LAYOUTGAN: GENERATING GRAPHIC LAYOUTS WITH WIREFRAME DISCRIMINATOR

Anonymous authors
Paper under double-blind review

ABSTRACT

Layouts are important for graphic design and scene generation. We propose a novel generative adversarial network, named as LayoutGAN, that synthesizes graphic layouts by modeling semantic and geometric relations of 2D elements. The generator of LayoutGAN takes as input a set of randomly placed 2D graphic elements and uses self-attention modules to refine their semantic and geometric parameters jointly to produce a meaningful layout. Accurate alignment is critical for good layouts. We thus propose a novel differentiable wireframe rendering layer that maps the generated layout to a wireframe image, upon which a CNN-based discriminator is used to optimize the layouts in visual domain. We validate the effectiveness of LayoutGAN in various experiments including MNIST digit generation, document layout generation, clipart abstract scene generation and tangram graphic design.

1 INTRODUCTION

Graphic design is an important visual communication tool in our modern world, encompassing everything from book covers to magazine layouts to web design. Whereas methods for generating realistic natural-looking images have made significant progress lately, e.g., with Generative Adversarial Networks (GANs) Karras et al. (2017), methods for creating designs are far more primitive. This is, in part, due to the difficulty of finding data representations suitable for learning. Graphic designs are normally composed by scalable primitive objects, such as polygons, curves and ellipses, instead of pixels laid on a regular lattice. The crucial semantic meaning of graphic design depends on the presence of elements, their attributes and their relations to other elements, which can be reflected by the layout of these elements on a page. The visual perception of design depends on the arrangement of these elements; misalignment of two elements of just a few millimeters can ruin the design. Training from images of designs using conventional GANs synthesizes the layouts in pixel-level, thus do not precisely characterize dependence among different elements and would be unlikey to capture arrangement and layout styles well. Modeling such highly-structured data using neural networks is of great interest as they usually represent human abstract knowledge about the visual world Zitnick & Parikh (2013); Song et al. (2017) and how these knowledge are expressed via documents and designs Deka et al. (2017a); Yang et al. (2017).

Instead of generating pixel-level images like conventional GANs, this paper introduces LayoutGAN, a novel GAN to explore the synthesis process of structured data, which directly synthesizes a set of relational graphic elements representing semantic layouts for graphic design. In our network, each element is represented by its class label and its geometric parameters, e.g., polygon keypoints. The generator takes as input graphic elements with randomly-sampled classes and geometric parameters, and arranges them in a design; the output is the contextually refined class probabilities and geometric parameters of the design elements. The generator has the desirable property of being permutation-invariant: it will generate the same layout if we re-order the input elements.

Following up the synthesized and real structured data, we propose two kinds of discriminator networks. The first one, is similar in structure to the generator: it operates directly on the class probabilities and geometric parameters of the elements, and learns to extract the semantic and geometric relations among different elements for layout optimization from graphic domain. Though effective, it is not sensitive enough to the misalignment and occlusion of different elements, which is crucial to graphic layout designs. The second one is in visual domain. Like a human viewer who judge a graphic design by looking at its rasterized image, the spatial-semantic relations among different

elements can be well reflected in visual domain by mapping them to 2D layouts. Then CNNs can be used for layout optimization as they are specialized in distinguishing visual patterns including but not limited to misalignment and occlusion. However, the key challenge is how to map the geometric parameters to pixel-level layouts differentiably. Apparently, one trial is to render the graphic elements into bitmap masks using STN Jaderberg et al. (2015). But we found the filled pixels within the design elements such as polygons cause occlusion issue and are ineffective for back-propagation (for example, a small polygon hides inside a larger one). We experimented with bitmap mask rendering but it was not successful. In this paper, we propose a novel differentiable wireframe rendering layer that raterizes both synthesized and real structured data of graphic elements into wireframe images, upon which a standard CNN can be used to optimize the layout across the visual and the graphic domain. The wireframe rendering discriminator has several advantages. First, convolution layers are very good at extracting spatial patterns of images so that they are more sensitive to alignment. Second, the rendered wireframes make it visible how two elements may overlap and thus the network is alleviated from inferring the occlusions that may occur in other renderings such as masks.

We evaluate the LayoutGAN for several different tasks. In the MNIST experiments, we represent each digit by a set of points, which is the simplest geometric form to generate digits for sanity test and evaluate the output with inception scores Salimans et al. (2016). In document experiments, we generate page layouts with a set of semantically-labeled bounding boxes, and compare with the results from conventional GANs. In clipart abstract scene experiments, we show our LayoutGAN can accurately model objects' pairwise semantic and geometric relations by arranging the layout of a set of specific scene elements to synthesize clipart abstract scenes. We further demonstrate that our method can generalize to more complex elements in tangram graphic design and show how our LayoutGAN can be used to optimize randomly-perturbed layouts. In summary, this work makes the following contributions. 1. Unlike conventional GANs that generates pixel-level images structurally, the proposed LayoutGAN directly synthesizes structured data, that is a set of graphic elements to form meaningful layouts, which is totally resolution free. 2. A novel differentiable wireframe rendering layer is proposed to rasterize graphic elements to wireframe images, allowing the discriminator to judge alignment from discrete element arrangements.

2 Related Work

Structured data generation. Convolutional networks have been shown successful for generating data in regular lattice, such as images Radford et al. (2015), videos Vondrick et al. (2016) and 3D volumes Yan et al. (2016); Wu et al. (2016). When generating highly-structured data, such as texts Donahue et al. (2015) and programs Reed & De Freitas (2015), recurrent networks are often the first choice Sutskever et al. (2014), esepcially equipped with attention Bahdanau et al. (2014) and memory modules Graves et al. (2014). Recently, researchers show that convolutional networks can be also used to synthesize sequences van den Oord et al. (2016); Van Den Oord et al. (2016) using auto-regressive models. However, in many cases, a datum cannot be expressed as a sequence Vinyals et al. (2015), but just a set of elements, e.g. point clouds. Fan et al. (2017) propose a point set generation network for synthesizing 3D point clouds of the object shape from a single images. It is further paired with a point set classification network Qi et al. (2017) for auto-encoding 3D point clouds Achlioptas et al. (2017). Our work extends the set repsentation to more general priminitive objects, i.e. semantically-labeled polygons. Meanwhile, researchers also model the structured data of connected elements using graph convolutions Kipf & Welling (2016).

Data-driven graphic design. Automating layout is a classic problem in graphic design Hurst et al. (2009). O'Donovan et al. (2014a) formulate an energy function by assembling various heuristic visual cues and design principles to optimize single-page layouts, and extend this to an interactive tool O'Donovan et al. (2015). The model parameters are learned from a small number of example designs. Pang et al. (2016) optimize layout for desired gaze direction. Deka et al. (2017a) collect a mobile app design database for harnessing data-driven applications and present preliminary results of learning similarities of pixel-level textural/non-textual masks for design search, but do not learn models from this data. Swearngin et al. (2018) propose an interactive system that converts example design screenshots to vector graphics for designers to re-use and edit. Bylinskii et al. (2017) analyze the visual importance of graphic designs and use saliency map as the driving force to assist retargeting and thumbnailing. Previous methods have learned models for other graphic design elements, such as fonts O'Donovan et al. (2014b) and colors O'Donovan et al. (2011). These are orthogonal to the layout problem, and could be combined in future work. No previous method has learned to create design or layout from large datasets, and no previous work has applied GANs to layout.

