently is arguably the most foundational task in data analysis. The key challenge in comparing layouts is that it is not purely a task of visual comparison — it depends critically on inference and reasoning about *structures*, which are expressed by the semantics and organizational arrangements of the elements or subdivisions which compose a layout. Hence, none of the well-established image-space metrics, whether model-driven, perceptual, or deeply learned, are best suited to measure structural layout similarity. Frequently applied similarity measures for image segmentation such as IoUs and F1 scores all perform pixel-level matching "in place" — they are not structural and can be sensitive to element misalignments which are *structure-preserving*.

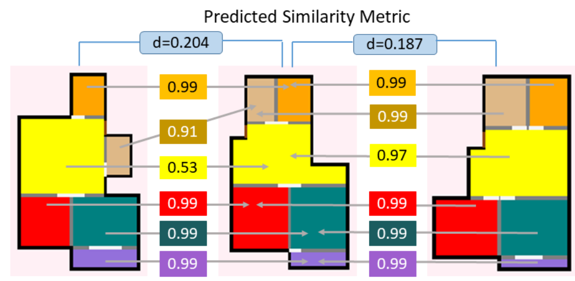


Figure 1. LayoutGMN learns a structural layout similarity metric between floorplans and other 2D layouts, through *attention-based neural graph matching*. The learned attention weights (numbers shown in the boxes) can be used to match the structural elements.

In this work, we develop a deep neural network to predict structural similarity between two 2D layouts, e.g., floorplans or UI designs. We take a predominantly structural view of layouts for both data representation and layout comparison. Specifically, we represent each layout using a directed, fully connected graph over its semantic elements. Our network learns structural layout similarity via neural graph matching, where an *attention-based graph matching network* [27] is designed under a *triplet network* setting. The network, coined LayoutGMN, takes as input a triplet of layout graphs, composed together by one pair of anchor-positive and one pair of anchor-negative graphs, and performs intra-graph message passing and cross-graph information communication per pair, to learn a graph embedding for layout similarity prediction. In addition to returning a metric, the attention weights learned by our network can also be used to match the layout elements; see Figure 1.

To train our triplet network, it is natural to consider human labeling of positive and negative samples. However, it

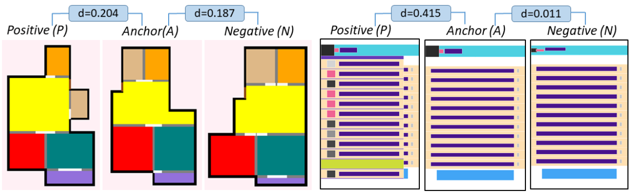


Figure 2. Structure matching in LayoutGMN "*neutralizes*" IoU feedback. In each example (left: floorplan; right: UI design), a training sample N labeled as "Negative" by IoU is more structurally similar to the anchor (A) than P, a "Positive" sample. With structure matching, our network predicts a smaller A-to-N distance than A-to-P distance in each case, which contradicts IoU.

is well-known that subjective judgements by humans over structured data such as layouts are often unreliable, especially with non-experts [45, 2]. When domain experts are employed, the task becomes time-consuming and expensive [45, 2, 14, 9, 20, 41], where discrepancies among even these experts still remain [14]. In our work, we avoid this issue by resorting to *weakly supervised* training of LayoutGMN, which obtains positive and negative labels from the training data through thresholding using layout IoUs [30].

The motivations behind using IoU for training are threefold, despite its shortcomings for structural matching. First, IoU does hold merits as one of the most wide-used layout similarity measures [30, 15]. Second, IoU is *objective* and much cheaper to obtain compared to expert annotations. Finally and most importantly, our network has a built-in inductive bias to enforce structural correspondence, via inter-graph information exchange, when learning the graph embeddings. The *structural bias* introduced can effectively compensate for the lack of structure awareness in the IoU-based triplet loss. In Figure 2, we illustrate the effect of the structural bias on the metric learned by our network.

We evaluate our network on retrieval tasks over large datasets of floorplans and UI designs, via Precision@k scores, and investigate the stability of the proposed metric by checking retrieval consistency and top-1 retrieved results. Overall, retrieval results by LayoutGMN better match human judgement of structural layout similarity compared to both IoUs and other baselines including a state-of-the-art method based on graph neural networks [30]. Finally, we show a label transfer application for floorplans enabled by the structure matching learned by our network.

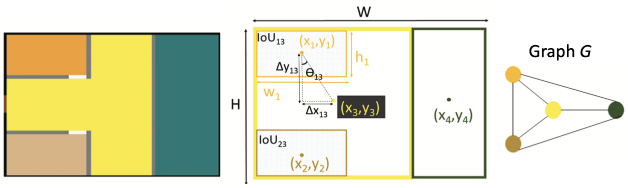
2. Related Work

Layout analysis. Early works [18, 3] on document analysis involved primitive heuristics to analyse document structures. Organizing a large collection of such structures into meaningful clusters requires a distance measure between layouts, which typically involved content-based heuristics [34] for documents and constrained graph matching algorithm for floorplans [40]. An improved distance measure relied on rich layout representation obtained using autoencoders [7, 29], operating on an entire UI layout. Although such models capture rich raster properties of layout images, layout structures are not modeled, leading to noisy recommendations in contextual search over layout datasets.

Layout generation. Early works on synthesizing 2D layouts relied on exemplars [16, 23, 37] and rule-based heuristics [33, 38], and were unable to capture complex element distributions. The advent of deep learning led to generative models of layouts of floorplans [42, 15, 5, 32], documents [25, 11, 47], and UIs [7, 6]. Perceptual studies aside, evaluation of generated layouts, in terms of diversity and generalization, has mostly revolved around IoUs of the constituent semantic entities [25, 11, 15]. While IoU provides a visual similarity measure, it is expensive to compute over a large number of semantic entities, and is sensitive to element positions within a layout. Developing a tool for structural comparison would perhaps complement visual features in contextual similarity search. In particular, a learning-based method that compares layouts structurally can prove useful in tasks such as layout correspondence, component labeling and layout retargeting. We present a Layout Graph Matching Network, called LayoutGMN, for learning to compare two graphical layouts in a structured manner.

