Content-aware Generative Modeling of Graphic Design Layouts

XINRU ZHENG*, City University of Hong Kong
XIAOTIAN QIAO*, City University of Hong Kong
YING CAO, City University of Hong Kong
RYNSON W.H. LAU, City University of Hong Kong

(a) Inputs (b) Content-aware layout generation (c) Adding user constraint

Fig. 1. Our probabilistic generative model can support content-aware layout generation. Given input images, design category, and keywords that summarize the text contents (which are either automatically extracted from the input text or directly provided by the user), as shown in (a), our method automatically generates multiple layouts that fit the visual and textual contents (b). The user may optionally express his/her design intent by roughly sketching some elements on a page, e.g., adding two image elements by sketching two green regions I_1 and I_2 and a headline element by sketching a red region H, as shown in the small diagram in (c). Our method will then generate a layout that matches the user's intent, i.e., two images and a header placed at the specified locations, as shown in the large diagram in (c). The input images are from Pexels.

Layout is fundamental to graphic designs. For visual attractiveness and efficient communication of messages and ideas, graphic design layouts often have great variation, driven by the contents to be presented. In this paper, we study the problem of content-aware graphic design layout generation. We propose a deep generative model for graphic design layouts that is able to synthesize layout designs based on the visual and textual semantics of user inputs. Unlike previous approaches that are oblivious to the input contents and rely on heuristic criteria, our model captures the effect of visual and textual contents on layouts, and implicitly learns complex layout structure variations from data without the use of any heuristic rules. To train our model, we build a large-scale magazine layout dataset with fine-grained layout annotations and keyword labeling. Experimental results show that our model can synthesize high-quality layouts based on the visual semantics of input images and keyword-based summary of input text. We also demonstrate that our model internally learns powerful features that capture the subtle interaction between contents and layouts, which are useful for layout-aware design retrieval.

CCS Concepts: • **Computing methodologies** → **Computer graphics**; *Machine learning approaches*; *Neural networks.*

Additional Key Words and Phrases: Graphic design, layout, content-aware, deep generative networks

ACM Reference Format:
Xinru Zheng, Xiaotian Qiao, Ying Cao, and Rynson W.H. Lau. 2019. Content-aware Generative Modeling of Graphic Design Layouts. *ACM Trans. Graph.* 38, 4, Article 133 (July 2019), 15 pages. https://doi.org/10.1145/3306346.3322971

1 INTRODUCTION

Layout is at the core of graphic designs, including magazines, posters, comics and webpages. A high-quality layout can benefit information presentation, guide reader attention and enhance visual attractiveness [Stribley 2015; Ying 2014]. Graphic design layout problems are receiving a growing interest in the graphics community in recent years. Some prior works try to model graphic design layouts for layout generation guided by style, perception and aesthetics [Cao et al. 2012, 2014; O'Donovan et al. 2014; Pang et al. 2016].

In graphic designs, layouts are especially created to frame contents (e.g., images and text) in order to present messages and ideas quickly and clearly. Therefore, rich layout variations in graphic designs are largely driven by the visual and textual contents to be presented [Prust 2010]. In other words, generating an effective graphic design layout requires understanding the visual content

*Xinru Zheng and Xiaotian Qiao are joint first authors.
Ying Cao is the corresponding author. This work was led by Rynson Lau.

Authors' addresses: Xinru Zheng, Department of Computer Science, City University of Hong Kong, xrzheng22@gmail.com; Xiaotian Qiao, Department of Computer Science, City University of Hong Kong, xt.qiao@my.cityu.edu.hk; Ying Cao, Department of Computer Science, City University of Hong Kong, caoying59@gmail.com; Rynson W.H. Lau, Department of Computer Science, City University of Hong Kong, rynson.lau@cityu.edu.hk.

0730-0301/2019/7-ART133 $15.00
https://doi.org/10.1145/3306346.3322971

(a) (b) (c) (d)

Fig. 2. Influence of content on layout. All example pages shown here are from the fashion category: (a) and (b) on interviews; (c) on brief biographies; (d) on fashion trends. The pages are from ELLE (© Hearst Magazine Media).

of image elements and the meaning of text elements in the design, which largely encode the topic (e.g., makeup or health), style (e.g., dynamic or formal) and purpose (e.g., promoting a product or illustrating an idea) of the design. For example, although all four pages in Figure 2 belong to the "fashion" category, they exhibit very different layouts because of different contents. The contents of (a) and (b) are both about interviews, and thus have layouts with a dominant image element. In contrast, the content of (c) is about brief biographies of multiple persons, resulting in a layout with multiple columns to present information. The content of (d) is about brief introductions to several fashion trends, and is arranged in a more irregular and creative way than that of (c). Unfortunately, existing layout methods only focus on modeling layout generative rules under some high-level semantic factors (e.g., perceptual importance of images or attention transition between image and text elements). They do not consider the effect of visual and textual contents upon layouts.

In this work, we take a step towards investigating how visual and textual contents affect graphic design layouts and modeling graphic design layouts conditioned on the contents to be presented. We focus our study on one popular and important type of graphic design: magazines. This is because magazines cover a variety of image and text contents, such as biographies, fashion shows and recipes, which exhibit a *diverse* range of layouts with rich and complex arrangement of images and text for different purposes. In addition, magazines (which organize images and text for information communication) are sufficiently *representative*, in terms of layout, of most types of graphic designs (e.g., posters, advertisements and webpages).

We propose a probabilistic generative framework for modeling content-aware graphic design layouts. Our model learns a conditional distribution of graphic design layouts on visual/textual contents and high-level design attributes. To account for rich layout variations in graphic designs, we take advantage of Generative Adversarial Networks (GANs) to model the complex layout distribution, and introduce a semantic embedding network to encode multi-modal contents and the structural/categorical attributes of designs. Our model *implicitly* learns layout structures and design principles from data, without using any heuristics as in existing works. Further, our model automatically extracts visual features that capture the subtle interaction between contents and layouts.

To train our model, we have constructed a large-scale magazine layout dataset consisting of fine-grained semantic layout annotations and keyword-based summary of text contents. Our dataset covers a wide range of magazine categories (including fashion, food, news, science, travel and wedding) with rich layout variations.

We demonstrate that our model can naturally support *content-aware layout generation*. Given input images, design category, and keywords summarizing the text contents (which can be either automatically extracted from the full text or directly provided by the user), our layout generation method can automatically generate multiple plausible layouts for the user, as shown in Figure 1(b). The user can further express his/her design intent by roughly sketching some elements on a page. Our method can then generate a layout that matches the user's intent, as shown in Figure 1(c). With a layout-aware design retrieval experiment, we further show that our model can learn features that capture the subtle interaction between contents and layouts.

In summary, this paper has the following main contributions:

- To the best of our knowledge, we make the first effort to study the problem of graphic layout generation conditioned on visual and textual semantics of user inputs. To this end, we propose the first content-aware deep generative model for graphic design layouts, which is able to synthesize diverse graphic design layouts based on visual and textual features.
- We contribute a large-scale magazine layout dataset with rich semantic annotations including categories, fine-grained semantic layouts and keywords summarizing the text contents.
- We demonstrate how our model can be applied to automatic and constrained layout synthesis based on visual and textual contents to produce high-quality layout designs. We also show that our model can internally learn visual features that capture how contents and layouts interact in graphic designs, which is useful for layout-aware graphic design retrieval.

2 RELATED WORKS

Graphic Design Layout. The importance of layout in graphic designs has motivated a lot of research efforts on layout synthesis. Early works on document layout are mainly based on templates [Damera-Venkata et al. 2011; Hurst et al. 2009; Jacobs et al. 2003; Schrier et al. 2008]. These templates are manually designed according to some design principles. A layout can be produced by selecting a template that best fits the given content. However, such a predefined, limited set of templates cannot describe the rich variation of graphic design layouts well. In addition, the design of these templates usually requires a lot of professional knowledge and manual efforts.

In recent years, we have witnessed a rising interest in graphic design layout problems in the computer graphics community. For example, Kumar et al. [2011] propose an example-based retargeting algorithm to transfer the content of a webpage to the layout of another. Cao et al. [2012] propose statistical style models for creating stylistic comic layouts from input images and semantics. Cao et al. [2014] further compose comic elements to direct reader attention by inferring a probabilistic graphical model. O'Donovan et al. [2014; 2015] arrange the input contents of a single-page graphic design by optimizing an energy function defined by some visual design principles. Recently, Pang et al. [2016] propose user attention models for optimizing input web designs in order to guide user attention along designer-specified input paths. Todi et al. [2016] study the layout problem of general graphic designs, and propose a layout

optimizer to improve hand-drawn layouts based on some visual design rules. Yang et al. [2016] propose a system to generate visually appealing visual-textual presentation layouts by making use of topic-dependent layout templates created by domain experts.

All these existing methods do not consider the effect of input visual and textual semantics upon layouts, and therefore produce results that are not adapted to the contents to be laid out. Our model explicitly accounts for the subtle impact of contents upon layouts, which has not been explored before. In addition, while these works produce layouts based on some design rules, our model can capture the space of rich layout structures purely from data, without using any hand-crafted rules.

Deep Generative Models. Deep generative models learn to understand data through generation. Deep Belief Networks (DBNs) and Deep Boltzmann Machines (DBMs) are a family of generative models that have been used to synthesize shapes [Eslami et al. 2014; Hinton et al. 2006; Wu et al. 2015]. However, such generative models have low convergence and high computational cost. Recently, variational autoencoders (VAEs) [Kingma and Welling 2014; Rezende et al. 2014] and generative adversarial networks (GANs) [Goodfellow et al. 2014] have shown great promise for learning deep generative models. VAEs learn an approximate inference mechanism, but the generated samples tend to be blurry. In contrast, GANs have achieved impressive results in image synthesis [Isola et al. 2017; Radford et al. 2015; Zhu et al. 2016]. However, standard GANs lack an inference mechanism, which helps reason about the latent features of data. Some techniques have been proposed to integrate inference in GANs [Brock et al. 2017; Donahue et al. 2017; Dumoulin et al. 2017] and applied to encode/synthesize 3D shapes [Li et al. 2017]. While previous works focus on modeling 2D images and 3D shapes, we take advantage of the GAN framework to learn the distribution of graphic design layouts, and introduce a semantic embedding network to model multi-modal contents. Concurrent with our work, LayoutGAN [Li et al. 2019] uses a GAN-based framework to refine the layout of randomly placed 2D graphic elements into a good quality one. While similar in using GANs, their approach is content-agnostic and thus cannot be easily adapted to address our content-aware layout generation problem.