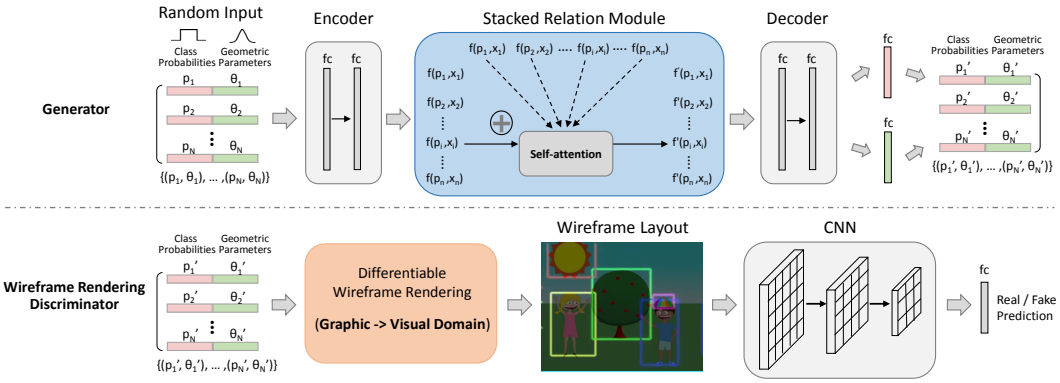


Figure 1: Overall architecture of LayoutGAN. The generator takes as input graphic elements with randomly sampled class labels and geometric parameters from Uniform and Gaussian distribution respectively. An encoder embeds the input and feeds them into the stacked relation module, which refines the embedded features of each element in a coordinative manner by considering its semantic and spatial relations with all the other elements. Finally, a decoder decodes the refined features back to class probabilities and geometric parameters. The wireframe rendering discriminator feeds the generated results to a differentiable wireframe rendering layer which raterizes the input graphic elements into 2D wireframe images, upon which a CNN is applied for layout optimization.

3D scene synthesis. Interior scene synthesis and furniture layout generation draws great interest in graphics community. Early approaches focus on optimizaton of hand-crafted design principles Merrell et al. (2011) and learning statistical priors of pairwise object relationships Fisher et al. (2012) due to limited data. Kai Wang & Ritchie (2018) recently propose a sequential decision making approach to indoor scene synthesis. In each step, a CNN is trained to predict either location or category of one object by looking at the rendered top-down views. This bears resemblance to our wireframe rendering discriminator in spirit of using convolutions to capture spatial patterns of layouts.

3 LayoutGAN

3.1 Overview

Instead of generating pixel-level images, the LayoutGAN synthesizes a set of (total number of N) relational graphic elements represented as $\{(p_1, \theta_1), \cdots, (p_N, \theta_N)\}$, where p is the class probability and $\theta = \left[\theta^1, ..., \theta^m\right]$ denote geometric parameters (total number of m) that can be in various forms for different graphic layouts. For example, $\theta = [X, Y]$ for 2D point set generation (MNIST digit) and $\theta = \left[X^1, Y^1, X^2, Y^2\right]$ for bounding box generation (document layout). It can be easily extended to other geometric forms depending on different tasks, for example, $\theta = [X, Y, S, F]$ for graphic layouts considering position, scale and flip simultaneously (clipart abstract scene). In conjunction with class probability p, LayoutGAN is flexible to synthesize graphic layouts of different elements with various geometric forms as needed.

To synthesize semantic layouts of relational graphic elements, we introduce new designs in both the generator and the discriminator of our LayoutGAN. The generator aims to learn a function G that takes as input layout $z = \{(p_1, \theta_1), \cdots, (p_N, \theta_N)\}$ consisting of initial graphic elements with randomly sampled class labels and geometric parameters $(p_i, \theta_i), (i = 1, \cdots, N)$, and outputs contextually refined $(p_i{}', \theta_i{}')$ for element i in z to form a new graphical layout $G(z) = \left\{(p_1{}', \theta_1{}'), \cdots, (p_N{}', \theta_N{}')\right\}$ which resembles a real graphic layout. Note that different from vanilla GANs where z represents a low-dimensional latent variable, our z represents initial random graphic layout that has the same structure as the real one. While the discriminator learns to capture the semantic and geometric relations among different elements for layout optimization from both the graphic domain and the visual domain. Next, we go into the details of the generator and discriminator design.

3.2 Generator Network Architecture

As shown in Figure 1, the generator takes as input a set of graphic elements with random class labels and geometric parameters sampled from Uniform and Gaussian distribution respectively. An encoder consisting of a multilayer perceptron network (implemented as multiple fully connected layers) first embeds the class label and geometric parameters of each graphic element. The relation

module implemented as self-attention module inspired by Wang et al. (2017) is then used to refine the embedded features of each graphic element contextually by modeling its semantic and spatial relations with all the other elements in the set. Denote $f(p_i, \theta_i)$ as the embedded feature of the graphic element i, its refined feature representation $f'(p_i, \theta_i)$ can be obtained through a contextual residual learning process, which is defined as:

$$f'(p_i, \theta_i) = W_r \frac{1}{N} \sum_{\forall j} H(f(p_i, \theta_i), f(p_j, \theta_j)) U(f(p_j, \theta_j)) + f(p_i, \theta_i). \tag{1}$$

Here j is the element index and the unary function U computes a representation of the embedded feature $f(p_j, \theta_j)$ of element j. The pairwise function H computes a scalar value representing the relation between elements i and j. Thus, all the other elements $j \neq i$ contribute to the feature refinement of element i by summing up their relations. The response is normalized by the total number of elements in the set, N. The weight matrix W_r computes a linear embedding, producing the contextual residual to be added on $f(p_i, \theta_i)$ for feature refinement. In our experiments, we define H as a dot-product:

$$H(f(p_i, \theta_i), f(p_j, \theta_j)) = \delta(f(p_i, \theta_i))^T \phi(f(p_j, \theta_j)), \tag{2}$$

where $\delta(f(p_i, \theta_i)) = W_\delta f(p_i, \theta_i)$ and $\phi(f(p_j, \theta_j)) = W_\phi f(p_j, \theta_j)$ are two linear embeddings. We stack $T = 4$ relation modules for feature refinement in our experiments. Finally, a decoder consisting of another multilayer perceptron network followed by two branches of fully connected layer with sigmoid activation are used to map the refined feature of each element back to class probabilities and geometric parameters respectively. Optionally, Non-Maximum Suppression (NMS) can be applied to remove duplicated elements.

3.3 Discriminator network architectures

The discriminator aims to distinguish between the synthesized and the real layouts. We have explored two approaches from graphic and visual domains respectively. For the graphic domain solution, denoted as the relation-based discriminator, is to directly extract the relations among different graphic elements in the parameter space for layout optimization. In contrast, the visual domain solution, denoted as the wireframe rendering discriminator, a differentiable wireframe rendering layer is proposed to map graphic elements to 2D wireframe images, thus CNNs are applied to optimize the layout from visual domain.