Structural similarity in 3D. Fisher et al. [8] develop Graph Kernels for characterizing structural relationships in 3D indoor scenes. Indoor scenes are represented as graphs, and the Graph Kernel compares substructures in the graphs to capture similarity between the corresponding scenes. A challenging problem of organizing a heterogeneous collection of such 3D indoor scenes was accomplished in [43] by focusing on a subscene, and using it as a reference point for distance measures between two scenes. Shape Edit Distance, SHED, [22] is another fine-grained sub-structure similarity measure for comparing two 3D shapes. These works provide valuable cues on developing an effective structural metric for layout similarity. Graph Neural Networks (GNN) [28, 21, 4, 36] model node dependencies in a graph via message passing, and are the perfect tool for learning on structured data. GNNs provide coarse-level graph embeddings, which, although useful for many tasks [39, 1, 17, 19], can lose useful structural information in contextual search, if each graph is processed in isolation. We make use of Graph Matching Network [27] to retain structural correspondence between layout elements.

GNNs for structural layout similarity. To the best of our knowledge, the recent work by Manandhar et al. [30] is the first to leverage GNNs to learn structural similarity of 2D graphical layouts, focusing on UI layouts with rectangular boundaries. They employ a GCN-CNN architecture on a graph of UI layout *images*, also under an IoU-trained triplet



(a) Layout image. (b) Layout features. (c) Layout graph.

Figure 3. Given an input floorplan image with room segmentations in (a), we abstract each room into a bounding box and obtain layout features from the constituent semantic elements, as shown in (b). These features form the initial node and edge features (Section 3.1) of the corresponding layout graph shown in (c).

network [13], but obtain the graph embeddings for the anchor, positive, and negative graphs independently.

In contrast, LayoutGMN learns the graph embeddings in a *dependent* manner. Through cross-graph information exchange, the embeddings are learned in the context of the anchor-positive (respectively, the anchor-negative) pair. This is a critical distinction to GCN-CNN [30], while both train their triplet networks using IoUs. However, since IoU does not involve structure matching, it is not a reliable measure of structural similarity, leading to labels which are considered "structurally incorrect"; see Figure 2.

In addition, our network does not perform any convolutional processing over layout images; it only involves eight MLPs, placing more emphasis on learning finer-scale structural variations for graph embedding, and less on image-space features. We clearly observe that the cross-graph communication module in our GMNs does help in learning finer graph embeddings than the GCN-CNN framework [30]. Finally, another advantage of moving away from any reliance on image alignment is that similarity predictions by our network are more robust against highly varied, non-rectangular layout boundaries, e.g., for floorplans.

3. Method

The Graph Matching Network (GMN) [27] consumes a pair of graphs, processes the graph interactions via an attention-based cross-graph communication mechanism and results in graph embeddings for the two input graphs, as shown in Fig 4. Our LayoutGMN plugs in the Graph Matching Network into a Triplet backbone architecture for learning a (pseudo) metric-space for similarity on 2D layouts such as floorplans, UIs and documents.

3.1. Layout Graphs

Given a layout image of height H and width W with semantic annotations, we abstract each element into a bounding box, which form the nodes of the resulting layout graph. Specifically, for a layout image I_1, its layout graph G_l is given by $G_l = (V, E)$, where the node set $V = \{\boldsymbol{v_1}, \boldsymbol{v_2}, ..., \boldsymbol{v_n}\}$ represents the semantic elements in the layout, and $E = \{\boldsymbol{e_{12}}, ..., \boldsymbol{e_{ij}}, .., \boldsymbol{e_{n(n-1)}}\}$, the edge set,

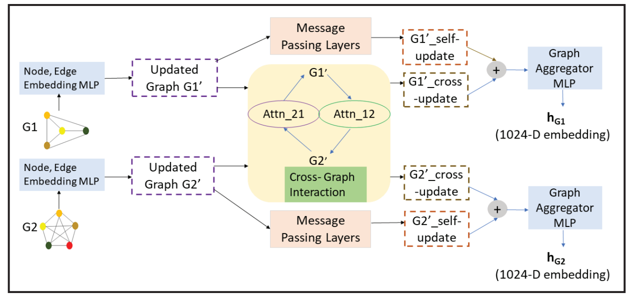


Figure 4. LayoutGMN takes two layout graphs as input, performs intra-graph message passing (Eq. 2), along with cross-graph information exchange (Eq. 3) via an attention mechanism (Eq. 5, also visualized in Figure 1) to update node features, from which final graph embeddings are obtained (Eq. 7).

represents the set of edges connecting the constituent elements. Our layout graphs are directed and fully-connected.

Initial Node Features. There exist a variety of visual and content-based features that could be incorporated as the initial node features; ex. the text data/font size/font type of an UI element or the image features of a room in a floorplan. For structured learning tasks as ours, we ignore such content-based features and only focus on the box abstractions. Specifically, similar to [11, 12], the initial node features contain *semantic* and *geometric* information of the layout elements. As shown in Fig 3, for a layout element k centered at (x_k, y_k), with dimensions (w_k, h_k), its geometric information is:

$$g_k = \left[\frac{x_k}{W}, \frac{y_k}{H}, \frac{w_k}{W}, \frac{h_k}{H}, \frac{w_k h_k}{\sqrt{WH}}\right].$$

Instead of one-hot encoding of the semantics, we use a learnable embedding layer to embed a semantic type into a 128-D code, s_k. A two-layer MLP embeds the 5×1 geometric vector g_k into a 128-D code, and is concatenated with the 128-D semantic embedding s_k to form the initial node features $U = \{\boldsymbol{u_1}, \boldsymbol{u_2}, ..., \boldsymbol{u_n}\}$.

Initial Edge Features. In visual reasoning and relationship detection tasks, edge features in a graph are designed to capture relative difference of the abstracted semantic entities (represented as nodes) [12, 44]. Thus, for an edge e_{ij}, we capture the spatial relationship (see Fig 3) between the semantic entities by a 8×1 vector:

$$\boldsymbol{e_{ij}} = \left[\frac{\Delta x_{ij}}{\sqrt{A_i}}, \frac{\Delta y_{ij}}{\sqrt{A_i}}, \sqrt{\frac{A_j}{A_i}}, U_{ij}, \frac{w_i}{h_i}, \frac{w_j}{h_j}, \frac{\sqrt{\Delta x^2 + \Delta y^2}}{\sqrt{W^2 + H^2}}, \theta\right],$$

where A_i is the area of the element box i; $U_{ij} = \frac{B_i \cap B_j}{B_i \cup B_j}$ is the IoU of the bounding boxes of the layout elements i, j; $\theta = atan2(\frac{\Delta y}{\Delta x})$ is the relative angle between the two components, $\theta \in [-\pi, \pi]$; $\Delta x_{ij} = x_j - x_i$ and $\Delta y_{ij} = y_j - y_i$. This edge vector accounts for the translation between the two layout elements, in addition to encoding their box IoUs, individual aspect ratios and relative orientation.