3 DATASET

Training our model requires a large and diverse graphic design dataset with ground-truth layout annotations. Although there are some publicly available datasets on magazines and technical journals with semantic segmentation [Antonacopoulos et al. 2009; Todoran et al. 2005], they contain only a few hundreds of pages, with very limited types of layouts. However, our goal is to characterize the rich layout variation, which requires a larger number of pages with diverse layouts. In addition, these existing datasets contain only page segmentation, without any representation for text contents, which is required to model layouts conditioned on text contents.

To this end, we have collected a corpus of 3,919 magazine pages from the Internet, covering 6 common categories, including fashion, food, news, science, travel and wedding. To the best of our knowledge, our dataset is an order of magnitude larger than any similar publicly available datasets. The numbers of pages for the 6 categories are 685, 753, 618, 509, 721, and 633, respectively. As these 6 categories of magazine pages cover a large variety of contents, they exhibit a rich layout variation. We annotate each page with 6 different semantic elements, including Text, Image, Headline, Text-over-image, Headline-over-image and Background. In addition, we also extract keywords from the text contents of each page to represent the text. Refer to Section 1.3 of the supplemental for the statistics and examples of our dataset.

3.1 Semantic Layout Annotation

In this work, we assume that a layout comprises 4 types (i.e., labels) of semantic elements: Headline, Text, Image and Background. We distinguish Headline elements from other Text elements since Headline elements play an important role on graphic design layout [Smith 2014]. We have two considerations for selecting these 4 types of semantic elements. First, they are common elements in graphic designs, and are also found to be the most frequently occurring ones in our collected dataset and can therefore describe layout variability of the dataset well. Second, other element types tend to come along with one of these 4 element types (e.g., subheads are always placed before body text and introductory paragraphs are always placed after headlines), are always placed at regular location (e.g., author credits), or are less common (e.g., pull quote). Without loss of generality, we regard the other element types as text elements in this paper. Further, to model layered representation in layouts, we introduce two other labels to represent two popular types of layering: Text-over-image and Headline-over-image, which represent the Text or Headline elements that entirely/partially overlap an Image element. For each page, we assign each pixel to one of $T = 6$ labels (Text, Image, Headline, Text-over-image, Headline-over-image, Background) to represent its layout.

Since manually labeling so many pages from scratch would be very time- and labor-consuming, we propose a semi-automatic mechanism to label our dataset. We first manually label a small subset of the pages and then use them to train a network to automatically segment the other pages in the dataset, assigning each pixel to one of the 6 labels. Motivated by the outstanding performance of fully convolutional neural networks (FCNs) for semantic segmentation [Long et al. 2015], we exploit the FCN for labeling our pages. After the FCN segmentation, each segmented result is improved via an automatic refinement step to remove noise and refine element boundaries. To ensure the quality of the annotations, we further manually rectify all the segmentations after the automatic refinement. With the manually refined segmentation as the ground truth, the pixelwise accuracy and region intersection over union (IoU) [Long et al. 2015] of the FCN segmentation over the whole dataset are 87% and 69%, respectively. After the automatic refinement, both metrics increase to 88% and 76%, respectively. Refer to Section 1.1 of the supplemental for more details.

3.2 Keywords Extraction

To model visual and textual contents in graphic designs, we need to extract both images and text from the designs. Images can be extracted directly from the segmentation. For text, since the full text in a magazine page may contain a lot of redundant information that

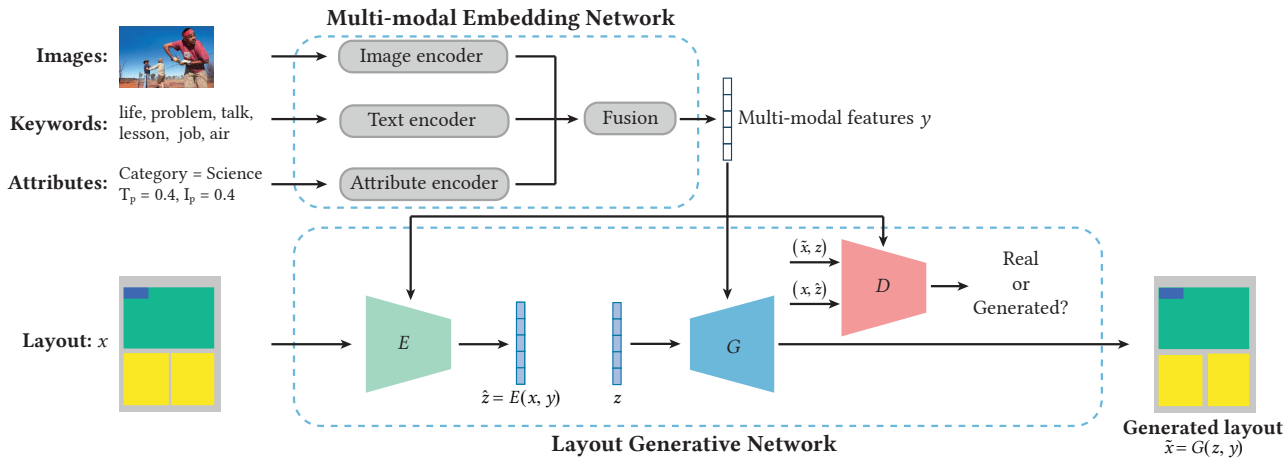

Fig. 3. The framework of our model. It has two main parts: a multi-modal embedding network and a layout generative network. The multi-modal embedding network learns the multi-modal features $\mathbf{y}$ from three inputs: visual contents (images), textual contents (keywords) and 3 high-level design attributes (design category, text proportion T_p, and image proportion I_p). These inputs are first sent to 3 independent encoders, i.e., image encoder, text encoder and attribute encoder, respectively, and then merged via a fusion module to obtain $\mathbf{y}$. The layout generative network learns a distribution of layouts conditioned on $\mathbf{y}$ and extracts content-aware features $\hat{\mathbf{z}}$. In particular, a layout encoder E maps a layout sample $\mathbf{x}$ to features $\hat{\mathbf{z}}$ conditioned on $\mathbf{y}$, a layout generator G maps a random vector $\mathbf{z}$ to a layout sample $\tilde{\mathbf{x}}$ conditioned on $\mathbf{y}$, and a discriminator D learns to distinguish joint pairs $(\mathbf{x}, \hat{\mathbf{z}})$ and $(\tilde{\mathbf{x}}, \mathbf{z})$ conditioned on $\mathbf{y}$. The input image is from Australian Geographic (© Barry Skipsey/Australian Geographic).

are irrelevant to layout modeling, we propose to extract keywords to compactly summarize and represent the full text. To this end, we use an OCR tool from the Google Cloud Platform to recognize the text on a page and then extract keywords using Rapid Automatic Keyword Extraction (RAKE) [Berry and Kogan 2010].

For each magazine category, we create a *keyword list* to represent the text of the category. This keyword list can then be used to help select relevant keywords from the input text. For example, words like "recipe" and "taste" are considered as meaningful keywords for a "food" magazine page, while words like "style" and "dress" are for a "fashion" page. To construct the keyword list for a category, we first use RAKE to extract the keywords from all pages of the same category in the dataset, and sort them according to their frequencies of occurrences. We then manually filter them to 100 words as the keyword list, by removing irrelevant or meaningless words to the design category. We show the top 10 keywords in each of the 6 category-specific keyword lists in Section 1.2 of the supplemental.

In runtime, given an input page, we again extract the keywords from the text content using RAKE and then remove those that are not in the corresponding keyword list. The resulting keywords are used to represent the text content.

3.3 Representing Layouts

Instead of using a parameterized layout representation (i.e., bounding boxes) as in previous works, we choose to use an image-based representation for layouts. In particular, we downsample each pixelwise layout segmentation to a compact image of size $H \times W$, by dividing the segmentation into a $H \times W$ grid of cells and assigning each cell with the label that the majority of pixels in the cell take. Note that such image-based representation is reminiscent of the widely used grid system in the graphic design field, and can be processed by CNNs naturally. We encode the value of each cell in the layout with a 3-dimensional binary vector, using $(0, 0, 1)$, $(0, 1, 0)$, $(0, 1, 1)$, $(1, 0, 0)$, $(1, 0, 1)$ and $(1, 1, 0)$, to represent the 6 labels. The unused values of the binary vector, i.e., $(0, 0, 0)$ and $(1, 1, 1)$, are considered as Background. An appropriate layout resolution is important. A higher resolution allows the model to capture more fine-level details of the layouts, but at a higher computational cost. Here, we set the layout size to 60×45 to allow our model to capture sufficient details, while still being efficient to train. Our chosen layout size can also help preserve the aspect ratio of 4/3, which is commonly used in most magazine pages.

4 OUR MODEL

Since our end goal is to learn to generate high-quality graphic design layouts by considering contents, we desire a model that has a high learning capacity to represent complex layout structure variations while capturing the dependency of layouts upon the contents. To this end, we propose a content-aware deep generative network architecture for layout generation. Since there may be many plausible layouts given a particular set of contents, instead of learning a deterministic mapping, our network learns a conditional distribution of layouts given visual and textual contents as well as the design attributes (i.e., design category, text proportion, and image proportion). Finally, we can sample our network to synthesize multiple different layouts according to the input contents.

