Content-aware Generative Modeling of Graphic Design Layouts

Supplementary Materials

Xinru Zheng* Xiaotian Qiao* Ying Cao Rynson W.H. Lau

City University of Hong Kong

Contents

1

[1] *Xinru Zheng and Xiaotian Qiao are joint first authors.

1 Our Dataset

1.1 Layout Annotations

1.1.1 FCN Details

Motivated by the outstanding performance of fully convolutional neural networks (FCNs) [Long et al. 2015] on semantic segmentation, we exploit FCNs for semantic layout annotation of our pages. The input to our FCN is an image of size $500 \times 500 \times 3$ (3 for RGB channels), and the output is a mask with a predicted class label for each pixel. Our FCN architecture is based on the FCN-8s network [Long et al. 2015]. We set the number of classes to 6. Since we can only train on a small set of examples, we reduce the number of units in the last two fully connected layers from 4096 to 1024 to prevent overfitting.

We train and test our FCN on a manually-labeled dataset, which is split into 498 pages for training and another 118 pages for testing. In order to alleviate overfitting and improve generalization, we augment our training and test splits by rescaling the height of each page to 384, 512, 640, 768, 896, and 1024, while maintaining the aspect ratio. In this way, we end up with a total of 2,988 pages for training and 708 pages for testing. We initialize the weights of the first 13 convolutional layers using the FCN-8s network pre-trained on PASCAl VOC, and fix the weights of the first 9 convolutional layers. The parameters of the FCN are learned by minimizing a cross-entropy loss. Our FCN is trained with stochastic gradient decent for 55 epochs, resulting in an accuracy of 93.6% on the test data.

To annotate the pages in our dataset, we rescale the height of each page in the dataset to 1,024 while maintaining the aspect ratio, and feed it into our FCN.

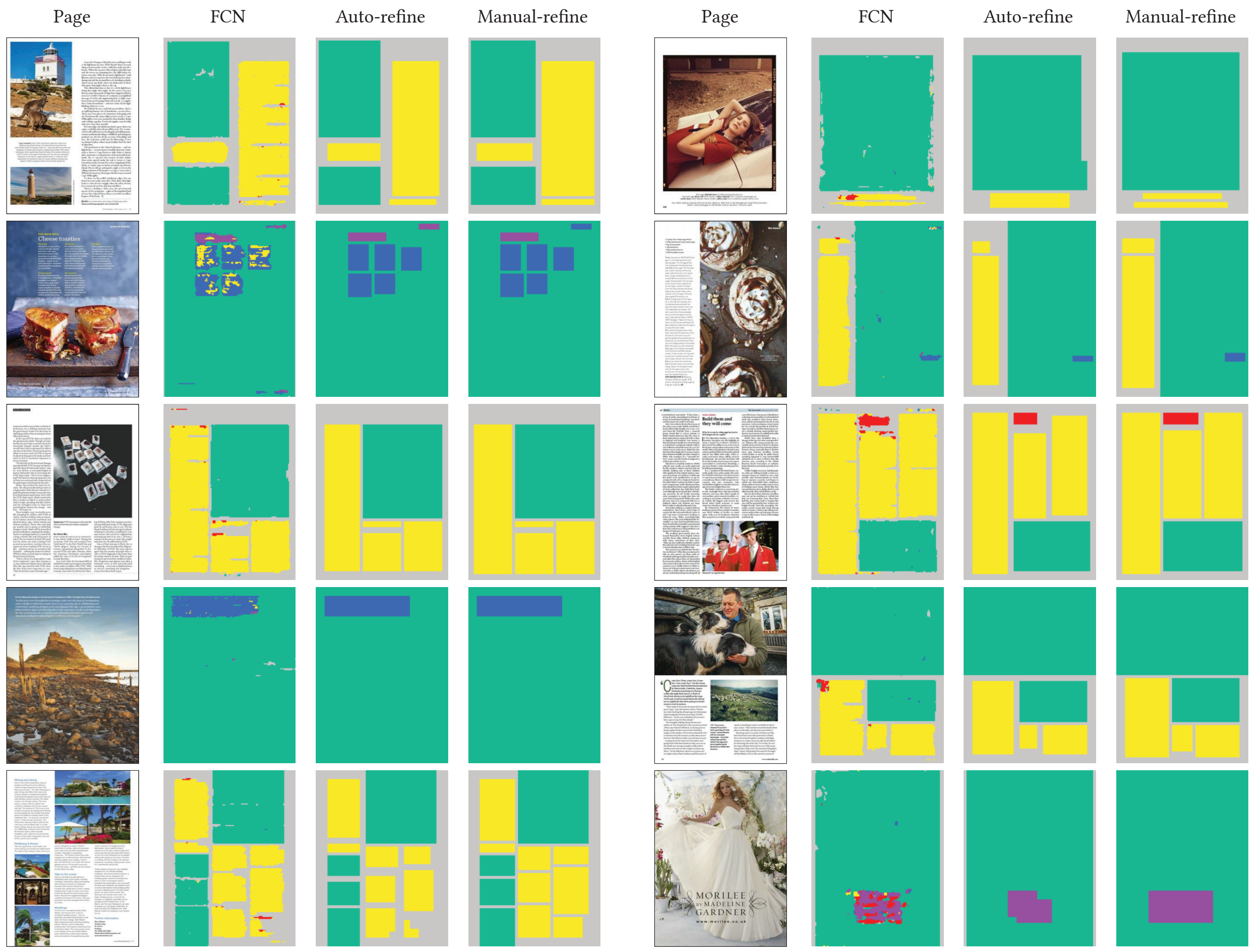

Figure 1: *Results of our FCN, automatic refinement (Auto-refine), and manual refinement (Manual-refine).*

1.1.2 Refinement of FCN Segmentation Results

After FCN segmentation, each segmentation result is refined via an automatic refinement step. In particular, given a FCN segmentation result, we first divide it into two masks: one with images and the other with texts and headlines. We then extract individual elements from these two masks via connected component labeling. The holes in the image mask are filled if they contain text elements that occupy larger than 40% of the hole area. We finally apply refinement operations to these two masks to remove noise and rectify element boundaries.

Our refinement starts with a boundary refinement method, to approximate the boundaries of elements with straight lines, which is detailed as follows. Let h and w be the height and width of the mask. We approximate an element using its bounding box, if its area is less than $a_s = 10S_r \times S_c$ (where $S_r = h/60$ and $S_c = w/45$) and the ratio of its area to the area of its bounding box is larger than $e = 95\%$. The boundaries of such elements are approximated with the corresponding bounding boxes. For each of the remaining elements, we detect the weak connections where the number of consecutive 1's are less than $2S_c$ in the horizontal direction and $2S_r$ in the vertical direction, and replace them with 0's to form a new mask. A set of new elements, referred to as *large elements*, are extracted from this new mask. For each of the large elements, we use a *boundary approximation method* to refine its boundaries. In particular, given an element, we detect each of its outer boundaries as a sequence of points, and smoothen it using 5-point moving average. For each of the smooth boundaries, we approximate it with a straight line based on clustering the coordinates of its points. More specifically, for a left or right boundary, we first cluster the horizontal coordinates of its points based on a threshold of 2. We restrict the maximum number of clusters to 3. Then, for the points of the same cluster, we replace their horizontal coordinate values with the mean horizontal coordinate value of the cluster. A top or bottom boundary is processed in the same way but based on vertical coordinates. In this way, the outer boundaries of all large elements are refined. For the inner boundaries of all large elements, we detect the holes and treat them as *inner elements*, resulting in a hole mask. The boundaries of these inner elements are then refined using the same method as the outer boundaries. After refining the boundaries of the elements, we update the labels of the elements. In particular, for each element in the mask with texts and headlines, we first compute the frequencies of occurrences of the 4 labels (i.e., Text, Headline, Text-over-image, and Headline-over-image) among all the pixels belonging to this element in the FCN segmentation result. We then assign the element with the most frequently occurring label. Finally, we merge the two masks to get automatic refinement result.

