# Towards Flexible Multi-modal Document Models

Naoto Inoue[1] Kotaro Kikuchi[1] Edgar Simo-Serra[2] Mayu Otani[1] Kota Yamaguchi[1]
[1]CyberAgent, Japan [2]Waseda University, Japan
{inoue_naoto, kikuchi_kotaro_xa}@cyberagent.co.jp ess@waseda.jp
{otani_mayu, yamaguchi_kota}@cyberagent.co.jp

## Abstract

*Creative workflows for generating graphical documents involve complex inter-related tasks, such as aligning elements, choosing appropriate fonts, or employing aesthetically harmonious colors. In this work, we attempt at building a holistic model that can jointly solve many different design tasks. Our model, which we denote by* FlexDM*, treats vector graphic documents as a set of multi-modal elements, and learns to predict masked fields such as element type, position, styling attributes, image, or text, using a unified architecture. Through the use of explicit multi-task learning and in-domain pre-training, our model can better capture the multi-modal relationships among the different document fields. Experimental results corroborate that our single FlexDM is able to successfully solve a multitude of different design tasks, while achieving performance that is competitive with task-specific and costly baselines.* [1]

## 1. Introduction

Vector graphic documents are composed of diverse multi-modal elements such as text or images and serve as the dominant medium for visual communication today. The graphical documents are created through many different design tasks, *e.g.*, filling in a background image, changing font and color, adding a decoration, or aligning texts. While skilled designers perform tasks based on their design knowledge and expertise, novice designers often struggle to make decisions to create an effective visual presentation. To assist such novice designers, interactive frameworks equipped based on models that learn design knowledge from completed designs have been proposed [12, 38]. Our present work proposes models that can be used in such systems, with a particular focus on developing holistic models that can flexibly switch between design tasks.

Design tasks are characterized by 1) the variety of possible actions and 2) the complex interaction between multi-modal elements. As discussed above, a designer can make almost any edit to the appearance of a vector graphic document, ranging from basic layout to nuanced font styling. While there have been several studies in solving specific tasks of a single modality, such as layout generation [3, 13, 23, 26, 30], font recommendation [56], or colorization [22, 40, 54], in realistic design applications, we believe it is essential to build a *flexible* model that can consider multiple design tasks in a principled manner to make automated decisions on creative workflow.

In this work, we refer to a certain attribute of an element as a *field* and formulate the various design tasks as a unified *masked field prediction*, which is inspired by the recent masked autoencoders [9, 15] and multi-task models [19, 36]. The key idea is to utilize masking patterns to switch among different design tasks within a single model; *e.g.*, element filling can be formulated as predicting all the fields of the newly added element. Our flexible document model, denoted by *FlexDM*, consists of an encoder-decoder architecture with a multi-modal head dedicated to handling different fields within a visual element. After pre-training with random masking strategy, we train FlexDM by explicit multi-task learning where we randomly sample tasks in the form of masking patterns corresponding to the target design task. We illustrate in Figs. 1 and 2 an overview of FlexDM, with emphasis on the correspondence between design tasks and masking patterns.

Through our carefully designed experiments, we show that our proposed FlexDM performs favorably against baselines in five design tasks using the Rico [7] and Crello [52] datasets. We also study how different modeling approaches affect the final task performance in the ablation study. Finally, we apply our framework to several previously studied design tasks with minimal modifications and show that the performance matches or even surpasses the current task-specific approaches.

Our contributions can be summarized in the following.

- We formulate multiple design tasks for vector graphic documents by masked multi-modal field prediction in a

[1]Please find the code and models at:
https://cyberagentailab.github.io/flex-dm.

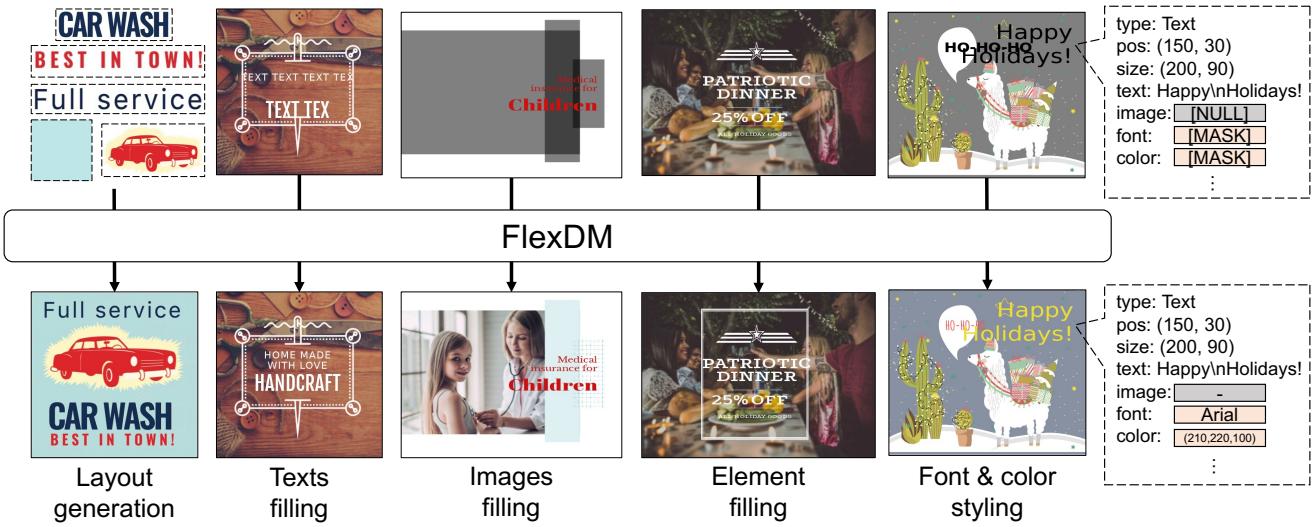


Figure 1. Examples of the design tasks that can be solved by our proposed FlexDM model, which is designed to process a vector graphic document consisting of an arbitrary number of elements (*e.g.*, text). Each element is composed of multi-modal fields indicating its attribute properties (*e.g.*, text content, position, font color, etc.).

set of visual elements.