Figure 3 shows our network architecture, which consists of two main parts: a multi-modal embedding network and a layout generative network. The multi-modal embedding network learns the visual features from images and textual features from text, and combines them with the features learned from the three high-level design attributes to form the multi-modal features for the layout generative network. The layout generative network learns a layout distribution

to describe large layout variation and extracts high-level features simultaneously, conditioned on the multi-modal features.

4.1 Multi-modal Embedding Network

Our multi-modal embedding network learns visual and textual features from image and text contents, and uses them to guide the generative modeling process of graphic design layouts. We also introduce three high-level design attributes to allow for a more fine-grained control over layout generation. Our multi-modal embedding network takes images, keywords and design attributes as inputs, and uses image, text and attribute encoders to produce visual, textual and attribute feature vectors, respectively. These feature vectors are then merged through a fusion layer (including 2 fully connected layers) to produce a 128-dimensional feature vector $\mathbf{y}$ to condition the layout generative network.

Image encoder. Given a page, we extract all image regions according to its segmentation labels. We then feed each image region to a pre-trained VGG16 model [Simonyan and Zisserman 2015] to extract image features. We use the $14 \times 14 \times 512$ output of the last convolutional layer as the image representation for each image element. All image representations on a page are first summed up to get a $14 \times 14 \times 512$ representation. We apply spatial global average pooling to form a 512-dimensional vector, which is then fed into 3 fully connected layers to produce a 128-dimensional image vector.

Text encoder. As stated in Section 3.2, given a page, we extract a list of keywords to represent the text contents of the page. Each keyword is projected to a 300-dimensional word embedding vector using word2vec [Mikolov et al. 2013]. All word embedding vectors for a page are summed up and then fed into 3 fully connected layers to produce a 128-dimensional text vector.

Attribute encoder. We consider three design attributes: design category, text proportion and image proportion. The design category is important for layout modeling as it can influence the content as well as how the content should be presented [Prust 2010; Saleem 2015]. Text and image proportions refer to the total area occupancies by text (including Text-over-image) and image elements over a page. They serve as high-level control signals to allow designers to optionally and intuitively express their preference on the overall density of text or images in the final layout, and are considered as soft control for our layout modeling.

We describe the category attribute using an integer ranging from 0 to 5 to represent the 6 different categories in our dataset. We empirically find that for over 98% of layouts in our dataset, the text proportion falls into $[0, 0.7]$ since it is unusual for text to cover the whole page without any empty space (Background). On the other hand, the image proportion can range between 0 and 1. Hence, we set the maximum values of text and image proportions to 0.7 and 1, and quantize the two proportions uniformly with an interval of 0.1 to 7 scales (from 0.1 to 0.7) and 10 scales (from 0.1 to 1), respectively, such that all three attributes can be encoded as one-hot vectors. Each vector is then duplicated 10 times to increase their significance [Wu et al. 2015], and fed into a fully connected layer to output a 48-dimensional attribute vector. Finally, all the outputs are fused through a fully connected layer to form a 32-dimensional attribute vector. Refer to Section 2.2 of the supplemental for the detailed architecture of the image encoder, text encoder, attribute encoder and fusion layer.

4.2 Layout Generative Network

The layout generative network is built upon the GAN [Goodfellow et al. 2014], which consists of a generator and a discriminator. The generator learns to generate samples of the same distribution as the training data, while the discriminator learns to determine whether a given sample is real or generated. In our layout generative network, the generator G maps a 128-dimensional latent vector $\mathbf{z}$ to a layout $G(\mathbf{z})$. The discriminator D outputs a confidence value to indicate whether a layout $\mathbf{x}$ is real or generated.

The standard GAN generates a layout from a sampled latent vector. However, it is desirable to infer this latent vector from an observed layout, be it partial or complete, via a learned mapping. Such mapping would allow us to incorporate user preference into the layout generation process (Section 6.3) as well as learn layout-aware features for a more meaningful design comparison (Section 6.5). To this end, we add an additional encoder, E, so that E can induce a distribution $p(\hat{\mathbf{z}}|\mathbf{x})$ to map a layout sample $\mathbf{x}$ from real layout distribution $p(\mathbf{x})$ to the feature space. Concurrently, the generator, G, induces a distribution $q(\tilde{\mathbf{x}}|\mathbf{z})$ to map samples from a prior distribution $q(\mathbf{z})$ to the layout space. We use a standard normal distribution as the prior distribution $q(\mathbf{z})$. On the other hand, as in [Donahue et al. 2017; Dumoulin et al. 2017], the discriminator D is trained to discriminate jointly in the layout and feature spaces, i.e., to distinguish the joint pairs $(\mathbf{x}, \hat{\mathbf{z}} = E(\mathbf{x}))$ and $(\tilde{\mathbf{x}} = G(\mathbf{z}), \mathbf{z})$, rather than only in the layout space ($\mathbf{x}$ and $\tilde{\mathbf{x}}$). In other words, our adversarial objective becomes: the generator and encoder are jointly trained to fool the discriminator by generating joint pairs either from the generator $(\tilde{\mathbf{x}} = G(\mathbf{z}), \mathbf{z})$ or encoder $(\mathbf{x}, \hat{\mathbf{z}} = E(\mathbf{x}))$ that are indistinguishable by the discriminator, while the discriminator learns to distinguish between these two types of joint pairs. The goal becomes to match the two joint distributions $p(\mathbf{x}, \hat{\mathbf{z}}) = p(\hat{\mathbf{z}}|\mathbf{x})\, p(\mathbf{x})$ and $q(\tilde{\mathbf{x}}, \mathbf{z}) = q(\tilde{\mathbf{x}}|\mathbf{z})\, q(\mathbf{z})$. The encoder processes a layout sample through a series of convolutional layers to produce two vectors, which represent the mean and standard deviation of a Gaussian distribution. A sample is then drawn from the Gaussian distribution and used as the feature vector. A detailed description of the generator, encoder and discriminator is given in Section 2.1 of the supplemental. Note that our layouts are represented as binary vectors. When used in our model, they are treated as real vectors with values in $[0, 1]$, as it is difficult for GANs to handle discrete inputs.

Conditional modeling. To model graphic design layouts conditioned on image and text contents, we feed the 128-dimensional multi-modal features $\mathbf{y}$ from the multi-modal embedding network into the layout generative network as conditional information. It is sent to the encoder, the generator and the discriminator, so that (1) encoder E induces a distribution $p(\hat{\mathbf{z}}|\mathbf{x}, \mathbf{y})$ to map a layout sample $\mathbf{x}$ from real layout distribution $p(\mathbf{x})$ to the feature space conditioned on $\mathbf{y}$, (2) generator G induces a distribution $q(\tilde{\mathbf{x}}|\mathbf{z}, \mathbf{y})$ to map samples from a prior distribution $q(\mathbf{z})$ to the layout space conditioned on $\mathbf{y}$, and (3) discriminator D learns to discriminate the input joint

pairs conditioned on $\mathbf{y}$. We feed $\mathbf{y}$ to generator G by directly concatenating $\mathbf{y}$ and $\mathbf{z}$ as the input to G. To apply $\mathbf{y}$ to encoder E and discriminator D, we first duplicate $\mathbf{y}$ along spatial dimensions to form a $60 \times 45 \times 128$ feature map. We then concatenate this feature map with $\mathbf{x}$ or $\tilde{\mathbf{x}}$ as the input to D and with the feature map of the fourth layer of E. Refer to Section 2.1 of the supplemental for details.

4.3 Loss Function

We follow the least squares GAN (LSGAN) [Mao et al. 2017] to formulate the loss L_{GAN}^{D} for discriminator D and the adversarial loss L_{GAN}^{G} for generator G as:

$$L_{GAN}^{D} = \frac{1}{2}\left(D\left(\mathbf{x}, E(\mathbf{x}, \mathbf{y}), \mathbf{y}\right) - 1\right)^2 + \frac{1}{2}\left(D\left(G(\mathbf{z}, \mathbf{y}), \mathbf{z}, \mathbf{y}\right)\right)^2, \quad (1)$$

$$L_{GAN}^{G} = \frac{1}{2}\left(D\left(G(\mathbf{z}, \mathbf{y}), \mathbf{z}, \mathbf{y}\right) - 1\right)^2. \quad (2)$$

To further improve the learning, we follow the VAE [Kingma and Welling 2014] to introduce two additional terms into our loss function: the reconstruction loss L_{rec} and the Kullback-Leibler (KL) divergence loss L_{KL} as:

$$L_{rec} = \|\mathbf{x} - G\left(E\left(\mathbf{x}, \mathbf{y}\right), \mathbf{y}\right)\|^2, \quad (3)$$

$$L_{KL} = D_{KL}\left(p\left(\hat{\mathbf{z}}|\mathbf{x}, \mathbf{y}\right) \| q\left(\mathbf{z}\right)\right), \quad (4)$$

where D_{KL} is the KL divergence. The reconstruction loss encourages cycle consistency for the encoder and generator (i.e., if a layout $\mathbf{x}$ is transformed to a feature vector $\hat{\mathbf{z}}$ via the encoder and then transformed back to a layout $\tilde{\mathbf{x}}$ via the generator, $\tilde{\mathbf{x}}$ should be similar to $\mathbf{x}$). The KL divergence loss forces the distribution of $\hat{\mathbf{z}}$ of the encoder $p\left(\hat{\mathbf{z}}|\mathbf{x}, \mathbf{y}\right)$ to be close to the distribution of $\mathbf{z}$ of the generator $q\left(\tilde{\mathbf{x}}|\mathbf{z}, \mathbf{y}\right)$ (i.e., standard normal distribution $q\left(\mathbf{z}\right)$), so that the feature vectors for both encoder and generator lie in approximately the same space. We have empirically found that the two additional terms can help improve the inference performance on the data distribution, where we need to use the encoder and generator jointly (i.e., first map the sketched layout $\mathbf{x}$ to $\hat{\mathbf{z}}$ via the encoder and then map $\hat{\mathbf{z}}$ to the inferred $\tilde{\mathbf{x}}$ via the generator, as in our content-aware layout generation discussed in Section 5).