Based on the automatic refinement result, we further refine the segmentation manually to address other remaining issues (e.g., incorrect element labels and incomplete element regions). Table 1 shows the pixel accuracy and region intersection over union (IoU) of the FCN segmentation results and those after automatic refinement, which are computed with the manually refined segmentation as the ground truth. We can observe that our automatic refinement improves the pixel accuracy and the IoU of the segmentation results, especially for Text-over-image and Headline-over-image. Figure 1 shows some results of FCN, automatic refinement and manual refinement.

Table 1: *Pixel accuracy and IoU of FCN (FCN) and automatic refinement (Auto-refine).*

Element type	Pixel accuracy (FCN)	Pixel accuracy (Auto-refine)	IoU (FCN)	IoU (Auto-refine)
Text	77%	78%	72%	75%
Image	92%	92%	87%	88%
Headline	59%	62%	38%	47%
Text-over-image	27%	59%	25%	57%
Headline-over-image	25%	47%	20%	38%
Background	92%	95%	75%	76%

1.2 Extracted Keywords

Table 2 shows the top 10 keywords in each category-specific keyword list.

Table 2: *The top 10 keywords for each category.*

Fashion	woman, love, life, friend, style, skin, wear, fashion, dress, top
Food	top, salt, stir, heat, place, taste, recipe, dish, cool, season
News	country, life, government, woman, man, decade, talk, power, state, child
Science	study, scientist, life, place, earth, researcher, university, decade, science, research
Travel	place, life, love, visit, house, country, garden, view, family, top
Wedding	wedding, guest, love, bride, style, couple, ceremony, dress, venue, family

1.3 Statistics of Our Dataset

Our dataset contains 3,919 magazine pages obtained from the Internet, covering 6 common categories, including fashion, food, news, science, travel and wedding. The numbers of pages for the 6 categories are 685, 753, 618, 509, 721, and 633, respectively. We encode the layout of each page with 6 different types of labels, including Text, Image, Headline, Text-over-image, Headline-over-image and Background. Pages from different categories exhibit diverse layouts, in terms of number, size and spatial location of page elements. We show the statistics of our dataset on the number, size and occupation percentage of the 5 element types (i.e., Text, Image, Headline, Text-over-image and Headline-over-image) in Figures 2, 3 and 4. From these figures, we can see that the magazine pages in our dataset have diverse layout structures. For example, some pages exhibit heavy text, while some pages have one or several large images alongside little text. In addition, the pages from different categories show different layout structures. For example, the pages from the fashion category tend to have larger images, while the pages from the news category often exhibit heavier text.

Figure 5 shows the spatial distributions of the 6 element types on a page for different categories in the dataset. The elements tend to be uniformly distributed over the entire page, especially for Text, Image and Text-over-image elements. This demonstrates the diversity of our dataset in terms of spatial layout of page elements. Figure 6 shows some examples from each magazine category in our dataset.

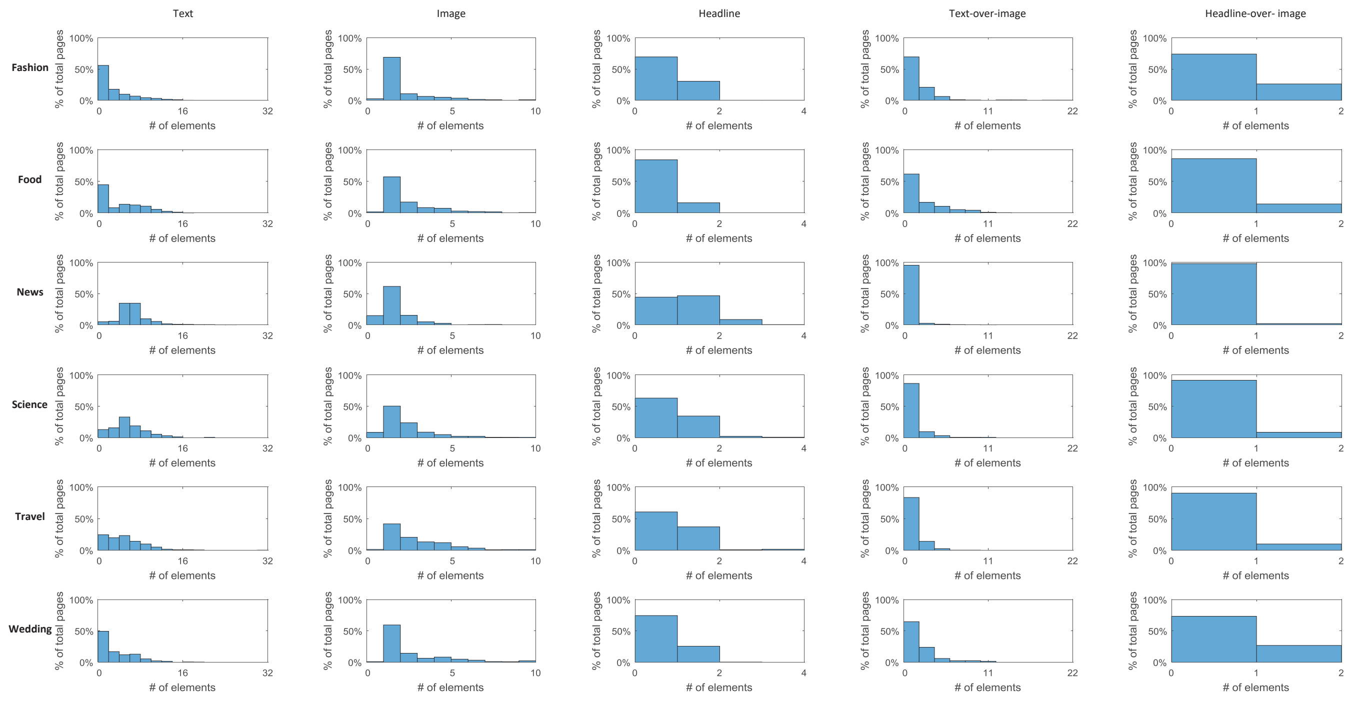

Figure 2: *Distribution of the number of elements for each of the* 5 *element types (from left to right) on a single page within each of the* 6 *magazine categories (from top to bottom). The vertical axis indicates the percentage of pages in the dataset.*