3.3.1 Relation-based discriminator

The relation-based discriminator embeds the input graphic elements and extracts their graphical relations for layout optimization from graphic domain. It takes as input a set of graphic elements represented by class probability and geometric parameters, and feeds them to an encoder consisting of a multilayer perceptron network for feature embedding $f(p_i, \theta_i)$ and then extract their global graphical relations among different elements $g(r(p_1, \theta_1), \cdots, r(p_N, \theta_N))$ where $r(p_i, \theta_i)$ is obtained from the similar relation module as that in generator but without shortcut connection, and g is a max-pooling function Qi et al. (2017). Thus, the global relations among all graphic elements can be modeled, upon which a classifier composed by a multilayer perception network with sigmoid activation is applied for real/fake prediction.

3.3.2 Wireframe rendering discriminator

To take advantage of CNNs to learn visual patterns for effcient layout optimization, a key issue is to map graphic elements to 2D images in a differentiable way, which is essentially a process of rasterization. Let's consider to rasterize a graphic layout with N elements, denoted as $\{(p_1, \theta_1), ..., (p_N, \theta_N)\}$ onto a target image $I(X^t, Y^t)$, where (X^t, Y^t) is the location in pre-defined regular grid. Assuming there are C semantic classes for each element, the target image I is thus of C channels. The pixel $I(X^t, Y^t)$ on the location (X^t, Y^t) in the rendered image can be calculated through class-wise maximum operation of the rendered class probability distributions on (X^t, Y^t) of all elements, generally formulated as:

$$I(X^t, Y^t) = \max_{i=1,\ldots,N} R((p_i, \theta_i), (X^t, Y^t)), \tag{3}$$

where $R((p_i, \theta_i), (X^t, Y^t))$ denotes the rendering process as:

$$R((p_i, \theta_i), (X^t, Y^t)) = p_i \cdot F((X^t, Y^t), \theta_i), \tag{4}$$

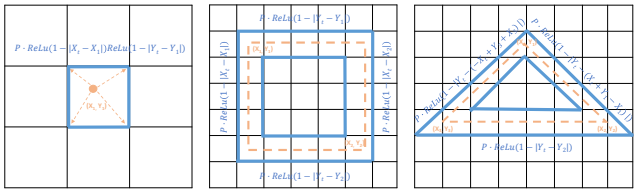


Figure 2: Wireframe rendering of different polygons (point, rectangle and triangle). The black grids represent grids of target image. The orange dots/dotted lines represent the graphic element mapped onto the image grid. The blue solid lines represent the rasterized wireframes expressed as differentiable functions of graphic elements in terms of both class probilities and geometric parameters.

where function F computes the rasterization, which varies for different geometric forms of graphic elements. In this paper, we focus on polygons $\theta_i = (X_i^1, Y_i^1, ..., X_i^K, Y_i^K)$ with K keypoints.

We start with the simplest geometric form, a single keypoint $\theta_i = (X_i^1, Y_i^1)$ for element i. We implement an interpolation kernel k for its rasterization. Its spatial rendering response on (X^t, Y^t) in the rendered image can be written as:

$$F((X^t, Y^t), (X_i^1, Y_i^1)) = k(X^t - X_i^1)k(Y^t - Y_i^1). \tag{5}$$

We adopt bilinear interpretation Johnson et al. (2016), corresponding to the kernel $k(d) = max(0, 1 - |d|)$ (implemented as ReLU activation), as shown in Figure 2. As $R((p_i, \theta_i), (X^t, Y^t))$ is a linear function of the class probability and the coordinates, gradients can be propagated backward in to them both. We validate such rendering design for MNIST digit generation, detailed experiments can be seen in Section 4.1.

We now consider more complex polygons. Assuming an element is a rectangle, or bounding box represented by its top-left and bottom-right coordinates $\theta = (X^1, Y^1, X^2, Y^2)$, which is very common in various designs. Specifically, considering a rectangle i with coordinates $(X_i^1, Y_i^1, X_i^2, Y_i^2)$, as shown in Figure 2, the black grids represent the locations in the rendered image and the orange dotted box represents the rectangle being rasterized in the rendered image. For a wireframe representation, only the points near the boundary of the dotted box (lie in blue solid line) are related to the rectangle, so its spatial rendering response on (X_t, Y_t) can be formulated as:

$$F((X^t, Y^t), (X_i^1, Y_i^1, X_i^2, Y_i^2)) = \max \begin{pmatrix} k(X^t - X_i^1)b(Y^t - Y_i^1)b(Y_i^2 - Y^t), \\ k(X^t - X_i^2)b(Y^t - Y_i^1)b(Y_i^2 - Y^t), \\ k(Y^t - Y_i^1)b(X^t - X_i^1)b(X_i^2 - X^t), \\ k(Y^t - Y_i^2)b(X^t - X_i^1)b(X_i^2 - X^t) \end{pmatrix}, \tag{6}$$

where $b(d) = \min(\max(0, d), 1)$ constraining the rendering to nearby pixels.

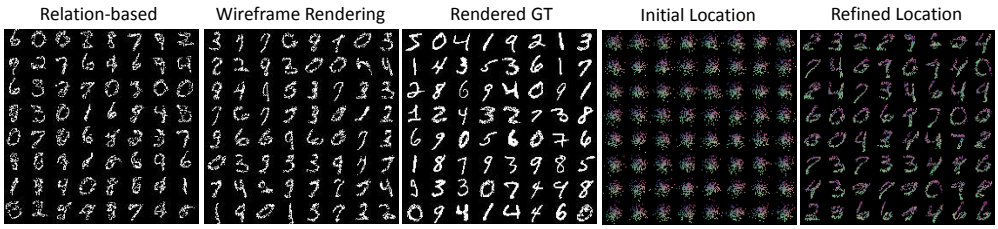
We further describe the wireframe rendering process of another geometric form, triangle. For triangle i represented by its three vertices' coordinates $\theta_i = (X_i^1, Y_i^1, X_i^2, Y_i^2, X_i^3, Y_i^3)$, its spatial rendering response on (X^t, Y^t) in the rendered image can be calculated as:

$$\begin{aligned} &F((X^t, Y^t), (X_i^1, Y_i^1, X_i^2, Y_i^2, X_i^3, Y_i^3)) = \\ &\max \begin{pmatrix} k(Y^t - \frac{(Y_i^2 - Y_i^1)\cdot(X^t - X_i^1)}{X_i^2 - X_i^1} - Y_i^1)b(X^t - X_i^1)b(X_i^2 - X^t), \\ k(Y^t - \frac{(Y_i^3 - Y_i^1)\cdot(X^t - X_i^1)}{X_i^3 - X_i^1} - Y_i^1)b(X^t - X_i^3)b(X_i^1 - X^t), \\ k(Y^t - \frac{(Y_i^3 - Y_i^2)\cdot(X^t - X_i^2)}{X_i^3 - X_i^2} - Y_i^2)b(X^t - X_i^3)b(X_i^2 - X^t) \end{pmatrix}, \end{aligned} \tag{7}$$

Through this wireframe rendering process, gradients can be propagated backward to both the class probability and geometric parameters of the graphic elements for joint optimization. A CNN consisting of 3 convolutional layers followed by a fully connected layer with sigmoid activation is then used for predicting real/fake graphic layouts.