To encourage the generator to produce diverse layouts, we add a variety loss $L_{variety}$ as in [Gupta et al. 2018]:

$$L_{variety} = \min_{\mathbf{k} \in 1,2,\cdots,K} \left\|\mathbf{x} - G(\mathbf{z}^{(\mathbf{k})}, \mathbf{y})\right\|^2, \quad (5)$$

where $\mathbf{k}$ is a hyperparameter. Given the input multi-modal features $\mathbf{y}$, we generate K layouts by randomly sampling K vectors $\{z^{(k)}\}_{k=1}^{K}$ from the distribution of z, and choose the generated layout that has the closest distance to the ground truth layout x. The loss enables the generator to thoroughly explore the true data distribution and encourages diversity of the generated samples.

Hence, the loss functions for the generator and encoder become:

$$L_G = L_{GAN}^{G} + L_{rec} + L_{variety}, \quad (6)$$

$$L_E = L_{rec} + L_{KL}. \quad (7)$$

4.4 Training Details

The whole network is trained end-to-end. A layout is first converted to a downsampled image-based representation ($60 \times 45 \times 3$), and then transformed to $64 \times 64 \times 3$ by padding with 0's. Our model is trained with the Adam optimizer [Kingma and Ba 2015] using the recommended configuration from DCGAN [Radford et al. 2015], i.e., $\beta_1 = 0.5, \beta_2 = 0.999, \epsilon = 10^{-8}$, a learning rate of 0.0002 and a mini-batch size of 128. In each iteration, we perform three parameter update steps to update: (1) the parameters of the discriminator using Eq. 1, (2) the parameters of the generator using Eq. 6, and (3) the parameters of the encoder using Eq. 7. Note that in each parameter update step, we also update the parameters of the multi-modal embedding network using the loss of the step. We have empirically found that the discriminator learns faster than the generator, which would inhibit the learning of the generator. To ensure sufficient training of all modules, in each iteration, we train the generator and encoder multiple times until the loss of the discriminator is larger than a threshold (1 in our experiments).

4.5 Runtime Layout Generation

To generate a layout with our network, we first convert the output of the generator to an initial layout ($60 \times 45 \times 3$) by simply removing padded pixels, and quantizing each value to 0 or 1.

GANs are known to have difficulties in reconstructing fine visual details. Although this is not a problem for our work since we aim to generate gross layout structures instead of photo-realistic images, it may still cause some visual artifacts in the generated layouts. To tackle this problem, as in [Kelly et al. 2018], we use a post refinement step to refine element boundaries and correct slight misalignments among the elements. In particular, we first extract individual elements from the initial layout via connected component labeling. To address jagged element boundaries, we apply a series of morphological image processing operations to approximate the element boundaries with straight lines. To address slight misalignments among some elements, we perform top/bottom/left/right alignment on them. To perform a top-alignment, we first cluster elements into one group if the top boundary coordinates of their bounding boxes differ by less than 2 cells. For the elements of the same group, we then adjust their top boundaries to align with the lowest top boundary to create sufficient spacing between the elements. Likewise, we perform bottom, left or right alignment in a similar way. Refer to Section 2.3 of the supplemental for details.

5 CONTENT-AWARE LAYOUT GENERATION

Our problem setting for layout generation is as follows. The user first provides the contents, i.e., images, keywords (or full text), and design category. If the user provides full text, keywords are extracted from it automatically as in Section 3.2. Our goal is to generate layouts that fit the given contents and user constraints (if any).

5.1 Automatic Layout Generation

Given input visual and textual contents, we propose an algorithm based on our model to automatically synthesize a layout that matches the inputs. Note that when multiple input images are provided by the user, the correspondence between the input images and the image elements in the generated layout is required in order to place the input images in the layout properly. To this end, we assume that the input images are provided in a ranking list based on their importance, with the first being the most important and the last being the

least. These input images are then automatically associated with the image elements in the layout according to the size difference of the image elements, by assigning more important input images to much larger image elements. In particular, given the user specified design category, our method first samples 16 sets of text and image proportion values from the empirical distributions of text and image proportions from the same design category in our dataset. For each set, we generate 32 layouts by sending 32 different random vectors drawn from the prior distribution $q(\mathbf{z})$ to the generator, resulting in 512 layouts in total. We then remove invalid layouts by applying the filtering criteria below: (1) the number of image elements is not the same as that of input images; (2) the aspect ratio of the image element (width/height) differs too much from that of the corresponding input image (larger than 1.3 or less than 0.7 in our implementation). The user is also allowed to specify his/her preferred text and/or image proportions in the generated layouts. If these proportions are given, the 512 layouts are generated from 512 different random vectors drawn from $q(\mathbf{z})$.

For the remaining layouts, we diversify them using Maximal Marginal Relevance (MMR) criterion [Carbonell and Goldstein 1998]. Specifically, we use the discriminator output (classification probability) as the quality score of the generated layouts, and use L_2 distance in a feature space (the mean vector from our encoder conditioned on the multi-modal features extracted from the input contents) to calculate the similarity score between layouts. The layout with the highest quality score is ranked the first and added to a rank list L. For each of the other layouts l, we calculate a ranking score: $r_l = Q_l - \max_{e \in L} S(l, e)$, where Q_l is the quality score of l and $S(l, e)$ is the similarity score between l and e. The layout with the highest ranking score is added to the rank list one at a time. By iteratively adding layouts to L, all the layouts are ranked by balancing between quality and similarity. Finally, we return the top 3 layouts as our generated layouts, which are filled with the given images and some random texts.

5.2 Adding User Constraints

The user can also exert control over the layout generation process by roughly sketching some elements on a page to indicate the approximate positions and sizes of some elements. Our method will then generate a layout that matches the user's constraints, as shown in Figure 1(c). Note that the sketched elements only reflect the user's design intent, and are therefore treated as soft constraints. The user will also need to indicate the corresponding images for the sketched image elements. Given the user constraints, our layout generation follows the steps discussed in Section 5.1, with one main difference: instead of using random vectors, we use the features extracted by the encoder from the input sketch, referred to as constraint features, to generate novel layouts.

Specifically, to obtain the constraint features, we first convert the input rough sketch to a partial layout $\mathbf{x}_s$ by representing each sketched element with a bounding box. We then send $\mathbf{x}_s$ to our layout encoder to obtain the constraint features $\hat{\mathbf{z}}_s$ conditioned on the input images, keywords and design category. $\hat{\mathbf{z}}_s$ capture the layout structure of the rough sketch, and are fed into the layout generator to generate novel, complete layouts $\tilde{\mathbf{x}}$ that match the sketched elements in terms of position and size.

To generate layouts, we constrain the sampled text and image proportions to be no less than their counterparts in the input sketch. We then obtain a total of 512 layouts and remove the layouts whose number of image elements is less than that of the input images or the difference in aspect ratio between an image element and its input image is large (as in the filtering criteria in Section 5.1). In addition, if the user provides some sketched Headline regions, we will remove layouts whose number of Headline elements is less than that of sketched Headline regions. When diversifying the layouts with MMR, we consider both the output of the discriminator and the similarity to the sketched layout to measure the quality of the generated layouts. Let s be an input sketch. The quality score of a generated layout l is computed as: $\alpha \times S(l, s) + Q_l$, where Q_l is the output of the discriminator given l and the input contents, $S(l, s)$ is the layout similarity score between l and s as in Section 5.1. α is set to 2 in our implementation. Finally, we return the top layout, which is automatically filled with the given images and some random text.

6 RESULTS AND EVALUATION

To evaluate our content-aware layout generation method, we consider two scenarios, automatic layout generation and constrained layout generation. In this section, we first present our evaluation results on automatic and constrained layout generation, and then look into the effect of visual and textual contents on layout generation. Finally, we explore using content-aware features learned by our model in a design retrieval task.

For evaluation, we have selected 60 magazine pages that are not seen by our model to form a test dataset. These pages are from the same 6 categories as in our dataset, showing various layout structures. Each page contains semantic layout segmentation and keywords, obtained using the methods discussed in Section 3.

6.1 Implementation Details

To visualize the generated layouts, the font size for both Text and Text-over-image elements is fixed, while the font size for both Headline and Headline-over-image elements is set to at least 3 times larger and varies according to the size of the corresponding regions. For Text-over-image and Headline-over-image, the font color can be white or black depending on the brightness of the corresponding image region, and the background of the Text and Headline is set to be transparent (or white) if it overlaps entirely (or partially) with the image. To fill an image element in a layout with its corresponding input image, we first detect a focal point (i.e., the center of the most salient region) of the image using [Heynemann et al. 2015], and then use it to guide the cropping of the image so that important image contents, e.g., the face of the key foreground character, are preserved as much as possible. Note that these operations are only used to visualize the generated layouts. Our layout generation process does not involve any heuristics rules. Refer to Section 3.1 in the supplemental for more details.

6.2 Automatic Layout Generation

Baselines. Since no existing works address the problem of content-aware graphic design layout generation, we compare our method with two simple yet reasonable content-aware baselines that are

Fig. 4. Results of our automatic layout generation method. In each case, the input images, keywords and design category are shown on the left. The layouts by two baseline methods (Baseline1, Baseline2), layout by our method (Ours), and the ground truth layout (Ground Truth) are shown on the right. For each layout, the Headline is filled with a sequence of A's in bold. The layout segmentation is shown at the bottom-right or top-left corner (yellow for Text, green for Image, red for Headline, blue for Text-over-image, purple for Headline-over-image and gray for Background). Note that in each case, the text and image proportions used in both the baselines and our method are obtained from the ground truth layout. The input images (from top to bottom) are from Australian Geographic (© Barry Skipsey/Australian Geographic), Club Med (© Club Med), MICHELLE BELLER PHOTOGRAPHY (© MICHELLE BELLER PHOTOGRAPHY).

based on nearest neighbor search. Given the same inputs as in our method, all the layouts that belong to the input design category are first selected as layout candidates for retrieval, which are then filtered by the number and aspect ratios of the input images. Finally, the two baselines compute a content embedding vector using our multi-modal embedding network and return the layout of the nearest neighbor in the content embedding space from the filtered layout candidates. To compute the embedding vector, for the first baseline (Baseline1), we use our image encoder with VGG weights to obtain a 512-dimensional image vector, and use the sum of word2vec embeddings to obtain a 300-dimensional keyword vector. We then concatenate the image and keyword vectors with three duplicated one-hot attribute vectors (i.e., category, text and image proportions) to form a content embedding vector. For the second baseline (Baseline2), we directly use our learned multi-modal embedding features as the content embedding vector for retrieval. While Baseline1 uses the content features pre-trained for other tasks to verify if a naive use of existing features works, Baseline2 uses the content features from our trained multi-modal embedding network to verify if a multi-modal embedding network alone works. Note that the baseline results are existing layouts created by professional designers, and are thus supposed to well respect visual design guidelines.