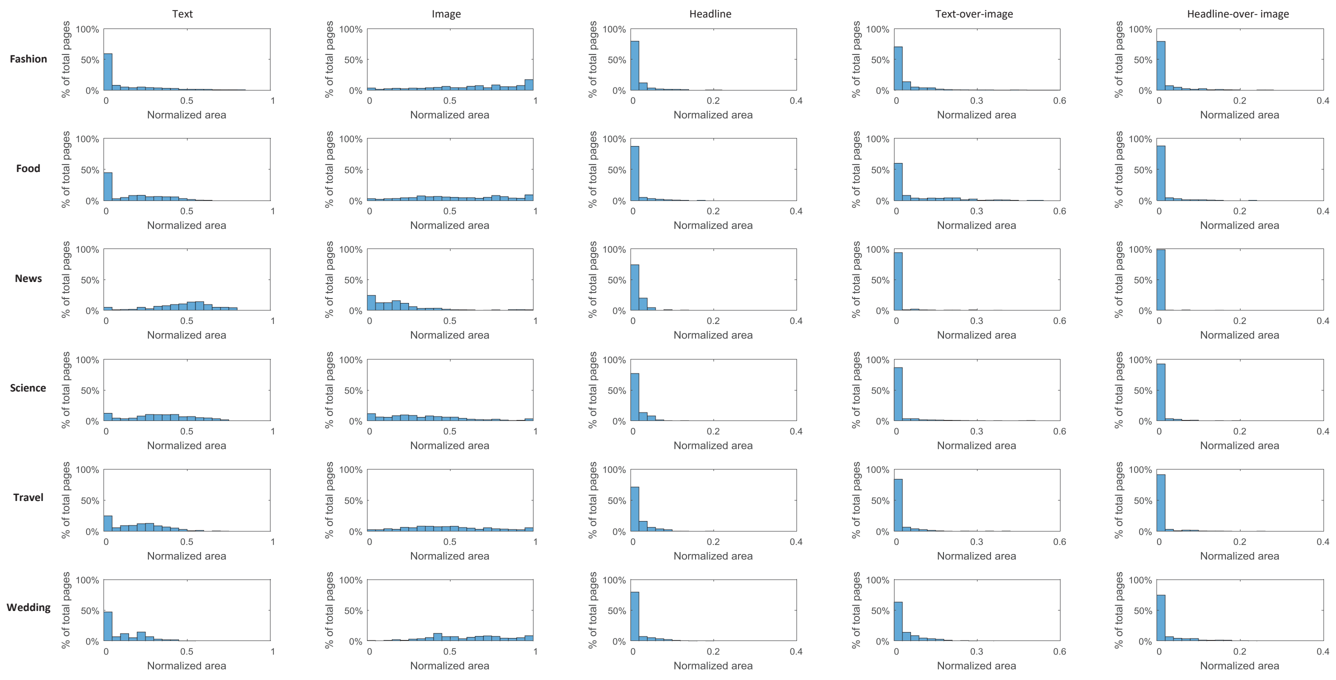

Figure 3: *Distribution of the normalized area for each of the* 5 *element types (from left to right) on a single page within each of the* 6 *magazine categories (from top to bottom). The vertical axis indicates the percentage of pages in the dataset.*

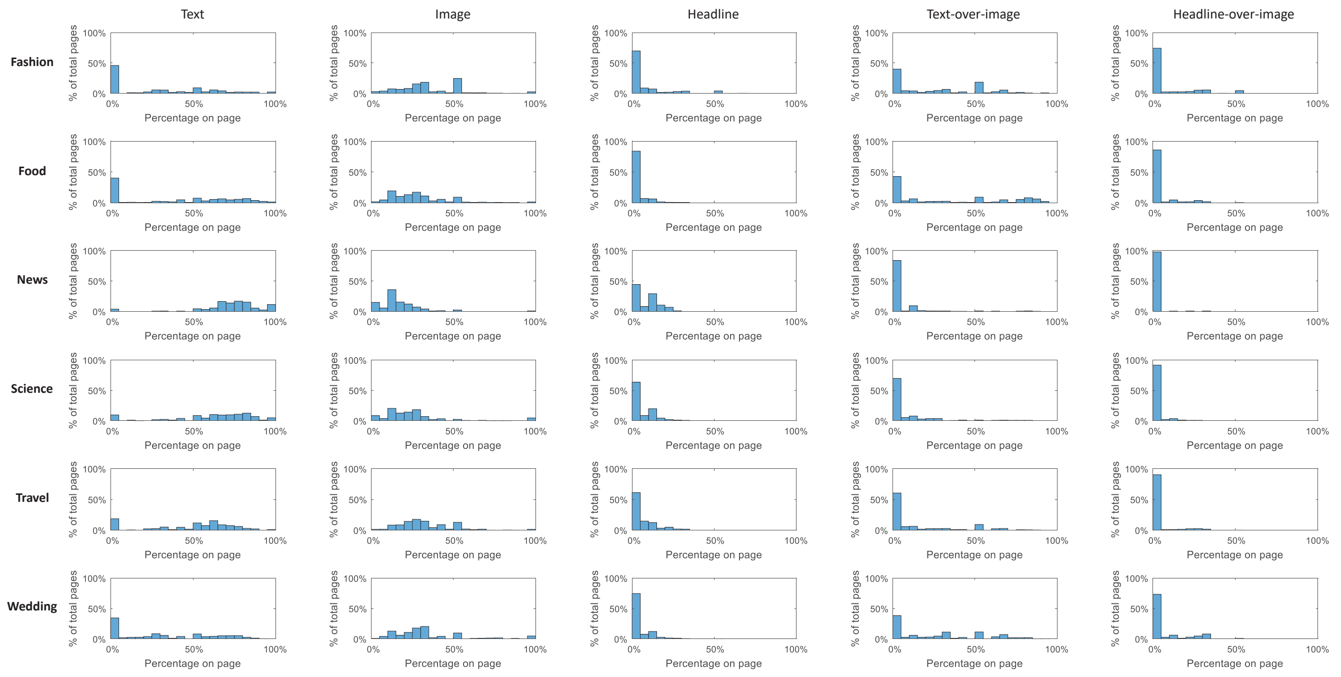

Figure 4: *Distribution of the occupation percentage for each of the* 5 *element types (from left to right) on a single page within each of the* 6 *magazine categories (from top to bottom). The vertical axis indicates the percentage of pages in the dataset.*

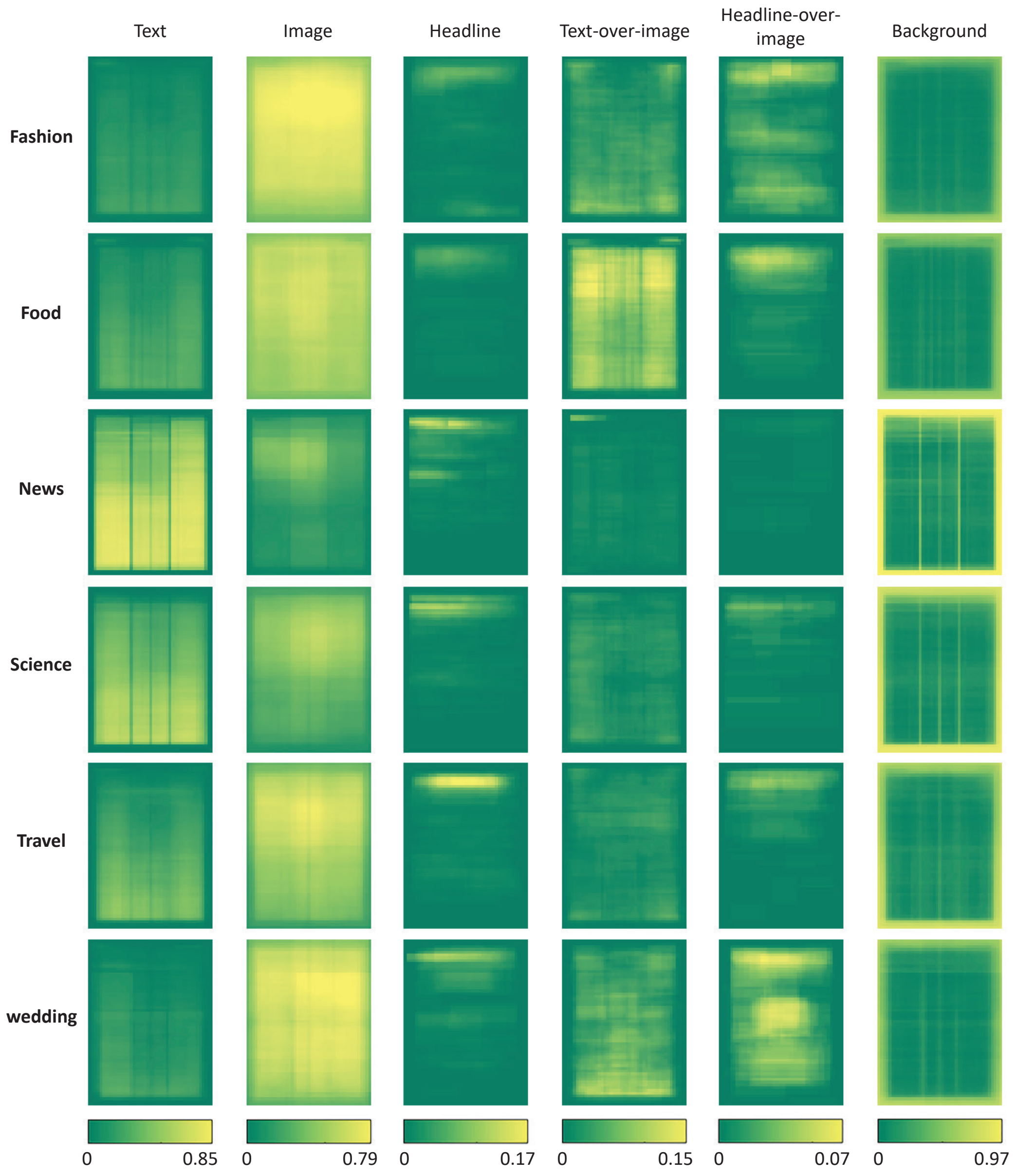

Figure 5: *Spatial distribution of the* 6 *element types (from left to right) on a page within each of the* 6 *magazine categories. The color of a pixel indicates the probability of an element type appearing at the pixel.*