3.2. Graph Matching Network

The graph matching module employed in LayoutGMN is made up of three parts: (1) node and edge encoders, (2) message propagation layers and (3) an aggregator.

Node and Edge Encoders. We use two MLPs to embed the initial node and edge features and compute their corresponding code vectors:

$$\begin{aligned} \boldsymbol{h_i}^{(0)} &= MLP_{node}(\boldsymbol{u_i}), \forall i \in U \\ \boldsymbol{r_{ij}} &= MLP_{edge}(\boldsymbol{e_{ij}}), \forall (i,j) \in E \end{aligned} \tag{1}$$

The above MLPs map the initial node and edge features to their 128-D code vectors.

Message Propagation Layers. The graph matching framework hinges on coherent information exchange between graphs to compare two layouts in a structural manner. The propagation layers update the node features by aggregating messages along the edges within a graph, in addition to relying on a graph matching vector that measures how similar a node in one layout graph is to one or more nodes in the other. Specifically, given two node embeddings $\boldsymbol{h}_i^{(0)}$ and $\boldsymbol{h}_p^{(0)}$ from two different layout graphs, the node updates for the node i are given by:

$$\boldsymbol{m}_{j\to i} = f_{intra}\left(\boldsymbol{h}_i^{(t)}, \boldsymbol{h}_j^{(t)}, \boldsymbol{r}_{ij}\right), \forall (i,j) \in E_1 \tag{2}$$

$$\boldsymbol{\mu}_{p\to i} = f_{cross}\left(\boldsymbol{h}_i^{(t)}, \boldsymbol{h}_p^{(t)}\right), \forall i \in V_1, p \in V_2 \tag{3}$$

$$\boldsymbol{h}_i^{(t+1)} = f_{update}\left(\boldsymbol{h}_i^{(t)}, \sum_j \boldsymbol{m}_{j\to i}, \sum_p \boldsymbol{\mu}_{p\to i}\right) \tag{4}$$

where f_{intra} is an MLP on the initial node embedding code that aggregates information from other nodes within the same graph, f_{cross} is a function that communicates cross-graph information, and f_{update} is an MLP used to update the node features in the graph, whose input is the concatenation of the current node features, the aggregated information from within, and across the graphs. f_{cross} is designed as an Attention-based module:

$$\begin{aligned} a_{p\to i} &= \frac{\exp(s_h(\boldsymbol{h}_i^{(t)}, \boldsymbol{h}_p^{(t)})}{\sum_p \exp(s_h(\boldsymbol{h}_i^{(t)}, \boldsymbol{h}_p^{(t)})} \\ \boldsymbol{\mu}_{p\to i} &= a_{p\to i}\left(\boldsymbol{h}_i^{(t)} - \boldsymbol{h}_p^{(t)}\right) \end{aligned} \tag{5}$$

where $a_{p\to i}$ is the attention value (scalar) between node p in the second graph and node i in the first, and such attention weights are calculated for every pair of nodes across the two graphs; s_h is implemented as the dot product of the embedded code vectors. The interaction of all the nodes $p \in V_2$ with the node i in V_1 is then given by:

$$\sum_p \boldsymbol{\mu}_{p\to i} = \sum_p a_{p\to i}\left(\boldsymbol{h}_i^{(t)} - \boldsymbol{h}_p^{(t)}\right) = \boldsymbol{h}_i^{(t)} - \sum_p a_{p\to i}\boldsymbol{h}_p^{(t)} \tag{6}$$

Intuitively, $\sum_p \boldsymbol{\mu}_{p\to i}$ measures the (dis)similarity between $\boldsymbol{h}_i^{(t)}$ and its nearest neighbor in the other graph. The pairwise attention computation results in stronger structural bonds between the two graphs, but requires additional computation. We use five rounds of message propagation, then the representation for each node is updated accordingly.

Aggregator. A 1024-D graph-level representation, $\boldsymbol{h}_G$, is obtained via a feature aggregator MLP, f_G, that takes as input, the set of node representations $\{\boldsymbol{h}_i^{(T)}\}$, as given below:

$$\boldsymbol{h}_G = MLP_G\left(\sum_{i\in V} \sigma(MLP_{gate}(\boldsymbol{h}_i^{(T)})) \odot MLP(\boldsymbol{h}_i^{(T)})\right) \tag{7}$$

Graph-level embeddings for the two layout graphs is similarly computed.

$$\begin{aligned} \boldsymbol{h}_{G_1} &= f_G(\{\boldsymbol{h}_i^{(T)}\}_{i\in V_1}) \\ \boldsymbol{h}_{G_2} &= f_G(\{\boldsymbol{h}_p^{(T)}\}_{p\in V_2}) \end{aligned}$$

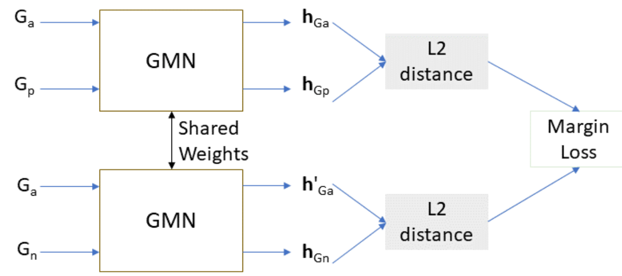


Figure 5. Given a triplet of graphs G_a, G_p and G_n corresponding to the anchor, positive and negative examples respectively, the anchor graph paired with each of other two graphs is passed through a Graph Matching Network (Fig 4) to get two 1024-D embeddings. Note that the anchor graph has different contextual embeddings $\boldsymbol{h}_{Ga}$ and $\boldsymbol{h'}_{Ga}$. LayoutGMN is trained using the margin loss (margin=5) on the L_2 distances of the two paired embeddings.