Qualitative results. Figure 4 shows our results (Ours), compared with the layouts by the two baseline methods (Baseline1 and Baseline2) and the ground-truth layouts (Ground Truth). For our method,

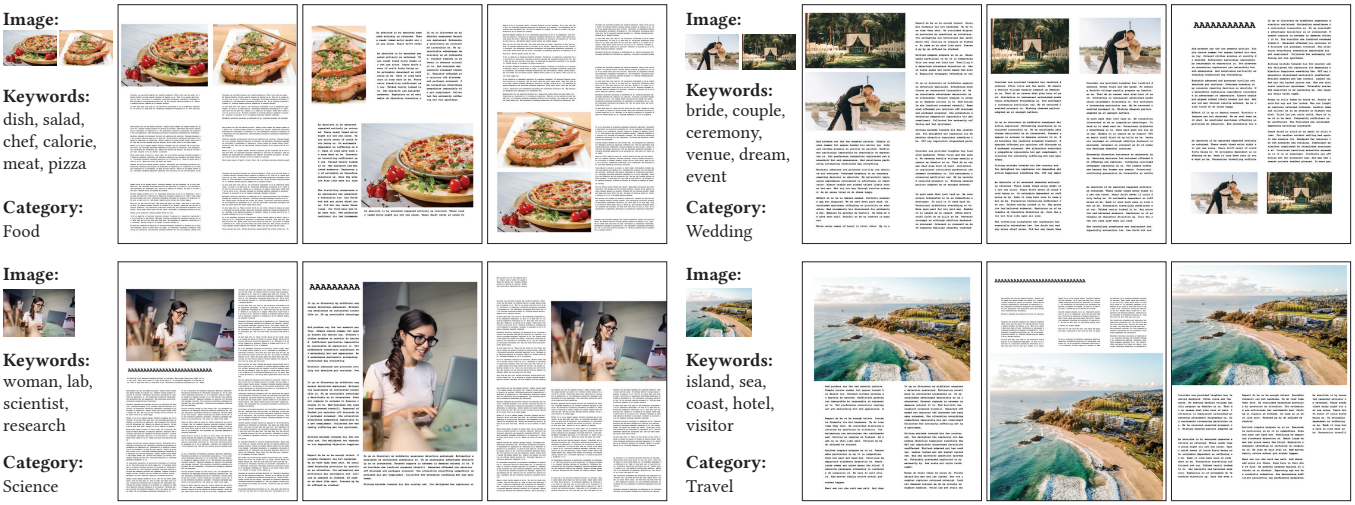

Fig. 5. Diversity of our results. Given each set of inputs, our method automatically generates 3 different layouts. The input images are from Pixabay and Pexels.

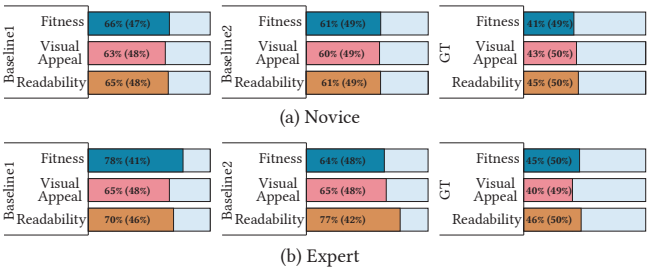

Fig. 6. Results of the user study comparing layouts generated by our method to those from the two baselines and the ground truth. We show the evaluation results by novices (a) and experts (b). In each column, we compare our results with those from Baseline1 (left), Baseline2 (middle), and Ground Truth (right). The color bar shows the percentage of times that our results are preferred over others. The number in the parenthesis denotes standard deviation. Evaluators (both novices and experts) consistently prefer our results over the baseline results, in terms of the 3 aspects (Chi-squared test, $p < 0.05$). Our results are generally comparable to the ground truth (Chi-squared test, $p > 0.05$), except for the fitness preference of novices (Chi-squared test, $p = 0.01$).

we show the top 1 layout among the generated layouts. For fair comparison, we set the text and image proportions for the two baselines and our method to be the same as those of the ground truth. The text and image proportions of the ground truth are computed as in Section 4.1. The layouts by all methods and the ground truth are created by filling them with random text and the input images. Since our focus here is on evaluating the layouts, for the ground truth, we do not use the original pages for comparison as they may contain some font decorations, such as different font faces and colors, which likely bias human perception of the entire designs.

From Figure 4, we can see that our method can generate visually attractive layouts that better fit the input contents, as compared with the two baselines. For example, the input contents of the first row are about a science lesson. The layouts from the two baselines either have too little space for image presentation, or place the title in the middle of the page. In contrast, our layout is more similar to the ground truth layout in terms of how to compose images and text visually and functionally. In addition, our layouts consistently exhibit a better balance between the roles of multiple images and text. For example, the input contents of the third row are about a wedding ceremony. Our layout has a clear arrangement of images and text, so as to elaborate the details of the wedding. In contrast, the layout from the first baseline inserts one image into the text region, and the layout from the second baseline overlays too much text on the larger image. Both layouts may cause confusion on how to follow the contents on the page.

Figure 5 shows more results generated by our method. Given the same set of inputs, our method is able to generate diverse layouts that fit the inputs. For example, the inputs for the bottom-right example are on traveling (related to visiting a place). Our method can generate diverse layouts with a large image and some empty space around the text, making readers focus on the image and feel relaxed. Refer to Section 3.2 of the supplemental for more results.

User study. We further perform a user study on Amazon Mechanical Turk (AMT) with 20 randomly selected pages from our test dataset. For each page, we extract its images, keywords and design category as inputs, and use its text and image proportions (calculated as in Section 4.1) to generate 3 layouts, each from our method (the top 1 layout), Baseline1 and Baseline2. Participants were recruited to evaluate the quality of the generated layouts via pairwise comparisons. For each test page, we created 3 comparison pairs (Ours vs. Baseline1, Ours vs. Baseline2, Ours vs. Ground Truth), resulting in a total of 60 pairs for comparison. In each comparison, two layouts were shown side by side in random order, and the corresponding inputs were also displayed. The participants were asked to select the one that they preferred, in terms of 3 aspects: fitness (how well a layout fits its given inputs), visual appeal, and readability. Each comparison was evaluated by 10 different workers. Each HIT consisted of 30 comparisons from 10 different inputs. For each worker, we duplicated 5 randomly selected comparisons for consistency

Fig. 7. Visual comparison of our method against a graphic design layout method [O'Donovan et al. 2014]. In each row, the results by [O'Donovan et al. 2014] and by our method from the same inputs are shown on the left. The results after changing the input image (the first row) and text (the second row) are shown on the right. The input image at the right of the first row is from Pexels. The rest are from [O'Donovan et al. 2014].

check. In addition, we also recruited another 3 experienced graphic designers for the evaluation.

Figure 6 shows the results. We can see that both novice (a) and expert (b) evaluators largely prefer our results over the baseline ones in all 3 aspects. The differences between our results and the ground truth in all 3 aspects are not statistically significant, meaning that our results are comparable to the ground truth. Refer to Section 3.2 of the supplemental for the layouts used in our user study.

Comparison to prior work. As discussed before, while there are several existing layout methods for different types of graphic designs, they are all oblivious to the input contents to be laid out and are therefore not able to produce tailored results for the contents. In addition, they assume that the input text is already segmented into text elements. While this is appropriate for comics (i.e., speech balloons) or posters (i.e., text blocks), this assumption may not be reasonable for graphic designs where the text is provided in the form of an article. Segmenting a full article into separate elements before the layout process is not straightforward as it is more a part of the layout process itself. Thus, a full comparison with these methods is impossible. However, we have made an attempt to visually compare our results with those from a state-of-the-art layout method for single-page graphic design [O'Donovan et al. 2014], which is based on optimizing some visual design principles, as shown in Figure 7. As can be seen, although our model is not explicitly forced to learn visual design guidelines, our results compare favorably with those by [O'Donovan et al. 2014], in terms of following well-known rules (e.g., alignment, visual balance and spacing). In addition, our method can generate different layouts as the input images and text change, while their method always produces the same layout, irrespective of the input contents.