4 Experiments

The implementation is based on TensorFlow Abadi et al. (2016). The network parameters are initialized from zero-mean Gaussian with standard deviation of 0.02. All the networks are optimized using Adam Kingma & Ba (2014) with a fixed learning rate of 0.00002. Detailed architectures can be found in the appendix.



(a) Comparing generated MNIST samples. (b) Point tracing (best viewed in color).

Figure 3: Results on MNIST digit generation.

Table 1: Inception scores for generated digits.

Methods	Score $\pm$ std
Relation	$6.53 \pm .09$
Wireframe	$7.36 \pm .07$
Real data	$9.81 \pm .08$

Table 2: Spatial analysis of document layout.

Methods	Overlap(%)	Alignment(%)
Relation	1.52	6.4
Wireframe	1.17	3.4
Real data	0.05	0.5

4.1 MNIST DIGITS GENERATION

MNIST is a handwritten digit database consisting of 60,000 training and 10,000 testing images. For each image, we extract the locations of 128 randomly-selected foreground pixels as the graphic representation so that digit generation can be formulated as layout generation of 2D points. In Figure 3a, each image shows 8×8 digits rendered from point layouts. The left and middle images are the generated samples by the LayoutGAN with wireframe rendering discriminator and the one with relation-based discriminator, respectively. The right image shows the digits rendered from ground-truth point layouts. One can see that the LayoutGAN with either discriminator well capture various digit patterns by collaboratively refining the coordinates of all the points through relation modeling. Particularly, the wireframe rendering discriminator helps generate more compact, better aligned point layouts. For quantitative evaluation, we train a multilayer perceptron network for digits classification, which achieves an accuracy of 98.91% on MNIST test set, and use it to compute the inception score Salimans et al. (2016) of 10,000 generated samples. As shown in Table 1, The LayoutGAN with wireframe rendering discriminator achieves a higher inception score than the one with relation-based discriminator. To shed light on the relational refinement process of LayoutGAN, we mark each input random point with a location-specific color, and track their refined locations in the generated layouts, as shown in Figure 3b. One can see that colors change gradually along the strokes, showing the network learns some contextually consistent local displacements of the points.

4.2 DOCUMENT SEMANTIC LAYOUT GENERATION

One document page consists of a number of regions with different semantic roles, such as heading, paragraph, table, figure, caption and list. Each region is represented by a bounding box. Modeling layouts of semantic regions is critical for document analysis, retargeting and synthesis. In real document data, these semantically-labeled bounding boxes are often well aligned according to some axises and their placement follows some particular patterns, such as heading always appears above paragraph or table (some document page examples can be found in the appendix). In this experiment, we focus on one-column layouts with no more than 9 bounding boxes that may belong to 6 semantic classes as mentioned above. We are interested how these spatial-semantic patterns can be captured by our network. For training data, we collect totally around 25,000 layouts from real document pages. We also implement a baseline method that represents the layout by semantic masks as used in Yang et al. (2017); Deka et al. (2017a). Particularly, we render all semantic regions into masks and train a DCGAN Radford et al. (2015) for generating mask layouts. Then we extract the connected mask regions of each semantic class and output enclosing bounding boxes of all regions. Figure 4 shows some representative generation results (each row consists of 6 samples, high-resolution results can be found in the appendix). The first row shows the results from DCGAN. One can see that the generated results suffer from misalignment among regions. In addition, different semantic blocks are in poor arrangement, showing the network cannot well capture the spatial-semantic patterns through layout rendering in pixel-level. The third row shows the results from LayoutGAN with wireframe rendering discriminator. We retrieve the most similar real layouts from the training set in the last row as references. It can be seen that LayoutGAN can well capture different document layout patterns with clear definitions of instance-level semantic bounding boxes.

To validate the advantage of wireframe rendering discriminator, we compare the generated results with those from the LayoutGAN with relation-based discriminator. As shown in the second row of

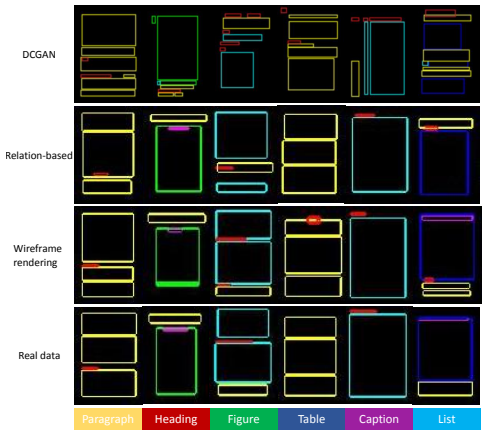


Figure 4: Document layout comparison.

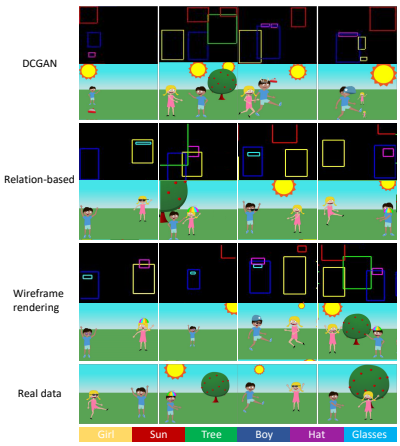


Figure 5: Clipart abstract scene generation.

Figure 4, the latter can also capture different layout patterns but sometimes suffers from overlapping and misalignment issues. As the bounding boxes in real document layouts are either left- or center-aligned with no overlappings with each other (except for caption with figure/table), we propose two metrics to quantitatively measure the quality of the generated layouts. The first one is overlapping index, which is the percentage of total overlapping area among any two bounding boxes inside the whole page. The second one is alignment index which is calculated by finding the minimum standard deviation of either the left or the center coordinates of all the bounding boxes. Table 2 provides quantitative comparisons of real layouts and synthesized layouts from LayoutGAN with two different discriminators. One can see that LayoutGAN with wireframe rendering discriminator achieves lower value in both overlapping index and alignment index than the one with relation-based discriminator, validating the superiority of the proposed wireframe rendering method for layout generation. Similar conclusion can also be drawn by analyzing the loss functions. We add shift perturbations to the bounding boxes of real layouts and feed them to both discriminators to examine their loss behaviors. Figure 6 visualizes the loss landscape of both discriminators corresponding to different extent of shift perturbation. One can see that the loss surface in terms of shift perturbation of the wireframe rendering discriminator is much smoother than that of relation-based discriminator.

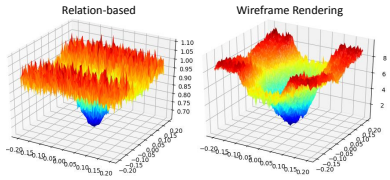


Figure 6: Discriminator loss landscapes.

Table 3: User study results for Clipart abstract scenes.