- We build a flexible model to solve various design tasks jointly in a single Transformer-based model via multi-task learning.
- We empirically demonstrate that our model constitutes a strong baseline for various design tasks.

## 2. Related Work

### 2.1. Vector Graphic Generation

There has been a growing interest in vector graphics to realize resolution/artifact-free rendering that is easy to interpret and edit, such as Scalable Vector Graphics (SVG) [8]. Modeling documents in a vector format is much more complex than the stroke or path level vector graphics [5, 14, 35] since each element contains multi-modal features such as text and image. CanvasVAE [52] tackles the document-level unconditional generation of vector graphics, but is not a multi-task model and cannot solve specific design tasks such as element filling. Doc2PPT [11] generates slides given a longer and more detailed multi-modal document, but it is a summarization task and cannot infer what is missing in the incomplete document.

Obtaining transferable representation for downstream tasks learned from multi-modal large-scale data is getting popular. Domains closest to our setting are document understanding [32, 49–51] and UI understanding [4, 16], where the data consist of elements with multi-modal attributes. Despite the generalizable representation, all the methods fine-tune different parameters for each downstream task (mainly in classification). In contrast, we aim to solve many essential tasks for design creation in a single model.

### 2.2. Multi-task Learning

Multi-task learning (MTL) [2, 6, 10] aims at solving different tasks at the same time while sharing information and computation among them, which is crucial for deployment. MTL methods achieve a good tradeoff between performance and computational cost by (i) multiple lightweight heads at the top of shared backbone [25, 55] and (ii) efficient use of task-specific parameters [33, 43, 44]. On the contrary, our model obtains the task information from the masking patterns of the input fields and we empirically show that extra task-specific parameters are not necessary.

Training a single model that generalizes to many different tasks has been a long-standing goal. 12-in-1 [37] and UniT [17] handle multiple tasks in vision and language domain with small task-specific parameters. In a more unified manner, Perceiver [20] and Perceiver IO [19] treat different modalities as the same data format, OFA [48] and Unified-IO [36] consider similar attempts in the sequence-to-sequence framework, resulting in a single model or architecture with no task-specific tuning. We are highly inspired by these works and explore how to unify the design tasks in vector graphic document domain.

### 2.3. Computational Assistance for Graphic Design

There is a long history of automatic graphic design [1, 34, 53]. Recent approaches rely on the learning-based formulation, where the primal focus is in predicting layouts given label sets [21, 30] or in an unconditional manner [3, 13], and avoids the manual design of the energy functions seen in

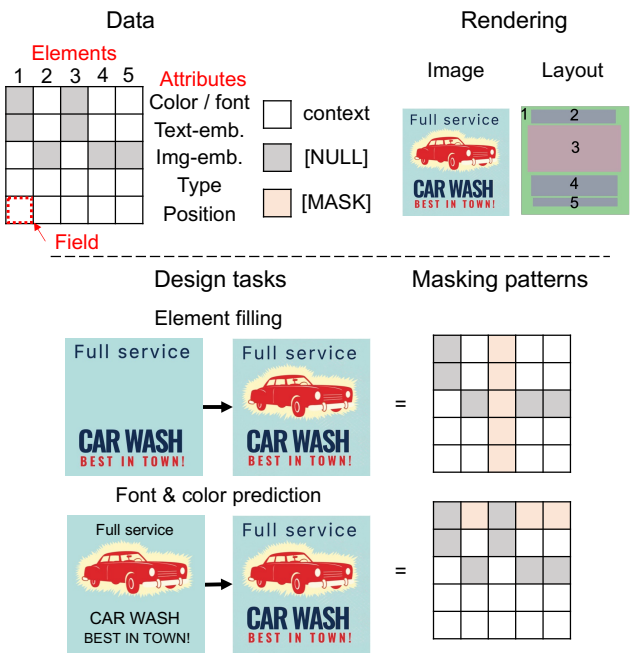


Figure 2. **Top**: example of a vector graphic document consisting of five elements. The array is used to illustrate the data structure of the document. Each column corresponds to a single visual element. Each row corresponds to an attribute or a group of attributes consisting the element. **Bottom**: Correspondence between design tasks and masking patterns for our masked field prediction.

the earlier work [39]. Some works additionally take positional/relational constraints [24, 27, 31] or textual descriptions [57] for finer design control, but are not applicable in a more complex scenario. In contrast, our multi-task approach solves many conditional tasks thanks to the flexible multi-modal fields in both inputs and targets.

Considering multi-modal features is essential to go beyond layout generation for intelligent graphic design assistance. Wang *et al*. [47] retrieve images from layout information and keywords for each element to obtain visually pleasing visual design by reinforcement learning. Zhao *et al*. [56] predict font properties of a text on a webpage over a background image considering metadata. Li *et al*. [29] predict position and size for a single text box over a background image considering saliency. We demonstrate that we can apply our flexible model to solve these tasks with almost no modification, and our model performs favorably against the task-specific well-tuned approaches.