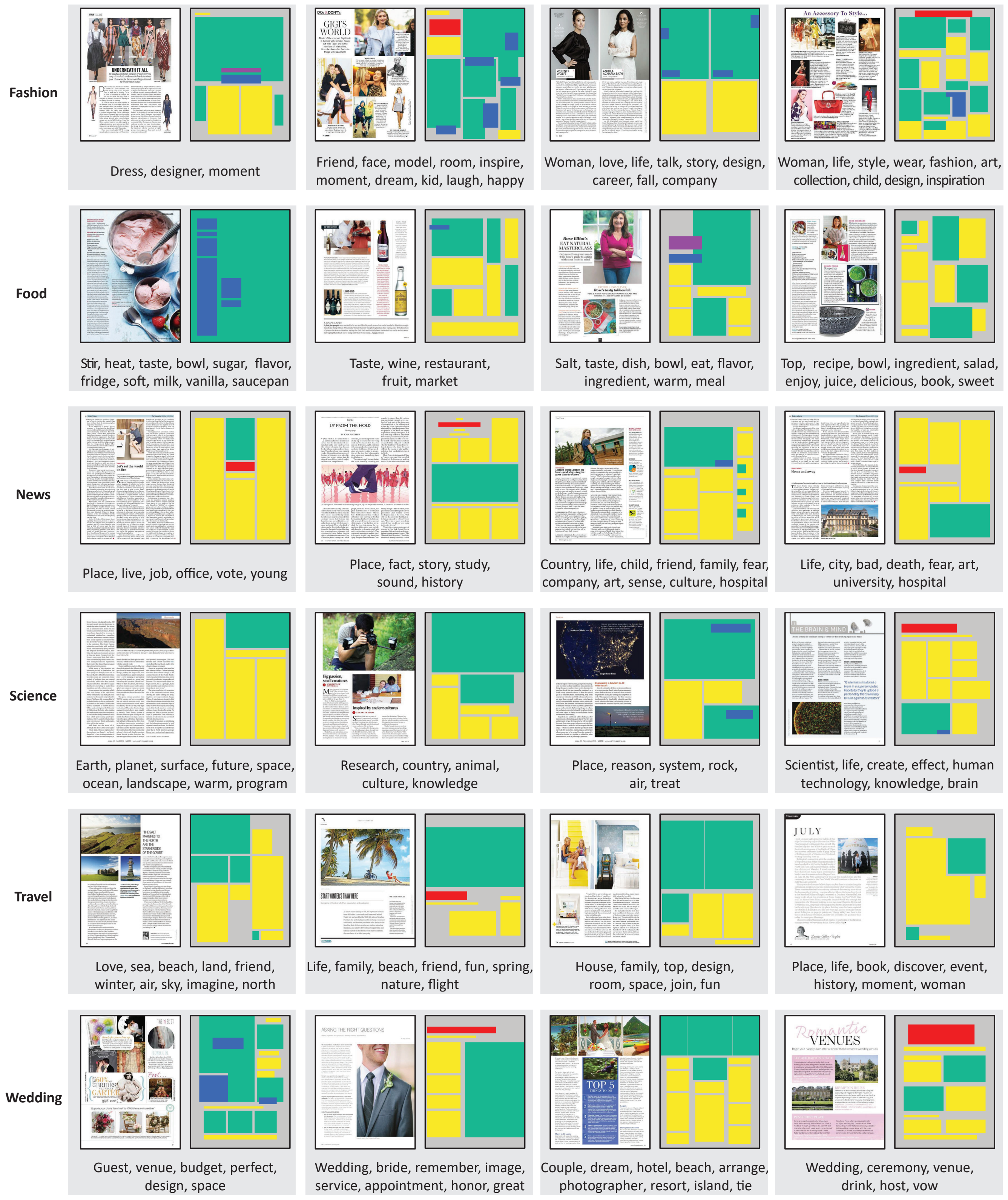
Fashion
Dress, designer, moment
Friend, face, model, room, inspire, moment, dream, kid, laugh, happy
Woman, love, life, talk, story, design, career, fall, company
Woman, life, style, wear, fashion, art, collection, child, design, inspiration
Food
Stir, heat, taste, bowl, sugar, flavor, fridge, soft, milk, vanilla, saucepan
Taste, wine, restaurant, fruit, market
Salt, taste, dish, bowl, eat, flavor, ingredient, warm, meal
Top, recipe, bowl, ingredient, salad, enjoy, juice, delicious, book, sweet
News
Place, live, job, office, vote, young
Place, fact, story, study, sound, history
Country, life, child, friend, family, fear, company, art, sense, culture, hospital
Life, city, bad, death, fear, art, university, hospital
Science
Earth, planet, surface, future, space, ocean, landscape, warm, program
Research, country, animal, culture, knowledge
Place, reason, system, rock, air, treat
Scientist, life, create, effect, human technology, knowledge, brain
Travel
Love, sea, beach, land, friend, winter, air, sky, imagine, north
Life, family, beach, friend, fun, spring, nature, flight
House, family, top, design, room, space, join, fun
Place, life, book, discover, event, history, moment, woman
Wedding
Guest, venue, budget, perfect, design, space
Wedding, bride, remember, image, service, appointment, honor, great
Couple, dream, hotel, beach, arrange, photographer, resort, island, tie
Wedding, ceremony, venue, drink, host, vow

Figure 6: *Some examples from each category in our dataset. Each example contains a magazine page (top left), a labeled layout (top right), and example keywords (bottom).*

2 Our Model

2.1 Architecture of the Layout Generative Network

Table 3 shows the architecture of the layout generative network in our model.

Layout encoder. The input of the encoder is a real layout $\mathbf{x}$, which is represented by a multi-channel image-based representation $(60 \times 45 \times 3)$ and transformed to $64 \times 64 \times 3$ by padding with 0's. The feature map of the fourth convolutional layer is first concatenated with a $60 \times 45 \times 128$ feature map (which is formed by duplicating the corresponding 128-dimensional multi-modal feature vector along the spatial dimensions) and then passed through two convolutional layers. Finally, the encoder produces two feature vectors of size 128 to represent the mean and standard deviation of a Gaussian distribution. A sample from this Gaussian distribution is used as a random latent vector $E(\mathbf{x}, \mathbf{y})$. Note that Batch Normalization (BN) is not applied to the first and last convolutional layers. The first four convolutional layers use Leaky ReLU as activation functions.

Layout generator. The input of the generator is a concatenation of a 128-dimensional random latent vector $\mathbf{z}$ from a standard Normal distribution and a 128-dimensional multi-modal feature vector $\mathbf{y}$. The output of the generator is a generated layout $G(\mathbf{z}, \mathbf{y})$ of size $64 \times 64 \times 3$, which is then converted to $60 \times 45 \times 3$ by removing the padded 0's. The first layer of the generator is a fully connected layer of size $8,192$, whose output is then reshaped to $4 \times 4 \times 512$ for further processing. All the remaining layers are deconvolutional layers with Batch Normalization and ReLU activation functions, except for the last layer, where Batch Normalization is not used and the activation function is Tanh.