Methods	Excellent (%)	Fair (%)	Poor (%)
DCGAN	6.7	38.8	54.5
Relation	17.2	50.3	32.5
Wireframe	37.3	48.0	14.7

4.3 CLIPART ABSTRACT SCENE GENERATION

For better visualizing the element's pairwise semantic and geometric relations captured by the LayoutGAN, we consider abstract scene generation, that is to synthesize abstract scenes by arranging a set of specific clipart scene elements (boy, girl, glasses, hat, sun, and tree in our experiments) tailored from the Clipart dataset Zitnick et al. (2016). To synthesize a reasonable abstract scene, we first use LayoutGAN to generate the layout of inner scene elements by representing each of them as semantically-labeled bounding box with normalized center coordinates, width and height, and flip attribute. Then, we render the corresponding clipart image of each scene element onto a background image according to the predicted position (center coordinates), scale (width and height) and flip attribute, forming synthesized abstract scenes. Note that it is challenging as accurate objects' pairwise spatial-semantic relations are required for reasonable scenes, for example, the glasses/hat should be exactly on the eyes/head of the boy/girl with matching scales and flips simultaneously. As shown in Figure 5, the first three rows show the generated layouts and corresponding rendered scenes by DCGAN, the LayoutGAN with relation-based discriminator and the one with wireframe rendering discriminator respectively. The last row shows some samples rendered from ground-truth scene layouts. It can be seen that compared to DCGAN and the LayoutGAN with relation-based discriminator, the one with wireframe rendering discriminator can capture the objects' pairwise relations more precisely, forming accurate scene layouts (for example, the glasses/hat accurately lies on the eyes/head of the person with varying scales and flips), validating the superiority of the pro-

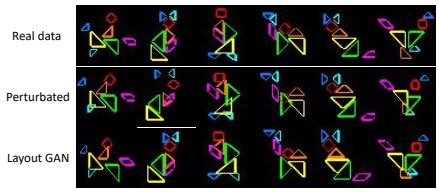


Figure 7: Optimizing perturbed tangram.

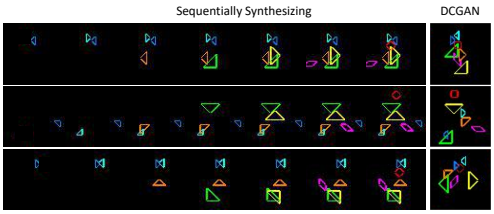


Figure 8: Baseline results of tangram design.

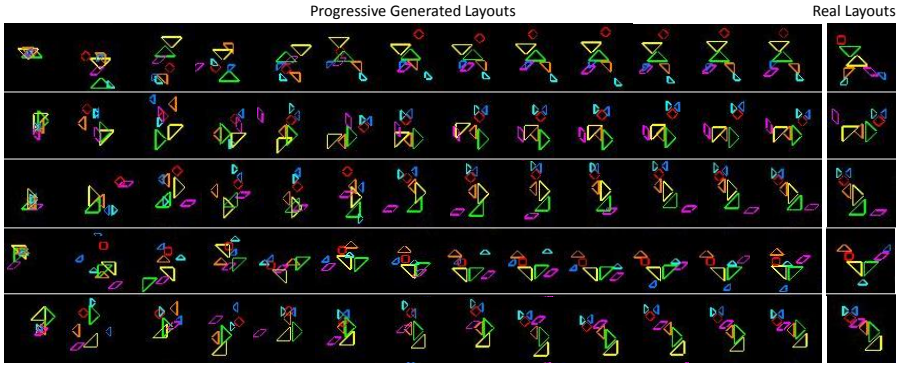


Figure 9: Training progression of tangram graphic design generation (from left to right).

posed wireframe rendering strategy. In addition, we conduct a user study involving 20 respondents for a subjective evaluation of the results. Specifically, we randomly sample a same number of abstract scenes synthesized by the above three models. A subject is asked to rate the synthetic scenes in terms of three scores (Excellent, Fair, Poor) by following three guidelines: 1) whether they have coherent overall structures, 2) whether different objects are placed in reasonable relative positions, scales and flips, and 3) whether there are duplicate objects. We compute the rating percentage of all sampled scenes for each model. As shown in Table 3, the results of LayoutGAN with the wireframe rendering discriminator receive better ratings, which suggests that they are preferred by respondents.

4.4 Tangram graphic design

The tangram is a dissection puzzle aiming to form a specific shape using seven pieces of flat shapes without overlapping. The seven pieces include two large right triangles, one medium right triangle, two small right triangles, one square and one parallelogram, which can be assembled to form a square of side one unit and having area one square unit when choosing a unit of measurement. We collect totally 149 tangram graphic designs including animals, people and objects. In our experiments, we consider eight rotation/reflection poses for each piece. Given the seven pieces initialized with random location and ground-truth pose and class, the LayoutGAN with wireframe rendering discriminator is trained to refine their configurations jointly to form reasonable layouts. Note that it is a challenging task due to the complex configurations and limited data. We perform two experiments. The first one is a perturbation recovering test, in which we first add some random perturbations to piece locations of real layouts and train a LayoutGAN to recover the true layouts. Figure 7 shows that our network is able to pull back the displaced shapes to the right locations, confirming that the network successfully figures out the relations between different graphic elements. We further train another LayoutGAN to generate tangram designs from purely random initialization. In Figure 9 (high-resolution results can be found in the appendix), each row represents the sampled progressive generated results and the retrieved similar real tangrams. We also present some generated results of DCGAN and a sequential approach that follows Wang et al. (2018) for comparison. One can see from Figure 8 that DCGAN cannot well model the spatial relations among different shapes while the sequential model suffers from accumulated error. In contrast, the LayoutGAN can generate meaningful tangrams like fox and person, although others may be hard to interpret, as shown in the penultimate row.

5 Conclusion

In this paper, we have proposed a novel LayoutGAN for generating layouts of relational graphic elements. Different from traditional GANs that generate images in pixel-level, the LayoutGAN can directly output a set of relational graphic elements. A novel differentiable wireframe rendering layer was proposed to rasterize the generated graphic elements to wireframe images, making it feasible to leverage CNNs as discriminator for better layout optimization from visual domain. Future works include adding a content representation, such as text, icon and picture to each graphic element.

REFERENCES

Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, et al. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*, 2016.

Panos Achlioptas, Olga Diamanti, Ioannis Mitliagkas, and Leonidas Guibas. Learning representations and generative models for 3d point clouds. *arXiv preprint arXiv:1707.02392*, 2017.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Zoya Bylinskii, Nam Wook Kim, Peter O'Donovan, Sami Alsheikh, Spandan Madan, Hanspeter Pfister, Fredo Durand, Bryan Russell, and Aaron Hertzmann. Learning visual importance for graphic designs and data visualizations. In *Proceedings of the 30th Annual ACM Symposium on User Interface Software & Technology*, 2017.

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*, UIST '17, pp. 845–854, New York, NY, USA, 2017a. ACM. ISBN 978-1-4503-4981-9. doi: 10.1145/3126594.3126651. URL `http://doi.acm.org/10.1145/3126594.3126651`.