3.3. Training

To learn a layout similarity metric, we borrow the Triplet training framework [13]. Specifically, given two pairs of layout graphs, i.e., anchor-positive and anchor-negative, each pair is passed through the same GMN module to get the graph embeddings in the context of the other graph, as shown in Fig 5. A margin loss based on the L_2 distance between the graph embeddings, as given in equation 8, is used to backpropagate the gradients through GMN.

$$\begin{aligned} L_{tri}(a,p,n) = max(0, \gamma + &\left\|\boldsymbol{h}_{G_a} - \boldsymbol{h}_{G_p}\right\|_2 \\ &- \|\boldsymbol{h'}_{G_a} - \boldsymbol{h}_{G_n}\|_2) \end{aligned} \tag{8}$$

4. Datasets

We use two kinds of layout datasets in our experiments: (1) UI layouts from the RICO dataset [7], and (2) floorplans from the RPLAN dataset [42]. After some data filtering , the size of the two datasets is respectively, 66261 and 77669.

In the absence of a ground truth label set and the need for obtaining the triplets in a consistent manner, we resort to using IoU values of two layouts, represented as multi-channel images, to ascertain their closeness. Given an anchor layout, the threshold on IoU values to classify another layout as positive, from observations, is 0.6 for both UIs and floorplans. Negative examples are those that have a threshold value of at least 0.1 less than the positive ones, avoiding the incorrect "negatives" during training. The train-test sizes for the aforementioned datasets are respectively: 7700-1588, 25000-7204. In the filtered floorplan training dataset [42], the distinct number of semantic categories/rooms across the dataset is nine and the maximum number of rooms per floorplan is eight. Similarly, for the filtered UI layout dataset [7], the number of distinct semantic categories is twenty-five and the number of elements per UI layout across the dataset is at most hundred.

5. Results and Evaluation

We evaluate LayoutGMN by comparing its retrieval results to those of several baselines, evaluated using human judgements. Similarity prediction by our network is efficient: taking 33 milliseconds per layout pair on a CPU. With our learning framework, we can efficiently retrieve multiple, sorted results by batching the database samples.

5.1. Baselines

Graph Kernel (GK) [8]. GK is one of the earliest structural similarity metrics, initially developed to compare indoor 3D scenes. We adopt it to 2D layouts of floorplans and UI designs. We input the same layout graphs to GK to get retrievals from the two databases, and use the best setting based on result quality/computation cost trade-off.

U-Net [35]. As one of the best segmentation networks, we use U-Net with the same parameter setting as in Pytorch, in a triplet network setting to auto-encode layout images. The input to the network is a multi-channel image with semantic segmentations. The network is trained on the same set of triplets as LayoutGMN until convergence.

IoU Metric. Given two multi-channel images, we use the IoU values between two layout images to get their IoU score, and use this score to sort the examples in the datasets to rank the retrievals for a given query.

GCN-CNN [30]. The state-of-the-art network for structural similarity on UI layouts is a hybrid network comprised of an attention-based GCN, similar to the gating mechanism in [28], coupled with a CNN. In this original GCN-CNN,

Method	Precision@k (%)		
	k=1 (↑)	k=5 (↑)	k=10 (↑)
Graph Kernel [8]	33.33	15.83	11.46
U-Net _Triplet [35]	27.08	10.83	7.92
IoU Metric	43.75	22.92	14.38
GCN-CNN_Triplet [30]	39.6	17.1	13.33
LayoutGMN	**47.91**	**22.92**	**15.83**
Graph Kernel [8]	27.27	15.15	12.42
U-Net _Triplet [35]	28.28	18.18	15.05
IoU Metric	33.84	24.04	17.48
GCN-CNN_Triplet [30]	37.37	22.02	17.02
LayoutGMN	**38.38**	**25.35**	**21.21**

Table 1. Precision scores for the top-k retrieved results obtained using different methods, on a set of randomly chosen UI and floorplan queries. The first set of five comparisons is for UI layouts, followed by floorplans.

the training triplets are randomly sampled every epoch, leading to better training due to diverse training data. In our work, for a fair comparison over all the aforementioned networks, we sample a fixed set of triplets in every epoch of training. The GCN-CNN network is trained on the two datasets of our interest, using the same training data as ours.

Qualitative retrieval results for GCN-CNN, IoU metric and LayoutGMN for a given query are shown in Figure 6.

5.2. Evaluation Metrics

***Precision@k* scores.** To validate the correctness of LayoutGMN as a tool for measuring layout similarity, we start by evaluating layout retrieval from a large database. A standard evaluation protocol for measuring the relevance of ranked lists is the *Precision@k* scores [31]. Given a query q_i from a query set $Q = \{q_1, q_2, q_3, ..., q_n\}$, we measure the relevance of the ranked lists $L(q_i) = [l_{i1}, l_{i2}, ..., l_{ik}, ...]$ using the precision scores, defined as:

$$P@k(Q, L) = \frac{1}{k|Q|} \sum_{q_i \in Q} \sum_{j=1}^{k} rel(L_{ij}, q_i), \tag{9}$$

where, *rel*(L_{ij}, q_i) is a binary indicator of the relevance of the returned element L_{ij} for the query q_i. In our evaluation, due to the lack of a *labeled* and *exhaustive* recommendation set for any query over the layout datasets employed, such a binary indication of relevance is determined by human subjects. Table 1 shows the $P@k$ scores for different networks described in Section 5.1 employed for the layout retrieval task. To get the precision scores, similar to [30], we conducted a crowd-sourced annotation study via Amazon Mechanical Turk (AMT) on the top-10 retrievals per query, for N ($N = 50$ for UIs and 100 for floorplans) randomly chosen queries outside the training set. 10 turkers were asked to indicate the structural relevance of each of