# 3. Approach

We first describe the formal definition of the vector graphic document and notations in Sec. 3.1. We then introduce the idea of masked field prediction and a model for it in Sec. 3.2 and Sec. 3.3. Finally, we describe how we train

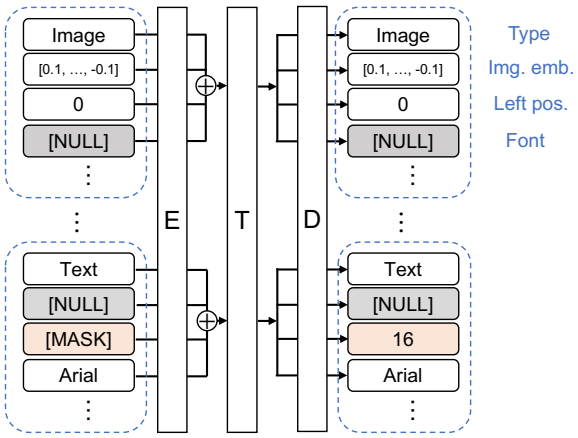


Figure 3. The architecture of FlexDM. E, T, and D are short for Encoder, Transformer blocks, and Decoder, respectively.

FlexDM in Sec. 3.4.

## 3.1. Preliminary

**Document Structure**: In this work, a vector graphic document $X$ consists of a set of elements $X = (X_1, X_2, \ldots, X_S)$, where $S$ is the number of elements in $X$. Each element $X_i$ consists of a set of multi-modal fields and denoted by $X_i = \{x_i^k \mid k \in \mathcal{E}\}$, where $\mathcal{E}$ indicates the indices for all the attributes. Each field $x_i^k$ can be either a categorical or numerical variable such as element type, position, text content, or image embedding. For ease of explanation, we illustrate $X$ by a 2D-array as shown in the top of Fig. 2. Note that the order in the array does not matter because $X$ is a set of sets. Since processing high-dimensional data such as raw images and texts during optimization is computationally intensive, we extract a low-dimensional numerical vector from such data for $x_i^k$ using pre-trained models.

**Special Tokens**: In a similar spirit to the masked language model [9], we use a few special tokens to represent $x_i^k$.
`[NULL]`: appears when $x_i^k$ is inevitably missing (*e.g.*, font type for an image element), or padding variable-length sequence within a mini-batch on training.
`[MASK]`: appears when $x_i^k$ is masked for prediction.

## 3.2. Masked Field Prediction

Given an incomplete document $X$ containing `[MASK]` as context, our goal is to predict values for all the fields filled with `[MASK]` and generate a complete document $\hat{X}$. We refer to this problem by *masked field prediction*, where a model has to predict the masked field considering the different multi-modal relations between the fields. While the masking approach is similar to the masked language model [9], there is a key distinction in that we process an order-less set of multi-modal items (*i.e.*, document $X$). For

this reason, we design our architecture to 1) efficiently capture inter-field relationships of vector graphic attributes, and 2) ensure that the model works without positional encodings commonly used to model an ordered sequence.

### 3.3. FlexDM Architecture

As shown in Fig. 3, our architecture consists of three modules; encoder, Transformer blocks, and decoder. Given a document, we first project a set of partially masked fields (*e.g.*, position or font) into embeddings using the encoder, and then feed the output to the intermediate Transformer blocks. The final decoder takes the transformed embeddings and projects them back to the original fields space. The Transformer blocks only process $S$ embeddings, which is efficient compared to architecture processing $S \times N$ fields with off-the-shelf Transformer [46] directly, when there are $N$ attributes. In the following, let us denote all model parameters by $\theta$.

**Encoder**: The encoder takes a document input $X$ and embeds it into $h^{\text{enc}} = \{h_1^{\text{enc}}, h_2^{\text{enc}}, \ldots, h_S^{\text{enc}}\}$ with element-wise operations. The encoder first maps each field $x_i^k$ to a fixed dimensional vector with $f^{\text{enc},k}$, and sums up all the fields in the element to produce a latent vector for the $i$-th element with:

$$h_i^{\text{enc}} = \sum_{k \in \mathcal{E}} f^{\text{enc},k}(x_i^k; \theta), \tag{1}$$

where $f^{\text{enc},k}$ is an embedding function that retrieves learnable dense embeddings for each category id if $x_i^k$ is a categorical variable, or a simple linear projection layer if $x_i^k$ is a numerical variable. We treat the special tokens (*i.e.*, `[NULL]` and `[MASK]`) in the same manner to the categorical variable.

**Transformer Blocks**: Transformer blocks take $h^{\text{enc}}$ as input and transform it to $h^{\text{dec}} = \{h_1^{\text{dec}}, h_2^{\text{dec}}, \ldots, h_S^{\text{dec}}\}$. We stack these intermediate blocks to process complex inter-element relations. Our model can stack any off-the-shelf Transformer layer to build up the blocks $f^{\text{trans}}$:

$$h^{\text{dec}} = f^{\text{trans}}(h^{\text{enc}}; \theta) \tag{2}$$

**Decoder**: The final decoder takes $h_{\text{dec}}$ and decodes them back into a document $\hat{X} = (\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_S)$, where $\hat{X}_i = \{\hat{x}_i^k \mid k \in \mathcal{E}\}$. We compute each $\hat{x}_i^k$ by a linear layer $f^{\text{dec},k}$ for both categorical and numerical variables:

$$\hat{x}_i^k = f^{\text{dec},k}(h_i^{\text{dec}}; \theta). \tag{3}$$

**Loss**: We train our model using reconstruction losses. Let us denote by $X^*$ the ground truth of the incomplete document $X$, and also denote by $M$ a set of tuples indicating the indices for `[MASK]` tokens in $X$. We define the loss function by:

$$\mathcal{L} = \sum_{(i,k) \in M} l^k(\hat{x}_i^k, x^{*k}_i), \tag{4}$$

where $l^k$ is the loss function for the $k$-th attribute. For each $l^k$, we use softmax cross-entropy loss for categorical variables and mean squared error for numerical variables.