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*, pp. 845–854. ACM, 2017b.

Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2625–2634, 2015.

Haoqiang Fan, Hao Su, and Leonidas Guibas. A point set generation network for 3d object reconstruction from a single image. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, volume 38, 2017.

Matthew Fisher, Daniel Ritchie, Manolis Savva, Thomas Funkhouser, and Pat Hanrahan. Example-based synthesis of 3d object arrangements. *ACM Transactions on Graphics (TOG)*, 31(6):135, 2012.

Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

N. Hurst, W. Li, and K. Marriott. Review of automatic document formatting. In *Proc. DocEng*, 2009.

Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *Advances in neural information processing systems*, pp. 2017–2025, 2015.

Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 4565–4574, 2016.

Angel X. Chang Kai Wang, Manolis Savva and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 2018.

Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016.

Paul Merrell, Eric Schkufza, Zeyang Li, Maneesh Agrawala, and Vladlen Koltun. Interactive furniture layout using interior design guidelines. *ACM Transactions on Graphics (TOG)*, 30(4):87, 2011.

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Color compatibility from large datasets. *ACM Trans. Graphics*, 2011.

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, 2014a.

Peter O'Donovan, Jānis Lībeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory font selection using crowdsourced attributes. *ACM Trans. Graphics*, 2014b.

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Designscape: Design with interactive layout suggestions. In *Proc. CHI*, 2015.

Xufang Pang, Ying Cao, Rynson W. H. Lau, and Antoni B. Chan. Directing user attention via visual flow on web designs. *ACM Trans. Graph.*, 35(6):240:1–240:11, November 2016.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Proc. Computer Vision and Pattern Recognition (CVPR), IEEE*, 1(2):4, 2017.

Alec Radford, Luke Metz, and Soumith Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

Scott Reed and Nando De Freitas. Neural programmer-interpreters. *arXiv preprint arXiv:1511.06279*, 2015.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems*, pp. 2234–2242, 2016.

Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, pp. 190–198. IEEE, 2017.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

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*, pp. 504. ACM, 2018.

Aaron Van Den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Oriol Vinyals, Alex Graves, et al. Conditional image generation with pixelcnn decoders. In *Advances in Neural Information Processing Systems*, pp. 4790–4798, 2016.

Oriol Vinyals, Samy Bengio, and Manjunath Kudlur. Order matters: Sequence to sequence for sets. *arXiv preprint arXiv:1511.06391*, 2015.

Carl Vondrick, Hamed Pirsiavash, and Antonio Torralba. Generating videos with scene dynamics. In *Advances In Neural Information Processing Systems*, pp. 613–621, 2016.

Kai Wang, Manolis Savva, Angel X Chang, and Daniel Ritchie. Deep convolutional priors for indoor scene synthesis. *ACM Transactions on Graphics (TOG)*, 37(4):70, 2018.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. *arXiv preprint arXiv:1711.07971*, 2017.

Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Advances in Neural Information Processing Systems*, pp. 82–90, 2016.

Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *Advances in Neural Information Processing Systems*, pp. 1696–1704, 2016.

Xiao Yang, Ersin Yumer, Paul Asente, Mike Kraley, Daniel Kifer, and C Lee Giles. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. *arXiv preprint arXiv:1706.02337*, 2017.

C Lawrence Zitnick and Devi Parikh. Bringing semantics into focus using visual abstraction. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pp. 3009–3016. IEEE, 2013.

C Lawrence Zitnick, Ramakrishna Vedantam, and Devi Parikh. Adopting abstract images for semantic scene understanding. *IEEE transactions on pattern analysis and machine intelligence*, 38 (4):627–638, 2016.

6 APPENDIX

6.1 REAL DOCUMENT PAGES

For better understanding the document semantic layouts, we show some layout samples and their corresponding real document pages, as shown in Figure 1.

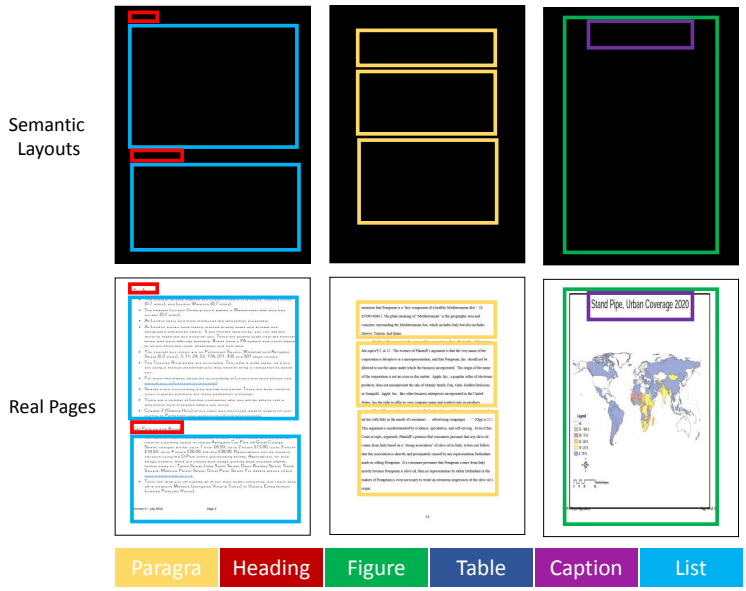


Figure 1: Visualization of document layout samples and their corresponding real document pages.

6.2 HIGH-RESOLUTION SYNTHESIZED DOCUMENT LAYOUTS

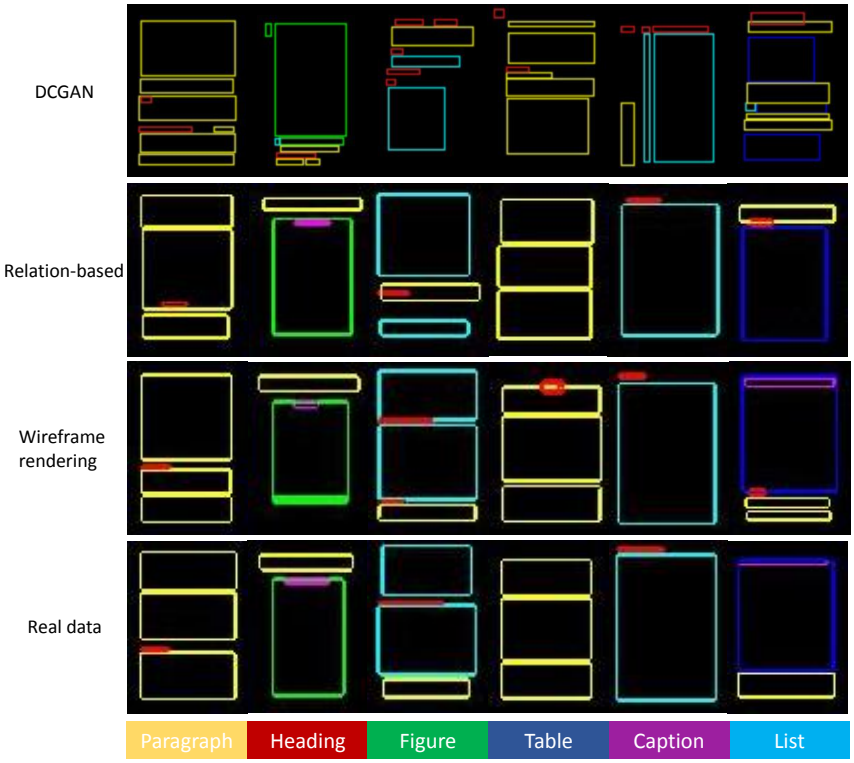


Figure 2: Document layout generation.