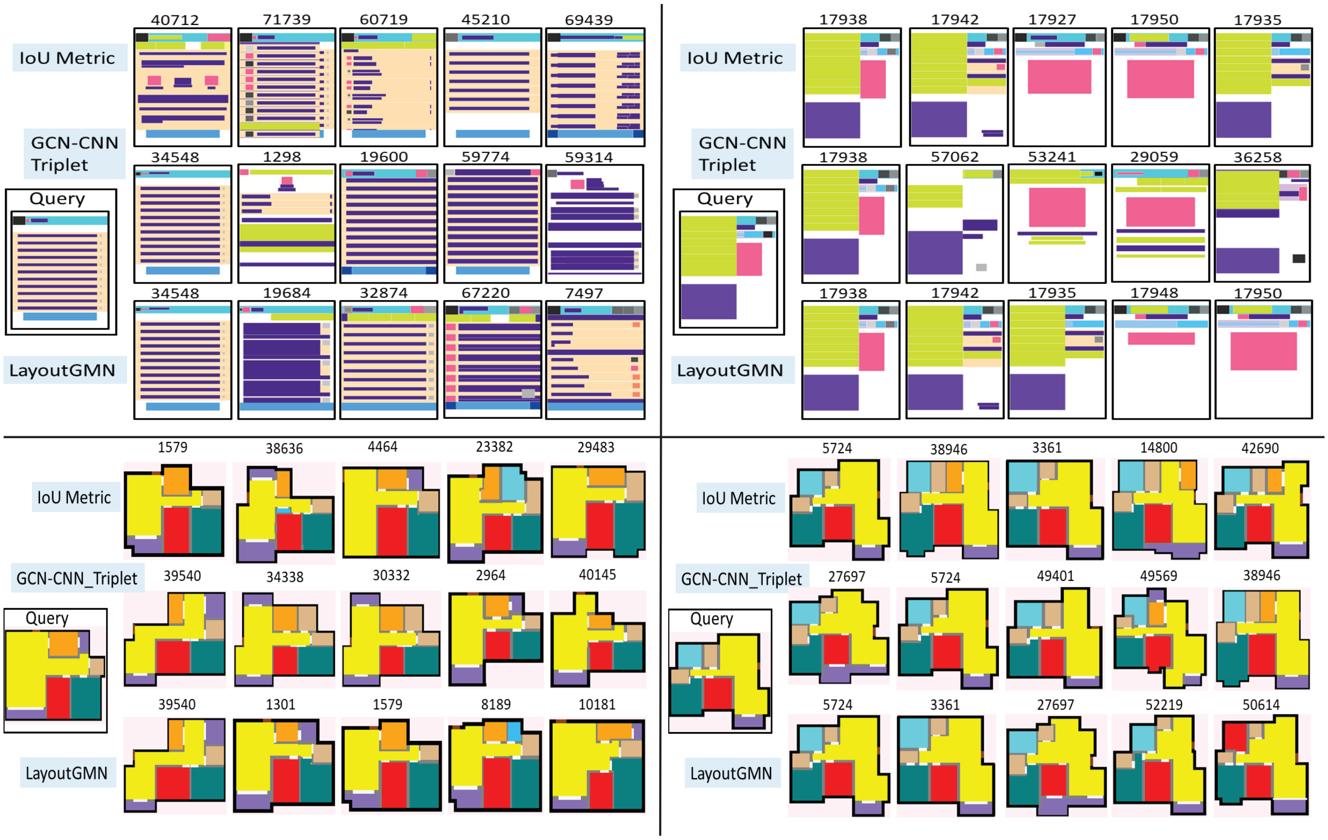


Figure 6. Top-5 retrieved results for an input query based on IoU metric, GCN-CNN_Triplet [30] and LayoutGMN. We observe that the ranked results returned by LayoutGMN are closer to the input query than the other two methods, although it was trained on triplets computed using the IoU metric. Attention weights for understanding structural correspondence in LayoutGMN are shown in Figure 1 and also provided in the supplementary material. UI and floorplan IDs from the RICO dataset [7] and RPLAN dataset [42] are indicated on top of each result. More results, along with results on document layouts, can be found in the supplementary material.

the top-10 results per query, without any specific instructions on what a structural comparison means. A result was considered relevant if at least 6 turkers agreed. For details on the AMT study, please see the supplementary material. We observe that LayoutGMN better matches humans' notion of structural similarity. [30] performs better than the IoU metric on floorplan data (+3.5%) on the top-1 retrievals and is comparable to IoU metric on top-5 and top-10 results. On UI layouts, the IoU metric is judged better by turkers than [30]. U-Net fails to retrieve structurally similar results because it overfits on the small amount of training data, and relies more on image pixels due to its convolutional structure. LayoutGMN outperforms other methods by at least 1% for all k, on both datasets. The precision scores on floorplans (bottom-set) are lower than on UI layouts perhaps because they are easier to compare owing to smaller set of semantic elements than UIs and turkers tend to focus more on the size and boundary of the floorplans in additional to the structural arrangements. We believe that when a lot of semantics are present in the layouts and are scattered (as in UIs), the users tend to look at the overall structure instead of trying to match every single element owing to reduced attention-span, which likely explains higher scores for UIs.

Overlap@k score. We also propose another measure to quantify the stability of retrieved results – *Overlap@k* score. Specifically, if Q_1 is a set of queries and Q_1^{top1} is the set of top-1 retrieved results for every query in Q_1, then

$$Ov@k(Q_1, Q_1^{top1}) = \frac{1}{k|Q_1|} \sum_{\substack{q_m \in Q_1 \\ q_p = top1(q_m)}} \sum_{j=1}^{k} (L_{mj}, \wedge L_{pj}) \tag{10}$$

where L_{ij} is the j^{th} ranked result for the query q_i, and $\wedge$ is the logical AND operation. Thus, ($L_{mj} \wedge L_{pj}$) is 1 if the j^{th} result for query $q_m \in Q_1$ and query q_p = top1(Q_1)$\in Q_1^{top1}$ are the same. This score measures the ability of the layout similarity metric to replicate the distance field implied by a query according to its top-ranked retrieval. In other words, retrieval stability can be measured by checking the consistency of retrievals for many (q_m, q_p) pairs. This

Method	Overlap@k (%)	
	k=5 (↑)	k=10 (↑)
IoU Metric	**50.6**	49.4
GCN-CNN_Triplet [30]	46.8	45.6
LayoutGMN	49.8	**49.8**
IoU Metric	30.42	30.8
GCN-CNN_Triplet [30]	43.2	46.8
LayoutGMN	**47.6**	**50.8**

Table 2. Overlap scores for checking the consistency of retrievals for a query and its top-1 retrieved result, over 50 such pairs. The first set of three rows are for UI layouts, followed by floorplans.