Discriminator. The input of the discriminator is a joint pair and $\mathbf{y}$, and the output of the discriminator is a confidence value indicating whether the input is real or generated. The joint pair can be either $(G(\mathbf{z}, \mathbf{y}), \mathbf{z})$ or $(\mathbf{x}, E(\mathbf{x}, \mathbf{y}))$. The multi-modal feature vector $\mathbf{y}$ is first duplicated along spatial dimensions to form a $60 \times 45 \times 128$ feature map, which is then concatenated with the layout (x or $G(\mathbf{z}, \mathbf{y})$) and passed through a series of convolutional layers. The output of the last convolutional layer is concatenated with the random latent vector ($E(\mathbf{x}, \mathbf{y})$ or $\mathbf{z}$), and passed through a fully connected layer to produce a 1-dimensional output. Similar to the architecture of the encoder, Batch Normalization is used for all the convolutional layers except for the first and last ones. The first four convolutional layers use Leaky ReLU as activation functions.

Table 3: *Hyperparameters of the layout generative network.*

Module	Operation	Kernel Size	Strides	BN	Activation	Output Size
Layout encoder Input: $\mathbf{x}$ ($64 \times 64 \times 3$), $\mathbf{y}$ (128) Output: $E(\mathbf{x}, \mathbf{y})$ (128)	Convolution	5	2	No	Leaky ReLU	$32 \times 32 \times 64$
	Convolution	5	2	Yes	Leaky ReLU	$16 \times 16 \times 128$
	Convolution	5	2	Yes	Leaky ReLU	$8 \times 8 \times 256$
	Convolution	5	2	Yes	Leaky ReLU	$4 \times 4 \times 512$
	Concatenation	Duplicate $\mathbf{y}$ spatially and concatenate the result with the output of the previous layer				$4 \times 4 \times 640$
	Convolution	4	1	No	No	128
		4	1	No	No	128
Layout generator Input: $\mathbf{z}$ (128), $\mathbf{y}$ (128) Output: $G(\mathbf{z}, \mathbf{y})$ ($64 \times 64 \times 3$)	Concatenation	Concatenate input $\mathbf{z}$ and $\mathbf{y}$				256
	Fully Connected	No	No	Yes	ReLU	$4 \times 4 \times 512$
	Deconvolution	5	2	Yes	ReLU	$8 \times 8 \times 256$
	Deconvolution	5	2	Yes	ReLU	$16 \times 16 \times 128$
	Deconvolution	5	2	Yes	ReLU	$32 \times 32 \times 64$
	Deconvolution	5	2	No	Tanh	$64 \times 64 \times 3$
Discriminator Input: $\mathbf{x}$ ($64 \times 64 \times 3$), $\mathbf{z}$ (128), $\mathbf{y}$ (128) Output: Real or not (1)	Concatenation	Duplicate $\mathbf{y}$ spatially and concatenate the result with $\mathbf{x}$				$64 \times 64 \times 131$
	Convolution	5	2	No	Leaky ReLU	$32 \times 32 \times 64$
	Convolution	5	2	Yes	Leaky ReLU	$16 \times 16 \times 128$
	Convolution	5	2	Yes	Leaky ReLU	$8 \times 8 \times 256$
	Convolution	5	2	Yes	Leaky ReLU	$4 \times 4 \times 512$
	Convolution	4	1	No	No	128
	Concatenation	Concatenate the output of the previous layer and $\mathbf{z}$				256
	Fully Connected	No	No	No	Leaky ReLU	1

2.2 Architecture of the Multi-modal Embedding Network

Table 4 shows the architecture of the multi-modal embedding network in our model.

Image encoder. We send each image to a pre-trained VGG16 model [Simonyan and Zisserman 2015], to extract a $14 \times 14 \times 521$ image representation. All the image representations within one page are summed up to get a $14 \times 14 \times 512$ representation. We then apply global average pooling forming a 512-dimensional vector. This vector is then fed into 3 fully connected layers, resulting in a 128-dimensional image vector.

Text encoder. For each page, there is a list of keywords. Each keyword is projected into a 300-dimensional word embedding vector using word2vec [Mikolov et al. 2013]. All the word embedding vectors for a page are summed up, and then fed into 3 fully connected layers to get a 128-dimensional text vector.

Attribute encoder. The inputs to the attribute encoder are 3 vectors, representing the category ($\mathbf{b}_1$), text proportion ($\mathbf{b}_2$) and image proportion ($\mathbf{b}_3$). Each vector is duplicated 10 times and fed into a fully connected layer, resulting in three output vectors ($\mathbf{b}_{o1}$, $\mathbf{b}_{o2}$, $\mathbf{b}_{o3}$). The output

Table 4: *Hyperparameters of the multi-modal embedding network.*

Module		Operation	Kernel Size	Strides	BN	Activation	Output Size
Image encoder		Fully Connected	No	No	No	ReLU	512
Input: (512)		Fully Connected	No	No	No	ReLU	256
Output: $\mathbf{i}$ (128)		Fully Connected	No	No	No	ReLU	128
Text encoder		Fully Connected	No	No	No	ReLU	256
Input: (300)		Fully Connected	No	No	No	ReLU	256
Output: $\mathbf{t}$ (128)		Fully Connected	No	No	No	ReLU	128
Attribute encoder	Category embedding	Duplication	Duplicate $\mathbf{b}_1$ 10 times				60
	Input: $\mathbf{b}_1$ (6). Output: $\mathbf{b}_{o1}$ (48)	Fully Connected	No	No	No	ReLU	48
	Text proportion embedding	Duplication	Duplicate $\mathbf{b}_2$ 10 times				70
	Input: $\mathbf{b}_2$ (7). Output: $\mathbf{b}_{o2}$ (48)	Fully Connected	No	No	No	ReLU	48
	Image proportion embedding	Duplication	Duplicate $\mathbf{b}_3$ 10 times				100
	Input: $\mathbf{b}_3$ (10). Output: $\mathbf{b}_{o3}$ (48)	Fully Connected	No	No	No	ReLU	48
	Concatenation	Concatenation	Concatenate $\mathbf{b}_{o1}, \mathbf{b}_{o2}, \mathbf{b}_{o3}$				144
	Input: $\mathbf{b}_{o1}$ (48), $\mathbf{b}_{o2}$ (48), $\mathbf{b}_{o3}$ (48). Output: $\mathbf{b}$ (32)	Fully Connected	No	No	No	ReLU	32
Fusion		Concatenation	Concatenate $\mathbf{i}, \mathbf{t}, \mathbf{b}$				288
Input: $\mathbf{i}$ (128), $\mathbf{t}$ (128), $\mathbf{b}$ (32)		Fully Connected	No	No	No	ReLU	256
Output: $\mathbf{y}$ (128)		Fully Connected	No	No	No	ReLU	128

vectors are then concatenated and passed through a fully connected layer to form a 32-dimensional attribute vector ($\mathbf{b}$).

Fusion layer. The image vector, text vector, and attribute vector are then jointly learned through a fusion layer (including 2 fully connected layers), resulting in a 128-dimensional multi-modal feature vector $\mathbf{y}$ to condition the layout generative network.

2.3 Refinement of the Generator Output

The output of the generator is a layout image ($64 \times 64 \times 3$), which is first transformed to $60 \times 45 \times 3$ by removing the padded 0's. Each value is then rounded to 0 or 1. In the grid of 60×45 cells, all unused values of the binary vector, i.e., $(1, 1, 1)$ and $(0, 0, 0)$, are considered as Background. A refinement step is then applied to this output layout to rectify element boundaries and address the slight misalignment between elements.