6.3 HIGH-RESOLUTION SYNTHESIZED CLIPART ABSTRACT SCENES

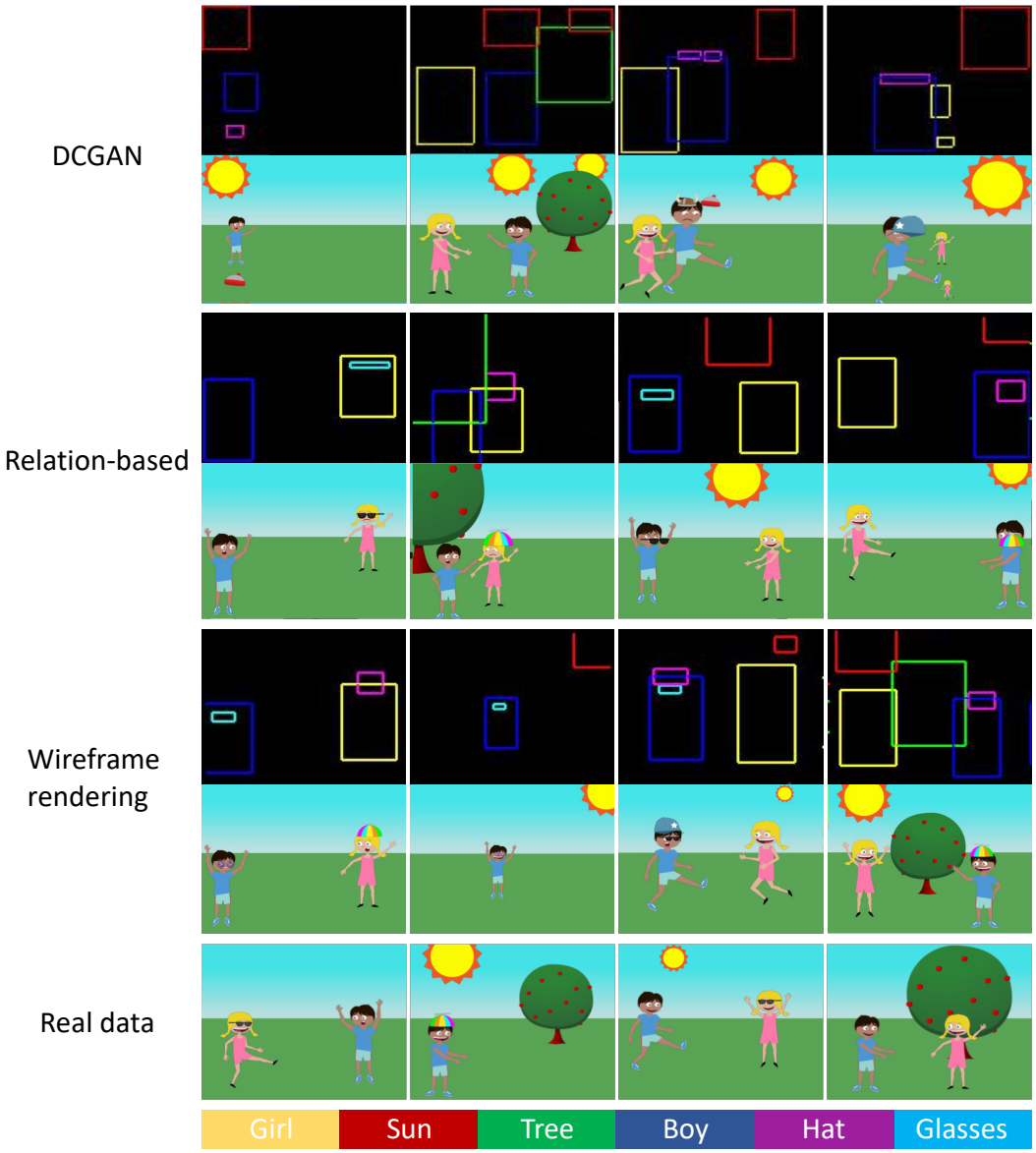


Figure 3: Clipart abstract scene generation.

6.4 HIGH-RESOLUTION SYNTHESIZED TANGRAM GRAPHIC DESIGNS

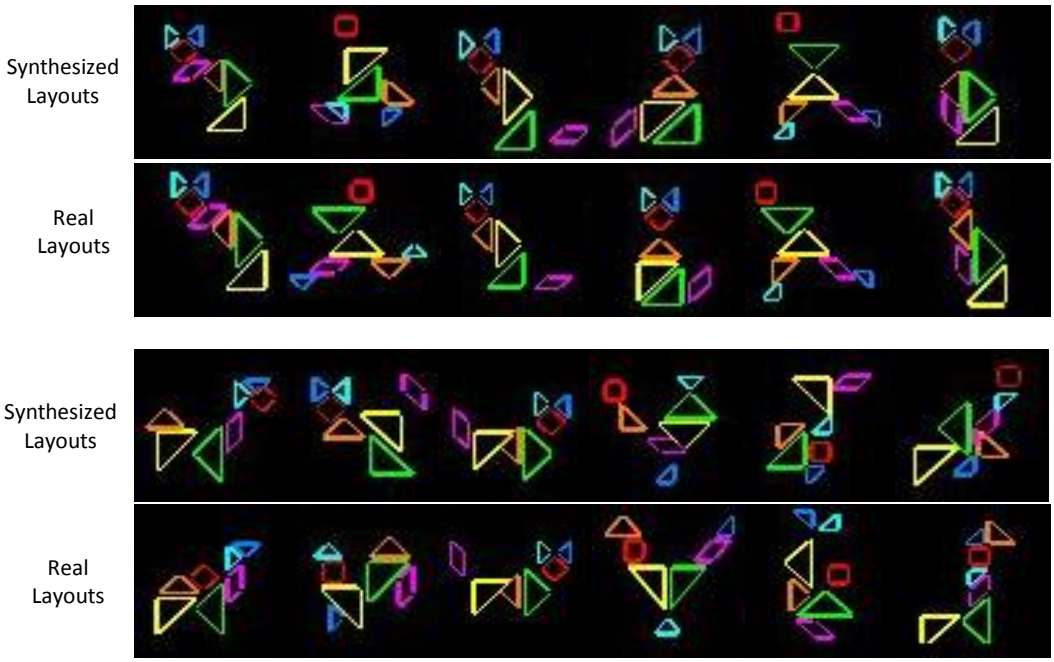


Figure 4: Tangram graphic design.