Method	Test Accuracy on Triplets	
	IoU-based (↑)	User-based (↑)
Graph Kernel [8]	90.09	90.73
U-Net _Triplet [35]	96.67	93.38
GCN-CNN_Triplet [30]	96.45	94.48
LayoutGMN	**98.96**	**95.80**
Graph Kernel [8]	92.07	95.60
U-Net _Triplet [35]	93.01	91.00
GCN-CNN_Triplet [30]	92.50	91.8
LayoutGMN	**97.54**	**97.60**

Table 3. Classification accuracy on test triplets obtained using IoU metric (IoU-based) and annotated by users (User-based). The first set of comparisons is for UI layouts, followed by floorplans.

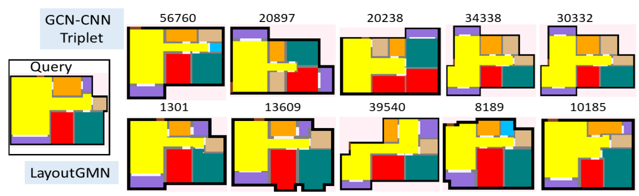


Figure 7. Retrieval results for the bottom-left query in Fig 6, when adjacency graphs are used. We observe, on most of the queries, that the performance of LayoutGMN improves, but degrades in the case of GCN-CNN [30] on floorplan data.

score makes sense only when the ranked results returned by a layout similarity tool are deemed reasonable, as assessed by the P@k scores. Table 2 shows the *overlap@k* scores for k=5,10 for IoU metric, GCN-CNN [30] and LayoutGMN on 50 such pairs. On UIs (first three rows), IoU metric has a slightly higher *Ov@5* score (+0.6%) than LayoutGMN. Also, it shares the largest *P@5* score with LayoutGMN, indicating that IoU metric has slightly better retrieval stability for the top-5 results. However, in the case of *Ov@10*, LayoutGMN has a higher score (+0.4%) than IoU metric and also has a higher *P@10* score than the other two methods, indicating that when top-10 retrievals are considered, LayoutGMN has slightly better consistency on the retrievals. As for floorplans (last three rows), Table 1 already shows that LayoutGMN has the best *P@k* scores. This, coupled with a higher *Ov@k* scores, indicate that on floorplans, LayoutGMN has better retrieval stability. In the supplementary material, we show qualitative results on the stability of retrievals for the three methods.

Classification accuracy. We also measure the classification accuracy of test-triplets as a sanity check. However, such a measure alone is not an appropriate one for correctness of a similarity metric employed in information retrieval tasks [31]. We present it alongside *Precision@k* and *Overlap@k* scores for a broader, informed evaluation, in Table 3. Since user annotations are expensive and time consuming (and hence the motivation to use IoU metric to get weak training labels), we only get user annotations on 452 triplets for both UIs and floorplans, and the last column of Table 3 reflects the accuracy on such triplets. LayoutGMN outperforms all the baselines by atleast 1.32%, on triplets obtained using both, IoU metric and user annotations.

5.3. Fully-connected vs. Adjacency Graphs

Following [30], we employed fully connected graphs for our experiments and observed that such graphs are a good design for training graph neural networks for learning structural similarity. We also performed experiments using adjacency graphs on GCN-CNN [30] and LayoutGMN, and observed that, for floorplans (where the graph node count is small), the quality of retrievals improved in the case of LayoutGMN, but degraded for GCN-CNN. This is mainly because GCN-CNN obtains independent graph embeddings for each input graph and when the graphs are built only on adjacency connections, some amount of global structural prior is lost. On the other hand, GMNs obtain better contextual embeddings by now matching the sparsely connected adjacency graphs, as a result of narrower search space; for a qualitative result using adjacency graphs, see Figure 7. However, for UIs (where the graph node count is large), the elements are scattered all over the layout, and no one heuristic is able to capture adjacency relations perfectly. The quality of retrievals for both the networks degraded when using adjacency graphs on UIs. More results can be found in the supplementary material.

5.4. Ablation Studies on Structural Representation

To evaluate how the node and edge features in our layout representation contribute to network performance, we conduct an ablation study by gradually removing these features. Our design of the initial representation of the layout graphs (Sec 3.1) are well studied in prior works on layout generation [11, 26], visual reasoning, and relationship detection tasks [12, 44, 30]. As such, we focus on analyzing LayoutGMN's behavior when strong structural priors viz., the edges, box positions, and element semantics, are ablated. Figure 8 shows qualitative results on top-5 retrieved results

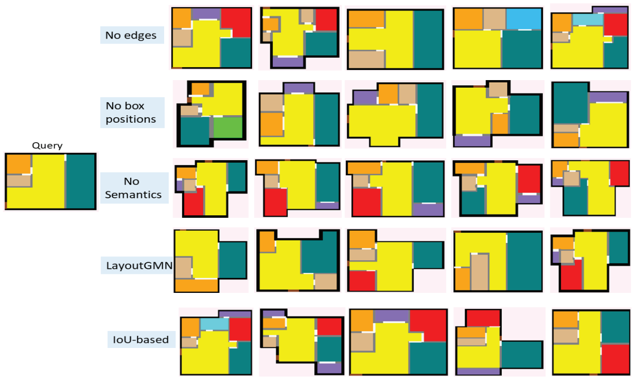


Figure 8. Top-5 retrieved results for a given query when structural priors (edges, box positions, and element semantics) are gradually removed from the input graphs.

for a given query when these structural priors in the training graphs are gradually removed.

Graph edges. When the edges of the graphs are not considered, i.e., when there is no message propagation within a graph, the only component that updates the node features is the attention-weighted node update (Eq. 4). Naturally, the structure encoding is lost in both, the query and the database sample, leading to random retrievals; see first row of Figure 8. **Effect of box positions:** The nodes of the layout graphs encode both the absolute box positions and the element semantics. When the position encoding information is withdrawn, arguably, the most important cue is lost. The resulting retrievals from such a poorly trained model, as seen in the second row of Figure 8, are noisy as semantics alone do not provide enough structural priors. **Effect of node semantics:** Next, when the box positions are preserved but the element semantics are not encoded, we observe that the network slowly begins to understand element comparison guided by the position info, but falls short of understanding the overall structure information. Finally, when all the above information is accounted for, we observe that the network better learns the structural information and even returns structurally sound results compared to the IoU metric.