To address the jagged edges of the elements, we first approximate the element boundaries with straight lines as follows. We first approximate an element using its bounding box, if its area is less than $a_s = 8$ and the ratio of its area to the area of its bounding box is larger than $e = 95\%$. Then, for each of the remaining elements, we detect the weak connections where the number of consecutive 1's are less than 4 in the horizontal and vertical directions, and replace them with 0's to form a new mask. A set of new elements, referred to as *large elements*, are extracted from this new mask. We then approximate the boundaries of all the large elements using the same boundary approximation method introduced in Section 1.1.2, except that we set the clustering threshold to 2 and limit the maximum number of clusters to 3.

To handle misalignment between elements, we first align the adjacent elements of the same type. To do this, we start by clustering the elements of the same type into one group if they are roughly aligned in the vertical or horizontal direction (i.e., the difference between their vertical or horizontal coordinates is smaller than 2 cells) and close to each other (i.e., their closest distance is smaller than 3 cells). We then construct a bounding box of size $h \times w$ for the group, and project the elements in the group onto the vertical and horizontal boundaries of the bounding box resulting in two binary vectors $u \in R^h$ and $v \in R^w$, respectively. An entry of u or v is set to 1 if the number of the element cells projected on it is larger than $h/2$ or $w/2$, and 0 otherwise. We reconstruct the layout of the elements in the group by taking the outer product of u and v. We further bring all the elements with slight misalignment into top/bottom/left/right alignment. To perform a top-alignment, we first cluster elements into one group if the top boundary coordinates of their bounding boxes differ no more than 2 cells. For the elements from the same group, we then adjust their top boundaries to align with the lowest top boundary to create sufficient spacing between elements. Likewise, we perform bottom, left and right alignments in a similar way.

Figure 7 shows some example outputs of the generator along with the refined outputs.

2.4 Hidden Unit Visualization of the Layout Encoder

We look into the hidden unit of the layout encoder in our model to probe what layout-related semantics that the representation has learned. We follow [Wu et al. 2016] to visualize the graphic design images and the parts of them that maximally activate specific hidden units in the fourth convolutional layer of the layout encoder. We choose this layer since it is the highest layer before fusing with multi-modal features and is thus supposed to capture content-agnostic layout semantics. Figure 8 shows the results. We can see that some hidden units prefer certain types of semantic elements on graphic design layout, e.g., the Headline-over-image elements at the first row, while some hidden units emerge to detect alignment between elements, as shown in the second row. These results demonstrate that our representation has learned some high-level concepts regarding graphic design layout.

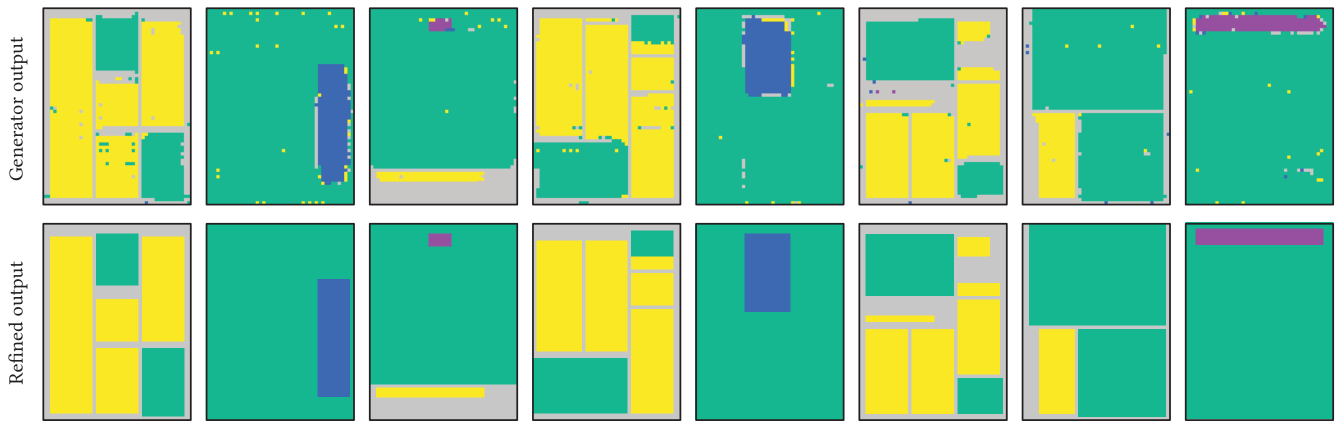

Figure 7: *Refinement of the generator outputs. Top: outputs from the generator. Bottom: the refined outputs.*

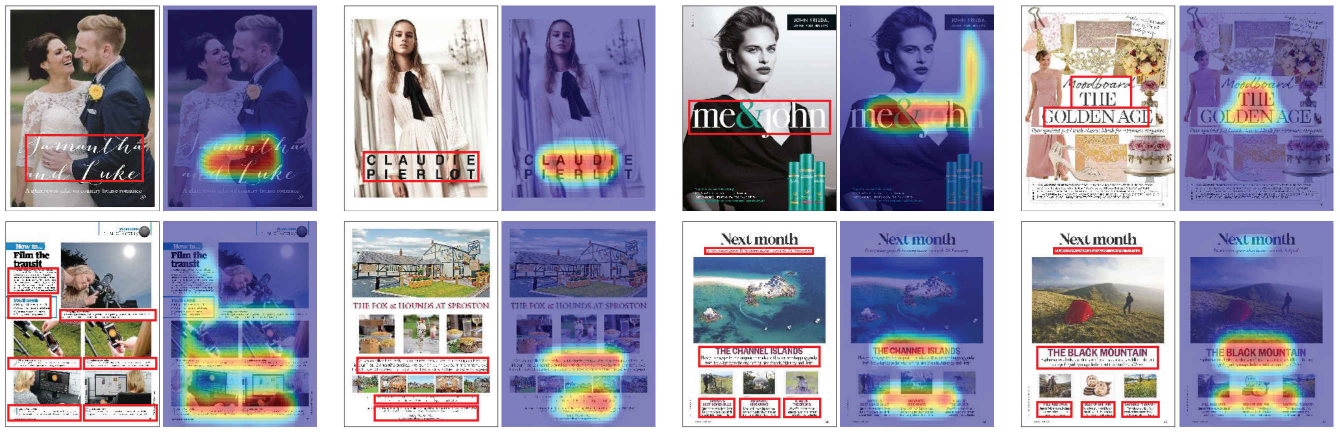

Figure 8: *Hidden unit visualization. In each row, we show four designs and the parts of them that maximally activate a specific hidden unit in the fourth convolutional layer of the layout encoder. Warmer colors represent higher activations.*

3 Results and Evaluation

3.1 Details of Filling Image Elements in the Generated Layouts

To fill an image element with an input image, a focal point (i.e., the center of the most salient region) is detected using the algorithm in thumbor [Heynemann et al. 2015] (a smart imaging service). We use this focal point to guide the cropping of the image so that the image is not distorted and the important image content is preserved as much as possible. In particular, if the aspect ratio of an image element is larger than that of the image, we scale the image to have the same width as the image element while keeping its aspect ratio, and align the centers of both. Since the scaled image has a larger height than that of the image element, we align them along the vertical direction by the center, top boundary or bottom boundary. To accomplish this, we equally divide the image into 3 vertical parts, i.e., top, middle and bottom. We align the centers of the image and its image element if the focal point falls into the middle part of the image. In case if the focal point falls into the top (or bottom) part, we align the top (or bottom) boundaries of both. Finally, we crop the image by preserving the part within the image element. If the aspect ratio of an image element is smaller than that of the image, we scale the image to have the same height as the image element while keeping its aspect ratio, and align the centers of both. We then determine, in the horizontal direction, how to align the centers/left boundaries/right boundaries of the input image and its image element in a similar way.