6.3 Constrained Layout Generation

Qualitative results. We compare our method with a retrieval-based baseline. Given input contents c and an input sketch s, the

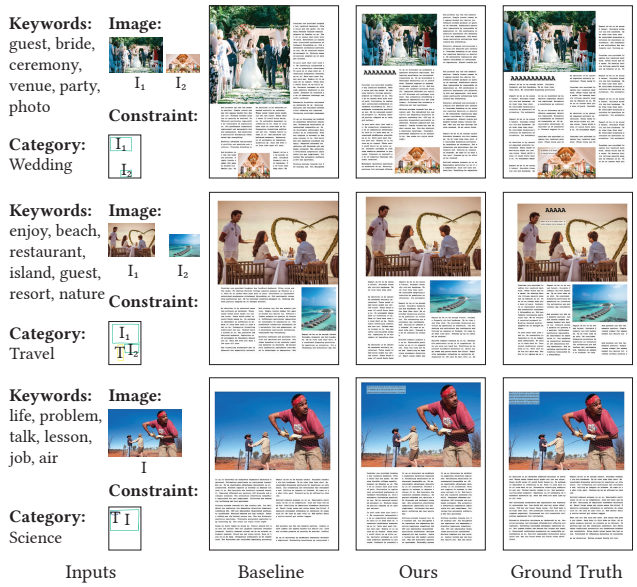

Fig. 8. Constrained layout generation results. In each case, the input contents and the input sketch that indicates the approximate positions and sizes of the desired elements in the output layouts ("T": Text element, "I": Image element, "H": Headline element, "$T\backslash I$": Text-over-image element, "$H\backslash I$": Headline-over-image element) are shown on the left. Results by the baseline (Baseline), our method (Ours), and the ground truth (Ground Truth) are shown on the right, where the Headline is filled with a sequence of A's in bold. Note that, in each case, the text and image proportions used in both our method and the baseline are obtained from the ground truth layout. The input images (from top to bottom) are from MICHELLE BELLER PHOTOGRAPHY (© MICHELLE BELLER PHOTOGRAPHY), Club Med (© Club Med), Australian Geographic (© Barry Skipsey/Australian Geographic).

baseline retrieves a layout from our training dataset based on content and layout similarity:

$$\arg\max_i \frac{L(\mathbf{f}_c, \mathbf{f}_{c_i})}{M} + \frac{\left[H(\mathbf{B}_s \odot \mathbf{g}_s, \mathbf{B}_s \odot \mathbf{g}_{l_i}) + H(\mathbf{B}_{l_i} \odot \mathbf{g}_s, \mathbf{B}_{l_i} \odot \mathbf{g}_{l_i})\right]}{2N}, \tag{8}$$

where i indexes over all pages that belong to the input category in our training dataset. l_i represents the layout of the i-th page. $\mathbf{f}_c$ is the content embedding vector of the input contents, which is obtained using Baseline1 in Section 6.2. $\mathbf{g}_{l_i} \in R^{H \times W}$ denotes a layout image of the i-th page, where the 6 element types are encoded as 6 different integers. $L(\cdot,\cdot)$ and $H(\cdot,\cdot)$ denote L_2 and Hamming distance, respectively. M denotes the numbers of elements in $\mathbf{f}_c$, and $N = H \times W$. $\odot$ is a point-wise multiplication. $\mathbf{B}_s$ is a binary map where the pixels within the sketched elements are set to 1 and the rest are set to 0. $\mathbf{B}_{l_i}$ is another binary map where all pixels of an element are turned on if this element overlaps with any element of the same type in the input sketch. Note that when computing the layout image of the input sketch $\mathbf{g}_s$, we set the cells outside the sketched elements to a value that does not belong to any of the 6 element labels. The two Hamming distances in Eq. 8 are to prevent the elements in a retrieved layout from over- or under-covering their corresponding elements in the sketched layout.

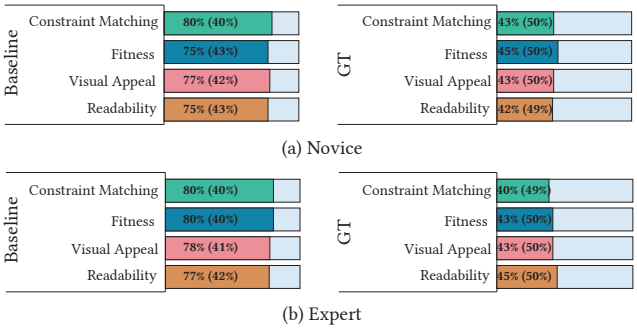

Fig. 9. Results of the user study comparing user-constrained layouts generated by our method to those from the baseline and the ground truth. We show the evaluation results by novices (a) and experts (b). In each column, we compare our results with Baseline results (left), and with Ground Truth (right). The color bar shows the percentage of times that our results are preferred over other results. The number in the parenthesis denotes standard deviation. The evaluators (both novices and experts) consistently prefer our results over the baseline results, in terms of the 4 aspects (Chi-squared test, $p < 0.001$). Our results are generally comparable to the ground truth (Chi-squared test, $p > 0.05$).

For evaluation, we randomly select 20 existing magazine pages from our test dataset. For each page, we select a random subset of image and text elements. Using their bounding boxes as a reference, we ask a human subject to manually sketch them to simulate an input sketch. We then generate layouts by feeding the simulated input sketch together with the selected image/text elements to our method and the baseline, and compare these results with the ground truth (i.e., the original page). For fair comparison, in each test case, the text and image proportions for our method and the baseline are set identical to those of the ground truth. Note that both our method and the baseline do not see the ground truth pages during training and test time. Figure 8 shows some of the results. The simulated input sketches are from elements of various labels and sizes. We can observe that our method can generate layouts that better match the input sketches and are more consistent with the ground truth in terms of presenting visual and textual contents, compared with those from the baseline. This is because our method can synthesize novel and plausible layouts from the constraint features, which capture the partial input sketches and the input contents. On the other hand, the baseline directly retrieves existing layouts from the dataset and fills them with the given images and some random texts. This may not fit the input contents well. Refer to Section 3.3 of the supplemental for more results.

User study. We perform another user study on AMT to evaluate the generated layouts via pairwise comparisons, as in Section 6.2. We use the same 20 pages as in the qualitative evaluation above. For each test page, we created 2 comparison pairs (Ours vs. Baseline, Ours vs. Ground Truth), resulting in a total of 40 pairs for comparison. We recruited 15 participants (6 experienced graphic designers and 9 novices), each of whom was asked to complete the 40 comparisons, plus 3 randomly selected, duplicate comparisons for consistency check. For each comparison, two layouts were shown side-by-side in random order, along with the inputs. The participants were asked to select the one that they preferred, in terms of 4 aspects: constraint matching (i.e., input sketch), fitness (how well a layout fits the given inputs), visual appeal, and readability. Refer to Section 3.3 of the supplemental for the layouts used in the user study.

Figure 9 shows the results. We can see that both novice (a) and expert (b) evaluators largely prefer our results over the baseline results in all 4 aspects, especially in terms of matching user constraints. The difference between our results and the ground truth is not statistically significant, which implies that our results are perceived to be of similar quality to the ground truth by both novices and experienced designers. In addition, we also compute the IoU (intersection over union) of Ours, Ground Truth and Baseline, with respect to the input sketches, to quantitatively measure the fitness between the user-drawn constraints and results. In particular, for each output layout, we first compute an element IoU between each sketched element and its corresponding element in the layout, and then a layout IoU by averaging all the element IoU's. Finally, an average IoU is computed across all the layouts. The average IoU's are 92.8%, 72.1% and 51.2% for Ground Truth, Ours, and Baseline, respectively. This also shows that our results better match the user constraints than the baseline results. Note that the average IoU for Ground Truth is not 100% as our manually drawn input sketches may not exactly match their counterparts in the original pages.

6.4 Effect of Input Contents on Layouts

Qualitative results. To investigate the effect of input contents on layout generation, we generate layouts by changing either the visual or the textual input. Figure 10 shows the results of two design categories, wedding and travel. We can see that changing the visual or textual content produces visually and functionally different layouts. For example, in the first column of the wedding category, if we change the image from a house to a couple, the result is changed from a more text-heavy layout (first row) to a layout with a large image and sparse text (second row), to give readers a stronger sense of romance. In the first column of the travel category, if we change the house image (first row) to an image of a sea scene (third row), the generated layout reduces the amount of text significantly with more white space around it. This gives readers more room to take a visual break, thereby making them feel more relaxed. In addition, in the last row of the travel category, if we change the text from food-related description (first column) to an introduction of a place (second column), the layout changes from a balanced image-text configuration to a large image with few lines of text below it for description. This would deliver a large impact to readers, giving them a stunning first impression of the place. Further, if we change the design category from wedding to travel, there is a large change in the layout style, since travel magazines often have a more dynamic structure with sparse text and large images in order to create a casual reading experience.

Quantitative results. To quantify the influence of input contents on the generated layouts, we perform an experiment to measure how much the layout changes as we vary the input contents. We consider three input factors, image, keyword and category, individually. Given a target input factor, we use our model to generate 10 groups of layouts. In each group, a set of layouts are generated by