5.5. Attention-based Layout Label Transfer

We present *layout label transfer*, via attention-based structural element matching, as a natural application of LayoutGMN. Given a source layout image I_1 with known labels, the goal is to transfer the labels to a target layout I_2. A straight-forward approach to establishing element correspondence is via maximum area/pixel-overlap matching for every element in I_2 with respect to all the elements in I_1. However, this scheme is highly sensitive to element positions within the two layouts. Moreover, raster-alignment (via translations) of layouts is non-trivial to formulate when the two layout images have different boundaries and structures. LayoutGMN, on the other hand, is robust to such boundary variations, and can be directly used to obtain element-level correspondences using the built-in attention mechanism that provides an attention score for every element-level match. Specifically, we use a *pretrained* LayoutGMN which is fed with two layout graphs, where the semantic encoding of all nodes is set to a vector of ones.

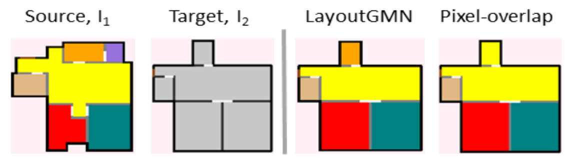


Figure 9. Element-level label transfer results from a source image I_1 to a target image I_2, using a *pretrained* LayoutGMN vs. maximum pixel-overlap matching. LayoutGMN predicts correct labels via attention-based element matching.

As shown in Figure 9, the *pretrained* LayoutGMN is able to find the correct labels despite masking the semantic information at the input. Note that when semantic information is masked at the input, such a transfer can not be applied to any two layouts. It is limited by a weak/floating alignment of I_1 and I_2, as seen in Figure 9.

6. Conclusion, limitation, and future work

With the advent of large-scale layout datasets, analysing and organizing layout data becomes crucial, where the first step is to develop an effective means to compare layouts. We present the first deep neural network to offer both metric learning of structural layout similarity and structural matching between layout elements. Extensive experiments demonstrate that our metric best matches human judgement of structural similarity for both floorplans and UI designs, compared to all well-known baselines.

The main limitation of our current learning framework is the requirement for strong supervision, which justifies, in part, the use of the less-than-ideal IoU metric for network training. An interesting future direction is to combine few-shot or active learning with our GMN-based triplet network, e.g., by finding ways to obtain small sets of training triplets that are both informative and diverse [24].

Another limitation of our current network is that it does not learn *hierarchical* graph representations or structural matching, which would have been desirable when handling large graphs. In addition, the graph embedding space learned by LayoutGMN may be worth a closer examination to assess its potential for generative modeling. Finally, we would like to explore applying our learning framework to other, more complex graph structured data.

References

[1] Oron Ashual and Lior Wolf. Specifying object attributes and relations in interactive scene generation. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4561–4569, 2019. 2

[2] Thorsten Brants. Inter-annotator agreement for a german newspaper corpus. In *International Conference on Knowledge Engineering and Knowledge Management*, 2000. 2

[3] Thomas M Breuel. High performance document layout analysis. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 209–218, 2003. 2

[4] Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017. 2

[5] Qi Chen, Qi Wu, Rui Tang, Yuhan Wang, Shuai Wang, and Mingkui Tan. Intelligent home 3d: Automatic 3d-house design from linguistic descriptions only. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12625–12634, 2020. 2

[6] Niraj Ramesh Dayama, Kashyap Todi, Taru Saarelainen, and Antti Oulasvirta. GRIDS: Interactive layout design with integer programming. In *Proceedings of the 2020 CHI Conference on Human Factors in Computing Systems*, pages 1–13, 2020. 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 ACM Symposium on User Interface Software and Technology*, pages 845–854, 2017. 2, 5, 6

[8] Matthew Fisher, Manolis Savva, and Pat Hanrahan. Characterizing structural relationships in scenes using graph kernels. In *ACM SIGGRAPH 2011 papers*, pages 1–12. 2011. 2, 5, 7

[9] Karën Fort, Maud Ehrmann, and Adeline Nazarenko. Towards a methodology for named entities annotation. 2009. 2

[10] Huan Fu, Bowen Cai, Lin Gao, Lingxiao Zhang, Rongfei Jia, Binqiang Zhao, and Hao Zhang. 3D-FRONT: 3D Furnished Rooms with layOuts and semaNTics, 2020. 1

[11] Akshay Gadi Patil, Omri Ben-Eliezer, Or Perel, and Hadar Averbuch-Elor. READ: Recursive autoencoders for document layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 544–545, 2020. 1, 2, 3, 7

[12] Longteng Guo, Jing Liu, Jinhui Tang, Jiangwei Li, Wei Luo, and Hanqing Lu. Aligning linguistic words and visual semantic units for image captioning. In *Proceedings of the 27th ACM International Conference on Multimedia*, pages 765–773, 2019. 3, 7

[13] Elad Hoffer and Nir Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015. 3, 4

[14] George Hripcsak and Adam Wilcox. Reference standards, judges, and comparison subjects: roles for experts in evaluating system performance. *Journal of the American Medical Informatics Association*, 9(1):1–15, 2002. 2

[15] Ruizhen Hu, Zeyu Huang, Yuhan Tang, Oliver van Kaick, Hao Zhang, and Hui Huang. Graph2Plan: Learning floorplan generation from layout graphs. *ACM Transaction on Graphics (TOG)*, 2020. 1, 2

[16] Nathan Hurst, Wilmot Li, and Kim Marriott. Review of automatic document formatting. In *Proceedings of the 9th ACM symposium on Document engineering*, pages 99–108, 2009. 2

[17] Justin Johnson, Agrim Gupta, and Li Fei-Fei. Image generation from scene graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1219–1228, 2018. 2

[18] Rangachar Kasturi. *Document image analysis*, volume 39. 2

[19] Nagma Khan, Ushasi Chaudhuri, Biplab Banerjee, and Subhasis Chaudhuri. Graph convolutional network for multi-label vhr remote sensing scene recognition. *Neurocomputing*, 357:36–46, 2019. 2

[20] Jin-Dong Kim, Tomoko Ohta, and Jun'ichi Tsujii. Corpus annotation for mining biomedical events from literature. *BMC bioinformatics*, 9(1):10, 2008. 2