3.2 Automatic Layout Generation

Figure 9 shows all the results used in the user study in Section 6.2 of the main paper.

3.3 Constrained Layout Generation

Figure 10 shows the remaining results used in the user study in Section 6.3 of the main paper.

3.4 Feature Learning

Figure 11 shows more retrieval results with VGG-based features (VGG), design features from [Bylinskii et al. 2017] and [Zhao et al. 2018], and our learned features (Ours).

References

BYLINSKII, Z., KIM, N. W., O'DONOVAN, P., ALSHEIKH, S., MADAN, S., PFISTER, H., DURAND, F., RUSSELL, B., AND HERTZMANN, A. 2017. Learning visual importance for graphic designs and data visualizations. In *Proc. ACM UIST*, 57–69.

HEYNEMANN, B., ESPINOLA, C., AND COSTA, F., 2015. Detection Algorithms. http://thumbor.readthedocs.io/en/latest/detection_algorithms.html.

LONG, J., SHELHAMER, E., AND DARRELL, T. 2015. Fully convolutional networks for semantic segmentation. In *Proc. CVPR*, 3431–3440.

MIKOLOV, T., SUTSKEVER, I., CHEN, K., CORRADO, G., AND DEAN, J. 2013. Distributed representations of words and phrases and their compositionality. In *Proc. NIPS*, 3111–3119.

SIMONYAN, K., AND ZISSERMAN, A. 2015. Very deep convolutional networks for large-scale image recognition. In *Proc. ICLR*.

WU, J., ZHANG, C., XUE, T., FREEMAN, B., AND TENENBAUM, J. 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *Proc. NIPS*, 82–90.

ZHAO, N., CAO, Y., AND LAU, R. 2018. What characterizes personalities of graphic designs? *ACM TOG 37*, 4, 1–15.

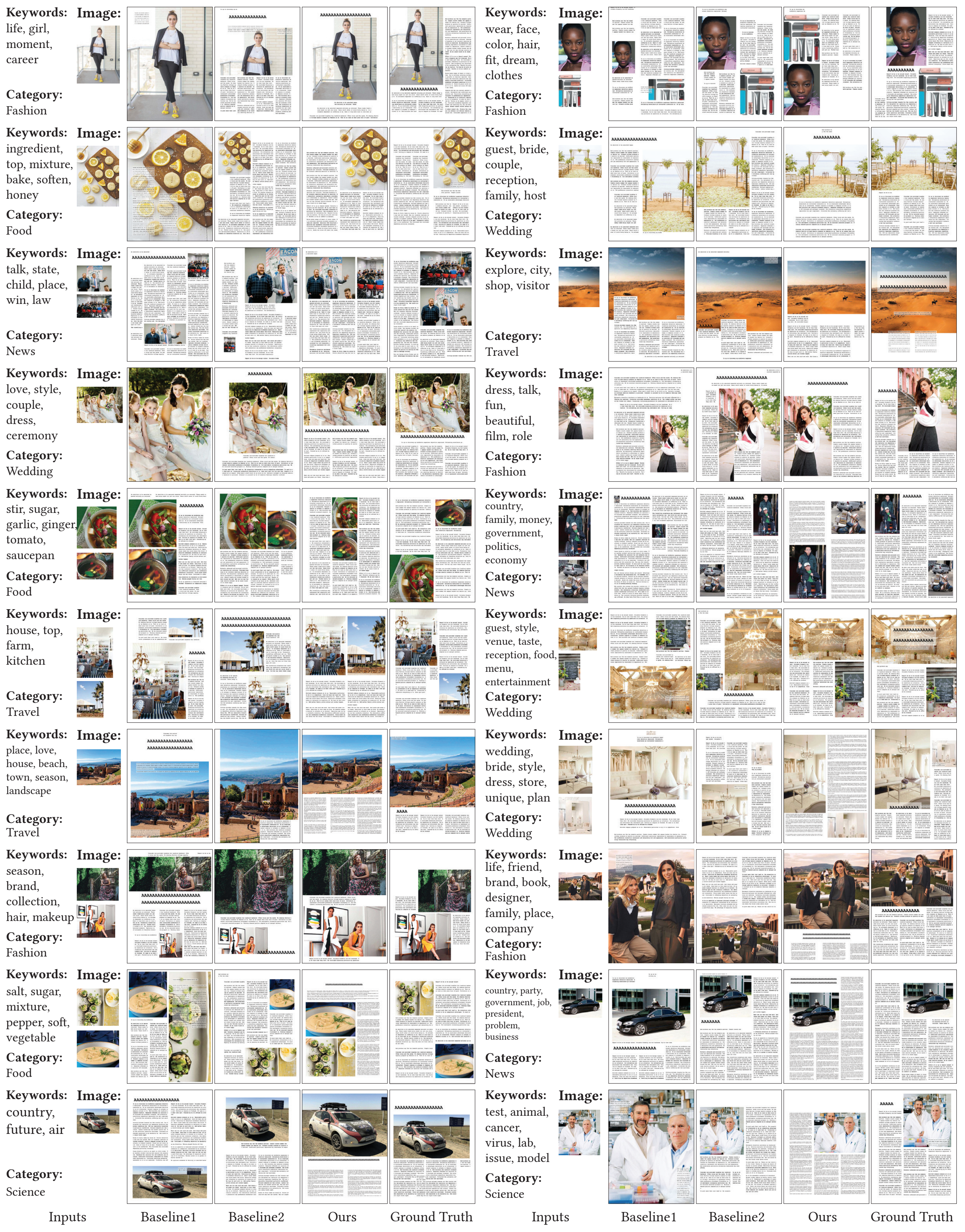
Keywords: life, girl, moment, career
Category: Fashion
Keywords: ingredient, top, mixture, bake, soften, honey
Category: Food
Keywords: talk, state, child, place, win, law
Category: News
Keywords: love, style, couple, dress, ceremony
Category: Wedding
Keywords: stir, sugar, garlic, ginger, tomato, saucepan
Category: Food
Keywords: house, top, farm, kitchen
Category: Travel
Keywords: place, love, house, beach, town, season, landscape
Category: Travel
Keywords: season, brand, collection, hair, makeup
Category: Fashion
Keywords: salt, sugar, mixture, pepper, soft, vegetable
Category: Food
Keywords: country, future, air
Category: Science
Keywords: wear, face, color, hair, fit, dream, clothes
Category: Fashion
Keywords: guest, bride, couple, reception, family, host
Category: Wedding
Keywords: explore, city, shop, visitor
Category: Travel
Keywords: dress, talk, fun, beautiful, film, role
Category: Fashion
Keywords: country, family, money, government, politics, economy
Category: News
Keywords: guest, style, venue, taste, reception, food, menu, entertainment
Category: Wedding
Keywords: wedding, bride, style, dress, store, unique, plan
Category: Wedding
Keywords: life, friend, brand, book, designer, family, place, company
Category: Fashion
Keywords: country, party, government, job, president, problem, business
Category: News
Keywords: test, animal, cancer, virus, lab, issue, model
Category: Science
Image:
Inputs
Baseline1
Baseline2
Ours
Ground Truth

Figure 9: *Automatic layout generation results used in the user study. In each case, the input images, keywords and design category are shown on the left. The layouts by the two baseline methods (Baseline1 and Baseline2), the top 1 layout by our method (Ours), and the ground truth layout (Ground Truth) are shown on the right. For each layout, the Headline element is filled with a sequence of A's in bold. 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.*

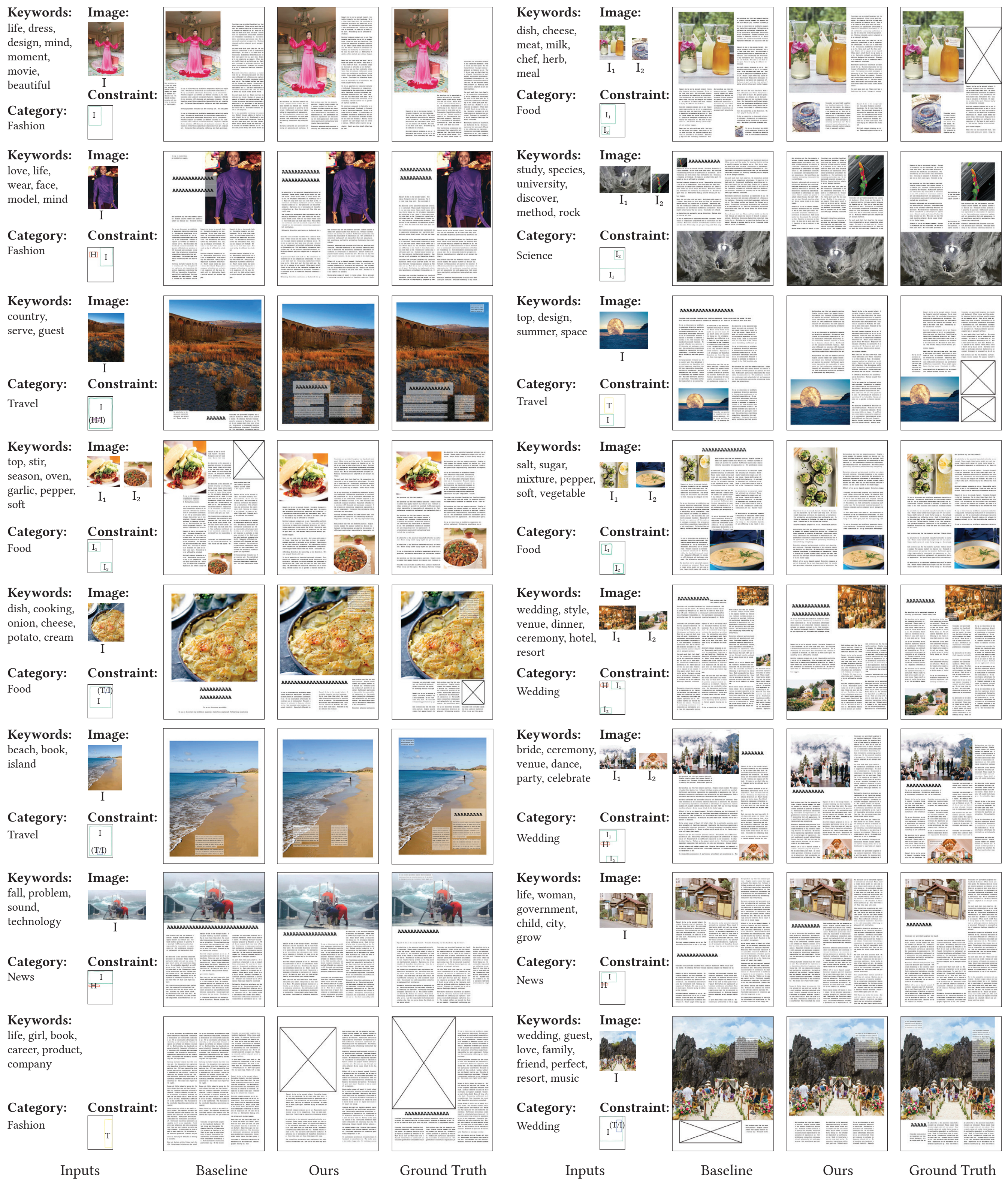
Keywords: life, dress, design, mind, moment, movie, beautiful
Image: I
Category: Fashion
Constraint:
Keywords: dish, cheese, meat, milk, chef, herb, meal
Image: I1 I2
Category: Food
Constraint:
Keywords: love, life, wear, face, model, mind
Image: I
Category: Fashion
Constraint:
Keywords: study, species, university, discover, method, rock
Image: I1 I2
Category: Science
Constraint:
Keywords: country, serve, guest
Image: I
Category: Travel
Constraint:
Keywords: top, design, summer, space
Image: I
Category: Travel
Constraint:
Keywords: top, stir, season, oven, garlic, pepper, soft
Image: I1 I2
Category: Food
Constraint:
Keywords: salt, sugar, mixture, pepper, soft, vegetable
Image: I1 I2
Category: Food
Constraint:
Keywords: dish, cooking, onion, cheese, potato, cream
Image: I
Category: Food
Constraint:
Keywords: wedding, style, venue, dinner, ceremony, hotel, resort
Image: I1 I2
Category: Wedding
Constraint:
Keywords: beach, book, island
Image: I
Category: Travel
Constraint:
Keywords: bride, ceremony, venue, dance, party, celebrate
Image: I1 I2
Category: Wedding
Constraint:
Keywords: fall, problem, sound, technology
Image: I
Category: News
Constraint:
Keywords: life, woman, government, child, city, grow
Image: I
Category: News
Constraint:
Keywords: life, girl, book, career, product, company
Category: Fashion
Constraint:
Keywords: wedding, guest, love, family, friend, perfect, resort, music
Image: I
Category: Wedding
Constraint:
Inputs
Baseline
Ours
Ground Truth
Inputs
Baseline
Ours
Ground Truth

Figure 10: *Constrained layout generation results used in the user study. In each case, on the left are the input contents and the input sketch indicating the approximate positions and sizes of the preferred elements in the output layouts ("T": Text element, "I": Image element, "H": Headline element, "(T/I)": Text-over-image element, "(H/I)": Headline-over-image element). Results by the baseline (Baseline), our method (Ours), and the ground truth (Ground Truth) are shown on the right, where additional image elements that are not specified by the input sketch are marked with a cross, and 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 the baselines and our method are obtained from the ground truth layout.*

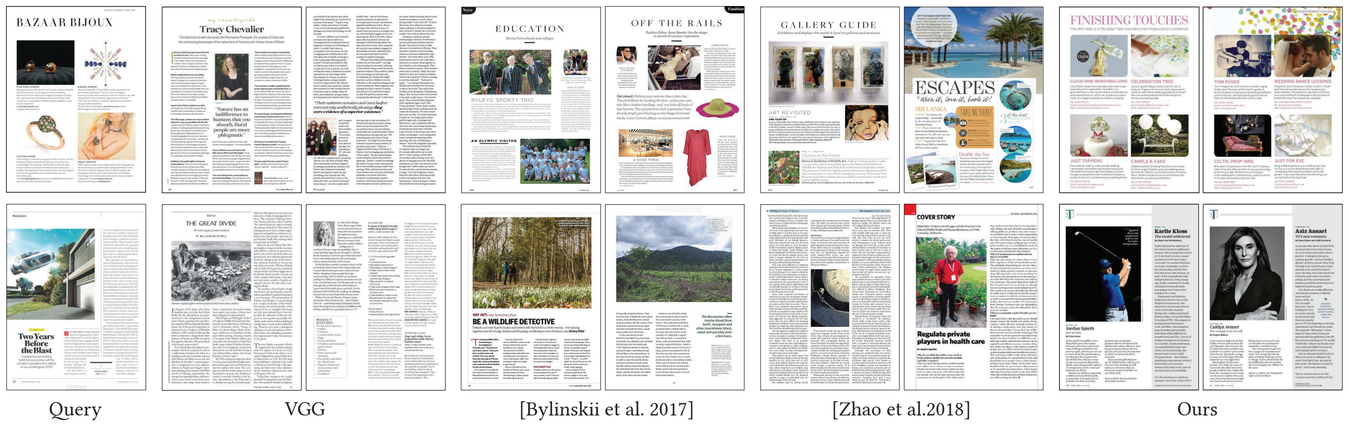

Figure 11: *Layout-aware design retrieval results. For each query design, a ranked list of similar designs in terms of both layout and content are retrieved using VGG-based features (VGG), design features from [Bylinskii et al. 2017] and [Zhao et al. 2018], and our learned features (Ours).*