### 3.4. FlexDM Training

Masked field prediction allows us to represent diverse design tasks having various input/output formats just by altering the masking pattern. The pattern can be both deterministic or stochastic. The bottom of Fig. 2 illustrates example tasks and the corresponding masking patterns. Although we can formulate arbitrary tasks with masked field prediction, we consider several subsets of representative design tasks for our evaluation and analyses in Sec. 4.

We describe typical masking patterns in the following. Note that fields already filled with `[NULL]` will never be replaced in priority to the masking operations. *Element masking* randomly selects elements and masks all the fields within the element; *i.e.*, we can formulate the element filling task by single element masking. *Attribute masking* randomly selects attributes and mask the fields across all the elements; *e.g.*, masking position and size of all the elements becomes layout prediction, and masking fonts becomes font prediction. *Random masking* strategy masks fields by some probability without considering the data structure, which is similar to BERT [9].

**Pre-training**: To learn the initial model, we employ a pre-training by ordinary random masking similar to the prevailing pre-training strategy of BERT [9]. One distinction is that our pre-training happens in the same, in-domain dataset, unlike the common setup where a model is pre-trained on a larger dataset in a different domain and then fine-tuned on a target task in a target dataset. We show in Sec. 4 that this in-domain pre-training moderately improves the final task performance.

**Explicit Multi-task Learning**: The random masking pre-training above is a solid baseline for any task. Radford *et al.* [42] hypothesize that this implicit multi-task training leads to the astonishingly strong zero-shot performance of large language models. However, the random masking strategy actually produces any task with an extraordinarily low probability as the number of attributes and elements increases. Instead, we employ the *explicit* masking strategy to maximize the performance on all the target tasks. During training we randomly sample a task from the target tasks, sample a complete document $X^*$, and make the triplet ($X$, $X^*$, $M$) by using the masking pattern associated with the task. We repeat this procedure to build each mini-batch when training FlexDM.

# 4. Experiments

## 4.1. Dataset

We mainly use two datasets containing vector graphic documents, Rico [7] and Crello [52], to evaluate FlexDM. We basically follow the setting used in [52]. Due to memory limitations, we discard documents having more than fifty elements. Position, size, and color information are discretized in order to enhance the implicit alignment of multiple elements. We describe the overview of each dataset.

**Rico [7]:** The dataset collects UI designs from mobile apps. We follow previous works [27, 30] and exclude elements whose labels are not in the most frequent 13 labels. We divide the dataset into 45,012 / 5,565 / 5,674 examples for train, validation, and test splits.

**Crello [52]:** The dataset provides design templates from an online design service. Crello contains various design formats such as social media posts, banner ads, blog headers, or printed posters. We divide the dataset into 18,738 / 2,313 / 2,271 examples for train, validation, and test splits. Please refer to the original paper [52] for the definition of each attribute. For image and text features, we extract 768-dimensional features using CLIP [41]. We also additionally extract categorical font information (called `Font`). We group the attributes into some groups based on their property. **TYPE** denotes `Type` attribute. **POS** denotes `Position` and `Size` attributes. **IMG** denotes `Image` attribute. **TXT** denotes `Text` attribute. **ATTR** denotes attributes not listed above, and these attributes have a large impact on fine-grained appearance.

## 4.2. Tasks

We carefully select tasks to evaluate how our model performs in various design tasks. We select evaluation tasks such that (i) they are practical, (ii) they have various combinations of input/output modalities, and (iii) the masking ratio is modest. We impose the masking ratio requirement because the extreme masking ratio makes the task too difficult or trivial to solve and makes the baseline comparison impossible.

**Element Filling (ELEM):** This task is to predict a new element that can enhance the document. We mask all the attributes of a single element in a complete document during training and evaluation.

**Attribute Prediction:** This task is to predict missing attributes at once in the document, which is very challenging. We apply attribute masking on a complete document to make the masked inputs during training and evaluation. We select an attribute group discussed in Sec. 4.1 and apply the attribute masking for all the attributes in the group. We consider each group-level prediction task as an individual task. Note that we do not consider TYPE prediction since it is too trivial and unrealistic. Therefore, we have two (POS and ATTR) and four (POS, ATTR, IMG, and TXT) attribute prediction tasks for Rico and Crello, respectively.

## 4.3. Evaluation Metrics

For each task, we quantitatively evaluate the reconstruction performance. The score $\mathcal{S}$ for each document is computed by:

$$\mathcal{S} = \frac{1}{|M|} \sum_{(i,k)\in M} s^k(\hat{x}_i^k, x^{*k}_i), \tag{5}$$

where $s^k \in [0, 1]$ is a scoring function for $k$-th attribute. If the attribute is categorical, $s^k$ is an indicator function that takes 1 if $\hat{x}_i^k$ and $x^{*k}_i$ are identical, otherwise 0. For image and text features that are the only numerical attributes in our experiments, we use cosine similarity in $[0, 1]$ scale.