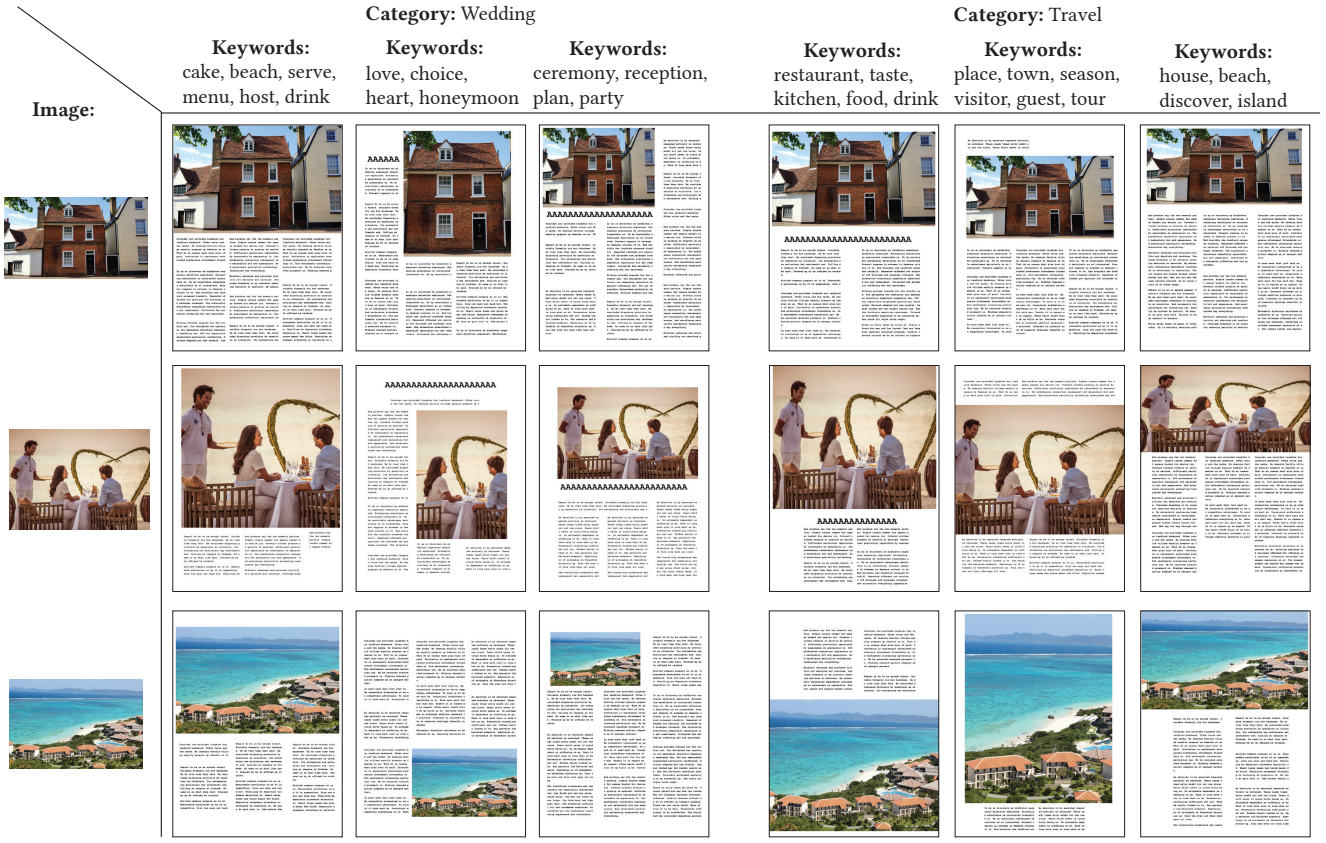

Fig. 10. Effect of visual and textual contents on layouts. From left to right, the input keywords are changed while keeping the image fixed. From top to bottom, the input keywords are the same while the images are changed. The input categories for the left three columns and the right three columns are wedding and travel, respectively. Note that the generated Headlines are filled with A's in bold, and all the examples are generated using the same random vector. The image in the first row is from Pexels, and the rest are from Club Med (© Club Med).

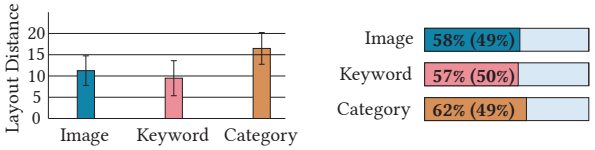

Fig. 11. Left: Quantitative results for measuring the impacts of three input factors (image, keyword and category) upon the generated layouts. For each input factor, we show the mean and standard deviation of layout distances when varying the factor. All the distance means are significantly different from zero (one sample t-test, $p < 0.05$). Right: Results of the user study investigating if people can recognize the influence of input contents upon the generated layouts. For each of the three input factors, participants were shown the content of the factor, a fit layout generated from the same content, and an unfit layout generated from a different, random content of the same factor. We report the fraction of times that participants prefer the fit layouts over the unfit layouts. The number in the parenthesis denotes standard deviation. For all factors, the preferences are statistically significant (Chi-square test, $p < 0.05$).

varying the target input factor while leaving the other input factors unchanged. We then measure the layout difference within each group, by embedding each layout into a content-independent layout feature space where we compute pairwise distances between the layouts via L_2 norm. We use the output of the fourth convolutional layer of layout encoder E in Figure 3 (the last layer before fusing with multi-modal content features $\mathbf{y}$) as the layout features, which capture high-level layout semantics (as demonstrated in Section 2.4 of the supplemental). Finally, we aggregate all the distances across different groups, resulting in a set of *layout distances*. In particular, for category, we use 6 different input categories, resulting in 150 unique layout distances. For image and keyword, we use 5 different inputs, giving 100 unique layout distances each.

Figure 11(left) shows the mean and standard deviation of layout distances for each factor. We note that the distance means are significantly different from zero (one sample t-test, $p < 0.05$) for all factors. This suggests that the impact of input contents on the generated layouts is statistically significant. We also note that the impacts of image and category are stronger than that of keyword.

User study. Finally, we conduct a user study to explore if the influence of input contents on the generated layouts are discernible to people. Again, we consider the three input factors, image, keyword and category, individually. Given a target input factor, we show participants the content of the target factor (e.g., "fashion"

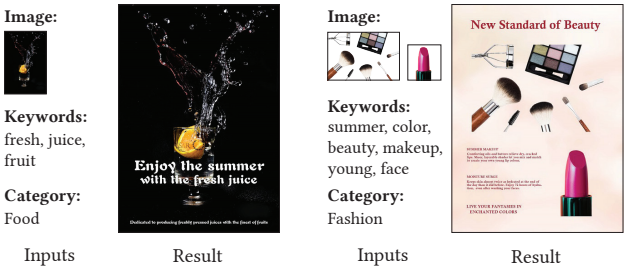

Fig. 12. Advertisements generated by our method. In each case, the input images, keywords, and design category are shown on the left, and the result is shown on the right. The input images are from Pexels.

for design category), alongside two layouts that are displayed side by side in random order. The two layouts are generated from our model by changing the target factor while keeping other factors fixed: a fit layout by using the content of the target factor and an unfit layout by using a different, randomly selected content of the target factor. We then ask the participants which layout better fits the content of the target factor. Our hypothesis here is that if people prefer the fit layouts consistently, it indicates that they are able to recognize the impact of the target input factor on layouts. For each input factor, we collect the responses from 12 participants on 20 different comparisons. Figure 11(right) shows the results. We can see that participants can distinguish between fit and unfit layouts w.r.t. the input category (Chi-square test, $p < 0.05$). Despite a relatively weaker effect of the input keyword on the generated layouts, participants still prefer the layouts that conform to the input set of keywords (Chi-square test, $p = 0.039$). This implies that keyword and design category can impact the generated layouts in different ways that are recognizable by people. The preference over the fit layouts w.r.t. the input image is also significant (Chi-square test, $p < 0.05$). These results suggest that people can perceive the influence of different input factors upon the generated layouts, and that all the inputs of our model are crucial to generating diverse layouts with human-distinguishable characteristics.

6.5 Feature Learning

By being trained to generate layouts, our model learns to map the given layouts into probabilistic latent features conditioned on the image and text contents of the layouts (i.e., $p(\hat{\mathbf{z}}|\mathbf{x}, \mathbf{y})$). We conjecture that in order to succeed in the content-aware layout generation task, the learned features should capture high-level knowledge of the subtle interaction between content semantics (i.e., what to present) and layout structures (i.e., how to present). We validate the effectiveness of our learned features through a layout-aware design retrieval task, which requires understanding layouts and contents jointly.

In the layout-aware design retrieval task, given a query design, our goal is to find other designs that are similar both structurally (layout) and semantically (content). The key to the success of this task is to define a powerful feature representation that allows for meaningful comparison between designs, in terms of both layout and content. For this task, we use the 128-dimensional mean vector from our encoder as our content-aware features, and perform retrieval using L_2 distance. We compare our features with three baselines. For the first baseline, we flatten the $14 \times 14 \times 512$ output of the last convolutional layer from a VGG16 model pre-trained on Imagenet. For the second baseline, we flatten the $64 \times 4 \times 3$ output of the last convolutional layer from the design feature network pre-trained on graphic designs [Zhao et al. 2018]. For the third baseline, we flatten the $4096 \times 10 \times 7$ output of the last convolutional layer from the GDI model (FCN-16s) trained on graphic designs [Bylinskii et al. 2017]. For all three baselines, instead of using the outputs of the last fully connected layers, we use the intermediate convolutional features to capture spatial information in the input designs, which is important for layout comparison. We observe that our features can return the results that are closer to the queries, in terms of both content and layout, as compared with the three baselines. This suggests that our model learns informative features that capture both contents and layouts jointly and their interactions for forming coherent designs. Refer to Section 2.4 of the supplemental for more details.

We also conduct an experiment to visualize the other hidden units of our layout encoder to investigate what semantics that they have learned. The results show that some of the hidden units capture high-level layout-related concepts, e.g., semantic elements and their alignment. Refer to Section 2.4 of the supplemental for more details.

6.6 Application to other Graphic Designs

Although our model is trained on magazine datasets, we argue that our model is generic enough in modeling general graphic design layouts. To demonstrate this, we directly apply our model, without re-training, to synthesize advertisement layouts, as shown in Figure 12. For better visualization, we manually select the font faces and colors used in the results, and add a color pattern to the background of the right example.

7 CONCLUSION AND DISCUSSION

In this paper, we have taken a step towards modeling graphic design layouts conditioned on the image and text contents to be presented. To this end, we have proposed a novel probabilistic generative framework for content-aware layout generation, and constructed a large and diverse magazine layout dataset with rich annotations, including fine-grained semantic layout annotations and keywords for text contents. We have demonstrated that our model naturally supports content-aware layout generation, with or without user input constraints. We analyze how the changes in visual and textual contents influence layout generation, and show that our model can learn the features that capture the interaction between contents and layouts, through a design retrieval task. Our dataset and code are available at our website[1].

We show a typical failure case of our method in Figure 13, where a Text-over-image element is placed at improper locations, occluding some important image regions. This may be because our model does not capture the spatial relationship between image saliency and Text-over-image elements well. One possible solution is to explicitly consider image saliency in our model, which is left as a future work.

Our end goal is to build a fully automated system that is able to translate high-level user inputs into professional graphic design layouts. We believe that there is still a long way from fully achieving

[1] https://xtqiao.com/projects/content_aware_layout/

Fig. 13. Typical failure cases of our method. Our layout generation method may generate unsatisfying layouts where the Text-over-image elements occlude semantically important parts of the input images. In these two examples, the text is placed on the human faces. The input images are from Pexels.

this objective, with a number of interesting questions to be answered in the future.