6.5 DETAILED ARCHITECTURE DESIGN OF THE GENERATOR

The generator takes as input a set of graphic elements with random class labels and geometric parameters sampled from Uniform and Gaussian distribution respectively. An encoder consisting of three fully connected layers first embeds the semantic and geometric parameters of each graphic element. Two cascaded relation modules implemented as self-attention module followed by a basic residual block (bottleneck building block) form one relation block. We cascade two such relation blocks (totally 4 relation modules) for contextual feature refinement. Finally, a decoder consisting of several fully connected layers followed by two heads (implemented as the fully connected layer) with sigmoid activation are used to decode the refined feature of each element back to class probability distributions and geometric parameters respectively.

6.6 SYNTHESIZED MOBILE APP LAYOUT DESIGNS

We further conduct a new experiment of mobile app layout design by using the RICO dataset Deka et al. (2017b). For better understanding the mobile app layouts, we first show some layout samples and their corresponding real mobile app screenshots, as shown in Figure 5.

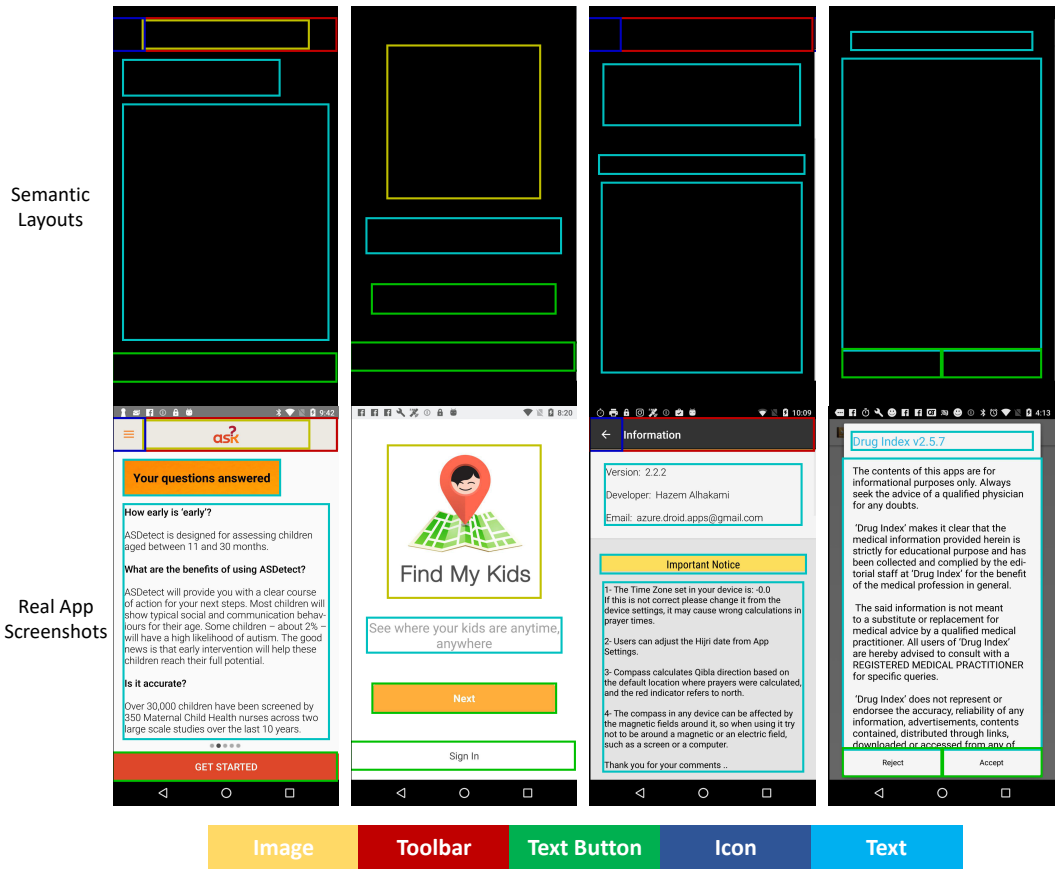


Figure 5: Visualization of mobile app layout samples and their corresponding real mobile app screenshots.

Figure 6 shows the results from LayoutGAN and the retrieved most-similar real layouts from the training set as references. Both the wireframe and the corresponding mask layouts are provided for better visualization.

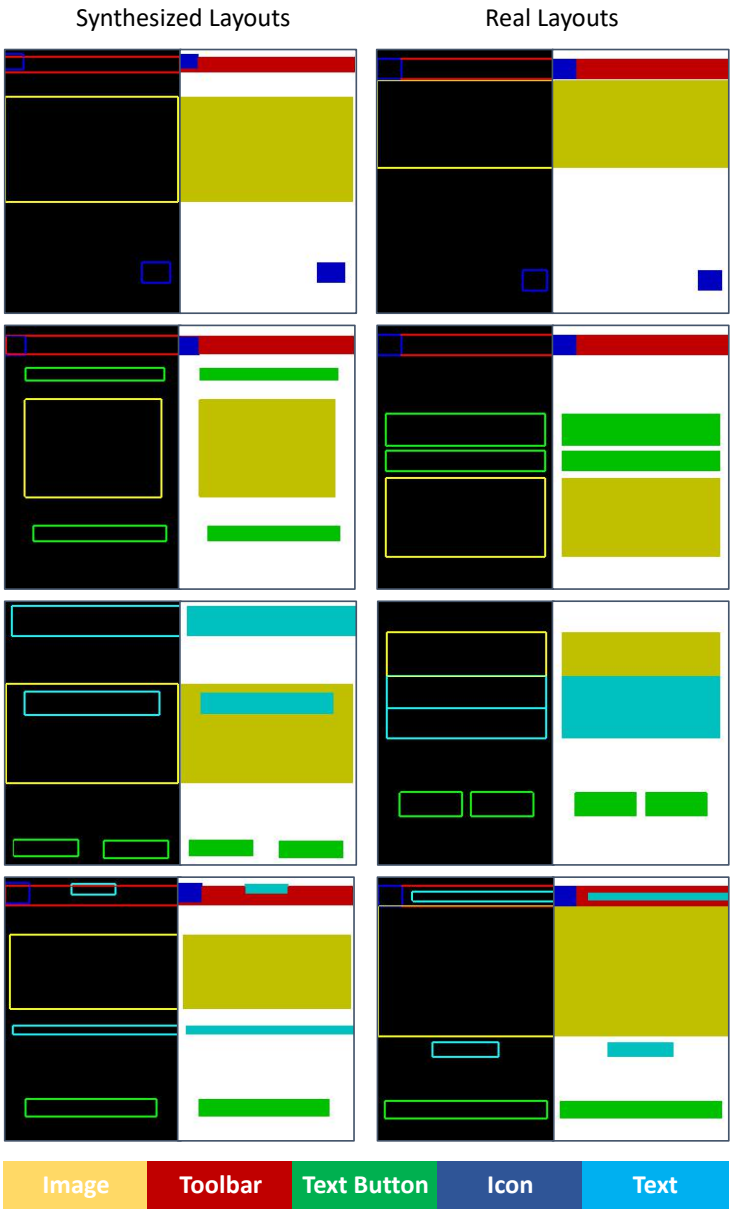


Figure 6: Mobile app layout design.