## 4.4. Training Details

We use 256-dimensional latent representations within the encoder, Transformer blocks, and decoder. For the Transformer blocks we use the one from DeepSVG [5]. We apply a dropout probability of 0.1 to all the dropout layers. We train the model with a batch size of 256 sequences for 500 epochs in all the experiments. We use Adam with learning rate of 1e-4, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and L2 weight decay of 1e-2. In experiments on Rico, we make FlexDM take positional embedding as the additional input, since otherwise the model is unable to distinguish elements having a completely similar set of attributes, which often occurs in POS prediction.

## 4.5. Quantitative Evaluation

We test three models based on our proposed framework to clarify the contribution of both explicit multi-task learning and pre-training.

**Ours-IMP**: As in the standard masked language modeling such as BERT [9], we randomly mask 15% of the fields during training. Since this randomized training is called implicit multi-task learning [42], we call it Ours-IMP.

**Ours-EXP**: All the tasks are explicitly and jointly trained in a single model by sampling the masking patterns corresponding to each task. For simplicity, $T$ tasks introduced in Sec. 4.2 are uniformly sampled in a mini-batch.

**Ours-EXP-FT**: This is our entire model. We use weights of the model trained on IMP, and fine-tune the model. The rest of the training is the same as Ours-EXP.

We compare these models with the following baselines, some of which are adapted from existing task-specific models to our multi-task, multi-attribute, and arbitrary masking setting with minimal modification.

**Expert**: We train the network individually for each task. Note that the number of the parameters used in this variant is $T$ times larger than our models.

| Dataset | Rico [7] | | | | Crello [52] | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Model | #par. | ELEM | POS | ATTR | #par. | ELEM | POS | ATTR | IMG | TXT |
| Most-frequent | 0.0x | 0.461 | 0.213 | 0.830 | 0.0x | 0.402 | 0.134 | 0.382 | 0.922 | 0.932 |
| BERT [9] | 1.0x | 0.517 | 0.238 | 0.847 | 1.0x | **0.524** | 0.155 | 0.632 | 0.935 | 0.949 |
| BART [28] | 1.2x | 0.515 | 0.220 | 0.714 | 1.2x | 0.469 | 0.156 | 0.615 | 0.932 | 0.945 |
| CVAE [21, 27] | 1.1x | 0.511 | 0.214 | 0.917 | 1.0x | 0.499 | 0.197 | 0.587 | 0.942 | 0.947 |
| CanvasVAE [52] | 1.2x | 0.437 | 0.192 | 0.790 | 1.2x | 0.475 | 0.138 | 0.586 | 0.912 | 0.946 |
| Ours-IMP | 1.0x | 0.505 | **0.259** | 0.923 | 1.0x | 0.483 | 0.197 | 0.607 | 0.945 | 0.949 |
| Ours-EXP | 1.0x | 0.540 | 0.226 | 0.937 | 1.0x | 0.499 | 0.218 | 0.679 | 0.948 | 0.952 |
| Ours-EXP-FT | 1.0x | **0.552** | 0.215 | **0.945** | 1.0x | 0.508 | **0.227** | **0.688** | **0.950** | **0.954** |
| Expert | 3.0x | 0.575 | 0.228 | 0.952 | 5.0x | 0.534 | 0.255 | 0.703 | 0.948 | 0.955 |

Table 1. Quantitative evaluation in two datasets. A higher score indicates the better performance. Top two results are highlighted in **bold** and underline, respectively. LGAN++ is short for LayoutGAN++.

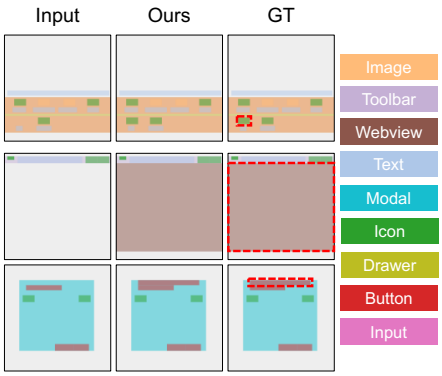


Figure 4. Results in element filling using Rico dataset. The red dotted box indicates the target element to be predicted.

**Most-frequent**: We calculate the statistics of the training dataset. For a categorical attribute, we count the occurrences and pick the most frequent category. For a numerical attribute, we compute the average because the numerical attributes that we use are only image and text features.
**BERT [9]**: We convert all the fields into a single sequence and process them with Transformer blocks. This evaluates the effect of element-wise embedding discussed in Sec. 3.3.
**BART [28]**: BART employs an encoder-decoder-based sequence-to-sequence model for pre-training text generation models by masked language modeling. We replace our Transformer blocks with the blocks from BART.
**CVAE [21,27]**: Recent methods for conditional layout generation such as LayoutVAE [21] and NDN [27] employ Conditional VAE [45] in an auto-regressive manner. We replace our Transformer block and decoder parts with CVAE variants used in [21,27] and predict the fields in an element-by-element manner. Note that the full version of NDN contains relation prediction and layout refinement modules in addition to CVAE modules. We omit the full NDN pipeline evaluation due to their specific approach.
**CanvasVAE [52]**: CanvasVAE is for an unconditional generation. Although direct comparison is impossible, we adapt CanvasVAE to our setting, similar to other baselines.