Encoding the input image dimension. Our approach does not take into account the input image dimension explicitly, and uses a post-filtering step to find the layouts that fit the dimensions of the input images. It would be interesting to condition our model directly on the input image dimension, e.g., by introducing an additional embedding branch to encode such information. This may speed up our layout generation process. However, a post-filtering step may still be needed, as there is no guarantee that the dimensions of the image regions in a generated layout will exactly match the input image dimensions due to the effects of other input factors.

Strong coupling between images and their layouts. Our network models the dependencies between overall semantics of the visual contents and the layout, but is unable to capture the exact correspondence between individual images and their corresponding configurations (e.g., size and location) in a layout. This is partly because we choose a symmetric function to pool all the image feature vectors in order to get a fixed image representation. To alleviate this limitation, we could provide a "shuffle" option for users to explore different ways of assigning input images to image regions in a generated layout, as in Adobe Spark [Spark 2018]. To enforce a stronger coupling between images and their layout properties, instead of directly reconstructing a full layout from our content representation, it would be interesting to explicitly map the feature vector of each image to an image layout (a binary mask encoding its spatial configuration in a layout) and compose all the image layouts for full layout generation.

Enforcing visual design principles. Our approach is purely data-driven and does not explicitly account for some visual design guidelines. We have shown that our current method can generate promising results. However, it may produce some layouts that do not strictly follow some design rules (e.g., slight misalignment). Since the visual design principles are widely known and studied, one solution to address this problem is to incorporate some priors on visual design principles into our network. Another solution is to use a more sophisticated rule-based optimization to refine our network output.

Multi-page layout. Our model only considers the layout of a single-page graphic design. However, in a magazine, there is likely a relationship among the layouts for a sequence of pages, either to provide a consistent appearance or to create a varying reading tempo through the entire magazine. To address this problem, we may combine recurrent neural networks (RNNs) to our model to deal with the temporal dependency of a layout sequence.

Human-machine co-creativity. While we have demonstrated that our results are readily usable for automatic production, it would be interesting to explore a collaborative scenario where both human and machine creativities can be involved in the design process. In particular, instead of using our outputs as end results, human designers may use them as prototypes or a starting point, and tweak them into more personalized layouts. We believe that such human-machine co-creativity has potential to inspire new approaches and expand the boundary of our imagination for graphic designs.

ACKNOWLEDGMENTS

We thank the anonymous reviewers for their valuable comments, and NVIDIA for generous donation of a Titan X Pascal GPU card for our experiments.

REFERENCES

Apostolos Antonacopoulos, David Bridson, Christos Papadopoulos, and Stefan Pletschacher. 2009. A realistic dataset for performance evaluation of document layout analysis. In *Proc. ICDAR*. 296–300.

Michael W Berry and Jacob Kogan. 2010. *Text Mining: Applications and Theory*. John Wiley & Sons.

Andrew Brock, Theodore Lim, JM Ritchie, and Nick Weston. 2017. Neural photo editing with introspective adversarial networks. In *Proc. ICLR*.

Zoya Bylinskii, Nam Wook Kim, Peter O'Donovan, Sami Alsheikh, Spandan Madan, Hanspeter Pfister, Fredo Durand, Bryan Russell, and Aaron Hertzmann. 2017. Learning visual importance for graphic designs and data visualizations. In *Proc. ACM UIST*. 57–69.

Ying Cao, Antoni Chan, and Rynson Lau. 2012. Automatic stylistic manga layout. *ACM TOG* 31, 6 (2012).

Ying Cao, Rynson Lau, and Antoni Chan. 2014. Look Over Here: Attention-Directing Composition of Manga Elements. *ACM TOG* 33, 4 (2014).

Jaime Carbonell and Jade Goldstein. 1998. The use of MMR, diversity-based reranking for reordering documents and producing summaries. In *Proc. ACM SIGIR*. 335–336.

Niranjan Damera-Venkata, Josép Bento, and Eamonn O'Brien-Strain. 2011. Probabilistic document model for automated document composition. In *Proc. ACM DocEng*. 3–12.

Jeff Donahue, Philipp Krähenbühl, and Trevor Darrell. 2017. Adversarial feature learning. In *Proc. ICLR*.

Vincent Dumoulin, Ishmael Belghazi, Ben Poole, Alex Lamb, Martin Arjovsky, Olivier Mastropietro, and Aaron Courville. 2017. Adversarially learned inference. In *Proc. ICLR*.

SM Ali Eslami, Nicolas Heess, Christopher Williams, and John Winn. 2014. The shape Boltzmann machine: A strong model of object shape. *IJCV* 107, 2 (2014), 155–176.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proc. NIPS*. 2672–2680.

Agrim Gupta, Justin Johnson, Li Fei-Fei, Silvio Savarese, and Alexandre Alahi. 2018. Social GAN: Socially Acceptable Trajectories with Generative Adversarial Networks. In *Proc. IEEE CVPR*.

Bernardo Heynemann, Cezar Espinola, and Fabio M. Costa. 2015. Detection Algorithms. http://thumbor.readthedocs.io/en/latest/detection_algorithms.html.

Geoffrey Hinton, Simon Osindero, and YeeWhye Teh. 2006. A fast learning algorithm for deep belief nets. *Neural Computation* 18, 7 (2006), 1527–1554.

Nathan Hurst, Wilmot Li, and Kim Marriott. 2009. Review of automatic document formatting. In *Proc. ACM DocEng*. 99–108.

Phillip Isola, JunYan Zhu, Tinghui Zhou, and Alexei Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proc. CVPR*.

Charles Jacobs, Wilmot Li, Evan Schrier, David Bargeron, and David Salesin. 2003. Adaptive grid-based document layout. *ACM TOG* 22, 3 (2003), 838–847.

Tom Kelly, Paul Guerrero, Anthony Steed, Peter Wonka, and Niloy J Mitra. 2018. FrankenGAN: Guided Detail Synthesis for Building Mass-Models Using Style-Synchonized GANs. *ACM TOG* (2018).

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proc. ICLR*.

Diederik Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *Proc. ICLR*.

Ranjitha Kumar, Jerry Talton, Salman Ahmad, and Scott Klemmer. 2011. Bricolage: example-based retargeting for Web design. In *Proc. ACM CHI*. 2197–2206.

Jun Li, Kai Xu, Siddhartha Chaudhuri, Ersin Yumer, Hao Zhang, and Leonidas Guibas. 2017. GRASS: Generative Recursive Autoencoders for Shape Structures. *ACM TOG* 36, 4 (2017).

Jianan Li, Tingfa Xu, Jianming Zhang, Aaron Hertzmann, and Jimei Yang. 2019. LayoutGAN: Generating Graphic Layouts with Wireframe Discriminator. In *Proc. ICLR*.

Jonathan Long, Evan Shelhamer, and Trevor Darrell. 2015. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*. 3431–3440.

Xudong Mao, Qing Li, Haoran Xie, Raymond YK Lau, Zhen Wang, and Stephen Paul Smolley. 2017. Least squares generative adversarial networks. In *Proc. ICCV*. 2813–2821.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*. 3111–3119.

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2014. Learning Layouts for Single-Page Graphic Designs. *IEEE TVCG* 20, 8 (2014), 1200–1213.

Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. 2015. DesignScape: Design with Interactive Layout Suggestions. In *Proc. ACM CHI*. 1221–1224.

Xufang Pang, Ying Cao, Rynson Lau, and Antoni Chan. 2016. Directing user attention via visual flow on web designs. *ACM TOG* 35, 6 (2016).

ZA Prust. 2010. *Graphic Communications*. Goodheart-Wilcox Publisher.

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

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. ICML*.

Neha Saleem. 2015. The key characteristics of a fashion magazine. http://nehasaleemmedia.weebly.com/blog/the-key-characteristics-of-a-fashion-magazine.

Evan Schrier, Mira Dontcheva, Charles Jacobs, Geraldine Wade, and David Salesin. 2008. Adaptive layout for dynamically aggregated documents. In *Proc. IUI*. 99–108.

Karen Simonyan and Andrew Zisserman. 2015. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*.

David Smith. 2014. Magazine Design Tips: Key Elements. https://www.envision-creative.com/magazine-design-tips-key-elements/.

Adobe Spark. 2018. https://spark.adobe.com/.

Mary Stribley. 2015. 10 Rules of Composition All Designers Live By. https://designschool.canva.com/blog/visual-design-composition/.

Kashyap Todi, Daryl Weir, and Antti Oulasvirta. 2016. Sketchplore: Sketch and explore with a layout optimiser. In *Proc. ACM DIS*. 543–555.

Leon Todoran, Marcel Worring, and Arnold WM Smeulders. 2005. The UvA color document dataset. *International Journal on Document Analysis and Recognition* 7, 4 (2005), 228–240.

Zhirong Wu, Shuran Song, Aditya Khosla, Fisher Yu, Linguang Zhang, Xiaoou Tang, and Jianxiong Xiao. 2015. 3d shapenets: A deep representation for volumetric shapes. In *Proc. IEEE CVPR*.

Xuyong Yang, Tao Mei, Ying-Qing Xu, Yong Rui, and Shipeng Li. 2016. Automatic generation of visual-textual presentation layout. *ACM TOMM* 12, 2 (2016), 33.

Charles Ying. 2014. Automating Layouts Bring Flipboard's Magazine Style To Web And Windows. https://techcrunch.com/2014/03/23/layout-in-flipboard-for-web-and-windows/?ncid=rss.

Nanxuan Zhao, Ying Cao, and Rynson Lau. 2018. What Characterizes Personalities of Graphic Designs? *ACM TOG* 37, 4 (2018), 1–15.

Jun-Yan Zhu, Philipp Krähenbühl, Eli Shechtman, and Alexei Efros. 2016. Generative visual manipulation on the natural image manifold. In *Proc. ECCV*. 597–613.