[21] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. 2017. 2

[22] Yanir Kleiman, Oliver van Kaick, Olga Sorkine-Hornung, and Daniel Cohen-Or. SHED: shape edit distance for fine-grained shape similarity. *ACM Transactions on Graphics (TOG)*, 34(6):1–11, 2015. 2

[23] Ranjitha Kumar, Jerry O Talton, Salman Ahmad, and Scott R Klemmer. Bricolage: example-based retargeting for web design. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pages 2197–2206, 2011. 2

[24] Priyadarshini Kumari, Ritesh Goru, Siddhartha Chaudhuri, and Subhasis Chaudhuri. Batch decorrelation for active metric learning. In *IJCAI-PRICAI*, 2020. 8

[25] Jianan Li, Tingfa Xu, Jianming Zhang, Aaron Hertzmann, and Jimei Yang. LayoutGAN: Generating graphic layouts with wireframe discriminator. In *International Conference on Learning Representations*, 2019. 1, 2

[26] Manyi Li, Akshay Gadi Patil, Kai Xu, Siddhartha Chaudhuri, Owais Khan, Ariel Shamir, Changhe Tu, Baoquan Chen, Daniel Cohen-Or, and Hao Zhang. GRAINS: Generative recursive autoencoders for indoor scenes. *ACM Transactions on Graphics (TOG)*, 38(2):1–16, 2019. 1, 7

[27] Yujia Li, Chenjie Gu, Thomas Dullien, Oriol Vinyals, and Pushmeet Kohli. Graph matching networks for learning the similarity of graph structured objects. In *ICML*, 2019. 1, 2, 3

[28] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. Gated graph sequence neural networks. 2016. 2, 5

[29] Thomas F Liu, Mark Craft, Jason Situ, Ersin Yumer, Radomir Mech, and Ranjitha Kumar. Learning design semantics for mobile apps. In *Proceedings of the 31st Annual ACM Symposium on User Interface Software and Technology*, pages 569–579, 2018. 2

[30] Dipu Manandhar, Dan Ruta, and John Collomosse. Learning structural similarity of user interface layouts using graph

networks. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020. 1, 2, 3, 5, 6, 7

[31] Christopher D Manning, Hinrich Schütze, and Prabhakar Raghavan. Chapter 8: Evaluation in information retrieval in "Introduction to Information Retrieval". pages 151–175. Cambridge university press, 2008. 5, 7

[32] Nelson Nauata, Kai-Hung Chang, Chin-Yi Cheng, Greg Mori, and Yasutaka Furukawa. House-gan: Relational generative adversarial networks for graph-constrained house layout generation. *Eur. Conf. Comput. Vis.*, 2020. 2

[33] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning layouts for single-page graphic designs. *IEEE transactions on visualization and computer graphics*, 20(8):1200–1213, 2014. 2

[34] Daniel Ritchie, Ankita Arvind Kejriwal, and Scott R Klemmer. d. tour: Style-based exploration of design example galleries. In *Proceedings of the 24th annual ACM symposium on User interface software and technology*, pages 165–174, 2011. 2

[35] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 5, 7

[36] Michael Schlichtkrull, Thomas N Kipf, Peter Bloem, Rianne Van Den Berg, Ivan Titov, and Max Welling. Modeling relational data with graph convolutional networks. In *European Semantic Web Conference*, pages 593–607. Springer, 2018. 2

[37] Amanda Swearngin, Mira Dontcheva, Wilmot Li, Joel Brandt, Morgan Dixon, and Andrew J Ko. Rewire: Interface design assistance from examples. In *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*, pages 1–12, 2018. 2

[38] Sou Tabata, Hiroki Yoshihara, Haruka Maeda, and Kei Yokoyama. Automatic layout generation for graphical design magazines. In *ACM SIGGRAPH 2019 Posters*, pages 1–2. 2019. 2

[39] Subarna Tripathi, Sharath Nittur Sridhar, Sairam Sundaresan, and Hanlin Tang. Compact scene graphs for layout composition and patch retrieval. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 0–0, 2019. 2

[40] Raoul Wessel, Ina Blümel, and Reinhard Klein. The room connectivity graph: Shape retrieval in the architectural domain. 2008. 2

[41] W John Wilbur, Andrey Rzhetsky, and Hagit Shatkay. New directions in biomedical text annotation: definitions, guidelines and corpus construction. *BMC bioinformatics*, 7(1):1–10, 2006. 2

[42] Wenming Wu, Xiao-Ming Fu, Rui Tang, Yuhan Wang, Yu-Hao Qi, and Ligang Liu. Data-driven interior plan generation for residential buildings. *ACM Transactions on Graphics (TOG)*, 38(6):1–12, 2019. 1, 2, 5, 6

[43] Kai Xu, Rui Ma, Hao Zhang, Chenyang Zhu, Ariel Shamir, Daniel Cohen-Or, and Hui Huang. Organizing heterogeneous scene collections through contextual focal points. *ACM Transactions on Graphics (TOG)*, 33(4):1–12, 2014. 2

[44] Ting Yao, Yingwei Pan, Yehao Li, and Tao Mei. Exploring visual relationship for image captioning. In *Proceedings of the European conference on computer vision (ECCV)*, pages 684–699, 2018. 3, 7

[45] Ziqi Zhang, Sam Chapman, and Fabio Ciravegna. A methodology towards effective and efficient manual document annotation: addressing annotator discrepancy and annotation quality. In *International Conference on Knowledge Engineering and Knowledge Management*, pages 301–315. Springer, 2010. 2

[46] Jia Zheng, Junfei Zhang, Jing Li, Rui Tang, Shenghua Gao, and Zihan Zhou. Structured3D: A Large Photo-realistic Dataset for Structured 3D Modeling. In *Eur. Conf. Comput. Vis.*, 2020. 1

[47] Xinru Zheng, Xiaotian Qiao, Ying Cao, and Rynson WH Lau. Content-aware generative modeling of graphic design layouts. *ACM Transactions on Graphics (TOG)*, 38(4):1–15, 2019. 1, 2

[48] Xu Zhong, Jianbin Tang, and Antonio Jimeno Yepes. Publaynet: largest dataset ever for document layout analysis. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1015–1022. IEEE, 2019. 1