Table 1 summarizes the performance of all the models. Our full model (Ours-EXP-FT) is almost comparable to Expert model while being much more efficient in the number of parameters. Ours-IMP exhibits moderate performance, resulting in a better initial weight for fine-tuning in Ours-EXP-FT. We can see that most of the compared baselines perform clearly worse compared to Ours-EXP. The result suggests that applying existing Transformer models for sequence modeling or conditional layout generation models is not enough in our challenging setting. POS-prediction in Rico is the exceptional case, where most of the methods fail because of the larger number of elements compared to the benchmark setup in the literature [24] (nine at maximum).

### 4.6. Qualitative Evaluation

We show the prediction quality of our full FlexDM (Ours-EXP-FT) for Rico dataset in the element-filling task in Fig. 4. For Rico, we show a color map indicating the position and type information. In Fig. 5, we show the prediction of our full FlexDM (Ours-EXP-FT) on all the target design tasks. For visualizing predicted low-dimensional image and text features, we conduct a nearest neighbor search to retrieve actual images and texts using the assets in the test subset, following CanvasVAE [52].

### 4.7. Ablation Study

In this section, we perform several ablation experiments in the Crello dataset, as shown in Tab. 2. We demonstrate that our design choices non-trivially affect the final performance of FlexDM.
**Task-specific Embedding**: The previous work [17] on unifying multiple tasks in a single Transformer uses small task-specific learnable query embedding to feed information of the current task explicitly. We append the query as $h_0^{\text{enc}}$ at the beginning of $h^{\text{enc}} = \{h_1^{\text{enc}}, h_2^{\text{enc}}, \dots, h_S^{\text{enc}}\}$ and train the model. The result suggests the benefit of the embedding is marginal. We conjecture that the model implicitly captures the task information from the masked inputs in our setting.
**Attention**: Here we study the importance of self-attention to model the inter-element relationship by training a model without self-attention. We increase the number of layers to eight to roughly match the total number of parameters with Ours-EXP. As expected, the result clearly suggests the importance of modeling the inter-element relationship.
**Additional Loss**: Our objective function in Eq. (4) only considers reconstruction. One may argue that incorporating

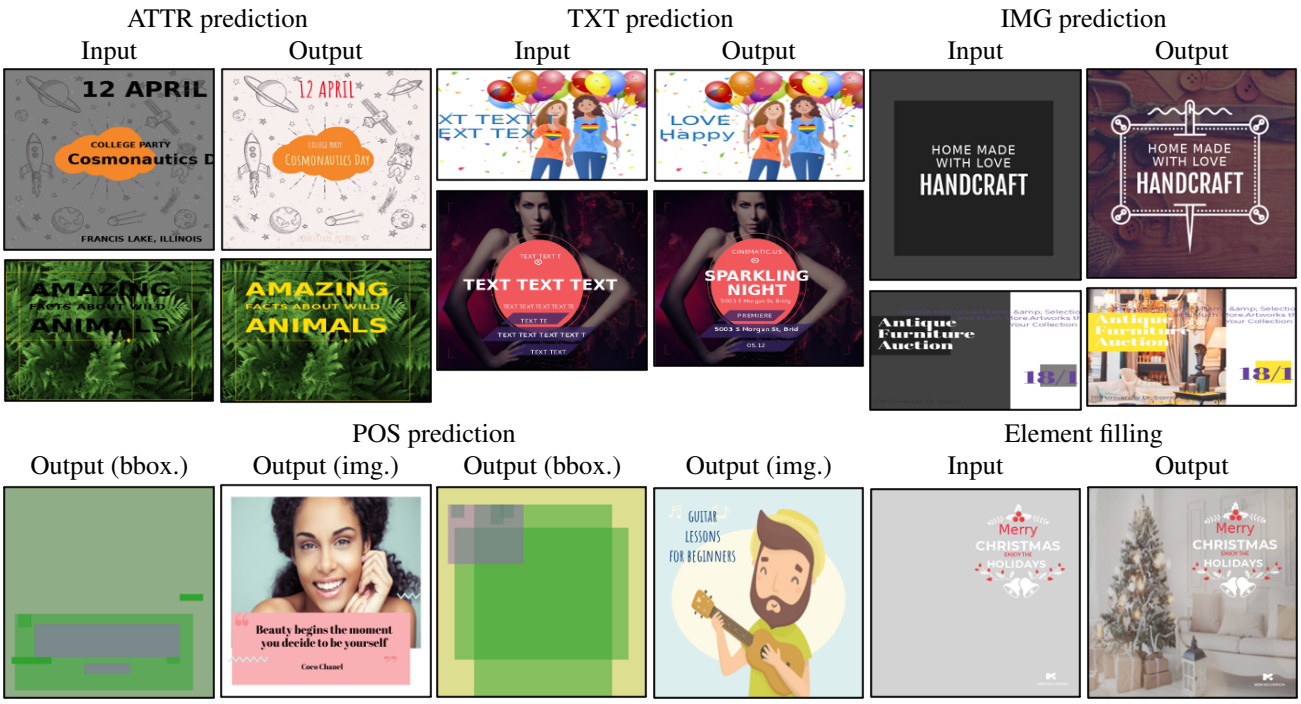


Figure 5. Prediction of FlexDM (Ours-EXP-FT trained on Crello). FlexDM jointly handles a large variety of design tasks with a single Transformer-based model. In the input of ATTR/TXT/IMG prediction, the target fields assigned `[MASK]` are visualized using fixed default values (*i.e.*, black for text color, gray for image and solid fill, 'TEXT' for text). In POS prediction, we additionally show the layout of the elements. The correspondence between the color and type of the element is as follows: green = *vector shape*, magenta = *image*, purple = *text*, yellow = *solid fill*. Best viewed with zoom and color.

Table 2. Ablation study results in Crello dataset. Top two results are highlighted in **bold** and underline, respectively.

| | Model | ELEM | POS | ATTR | IMG | TXT |
|---|---|---|---|---|---|---|
| | Ours-EXP | **0.499** | 0.218 | **0.679** | 0.948 | 0.952 |
| (i) | w/ task-ID | 0.496 | **0.222** | 0.674 | **0.949** | **0.953** |
| (ii) | w/o attention | 0.446 | 0.208 | 0.605 | 0.939 | 0.947 |
| (iii) | w/ adv. | **0.499** | 0.215 | 0.677 | 0.948 | 0.952 |

adversarial losses such as those used in LayoutGAN++ [24] could improve the model. While we tried our best in implementing and tuning the additional adversarial loss, we did not find a clear benefit in adversarial training.

## 4.8. Comparison with Task-specific Baselines

In this section, we show that our data-driven masked field prediction model can match or even surpasses task-specific approaches. We perform experiments in two tasks: 1) single text styling and 2) single text box placement. Since each task uses partially overlapping set of attributes, we train our model for each single task for fair comparison. Note that we are unable to compare to contextual images filling [47] discussed in Sec. 2.3 due to their task setup where they retrieve an image only from pre-defined sets used during training.

### 4.8.1 Single Text Styling

Zhao *et al*. [56] propose an MLP-based model to predict desirable font properties for a *single* text box (*i.e.*, font emb., color, and size), given context in web designs. We consider that each design is a document with one text and two image elements, and regard all the context information as attributes in the elements so that we can just apply FlexDM. We implement Zhao *et al*. [56] with the following minor difference, since the code is not publicly available. We quantize the color and size into 16 bins and 64 bins, respectively. We did not apply data augmentation using the external dataset, since the dataset used for the augmentation is not available. We show the results in Tab. 3. The metrics are accuracy for font color and size, and cosine similarity for font type, which is represented by a low-dimensional embedding. We can clearly see that our model is comparable to the task-specific model.

### 4.8.2 Single Text Box Placement

Li *et al*. [29] propose to predict the size and position of a single text box given a natural image and aspect ratio of

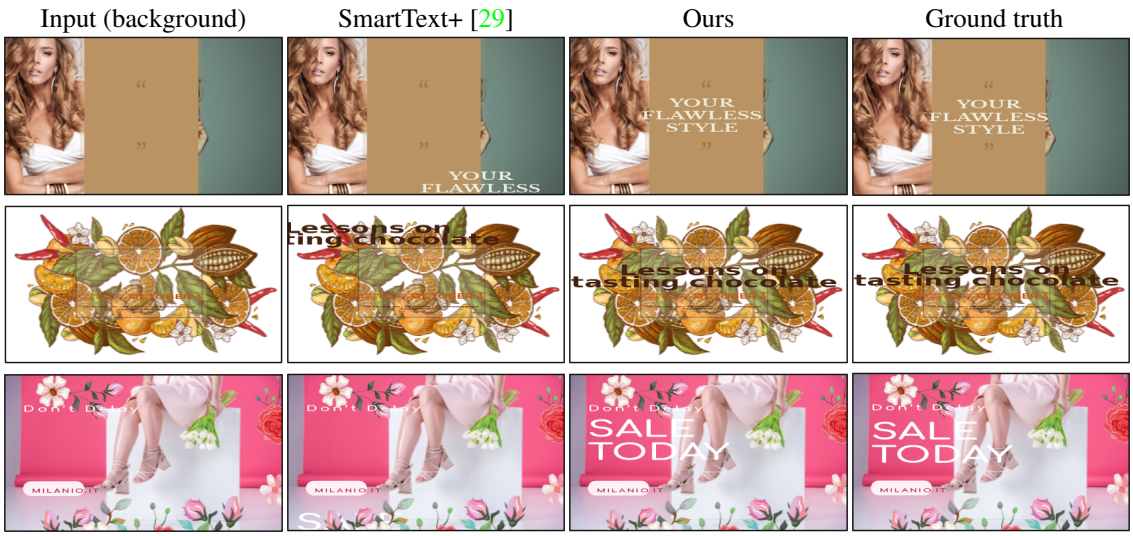


Figure 6. Qualitative comparison of single text box placement with SmartText+ [29]. Best viewed with zoom and color.

Table 3. Comparison of models for font properties prediction in CTXFont dataset [56]. The average and standard deviation of three runs are reported. The values are multiplied by 100x for visibility.

| Model | Color | Size | Emb. | Avg. |
|---|---|---|---|---|
| Zhao *et al*. [56] | 45.8$_{\pm 2.9}$ | 19.9$_{\pm 3.1}$ | **79.2**$_{\pm 0.5}$ | 48.2$_{\pm 1.2}$ |
| Ours | **54.2**$_{\pm 0.7}$ | **24.2**$_{\pm 0.1}$ | 77.7$_{\pm 1.3}$ | **52.0**$_{\pm 0.5}$ |

Table 4. Quantitative evaluation of models for single text box placement in Crello dataset. The samples are divided into two groups: no other text box available (Single) and some text boxes available as the context (Multiple).

| | Single | | Multiple | |
|---|---|---|---|---|
| | IoU ↑ | BDE ↓ | IoU ↑ | BDE ↓ |
| SmartText+ [29] | 0.047 | 0.262 | 0.023 | 0.300 |
| Ours | **0.357** | **0.098** | **0.110** | **0.141** |
| w/o image | 0.355 | 0.100 | 0.103 | 0.156 |
| w/o text | 0.350 | 0.106 | 0.086 | 0.178 |

the text box. We perform comparison in Crello dataset, since the dataset used for their model training and evaluation is not publicly available. We evaluate the performance in terms of the intersection over union (IoU) and boundary displacement error (BDE) [29]. As shown in the upper half of Tab. 4, our model clearly outperforms Li *et al*. [29]'s model. To measure the contribution of multi-modal features to the prediction, we exclude each of them and train the model. The results in the lower half of Tab. 4 suggest that those features contribute to the better performance. Some results are shown in Fig. 6.

## 5. Limitation and Discussion

As image and text generation quality is astonishingly improving, one may want to generate images and texts directly. However, retrieval-based generation is still a practical option. For instance, due to clients' requests, designers often need to use images from private collections or public photo stock services such as Adobe Stock or Shutterstock. Moreover, some people avoid using generated images or text as there are controversies about the legal and ethical issues of AI-generated images.

Our model does not support design tasks that cannot be framed as masked field prediction. We do not consider unconditional generation; *i.e*., generating a complete document without input. Extending FlexDM to an unconditional scenario requires us to apply a generative formulation instead of BERT-style masked modeling, and we leave such formulation as future work. However, we believe that our model nicely fits in a common application scenario where there exist initial design materials to start with.

The model's performance decreases when the input document has more elements. Whether bigger models or datasets alleviate the issue is worth investigating. Developing other evaluation metrics would be helpful for further analysis since current metrics simply evaluate reconstruction performance. In conditional generation, the input context may correspond to multiple possible outputs, especially when the input context is sparse (*e.g*., label sets). Modeling such variability as in layout generation models [18, 21, 24] would be an exciting direction.

## References


[1] Maneesh Agrawala, Wilmot Li, and Floraine Berthouzoz. Design principles for visual communication. *Communications of the ACM*, 54(4), 2011. 2

[2] Andreas Argyriou, Theodoros Evgeniou, and Massimiliano Pontil. Multi-task feature learning. In *NeurIPS*, 2006. 2

[3] Diego Martin Arroyo, Janis Postels, and Federico Tombari. Variational transformer networks for layout generation. In *CVPR*, 2021. 1, 2

[4] Chongyang Bai, Xiaoxue Zang, Ying Xu, Srinivas Sunkara, Abhinav Rastogi, Jindong Chen, and Blaise Agüera y Arcas. UIBert: Learning generic multimodal representations for ui understanding. In *IJCAI*, 2021. 2

[5] Alexandre Carlier, Martin Danelljan, Alexandre Alahi, and Radu Timofte. DeepSVG: A hierarchical generative network for vector graphics animation. In *NeurIPS*, 2020. 2, 5

[6] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997. 2

[7] Biplab Deka, Zifeng Huang, Chad Franzen, Joshua Hibschman, Daniel Afergan, Yang Li, Jeffrey Nichols, and Ranjitha Kumar. Rico: A mobile app dataset for building data-driven design applications. In *UIST*, 2017. 1, 5, 6

[8] Patrick Dengler, Erik Dahlström, Doug Schepers, Jon Ferraiolo, Anthony Grasso, Chris Lilley, Dean Jackson, Jonathan Watt, Cameron McCormack, and Jun Fujisawa. Scalable vector graphics (SVG) 1.1 (second edition). W3C recommendation, W3C, Aug. 2011. https://www.w3.org/TR/2011/REC-SVG11-20110816/. 2

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 1, 3, 4, 5, 6

[10] Theodoros Evgeniou and Massimiliano Pontil. Regularized multi–task learning. In *KDD*, 2004. 2

[11] Tsu-Jui Fu, William Yang Wang, Daniel McDuff, and Yale Song. Doc2ppt: Automatic presentation slides generation from scientific documents. In *AAAI*, 2022. 2

[12] Shunan Guo, Zhuochen Jin, Fuling Sun, Jingwen Li, Zhaorui Li, Yang Shi, and Nan Cao. Vinci: an intelligent graphic design system for generating advertising posters. In *CHI*, 2021. 1

[13] Kamal Gupta, Alessandro Achille, Justin Lazarow, Larry Davis, Vijay Mahadevan, and Abhinav Shrivastava. Layout-Transformer: Layout generation and completion with self-attention. In *ICCV*, 2021. 1, 2

[14] David Ha and Douglas Eck. A neural representation of sketch drawings. In *ICLR*, 2018. 2

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1

[16] Zecheng He, Srinivas Sunkara, Xiaoxue Zang, Ying Xu, Lijuan Liu, Nevan Wichers, Gabriel Schubiner, Ruby Lee, Jindong Chen, and Blaise Aguera y Arcas. ActionBert: Leveraging user actions for semantic understanding of user interfaces. In *AAAI*, 2021. 2

[17] Ronghang Hu and Amanpreet Singh. UniT: Multimodal multitask learning with a unified transformer. In *ICCV*, 2021. 2, 6

[18] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. LayoutDM: Discrete Diffusion Model for Controllable Layout Generation. In *CVPR*, 2023. 8

[19] Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. Perceiver IO: A general architecture for structured inputs & outputs. In *ICLR*, 2022. 1, 2

[20] Andrew Jaegle, Felix Gimeno, Andy Brock, Oriol Vinyals, