Design Order Guided Visual Note Layout Optimization

Xiaotian Qiao, Ying Cao, and Rynson W.H. Lau

Abstract—With the goal of making contents easy to understand, memorize and share, a clear and easy-to-follow layout is important for visual notes. Unfortunately, since visual notes are often taken by the designers in real time while watching a video or listening to a presentation, the contents are usually not carefully structured, resulting in layouts that may be difficult for others to follow. In this paper, we address this problem by proposing a novel approach to automatically optimize the layouts of visual notes. Our approach predicts the design order of a visual note and then warps the contents along the predicted design order such that the visual note can be easier to follow and understand. At the core of our approach is a learning-based framework to reason about the element-wise design orders of visual notes. In particular, we first propose a hierarchical LSTM-based architecture to predict a grid-based design order of the visual note, based on the graphical and textual information. We then derive the element-wise order from the grid-based prediction. Such an idea allows our network to be weakly-supervised, *i.e.*, making it possible to predict dense grid-based orders from visual notes with only coarse annotations. We evaluate the effectiveness of our approach on visual notes with diverse content densities and layouts. The results show that our network can predict plausible design orders for various types of visual notes and our approach can effectively optimize their layouts in order for them to be easier to follow.

Index Terms—visual note, design order, layout optimization

1 Introduction

Visual notes, which use a combination of graphical and textual (*i.e.*, content) elements to capture and share ideas, have become a popular way to create powerful visual narratives for a wide range of events (*e.g.*, conferences, workshops or meetings) [1], [2]. As stated in the dual coding theory [3] and the study by [4], the pairing of graphical and textual elements in visual notes can lead to a profound increase in retention and recollection. Therefore, visual notes can be versatile tools with many real applications. For example, they can be used to summarize the key-points of a presentation or drive engagement in a business event [5]. They can also serve as a key time saver in the design process when illustrating or exploring various ideas [6], [7], [8].

Visual notes are meant to be shared with others, and thus must be understandable by the intended readers. To achieve this, a clear and easy-to-follow layout structure is necessary. However, to create a visual note, designers often write down and draw ideas in real time as they are listening to a presentation/talk or watching a video. Thus, given the task of summarizing and abstracting the main points, designers typically lack the full picture of the contents and have limited time in planning the layout of the contents, resulting in some undesirable layouts that are difficult for average people to follow. Without arranging the elements in a clear way or using any guidance, readers will have to focus their concentration more on finding out how to read the content, instead of reading the content itself [9]. As an example, while one may have a pleasant reading experience on a well-structured visual note as shown in Figure 1(a), he/she may have a hard time figuring out which element to read on a visual note with an irregular layout structure as shown in Figure 1(b).

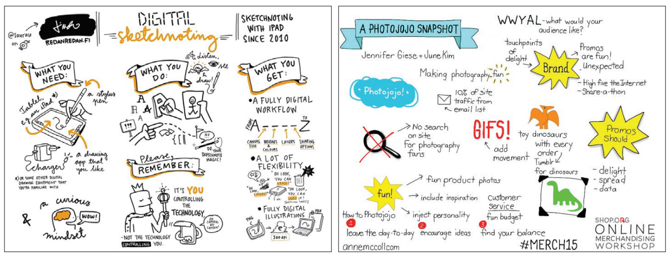


Fig. 1: Examples of visual notes with large layout variations. (a) A well-structured visual note. (b) An irregular-structured visual note that is difficult for readers to follow.

In this paper, we study the problem of layout optimization for visual notes. Given a visual note, we rearrange its elements so that it becomes more efficient to read and understand. Our basic idea is to infer the hidden sequential order of visual note elements that the designer expects readers to follow (referred to as the *design order* in our context). We can then refine the layout of the visual note based on the predicted design order, so that it becomes easier for readers to follow. The key challenge of this goal is to reason about the design order, which is vital to comprehending the contents and having a pleasant reading experience [10], [11]. As shown in Figure 1, visual notes have large variations in element location, shape and number. This hinders the use of any prior layout knowledge for order reasoning. In addition, comprehending the contents requires understanding not only high-level semantics of elements, but also the complex relationship between graphical and textual elements [12], [13], [14].

To address the challenge, we propose a learning-based frame-

- *X. Qiao is with the School of Computer Science and Technology, Xidian University, Xi'an, 710071, China. E-mail: qiaoxt1992@gmail.com.*
- *Y. Cao, and R.W.H. Lau are with City University of Hong Kong, Hong Kong SAR, China.*
- *Y. Cao is the corresponding author.*
- *R.W.H. Lau leads this project.*

work for predicting element-wise design orders of visual notes. However, instead of directly learning to predict element-wise order, which requires a lot of human supervision and suffers from learning inefficiency, we propose a novel, hierarchical Long Short-Term Memory (LSTM) network to first predict the design order over a grid of cells on the visual note at local- and global-levels, and then predict the design order over the elements. The introduction of this *grid-based*, *hierarchical* formulation allows for efficient end-to-end learning of our network from a set of visual notes with only coarse annotations. In addition, we propose to adaptively combine graphical and textual representations via a dynamic weighting scheme, which allows our model to learn an optimal strategy to selectively leverage graphical and textual information, for content understanding and order prediction.

We have conducted extensive qualitative and quantitative experiments to evaluate the effectiveness of our approach on a testing dataset with diverse content densities and layouts. We first show that our layout optimization approach can generate well-structured visual notes that have better readability than the original ones. We then demonstrate that our order prediction model can predict plausible design orders and is generic enough to handle other types of graphic designs (*e.g.*, webpages and magazines). Finally, we show that our model can be applied to layout rearrangement and re-targeting based on different template orders to produce various plausible visual notes.

In summary, our main contributions are:

- To our knowledge, we make the first effort to solve the automatic layout optimization problem of visual notes for better readability.
- We propose a learning-based framework to reason about the design order over elements on the visual note. It is based on a weakly supervised, hierarchical LSTM network that can learn to predict dense grid-based design order with only coarse annotations.
- We demonstrate that our model is generic enough to handle other types of graphic designs, and supports template-based rearrangement and re-targeting applications.

2 Related Work

To the best of our knowledge, we are the first to study the design order prediction problem for optimizing the layouts of visual notes. We review the most relevant works below.

Visual Scanpath Modeling. Predicting visual scanpaths (*i.e.*, temporal behaviors of eye movements) on natural images and webpages has been widely studied. Itti *et al.* [15] proposed the first model using winner-take-all (WTA) and inhibition of return (IoR) on saliency maps to simulate eye movements. Based on the Levy distribution of saccade magnitudes, stochastic models [16], [17] were proposed for scanpath prediction. Wang *et al.* [18] predicted eye gaze paths by modeling reference sensory responses, fovea periphery resolution discrepancy, and visual working memory. Liu *et al.* [19] integrated semantic and saliency information with a Hidden Markov Model (HMM) to estimate readers' scanpaths on images. Xia *et al.* [20] generated complete scanpaths by iteratively predicting fixations and updating the learned representation. Xia *et al.* [21] further proposed a saccadic model for webpages to investigate human dynamic eye movements. Siris *et al.* [22] proposed a model to leverage both bottom-up and top-down attention mechanisms to predict humans' attention shift on an input image. Reinforcement learning [23] and convolutional LSTM [24] were also used to learn visual attention behaviors.

Our ultimate goal of this work is fundamentally different from these prior works. Instead of predicting the reading orders (*i.e.*, the readers' actual scanpaths on visual notes), we aim to automatically infer the design orders (*i.e.*, the paths that the designers expect readers to follow). While the reading orders reflect readers' attention behaviors, the design orders represent designers' high-level intention upon readers.

Graphic Design Layout. Layout is a fundamental component in graphic design. Over the past few years, researchers have made impressive progress in layout synthesis for various types of graphic designs, such as documents, comics, magazines.

Early works developed layout models based on design templates and optimization techniques to find an optimal layout that could satisfy domain-specific criteria. Hurst *et al.* [25] reviewed related works on textual document formatting, including micro- and macro-typographic aspects of layouts. Jacobs *et al.* [26] adjusted online document layouts to different devices by encoding the layout properties in a set of templates. These templates require a lot of professional knowledge and manual efforts to construct. Gange *et al.* [27] provided three new techniques for finding a minimal height table layout, by treating table layout as a constrained optimization problem. O'Donovan *et al.* [28], [29] laid out single-page graphic designs by optimizing an energy function defined by some visual design principles. Cao *et al.* [30] proposed statistical style models for synthesizing comic layouts.

Recently, deep learning-based layout modeling become very popular. Zheng *et al.* [31] proposed a content-aware layout generation framework that could render layouts conditioned on the contents of the user inputs. Li *et al.* [32] proposed the LayoutGAN model to map a random layout (a set of elements with random class labels and geometric parameters) to a refined layout for layout generation. Lee *et al.* [33] proposed a graph neural network to generate design layouts by using a set of elements with user-specified attributes and constraints as inputs. Transformer-based networks [34], [35] were also applied to further improve the quality of the generated layouts. All these works aim at layout generation by learning the layout distribution implicitly from data. In contrast, we apply deep learning to infer the sequential design order on visual notes in a weakly-supervised manner, and use the predicted design order for layout optimization.

The most closely related to ours are the methods that aim to generate or optimize layouts to guide readers along designer-intended paths over elements on graphic designs. Cao *et al.* [36] proposed a probabilistic model to capture the relation among the artist guiding path, composition of comic elements and viewer attention, and used the model for attention-directing composition of comic elements. Pang *et al.* [37] proposed two user attention models to predict temporal behaviors of eye movements (*i.e.*, scanpaths) over webpages, and used them to optimize a webpage layout in order to guide readers along a designer-specified path. Li et al. [38] extended the LayoutGAN [32] to generate layouts conditioned on the predefined reading order of elements by introducing an order loss based on simple heuristics. In contrast to these works that assume the designer-intended path to be produced based on comic-specific heuristics [36] or given as an input by users [37], [38], we assume that such a designer-intended path (*i.e.*, design order) is unknown and learn to predict it directly from images of visual notes. Hence, these existing models cannot be directly applied to our problem. In addition, it is worth noting that

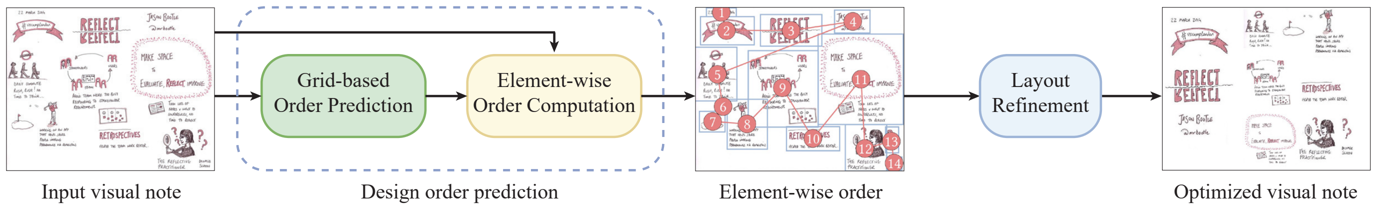


Fig. 2: Pipeline of the proposed method. Given an input visual note, the design order prediction stage predicts the design order over the elements, while the layout refinement stage optimizes the arrangement of the segmented elements based on the predicted design order, making the visual note easier to follow and understand.

although the models in [37] also predict the order of elements on a graphic design, they cannot be used to address our problem for two reasons. First, their models estimate local order between a pair of elements. In contrast, we aim to infer a global order of all elements. Second, training their models requires dense, element-wise order annotations, while training our model requires only coarse, grid-wise order annotations.

Visualization Layout Optimization. There is a line of research on optimizing layouts in information visualization. Gibson *et al.* [39] reviewed related works on graph layout techniques for information visualization. Some methods use optimization techniques for multidimensional projection layouts to avoid overlapping among nodes in a graph [40], [41], [42] or remove unnecessary empty space in a word cloud layout [43]. Gomez-Nieto *et al.* [44] proposed a new overlap removal mechanism for visualization layouts by considering both overlap removal and preservation of neighborhood structures. Gomez-Nieto *et al.* [45] further proposed an optimization method to build layouts from geometric primitives to address multiple concurrent requirements. Liu *et al.* [46] proposed a spatially coherent visualization technique to exploit similarity among a set of items to determine the spatial layout. Carrizosa *et al.* [47] presented an optimization model to visualize complex dynamic datasets by preserving the underlying structure and mental map. Yoghourdjian *et al.* [48] introduced an ultra-compact grid-based layout for node-link diagrams by exploring generic constrained optimization techniques. Unlike these works, we aim to optimize a visual note layout towards better reflecting an inferred design order that the designer wishes readers to follow, which has not been explored before.

3 Our Approach

As shown in Figure 2, our approach mainly consists of two stages: 1) design order prediction stage (Sections 3.1 and 3.2) – given an input visual note, we predict the design order over the elements; 2) layout refinement stage (Section 3.3) – we optimize the arrangement of the elements based on the predicted design order. The key to our approach is a weakly supervised, hierarchical grid-based order prediction model. Our model divides the visual note into grids of cells hierarchically and learns to predict the design orders over the grid cells, which are then used to further predict the design order over the elements. This grid-based order prediction model is independent of visual note segmentation, and can be trained on visual notes with only coarse annotations.

3.1 Grid-based Order Prediction

Given an input visual note, we aim to predict a plausible design order over the elements on the visual note. One naive solution is to learn to predict the order of elements directly. However, training a model for element-wise prediction requires element-level order annotations on a large number of visual notes, which is not available and expensive to collect. To tackle this problem, rather than training a model to predict the element-wise design order directly, we propose to first learn to predict an order over a regular grid of cells defined on the input visual note, and then use the predicted grid-based order to derive the element-wise order. This strategy requires only grid-wise order annotations, which are much cheaper to obtain. We formulate it as a sequence prediction problem, and make use of LSTMs to address the problem due to their encouraging performance in modeling long-term dependency in temporal data [49]. A naive strategy is to use a single LSTM-based network to predict the order of all the cells directly. This will, however, require dense grid-based order annotations, which are still expensive to obtain especially when the number of cells is large. It will also require a very deep network to model a long sequence of grid cells, which would make it susceptible to the gradient vanishing issue [50]. In other words, given a long sequence with many time steps, gradients can become very small during back-propagation, making the network difficult to train. Instead, we propose a weakly-supervised, hierarchical network for grid-based order prediction.

Figure 3 shows the network architecture. Given an input visual note, we first partition it into a *global-level* grid of $K \times K$ cells. Each cell in the global-level grid is further split into a *local-level* grid of $M \times M$ cells. As a result, the input visual note is divided into a grid of $N \times N$ cells in total, where $N = M \times K$. In order to predict the order over all $N \times N$ cells, we first propose a local order network (LocalOrderNet) to predict the design order over the cells in each local-level grid. All local-level order predictions are then passed to a global order network (GlobalOrderNet) to predict the design order over the cells in the global-level grid. Finally, we derive the element-wise order from the grid-based prediction.

There are three main advantages of our network architecture. First, the LocalOrderNets for the local-level grids share the same weights. This reduces the number of parameters to learn. Second, each LocalOrderNet or GlobalOrderNet only models the order of a fixed, small number of cells. It reduces the depth of the network, and thus partially alleviates the gradient diminishing issue. Third, our model can be trained end-to-end with only global-level order annotations, which require a much lower labeling cost. We describe the key modules of our model below.

3.1.1 *Local-level Order Prediction*

We use multiple LocalOrderNets to predict the design orders over local-level grids based on the contents inside the grids. The input to a LocalOrderNet is a local-level grid of $M \times M$ cells. The output is a temporal sequence of the cells, $\langle S_1, \ldots, S_t, \ldots, S_{M^2} \rangle$,

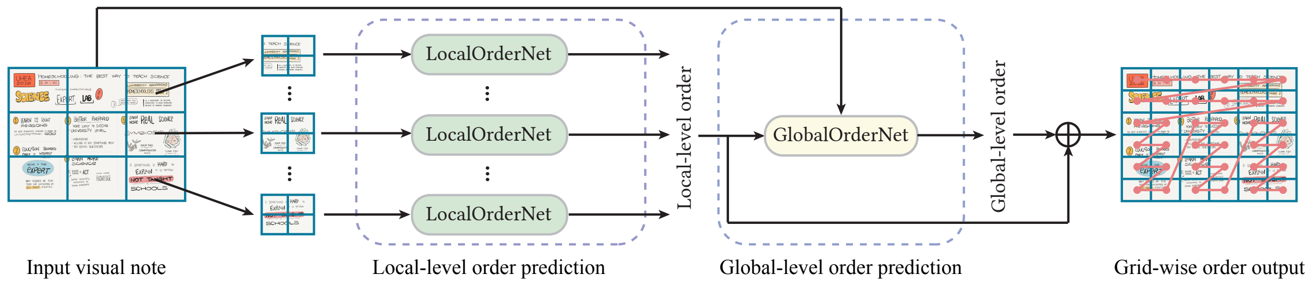


Fig. 3: Architecture of the grid-based order prediction model. The input visual note is first partitioned into a global-level grid of $K \times K$ cells ($K = 3$ here). Each cell in the global-level grid is further split into a local-level grid of $M \times M$ cells ($M = 2$ here). The local order network (LocalOrderNet) predicts the design order over the cells in each local-level grid. The predicted local-level orders are then fed into a global order network (GlobalOrderNet) to predict the design order over the cells in the global-level grid. Finally, the local-level and global-level order predictions are combined to generate the grid-wise order output.

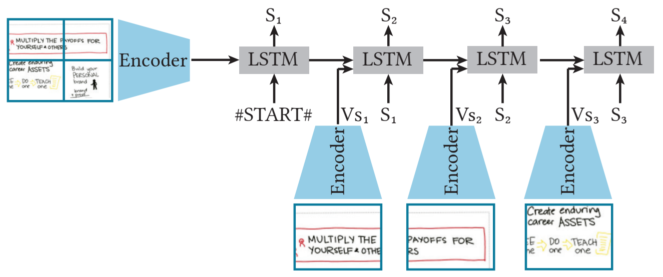


Fig. 4: Architecture of the LocalOrderNet. Given a local-level grid of $M \times M$ cells ($M = 2$ here) as input, the whole grid and individual cells are fed to the encoders to obtain the visual representation and then to the LSTM network to output a temporal sequence of the cells.

where S_t is a one-hot vector representing the index of the predicted cell at time step t. Figure 4 illustrates the architecture of the LocalOrderNet.

To model the long-term dependency of the temporal sequence (*i.e.*, which cell to look at next depends on the cells that we have already looked at) while alleviating the gradient vanishing and exploding problems, we opt for a LSTM-based network [51] to generate a temporal sequence based on the graphical representations of the input grid. A LSTM unit has three gates: input gate, forget gate and output gate. These gates are used to control what information to be added, removed and allowed to flow through. Given input x_t at current time step t along with hidden state h_{t-1} and cell state c_{t-1} from the previous time step, the output o_t, cell state c_t and hidden state h_t are updated using the following recurrent formulas:

$$\begin{aligned} i_t &= \sigma\left(W_i x_t + U_i h_{t-1} + b_i\right), \\ f_t &= \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right), \\ o_t &= \sigma\left(W_o x_t + U_o h_{t-1} + b_o\right), \\ \widetilde{c}_t &= \tanh\left(W_c x_t + U_c h_{t-1} + b_c\right), \\ c_t &= f_t \odot c_{t-1} + i_t \odot \widetilde{c}_t, \\ h_t &= o_t \odot tanh\left(c_t\right), \end{aligned} \tag{1}$$

where i_t, f_t and o_t are the input gate, forget gate and output gate, respectively. $\{W_i, W_f, W_o, Wc\}$ and $\{U_i, U_f, U_o, U_c\}$ are the weight matrices, while $\{b_i, b_f, b_o, b_c\}$ are the biases. They can be trained using the back propagation through time algorithm (BPTT) [52]. $\odot$ is element-wise multiplication. At the first time step, the graphical representation of the input grid, along with a special token START encoded as a zero vector, are passed to the network to output the probabilities of the cells being visited first. After that, at each time step t, the input x_t is a concatenated vector of S_{t-1} and $V_{S_{t-1}}$, where S_{t-1} is a one-hot vector indicating the index of the predicted cell at time step $t-1$, and $V_{S_{t-1}}$ is the graphical representation of the predicted cell at time step $t-1$. The output of the LocalOrderNet is a sequence of probabilities for all the cells in the local-level grid, $O_j \in R^{M \times M}$, where j is the local-level cell index.

To extract visual representations from the input grid and cells, we build an encoder based on VGG-16 [53]. Specifically, we remove all the fully-connected layers and add an additional 3×3 convolutional layer, followed by a fully connected layer of size 81 to generate the visual representation. We initialize the network weights by a pre-trained VGG-16 model on ImageNet [54].

3.1.2 Global-level Order Prediction

We use a GlobalOrderNet to predict the design order over the global-level grid of $K \times K$ cells based on both the contents inside the grid and the local-level orders produced by the LocalOrderNets. The GlobalOrderNet has the same architecture as LocalOrderNet except for the input. Here, at each time step t, the input of a GlobalOrderNet is denoted as $x_t = W\left[S_{t-1}; C_{S_{t-1}}; O_{S_{t-1}}\right]$, where S_{t-1} is a one-hot vector representing the index of the cell at time step $t-1$, $C_{S_{t-1}}$ is the content representation for S_{t-1}, and $O_{S_{t-1}} \in R^{M \times M}$ is a flattened vector of the predicted local-level order probabilities. W is a linear embedding matrix that projects a vector into a 90-dimensional vector x_t.

Dynamic Weighting Scheme. We observe that compared with the cells in a local-level grid, the cells in the global-level grid have a larger spatial extent, and thus contain more meaningful graphical and textual elements. Therefore, for $C_{S_{t-1}}$, instead of solely relying on the graphical representation of the cell, we utilize both graphical and textual information to serve as a condition for our GlobalOrderNet. This scheme is inspired by the observation that in graphic designs, graphical and textual elements usually interact in complex ways and are in a *varying* relationship to convey messages [12], [14], [31]. For example, in some cases, either graphical or textual elements play a dominant role, such that viewers have no difficulty in understanding the contents by looking at only one of the two types. In some other cases,

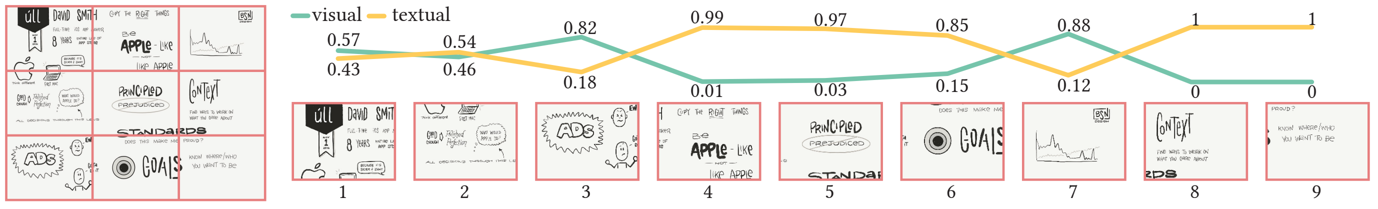


Fig. 5: Relative image-text importance predicted by our model. Given a visual note on the left, our model predicts the relative importance of the visual and textual elements in each cell.

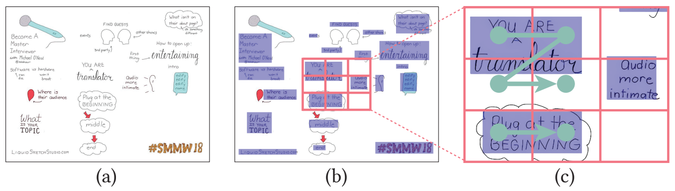


Fig. 6: Local-level order regularization. Given an input visual note (a), we detect text regions on it (purple boxes in (b)). For a local-level grid of $M \times M$ cells ($M = 3$ here) on the visual note (*e.g.*, the red grid in (b)), we first identify the text regions that cover more than one cell as the target regions (*i.e.*, the two purple regions on the left in (c)). For each target region, the pre-defined text reading order is then used as the ground truth order of the cells overlapping it (*i.e.*, the green trajectories in (c)).

however, they may be complementary to each other, such that the graphical or textual elements alone are ambiguous without the other. The objective of this dynamic weighting scheme is to allow our model to account for different relationships between graphical and textual elements by adapting the weights for the graphical and textual representations, *e.g.*, setting the weights for complementary graphical and textual elements to be similar so that both can be used to understand the underlying message.

Given an input cell, we extract a graphical representation $V_t \in R^{h\times w\times c}$ and a textual representation $T_t \in R^{h\times w\times c}$. The content representation $C_t \in R^{h\times w\times c}$ is then computed as:

$$C_t = D_c(M_t) \odot T_t + (1 - D_c(M_t)) \odot V_t, \tag{2}$$

where $\odot$ is element-wise multiplication, $M_t \in [0,1]^{h\times w}$ is a weight map to select textual features for use at each location. $D_c(\cdot)$ is a function that duplicates M_t c times along the feature channel. Each value in M_t can be interpreted as the relative importance of the textual representation at that location. M_t is computed via a dynamic weighting scheme:

$$M_t = f(h_{t-1}, V_t, T_t), \tag{3}$$

where h_{t-1} is the hidden state at time step $t-1$. Note that M_t essentially weights the contributions of the graphical and textual representations in the content representation. f is a multi-layer perception (MLP) to generate the weight map, which allows our model to learn an optimal mechanism to *adaptively* leverage graphical and textual information for content understanding. The proposed dynamic weight scheme provides some degree of interpretability for our model. In Figure 5, we visualize image-text importance prediction by our model on a visual note. For each cell, we compute the importance scores for the visual and textual elements based on our predicted weight map (*i.e.*, M_t in Eq. 3). Specifically, given a cell, we sum up the weight maps of its visual and textual representations to obtain the visual and textual weights, which are then normalized to get the importance scores of visual and textual elements in the cell. By visualizing these importance maps, we can get some clues of what information the model is looking at for its prediction and, hence, have some understanding about why our model works/fails in some cases.

To compute the textual representation, we first detect and recognize words using the OCR API for handwriting from the Microsoft Azure Platform and treat the word regions (bounding boxes detected by the OCR) as textual elements. For each textual element, we then project each word inside it into a 300-dimensional word embedding vector using word2vec [55], sum up the vectors of all words and send the result to a 2-layer MLP to get a 50-dimensional element-wise vector. Finally, we build a $h\times w\times 50$ textual representation by assigning all the pixels inside each textual element with an element-wise vector, and setting all remaining pixels to zeros.

To compute the graphical representation, we first remove the textual elements from the input visual note by filling the textual pixels with the background color of the visual note. We then extract a vector from the resulting visual note using an encoder that is based on a pre-trained classification network [56]. In particular, on top of the last convolutional layer of the network, we add a global average pooling layer, followed by two fully connected layers to output a 50-dimensional vector. Finally, we build a $h \times w \times 50$ graphical representation in the same way as for the textual representation, by assigning all the non-background pixels with the vector, and setting the other pixels to zeros.

3.1.3 Loss Function

For training, we define a joint loss to supervise both local-level and global-level order predictions.

Global-level Order Loss. This loss supervises the global-level order predictions using the order annotations in our datasets. It minimizes the negative log likelihood of the ground-truth temporal sequence $\langle \hat{S}_1, \ldots, \hat{S}_t, \ldots, \hat{S}_{K\times K}\rangle$:

$$L_G = -\sum_{t=1}^{K} \log p_t(\hat{S}_t), \tag{4}$$

where $p_t(\hat{S}_t)$ is the predicted probability of $\hat{S}_t$ at time step t.

Local-level Order Regularization Loss. Since we only have supervision on global-level order predictions, the local-level order predictions are under-constrained. Hence, we introduce an additional loss to provide weak guidance to the LocalOrderNet during training. Our observation is that the cells belonging to a single text region (*e.g.*, a paragraph) are likely to be read according to some common reading patterns that most people follow. We thus define a text-based order regularization loss for each local-level grid by forcing the local-level order prediction within a text region to align with a pre-defined reading pattern (*i.e.*, first from left to right and then top to bottom in our context). In particular, as shown in Figure 6, we use the text detection model in [57] to automatically detect a set of text regions on a visual note. For each local-level grid k of $M \times M$ cells, we identify all the text regions that overlap within it by more than 1 cell and refer to them as the *target regions*. For each target region, we order the grid cells overlapping it first from left to right and then top to bottom to get the target sequence of those cells, $< \hat{Y}_1, \ldots, \hat{Y}_i \ldots, \hat{Y}_n >$, where $\hat{Y}_i$ is a one-hot vector indexing the cells in the grid and n is the number of cells overlapping the target region. We define a loss for each target region l_{rgn} and then sum up all the region losses to get our local-level order regularization loss L^k_{reg}. The loss for a target region is formulated as:

$$l_{rgn} = -\sum_{t=t_0}^{T} \log p_t(\hat{Y}_{t-t_0+1}), \quad (5)$$

where $T = min(t_0 + n, M \times M)$ and t_0 is the time step of $\hat{Y}_1$ in the predicted local-level sequence. $p_t(\hat{Y}_{t-t_0+1})$ is the predicted probability of cell $\hat{Y}_{t-t_0+1}$ at time step t. Note that our regularization loss encourages the order prediction in a local-level grid to locally match the predefined text reading order in each target region.

Our final loss is a weighted combination of the two losses:

$$L_{total} = \alpha L_G + \beta \sum_k L^k_{reg}, \quad (6)$$

where α and β are the weighting factors, and k iterates over the local-level grids of the visual note.

3.1.4 Training Dataset

To train our network, we need a dataset of visual notes with ground-truth grid-based order annotations. Hence, we construct the first visual note dataset with global-level order annotations. It contains a total of 1,500 visual notes. To create this dataset, we first collect a number of visual notes from the Internet (*e.g.*, Flickr and Google) using “visual note” as the keywords. We then manually select representative visual notes with different element types, sizes, numbers, locations and layouts, so that the visual notes in our dataset have diverse content densities and layouts.

For each visual note, we divide it uniformly into a global-level grid of 3×3 cells, and distribute all the visual notes among 3 designers uniformly, so that each note is annotated by one designer. The designers were instructed to label the design order over the global-level grid. The average time of annotating one visual note is around $30s$. In addition, to compute the local-level order regularization loss in Section 3.1.3, we first use the detection model in [57] to automatically locate a set of text regions on the visual note. We then associate the cells with their corresponding target regions based on the amount of overlap. It should be noted that the target regions do not need to be precise nor consistent, since we only utilize them to provide a weak guidance to constrain the learning of the LocalOrderNets, as discussed in Section 3.1.3.

3.2 Element-wise Order Computation

Once the grid-based order is obtained, we reason about the order of the elements. We first need to identify the elements in the input visual note. However, the definition of an element in a visual note can be ambiguous: *e.g.*, it can be an object drawing, a text description or a combination of both. Hence, visual note segmentation is a very challenging problem, which is beyond the scope of the paper. In a real use case, we assume that the elements in an input visual note can be either labeled by the user manually or by an off-the-shelf segmentation method automatically. If an automatic element labeling method is used, any element segmentation ambiguity can then be fixed by a few user clicks via a well-designed interactive interface. In our evaluation, we adopt a simple bottom-up approach to visual note segmentation, without any further user interactions. It consists of three steps: element extraction, merging, and refinement. In the first step, we extract an initial set of elements from the input visual notes. Specifically, we first apply morphological operations (*i.e.*, horizontal and vertical dilations) to the binarized version of the visual note to connect nearby foreground pixels, and then apply a connected component labeling algorithm to obtain a set of connected components with distinct labels, which serve as the initial elements. In the second step, two neighboring elements are merged into a new element if the bounding boxes of two elements have an overlapping ratio higher than 0.3. In the third step, we further refine the resulting elements to handle over-segmentation, producing the final segmentation of the visual note. In particular, we repeat the first step by increasing the kernel size of the structuring element for the dilation so that larger connected components can be formed, and then repeat the second step to generate a set of new element bounding boxes. In each newly generated bounding box, two elements obtained in the original second step are further merged if their closest distance is smaller than a threshold (*i.e.*, 3 pixels in our implementation). Finally, we obtain a set of segmented elements with corresponding labels and bounding boxes.

To reason about the order of the segmented elements, we first map the elements to the cells on the visual note and then compute the element-wise order through an optimization. The mapping is achieved simply by computing an overlap ratio (*i.e.*, the area of intersection between a cell and an element, divided by the minimum area of the cell and the element). A cell belongs to an element if the overlap ratio is larger than 0.5. For those elements without any assigned cells, we assign a cell belonging to such an element if the center location of the element is within the cell. As a result, each element is associated with a number of cells with predicted orders.

Our optimization is summarized as follows. When a cell covers multiple elements, we simply order the elements from left to right and then top to bottom. When an element covers more than one cell, which we have found to be very common, we cast the element order computation as an optimization problem and define a global ranking distance (GRD) as the objective function to minimize the cell ranking distance as:

$$GRD = \frac{1}{|S|} \sum_{m,n \in S} d\left(X_m, X_n\right), \quad (7)$$

where S is a set of elements. $d\left(\cdot\right)$ is the cell ranking distance between a pair of elements X_m and X_n, and is defined by the number of cell pairs whose orders are different from the predicted order as:

$$d\left(X_m, X_n\right) = \sum_{i \in C_m} \sum_{j \in C_n} \left| \delta^{m,n}_{i,j} - \delta^{gt}_{i,j} \right|, \quad (8)$$

where C_m and C_n are the sets of cells in elements m and n, respectively. $\delta^{m,n}_{i,j}$ is 1 if grid cell i in element m is visited before

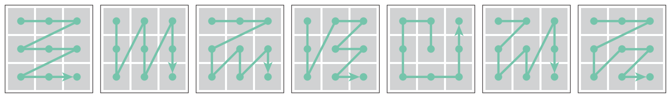


Fig. 7: Some common global-level design order patterns.

grid cell j in element n; otherwise, $\delta_{i,j}^{m,n}$ is 0. $\delta_{i,j}^{gt}$ is the predicted order by our model between grid cells i and j.

To perform the optimization, we first assign each element with an initial order value using the mean order of the grid cells belonging to the element. We then generate a ranked list of elements by sorting the initial values. We optimize the element-wise order using the Metropolis-Hastings algorithm [58], [59] to explore the solution space. The order value of the element is updated in each iteration to a random value from the ranking orders. The optimization process stops when changes are small or a maximum number of iterations is reached.

3.3 Layout Refinement

Given a visual note and its predicted design order, we would like to adjust the visual note layout, *i.e.*, rearrange the elements, to improve its readability. Our hypothesis is that by aligning with the predicted design order, viewers will have a higher chance of reading the elements in the order expected by the artist, resulting in a more readable visual note. Concretely, given a visual note and the global-level and element-wise design orders that are predicted in Section 3.1.2 and Section 3.2, respectively, we refine the layout of the visual note by rearranging its elements in such a way that the refined layout is able to reflect the predicted design order better, while adhering to common visual aesthetic criteria.

Our basic idea is to use the global-level order as a global guiding path that people can easily recognize and follow, and place the ordered elements sequentially along the path. We start by aligning the element-wise order with the global-level order. To do this, we cluster the cells in the global-level grid into horizontal (vertical) groups, so that the cells in the horizontal (vertical) group are supposed to be read consecutively along a horizontal (vertical) direction according to the predicted global-level order. As an example, we consider the leftmost global-level design order pattern in Figure 7. The cells in this gird can be clustered into three groups. Each row with three cells represents one group. We fill each group sequentially by placing the element according to the element-wise order. For a horizontal (vertical) group, the elements are placed along the direction specified by the group cell order at a predefined uniform space, with the center y (x) coordinates of the elements aligned with that of the group cells. If an element, after being placed, falls outside the current group boundary by a large amount, it will be used to fill the next group.

To build a layout structure that can better reveal the global-level design order, we need to create adequate space between the elements. To this end, we further adjust the locations of the elements as follows. We push the elements within the same group closer and those from different groups further apart. In particular, for a horizontal group, we first make its elements closer by moving them slightly towards the center element of the group. We then align the elements with the top/center/bottom of the group if the group is at the top/center/bottom row of the global-level grid, to increase the between-group space. Likewise, for a vertical group, we use the same method to move its element closer and align its elements with the left/center/right of the group if the group is at the left/center/right column of the global-level grid. We further refine the layout of the elements to conform to some visual design principles via an optimization. The objective function includes three energy terms defined in [28], including alignment, white space and overlap. We minimize the objective function using the Metropolis-Hastings algorithm [58], [59], and update the position and size of a randomly selected element at each iteration.

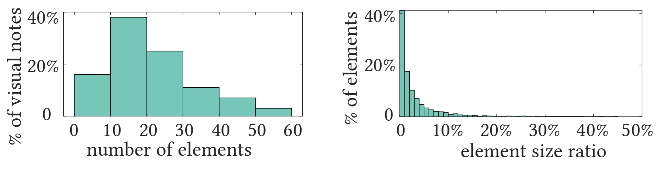


Fig. 8: Statistical summary of the benchmark dataset. Left: the distribution of the number of elements on a visual note. Right: the distribution of the element size ratio (*i.e.*, the area of an element divided by that of the visual note).

4 Results and Evaluation

In this section, we first qualitatively and quantitatively evaluate the performance of our design order prediction model, and apply our order prediction model to handle other types of graphic designs (*e.g.*, webpages and magazines). We then qualitatively and quantitatively evaluate the effectiveness of our layout optimization approach on various visual notes, and perform a user study to investigate whether our results are easier to follow by readers. Finally, we show that our model can be used for layout rearrangement and re-targeting to generate various layouts based on different input template orders.

Implementation Details. The inputs to the encoders in both LocalOrderNet and GlobalOrderNet are rescaled to 288×288. The number of hidden units in each LSTM is 512. Our model is trained with the Adam optimizer at a learning rate of 0.0002, a mini-batch size of 128, and decay parameters of $\beta_1 = 0.9$ and $\beta_2 = 0.999$. The weighting factors for the loss function are $\alpha = 0.5$ and $\beta = 1$. The LSTM weights are initialized with random orthogonal matrices, and other weights are randomly initialized. During testing, we combine both local- and global-level order predictions to output the grid-wise order. In particular, at each time step, we sample the most likely cell and use it as input to the next time step. Inspired by [60], we apply the beam search algorithm in the global-level order prediction to improve the prediction accuracy. The prediction process via beam search would terminate when all the cells have been assigned with a specific order.

To increase the diversity and number of our training examples and to reduce overfitting, we augment our training dataset by synthesizing new visual notes. This is done by resizing each visual note to 1/2 and 2/3 of its original size, and putting the resized visual note at random locations within an empty canvas of the same as the original visual note. As a result, in the synthesized visual note, the order of the cells in the global-level grid may be different from that in the original visual note. We also slightly perturb the sizes of randomly selected elements on each visual note by a random scaling factor between 0.8 and 1.2. This produces a total of 13,500 visual notes for training.

Benchmark Dataset. To evaluate our approach, we have constructed a benchmark dataset of visual notes with ground truth

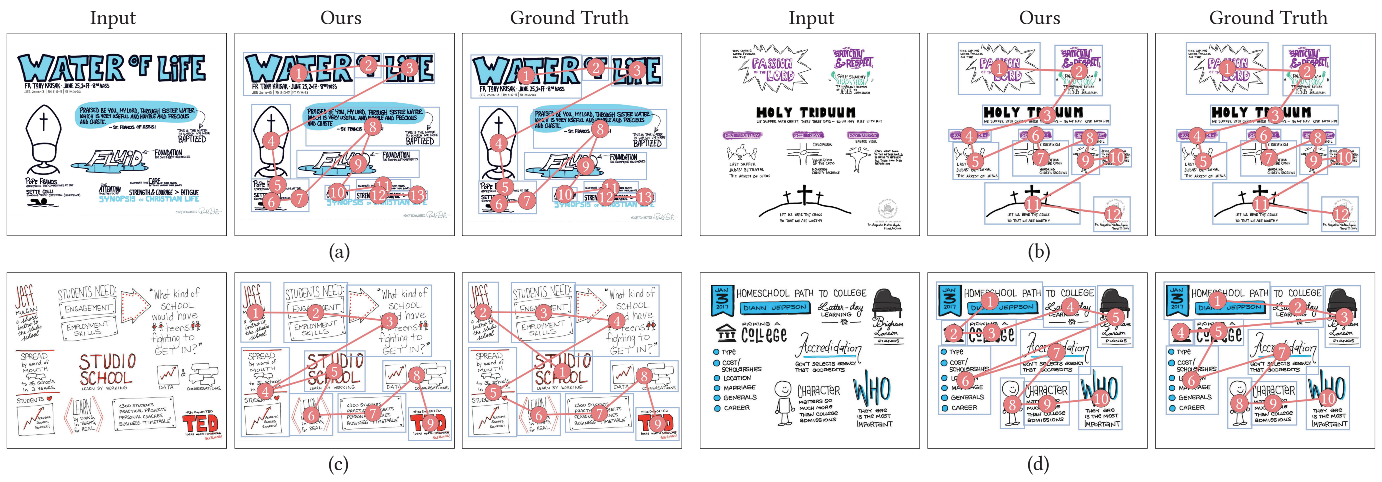


Fig. 9: Results of element-wise design order prediction. The blue boxes on each visual note represent elements, while the numbers and red trajectory represent the predicted design order over the elements.

design orders. We find that the design process of some visual notes is recorded in videos (*e.g.*, whiteboard animations), from which their design orders can be directly derived. Thus, we collect 74 visual notes with accompanying videos, and take the temporal order of elements in each video as the ground truth design order of its corresponding visual note. We have also recruited a designer to annotate the element-wise design order of 26 visual notes without videos. This designer is one of the 3 designers who annotate the global-level order of the training dataset. In this way, we obtain a benchmark dataset of 100 visual notes with ground-truth element-wise design order annotations. Figure 8 shows the statistics of the benchmark dataset. The majority of visual notes have about 10 to 30 elements, and most of the elements cover less than 10% of their visual notes.

4.1 Evaluation of Design Order Prediction

4.1.1 Qualitative Results

Figure 9 shows the prediction results on our benchmark dataset. Predicting element-wise design orders on visual notes can be quite challenging and even ambiguous. Although our model is trained only with coarse annotations (*i.e.*, global-level orders of 3×3 cells), the prediction results show that our method can handle highly irregular visual notes and favorably predict element-wise design orders that are similar to the ground truth. For example, in Figures 9(a) and 9(b), our predictions are exactly the same as the ground truth labeled by the designers. Even though our predictions may sometimes fail to exactly match the ground truth, they still look plausible, as shown in Figures 9(c) and 9(d).

4.1.2 Quantitative Metrics

To quantitatively evaluate the performance of our prediction results, we use three metrics that have been adopted in prior works for visual scanpath comparison, including similarity score (based on the string-edit distance stated in Section 4.2.3), pair-wise order matching rate (POMR), and mean minimum distance (MMD). POMR [37] is a local metric to evaluate the percentage of element pairs whose predicted orders are consistent with the ground-truth. MMD [18] is used to measure scanpath similarity in pixel space. Specifically, it divides a sequence of 2D points $< x_1, x_2, \ldots, x_T >$ into segments $\{C^k(t)\}_{t=1}^{n_k}$, where $C^k(t) =< x_t, \ldots, x_{t+k-1} >$ and $n_k = T - k + 1$. For each segment $C^k(t)$ in a predicted sequence, its closest segment in the ground truth is found in L2 sense and their distance is denoted as $d_k(t)$. MMD is defined as: $\frac{1}{n_k}\sum_{t=1}^{n_k} d_k(t)$. In our experiment, we vary k from 2 to 5 and report the average of MMD values as Avg. MMD.

	Edit dist.	POMR	Avg. MMD
Random	25%	32%	87
Naive Vertical pattern	38%	68%	80
Naive Horizontal pattern	50%	75%	76
No local-level cell representation	59%	81%	75
Local-level grid representation at all steps	58%	80%	74
No dynamic weighting	65%	87%	71
No local-level order regularization	62%	83%	73
Google OCR	**66%**	**89%**	68
ResNet101 encoder	**66%**	**89%**	**67**
Ours (full)	**66%**	**89%**	68

TABLE 1: Comparison of our model against the baselines (top group) and alternative architectures of our model (bottom group) using edit dist., POMR, and Avg. MMD. Best results are highlighted in bold.

Since there are no prior works about design order prediction on visual notes, we compare our approach with several baselines and variants of our model:

- **Random order generation.** For all the elements in a visual note, a random order of these elements is generated.
- **Naive order generation.** According to the psychological analysis on reading patterns [61], humans tend to follow two major reading orders, either from left to right or from top to bottom. Hence, we have two naive effective methods. One is the vertical pattern, *i.e.*, first from top to bottom and then left to right. The other is the horizontal pattern, *i.e.*, first from left to right and then top to bottom.
- **No local-level cell representation.** In our LocalOrderNet, only the graphical representation of the grid is provided at the first time step. No graphical representations are provided for the cells at the remaining time steps.
- **Local-level grid representation at all steps.** In our LocalOrderNet, the graphical representation of the grid is provided at all time steps.
- **No dynamic weighting.** In our GlobalOrderNet, we sum up

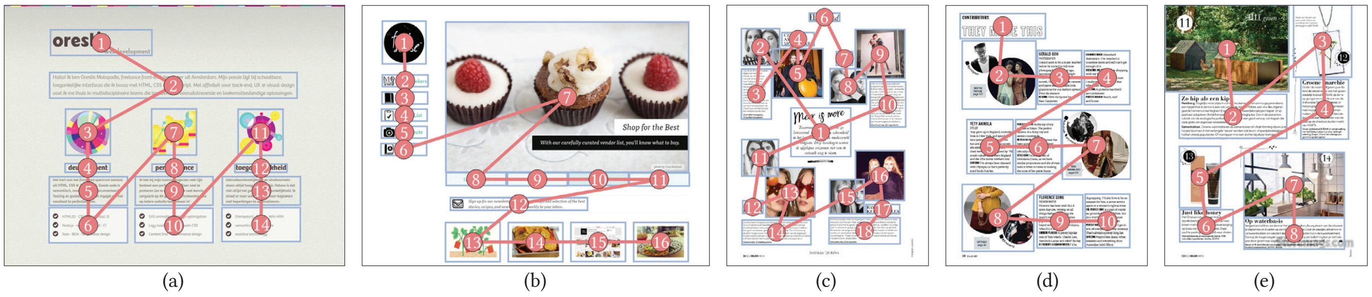


Fig. 10: Design order prediction on general graphic designs. (a) and (b) are webpages, while (c) to (e) are magazine pages.

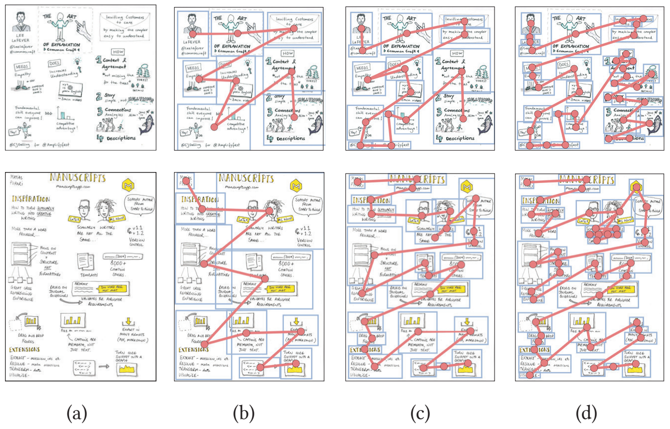


Fig. 11: Design order prediction results with different degrees of visual note segmentation. Given an input visual note (a), we show three degrees of segmentation and corresponding element-wise design order predictions.

image and textual representations directly.

- **No local-level order regularization.** We train our model without the local-level order regularization loss.
- **Google OCR.** We use a different OCR API (*i.e.*, the OCR tool from the Google Cloud Platform) for text recognition and extract textual representation in the same way as described in Section 3.1.2.
- **ResNet101 encoder.** We replace VGG-16 [53] with ResNet101 [62] to extract visual representations from the input grid and cells.

Table 1 shows the results. For all metrics, our method performs favorably against all baselines of our model by a large margin. The naive methods do not work well mainly because most visual notes do not have grid-like, regular layouts, resulting in naive scanning patterns being misaligned with the ground truth design orders most of the time. It is interesting to note that the horizontal pattern has a better performance than the vertical pattern. It seems that designers tend to set the order patterns to be more from left to right. Our model also outperforms its two variants, without cell representation and with only local-level grid representation in LocalOrderNet, which justifies the design of our network architecture. When our dynamic weighting scheme is disabled, the performance drops slightly, confirming the advantage of our scheme for adaptively fusing graphical and textual representations. When training the network without the local-level order regularization loss, the performance also drops, indicating the importance of incorporating additional guidance for the LocalOrderNet. We find that using different OCR algorithms have a similar performance on the visual notes. When the encoder is changed from VGG-16 to ResNet101, only the Avg. MMD score improves slightly, indicating that different visual encoders do not have much influence on the design order prediction result.

4.1.3 Generalization to Other Graphic Designs

We further demonstrate the applicability of our design order prediction model to other types of graphic designs. However, as our model is trained on visual notes and there is a large visual gap between visual notes and other graphic designs, we need to minimize the visual gap between them. To do this, we first convert the images in the original graphic designs into graphical elements via an edge detection method [63], and extract the texts and re-synthesize them as handwritten texts using a recurrent neural network [64]. We then apply our method to predict the design orders of the pre-processed designs. Figure 10 shows the qualitative results on 2 webpages and 3 magazine pages. Although our model did not see such types of designs during training, it can still make reasonable predictions. It is interesting to note that our method can reason about the design order based on the local and global relationships between graphical and textual elements, instead of just following a top-to-bottom or left-to-right reading pattern. For example, in Figure 10(a), each of the three icons at the center of the page and the corresponding text below it constitute a repeated vertical layout, forming three columns of information. Our model captures these layout cues and accurately predicts the design order to move along each column, rather than traversing the three icons horizontally. In addition, in Figure 10(c), our model predicts that the center element is designed to stand out and thus should be read first, before the two pairs of symmetrical image-text combinations (above and below it).

4.1.4 Robustness to Segmentation

As described in Section 3.2, visual note segmentation is a very challenging problem. Automatic segmentation methods may result in over-segmentation (*i.e.*, a single element is divided into multiple elements) or under-segmentation (*i.e.*, many unrelated elements are grouped into a single element). In case of under-segmentation or over-segmentation, since the sizes of most segmented elements are larger than the size of a cell in the local-level grid on the visual note, our approach can predict the design order over most elements favorably. To test the robustness of our approach to visual note segmentation, we generate segmentation results of different degrees (from coarse to fine) on an input visual note and compute element-wise design order based on them. Figure 11

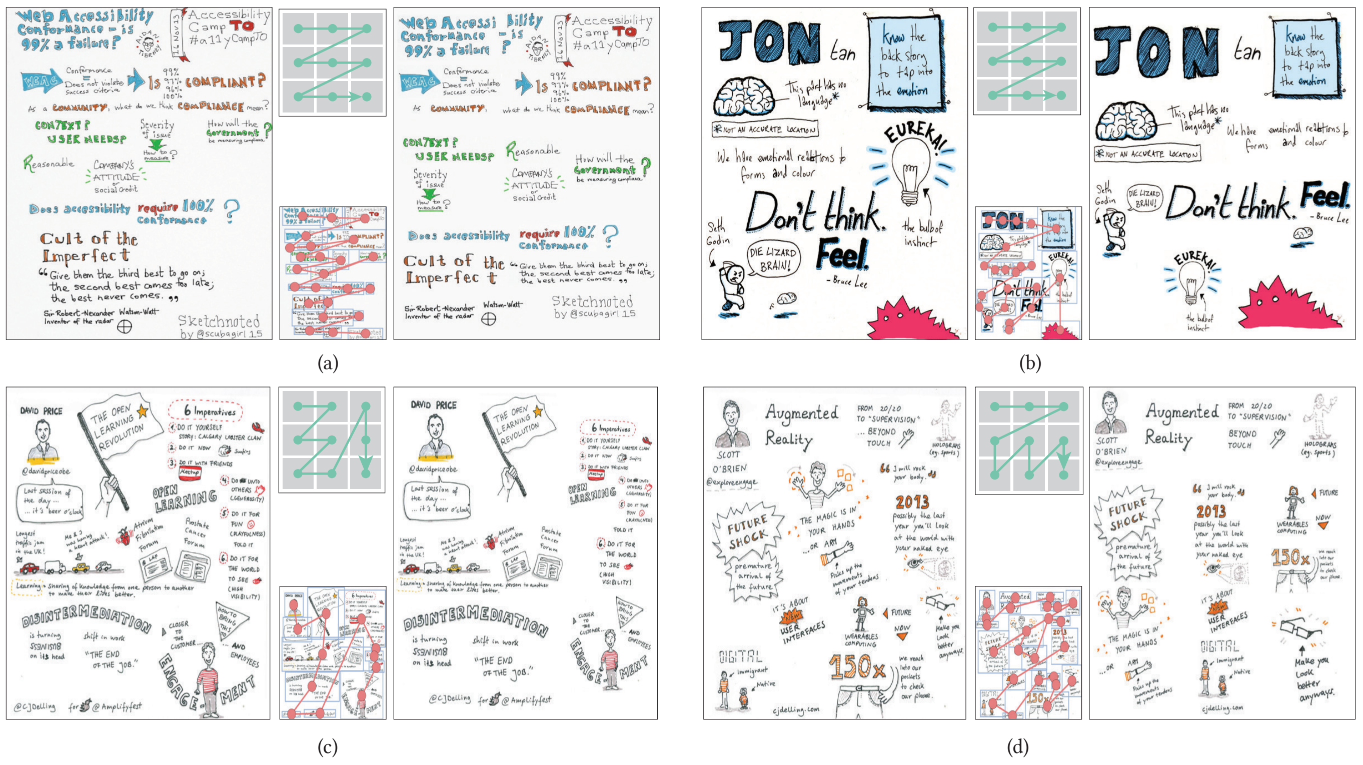


Fig. 12: Layout optimization results. For each example, given an input visual note (left), we predict its grid-based global design order (upper middle) and element-wise design order (lower middle). Based on the predicated design order, the layout of the visual note is automatically optimized such that it is easier for readers to follow along (right).

	Alignment Error	Size Error	Position Error
Input	0.21	0.11	0.18
Baseline	0.14	0.07	0.15
Ours	**0.09**	**0.04**	**0.07**

TABLE 2: Quantitative comparison of layout optimization results by different methods. Best results are marked in bold.

shows the results. We can see that our approach is able to predict plausible element-wise design orders with different degrees of segmentation.

4.2 Evaluation of Layout Optimization

4.2.1 Qualitative results

Figure 12 shows four example visual notes along with their optimized versions. Compared with the original visual notes whose reading orders can be ambiguous, our refined visual notes respect the design orders well and are easier to follow, providing readers with a much smoother reading experience. For example, in the optimized visual note in Figure 12(a), the elements are horizontally distributed based on the predicted design order. Clear empty space is created to allow readers to easily interpret the elements in one line as a group. In contrast, in the original visual note, it is non-trivial for readers to immediately figure out the order that these elements should be read, which may slow down their reading pace and hinder them from understanding the underlying message of the visual note.

4.2.2 Quantitative results

We conduct a quantitative experiment on the aforementioned benchmark. We recruit three professional designers to select good

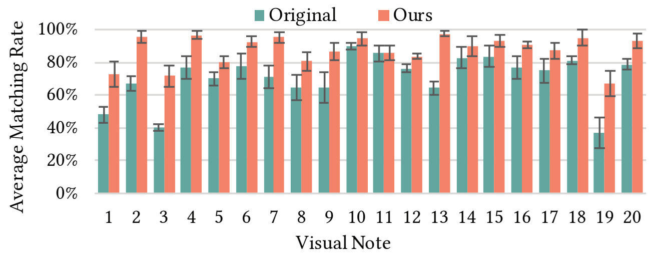


Fig. 13: Results of the user study on layout optimization. We show the average matching rates of the reading orders by the participants with the design orders labeled by the designers on both the original visual notes (Original) and optimized visual notes by our approach (Ours). The error bars represent the standard deviation of matching rates for each visual note. Overall, Ours has an average matching rate of $87\% \pm 2\%$, which is significantly higher than $71\% \pm 3\%$ of Original (paired t-test, $p < 0.01$).

visual note designs from the benchmark. Designers were asked to select the visual notes with good layout structures. Only the visual notes selected by more than two designers will be used for the quantitative experiment. Finally, we have selected 28 well-structured visual notes in the benchmark. For each selected visual note, we perturb its layout by applying random positional offset and scaling to a set of randomly selected elements. The perturbed layout is then sent into our method as input, and the original layout is regarded as the ground truth. We use three evaluation metrics: alignment error, size error, and position error. The *alignment error* is the average alignment difference between corresponding

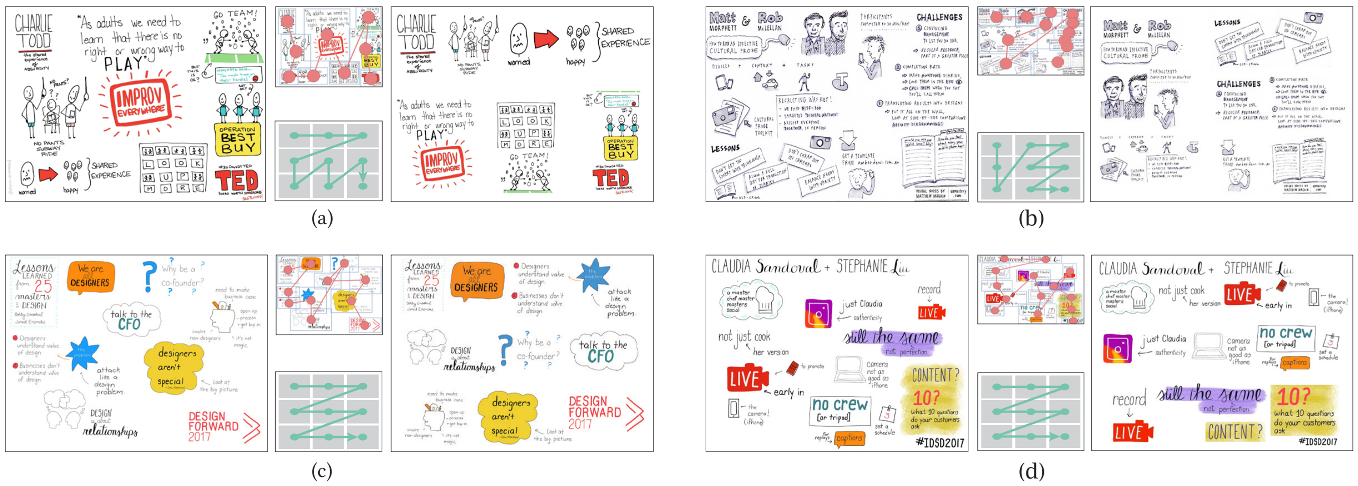

(a) (b) (c) (d)

Fig. 14: Template-based rearrangement. For each example, given an input visual note (left), we predict its element-wise design order (upper middle). Given an order pattern template (lower middle), the layout of the visual note is automatically rearranged such that it conforms to the desired pattern in the template while easier for readers to follow (right).

elements in a generated layout and the ground truth. We consider left, right, center, bottom, and top alignment. The *size error* is the average size difference between corresponding elements in a generated layout and the ground truth. The *position error* is the average L2 distance between the centers of corresponding elements in a generated layout and the ground truth. For comparison, we use the optimization method in our layout refinement step as a baseline, which optimizes input layouts to conform to some design principles without considering design order.

Table 2 shows the results. We can see that the proposed method significantly improves the input layouts, suggesting that it can effectively optimize the structure of input visual notes. In addition, our method outperforms the baseline (without considering design order), which confirms the importance of design order reasoning in the layout refinement task.

We further analyze the computation time of our method for design order prediction and layout optimization. The training of our grid-based design order prediction network takes about 86 hours on a PC with an i7 3GHz CPU, 16GB RAM and a Titan X GPU, while the testing stage of our network takes about $0.231s$ to predict the order of a visual note. For layout optimization, we incorporate the predicted design order and visual design principles into a constrained optimization problem that takes about 3 minutes for a visual note.

4.2.3 User Study

We conduct a user study to evaluate if the optimized visual notes are easier for readers to follow than their original ones. We note that a visual note is considered as easier to follow if the readers' reading paths align well with the design order of the visual note. To measure the similarity between a given reading order and a ground truth design order, we use the string-edit distance [65], which is used in eye-tracking research for comparing scanpaths [66]. The similarity score $S(d_i, d_j)$ between two sequences of elements d_i and d_j on a visual note is defined as:

$$S(d_i, d_j) = 100 \cdot (1 - \frac{Dist(d_i, d_j)}{L}), \quad (9)$$

where $Dist(d_i, d_j)$ is the distance between sequences d_i and d_j by transforming one of them to the other with a minimum number of editing operations, including insertion, deletion and substitution. L is the number of elements on a visual note.

For each visual note, we average the similarity scores of all viewers to obtain an *average matching rate*. We randomly select 20 visual notes from our benchmark dataset. The 20 participants that we recruited have different ages (26 on average) and are from both genders (13 men and 7 women). All of them have no prior experiences in reading visual notes. Each participant was asked to read 20 different visual notes, half of which were selected randomly from the original visual notes and the other half were selected from the optimized visual notes. Each participant could only read the contents of the same visual note once (either the original version or the optimized version). The maximum time of reading one visual note was 3 minutes. For each visual note, a total of 10 different participants were recruited and were instructed to mark their reading orders over the segmented elements. Unlike previous studies on visual attention, we did not use an eye-tracker to capture the participants' reading orders, as eye-tracking data can be noisy without a highly controlled setting, making it difficult to infer a reliable and clear reading order for our study.

Figure 13 shows the average matching rates for both the original and optimized visual notes. We can see that as compared with the original visual notes, the optimized visual notes have significantly higher average matching rates. This implies that our results are easier for readers to follow and the contents are easier to understand. For the visual notes with irregular layouts (*e.g.*, 1, 2, 3 in Figure 13), our optimized results have much higher average matching rates than their original counterparts. For some visual notes with reasonably clear layouts (*e.g.*, 10, 11 in Figure 13), the advantage of our approach is less apparent since the original visual notes already have reasonably good layouts.

4.3 Template-based Rearrangement and Re-targeting

Our approach can naturally support automatic template-based visual note rearrangement and re-targeting. Given a visual note and an order pattern template as input, we generate an output visual note that adheres to the order pattern specified by the template. The templates are used to pre-define some common design order

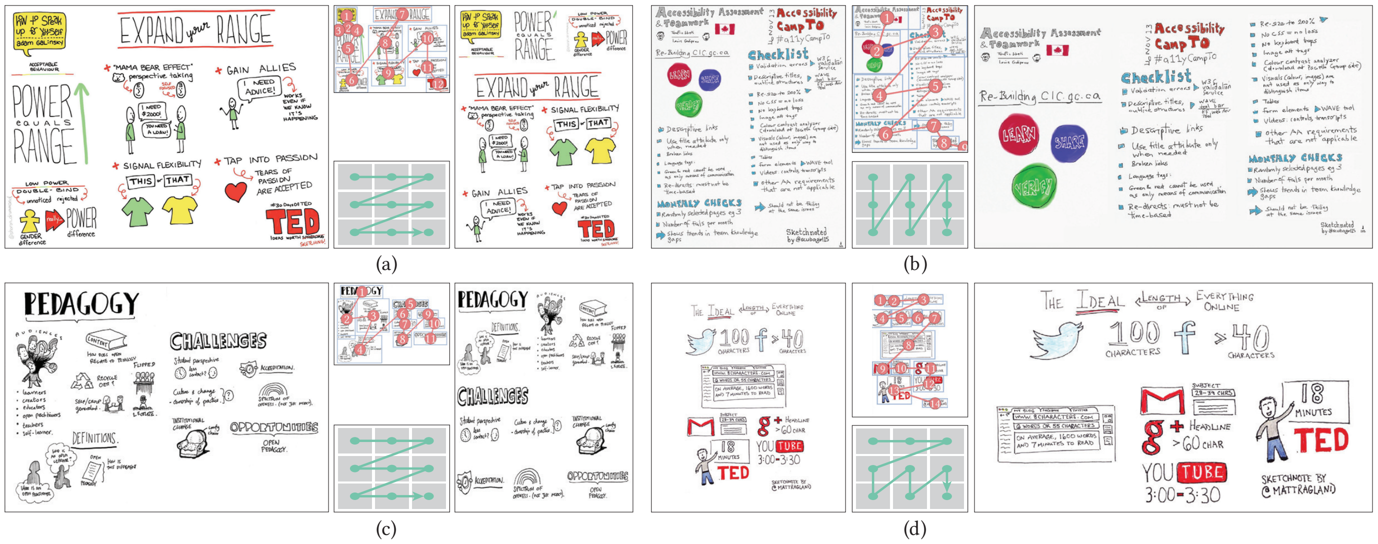


Fig. 15: Template-based re-targeting. In each example, given the input visual note (left), we first predict its element-wise design order (upper middle). Given a target display size and an order pattern template (lower middle), our approach can then re-target the input visual note to the target display size (right), while still ensuring the result to be easy to read according to our predicted design order.

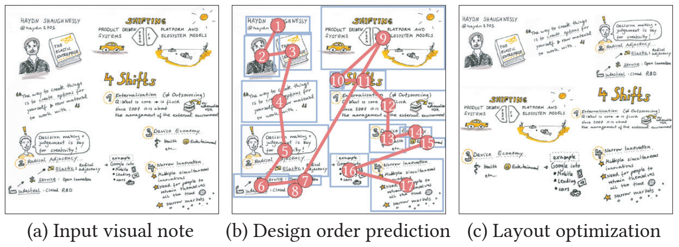


Fig. 16: Failure case. Given a complex visual note (a), our method may fail to give reasonable design order prediction (b) and thus generate a confusing layout (c), as it does not explicitly consider the order number cue on the note (*i.e.*, the numbers in yellow dotted circles).

patterns. The canvas space of each template is divided into 3 × 3 cells, identical to the global-level grid in Section 3.1.2.

Our method proceeds in two steps: an initial layout estimation step and a layout refinement step. In the initial layout estimation step, we predict the element-wise order and global-level order for the input visual note, and align the two orders by rearranging the ordered elements in the 3 × 3 global-level grid according to the global-level order, as done in Section 3.3. We then assign each rearranged element to a global-level grid cell based on the amount of overlap. Since both the global-level order and template order are expressed by an ordered list of 3 × 3 cells, we can establish a one-to-one mapping between the cells in the two orders based on the cell order indices. With the element-to-cell assignment and the mapping, the initial positions of the elements in the output visual note can be determined. Specifically, for a global-level grid cell centered at x^j, we move the centroid coordinates x_i^j of each element belonging to it to its corresponding template cell centered at y^j via $y^j + (x_i^j - x^j)$. In the layout refinement step, we follow the optimization-based refinement method in Section 3.3.

Figure 14 shows some template-based rearrangement results. Compared with the original visual notes, our rearranged visual notes respect the input template patterns well. For example, in the rearranged visual note in Figure 14(c), although the template order is totally different from the original design order, our method can still rearrange the elements on the visual note to follow the template pattern and design rules. In addition, compared to the original visual note, the rearranged one contains sufficient space between elements and is easier for readers to figure out which element to read first and which to follow next.

Figure 15 shows that our method can also be used to re-target a visual note to a different display size, while preserving its design order. For example, the original visual note in Figure 15(b) was designed for portrait mode, whose reading order is supposed to be first from left to right and then top to down (as reflected by our predicted design order). Our method can re-target it into a landscape mode with reading order first from top to down and then left to right, while ensuring the elements to be read according to its original design order.

5 Conclusion

In this paper, we have proposed an approach to automatically optimizing the layouts of visual notes to improve their readability. To this end, we have made the first attempt to predict the design order on visual notes (*i.e.*, the order of elements that the designer expects readers to follow). Once the design order is obtained, our method optimizes the layout of the visual note to be aligned with the predicted design order by rearranging the elements. To predict the design order, we first propose a learning-based framework to hierarchically predict the cell order of a grid defined over the visual note, and then use an optimization to obtain an element-wise design order. Our framework is weakly supervised, and can be efficiently trained with only global-level grid-based order annotations. The results show that our approach can predict plausible design orders and generate well-arranged visual notes that have better readability than the original ones.

Limitations and future works. There are several limitations in our approach for possible future works:

- Designers sometimes use symbols (*e.g.*, order numbers and arrows) to guide readers on how to read the visual notes.

Without considering such a guidance cue explicitly, our model may fail to exploit these small yet important symbols, and thus incorrectly predict the design orders for some visual notes that rely heavily on this guidance cue to determine the order. For example, in Figure 16, the element predicted as 12 should be read before element 5, as indicated by the sequence numbers provided by the designer. Given the input visual note, our grid-based order prediction model incorrectly predicts the global-level order to be in a vertical pattern based on the overall visual appearance and semantics, while the numbers on the visual note indicate that the global-level order should be in a horizontal pattern. It would be interesting to explore how modeling this kind of guidance cues explicitly could help improve our order prediction.

- Although we have shown that our method can give promising results on diverse visual notes, our approach may fail to produce plausible results when an input visual note is extremely over-segmented (e.g., containing more than 80 segmented elements). In addition, some users may not be satisfied with the automatic segmentation results since visual note elements are subjective to define. For example, in Figure 16(b), some users may want to split the element predicted as 9 into a set of finer elements, while others may want to group the elements predicted as 1, 2 and 3 into a single element. An interesting direction for future investigation would be to come up with a simple and intuitive interaction mechanism to incorporate user preferences.
- While our design order prediction stage contains a new deep learning model to learn the sequential order of elements from data, the layout refinement stage still follows the traditional heuristic-based approach. We believe that a more advanced layout refinement approach could further improve the performance. In addition, it would be an interesting future work to train the whole pipeline end-to-end by converting the optimization stage into a differentiable neural network.
- While we have shown the effectiveness of our design order prediction method on visual notes with various layouts, it may have difficulty in making reasonable order predictions for the visual notes with high-density contents and very irregular layouts. This is partly due to the fact that our grid-based design order prediction model has only weak, global-level supervision, which makes it difficult to resolve some local-level order ambiguities. We would like to explore using other unsupervised or self-supervised losses to provide stronger supervisory signals to our model, while minimizing human intervention in model training.
- Our current implementation uses a uniform splitting strategy to divide an input visual note into two levels of grids (*i.e.*, local-level and global-level). We would like to explore a content-adaptive grid generation strategy that can preserve text and pictures well, *e.g.*, by making use of a segmentation method. The hierarchical framework can also be easily extended to deal with multi-levels, *e.g.*, by further splitting each cell in the local-level grid. These would help improve design order reasoning if the visual note is dense and irregular.

Acknowledgments

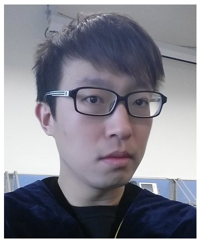
We thank all reviewers for the carefully reading of our paper and the constructive comments. This project is in part supported by a General Research Fund from RGC of Hong Kong (RGC Ref.: 11205620).

References

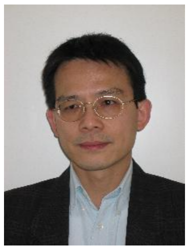
[1] D. Neill. (2012) Sketchnotes of the preface to the elements of graphic design. [Online]. Available: http://www.thegraphicrecorder.com/2012/06/13/sketchnotes-of-the-preface-to-the-elements-of-graphic-design/
[2] A. Lai. (2016) Sketchnoting - an introduction to visual notetaking. [Online]. Available: https://spark.adobe.com/page/cv2DaPKXwHRY2/
[3] A. Paivio, "Dual coding theory: Retrospect and current status," *Canadian Journal of Psychology*, vol. 45, no. 3, pp. 255–287, 1991.
[4] W. Hockley, "The picture superiority effect in associative recognition," *Memory & Cognition*, vol. 36, no. 7, pp. 1351–1359, 2008.
[5] R. Dimeo. (2015) Sketchnoting (and scientific talks). [Online]. Available: http://essentiallyelastic.tumblr.com/sketchnotes
[6] M. Baskinger, "Pencils before pixels: a primer in hand-generated sketching," *Interactions*, vol. 15, no. 2, pp. 28–36, 2008.
[7] A. Bresciani. (2013) Use pen-and-paper wireframe tools to create incredible user experiences. [Online]. Available: http://www.alessiobresciani.com/digital-marketing/use-pen-and-paper-wireframe-tools-to-create-incredible-user-experiences/
[8] O. Lindberg. (2018) Makayla lewis on the power of sketchnoting in ux design. [Online]. Available: https://theblog.adobe.com/makayla-lewis-power-sketchnoting-ux-design/
[9] E. Carlton. (2016) Sketchnote basics: Layout. [Online]. Available: https://medium.com/@emilyacarlton/sketchnote-basics-layout-dd375f59e58d
[10] C. Malamed, *Visual language for designers: principles for creating graphics that people understand*. Rockport Publishers, 2009.
[11] M. Rohde, *The sketchnote handbook: the illustrated guide to visual note taking*. Peachpit Press, 2013.
[12] S. McCloud, "Making comics: Storytelling secrets of comics, manga and graphic novels," *New York*, 2006.
[13] W. Eisner, *Comics and sequential art: Principles and practices from the legendary cartoonist*. WW Norton & Company, 2008.
[14] B. Kittle. (2015) Design principle no. 3 – what is design flow and how can i use it? [Online]. Available: https://www.cdgi.com/2015/01/design-principle-no-3-design-flow/
[15] L. Itti, C. Koch, and E. Niebur, "A model of saliency-based visual attention for rapid scene analysis," *IEEE TPAMI*, vol. 20, no. 11, 1998.
[16] D. Brockmann and T. Geisel, "The ecology of gaze shifts," *Neurocomputing*, vol. 32, pp. 643–650, 2000.
[17] G. Boccignone and M. Ferraro, "Modelling gaze shift as a constrained random walk," *Physica A: Statistical Mechanics and its Applications*, vol. 331, no. 1, 2004.
[18] W. Wang, C. Chen, Y. Wang, T. Jiang, F. Fang, and Y. Yao, "Simulating human saccadic scanpaths on natural images," in *CVPR*, 2011.
[19] H. Liu, D. Xu, Q. Huang, W. Li, M. Xu, and S. Lin, "Semantically-based human scanpath estimation with hmms," in *ICCV*, 2013, pp. 3232–3239.
[20] C. Xia, J. Han, F. Qi, and G. Shi, "Predicting human saccadic scanpaths based on iterative representation learning," *IEEE TIP*, 2019.
[21] C. Xia and R. Quan, "Predicting saccadic eye movements in free viewing of webpages," *IEEE Access*, vol. 8, 2020.
[22] A. Siris, J. Jiao, G. K. Tam, X. Xie, and R. W. Lau, "Inferring attention shift ranks of objects for image saliency," in *CVPR*, 2020.
[23] M. Jiang, X. Boix, G. Roig, J. Xu, L. Van Gool, and Q. Zhao, "Learning to predict sequences of human visual fixations," *IEEE TNNLS*, vol. 27, no. 6, pp. 1241–1252, 2016.
[24] Z. Chen and W. Sun, "Scanpath prediction for visual attention using ior-roi lstm," in *IJCAI*, 2018.
[25] N. Hurst, W. Li, and K. Marriott, "Review of automatic document formatting," in *DocEng*, 2009.
[26] C. Jacobs, W. Li, E. Schrier, D. Bargeron, and D. Salesin, "Adaptive grid-based document layout," *ACM TOG*, 2003.
[27] G. Gange, K. Marriott, P. Moulder, and P. Stuckey, "Optimal automatic table layout," in *DocEng*, 2011.
[28] P. O'Donovan, A. Agarwala, and A. Hertzmann, "Learning layouts for single-pagegraphic designs," *IEEE TVCG*, vol. 20, no. 8, 2014.
[29] ——, "Designscape: Design with interactive layout suggestions," in *ACM CHI*, 2015, pp. 1221–1224.
[30] Y. Cao, A. B. Chan, and R. W. Lau, "Automatic stylistic manga layout," *ACM TOG*, vol. 31, no. 6, 2012.
[31] X. Zheng, X. Qiao, Y. Cao, and R. W. Lau, "Content-aware generative modeling of graphic design layouts," *ACM TOG*, vol. 38, no. 4, 2019.

[32] J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu, "Layoutgan: Synthesizing graphic layouts with vector-wireframe adversarial networks," *IEEE TPAMI*, 2020.
[33] H.-Y. Lee, L. Jiang, I. Essa, P. B. Le, H. Gong, M.-H. Yang, and W. Yang, "Neural design network: Graphic layout generation with constraints," in *ECCV*, 2020.
[34] D. M. Arroyo, J. Postels, and F. Tombari, "Variational transformer networks for layout generation," in *CVPR*, 2021.
[35] K. Gupta, J. Lazarow, A. Achille, L. S. Davis, V. Mahadevan, and A. Shrivastava, "Layouttransformer: Layout generation and completion with self-attention," in *ICCV*, 2021.
[36] Y. Cao, R. W. Lau, and A. B. Chan, "Look over here: Attention-directing composition of manga elements," *ACM TOG*, vol. 33, no. 4, 2014.
[37] X. Pang, Y. Cao, R. W. Lau, and A. B. Chan, "Directing user attention via visual flow on web designs," *ACM TOG*, vol. 35, no. 6, 2016.
[38] J. Li, J. Yang, J. Zhang, C. Liu, C. Wang, and T. Xu, "Attribute-conditioned layout gan for automatic graphic design," *IEEE TVCG*, 2020.
[39] H. Gibson, J. Faith, and P. Vickers, "A survey of two-dimensional graph layout techniques for information visualisation," *Information visualization*, 2013.
[40] T. Dwyer, K. Marriott, and P. J. Stuckey, "Fast node overlap removal," in *Graph Drawing*, 2005.
[41] H. Strobelt, M. Spicker, A. Stoffel, D. Keim, and O. Deussen, "Rolled-out wordles: A heuristic method for overlap removal of 2d data representatives," *Computer Graphics Forum*, 2012.
[42] W. E. Marcílio-Jr, D. M. Eler, R. E. Garcia, and I. R. Venturini Pola, "Evaluation of approaches proposed to avoid overlap of markers in visualizations based on multidimensional projection techniques," *Information Visualization*, 2019.
[43] Y. Wu, T. Provan, F. Wei, S. Liu, and K.-L. Ma, "Semantic-preserving word clouds by seam carving," in *Computer Graphics Forum*, 2011.
[44] E. Gomez-Nieto, F. San Roman, P. Pagliosa, W. Casaca, E. Helou, M. C. F. de Oliveira, and L. G. Nonato, "Similarity preserving snippet-based visualization of web search results," *IEEE TVCG*, 2013.
[45] E. Gomez-Nieto, W. Casaca, D. Motta, I. Hartmann, G. Taubin, and L. G. Nonato, "Dealing with multiple requirements in geometric arrangements," *IEEE TVCG*, 2015.
[46] X. Liu, Y. Hu, S. North, and H.-W. Shen, "Correlatedmultiples: Spatially coherent small multiples with constrained multi-dimensional scaling," in *Computer Graphics Forum*, 2018.
[47] E. Carrizosa, V. Guerrero, and D. R. Morales, "Visualization of complex dynamic datasets by means of mathematical optimization," *Omega*, 2019.
[48] V. Yoghourdjian, T. Dwyer, G. Gange, S. Kieffer, K. Klein, and K. Marriott, "High-quality ultra-compact grid layout of grouped networks," *IEEE TVCG*, 2016.
[49] Z. C. Lipton, J. Berkowitz, and C. Elkan, "A critical review of recurrent neural networks for sequence learning," *arXiv:1506.00019*, 2015.
[50] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *ICML*, 2013, pp. 1310–1318.
[51] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.
[52] P. Werbos, "Backpropagation through time: what it does and how to do it," *Proc. IEEE*, vol. 78, no. 10, pp. 1550–1560, 1990.
[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *ICLR*, 2015.
[54] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *CVPR*, 2009, pp. 248–255.
[55] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *NIPS*, 2013, pp. 3111–3119.
[56] H. Zhang, S. Liu, C. Zhang, W. Ren, R. Wang, and X. Cao, "Sketchnet: Sketch classification with web images," in *CVPR*, 2016, pp. 1105–1113.
[57] Z. Tian, W. Huang, T. He, P. He, and Y. Qiao, "Detecting text in natural image with connectionist text proposal network," in *ECCV*, 2016.
[58] N. Metropolis, A. Rosenbluth, M. Rosenbluth, A. Teller, and E. Teller, "Equation of state calculations by fast computing machines," *The Journal of Chemical Physics*, vol. 21, no. 6, pp. 1087–1092, 1953.
[59] W. K. Hastings, "Monte carlo sampling methods using markov chains and their applications," *Biometrika*, vol. 57, no. 1, pp. 97–109, 1970.
[60] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *CVPR*, 2015, pp. 3156–3164.
[61] J. Nielsen. (2006) F-shaped pattern for reading web content. [Online]. Available: https://www.nngroup.com/articles/f-shaped-pattern-reading-web-content-discovered/
[62] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *CVPR*, 2016.
[63] S. Xie and Z. Tu, "Holistically-nested edge detection," in *ICCV*, 2015, pp. 1395–1403.
[64] A. Graves, "Generating sequences with recurrent neural networks," *arXiv:1308.0850*, 2013.
[65] V. Levenshtein, "Binary codes capable of correcting deletions, insertions, and reversals," in *Soviet Physics Doklady*, vol. 10, no. 8, 1966.
[66] J. Heminghous and A. Duchowski, "icomp: a tool for scanpath visualization and comparison," in *APGV*, 2006.

Xiaotian Qiao received the Ph.D. degree in computer science from City University of Hong Kong, and the M.Sc. and B.Eng. degrees in information and communication engineering from Zhejiang University, China. His research interests include computer vision and computer graphics.



Ying Cao received the Ph.D. degree in computer science from the City University of Hong Kong, and the M.Sc. and B.Eng. degrees in software engineering from Northeastern University, China. His research generally lies in computer graphics and computer vision. His primary research interest is data-driven graphic design.



Rynson W.H. Lau received his Ph.D. degree from University of Cambridge. He was on the faculty of Durham University, and is now with City University of Hong Kong.

Rynson serves on the Editorial Board of the International Journal of Computer Vision (IJCV). He served as the Guest Editor of a number of journal special issues, including ACM Trans. on Internet Technology, IEEE Trans. on Multimedia, IEEE Trans. on Visualization and Computer Graphics, and IEEE Computer Graphics & Applications. He also served in the committee of a number of conferences, including Program Co-chair of ACM VRST 2004, ACM MTDL 2009, IEEE U-Media 2010, and Conference Co-chair of CASA 2005, ACM VRST 2005, ACM MDI 2009, ACM VRST 2014. Rynson's research interests include computer graphics and computer vision.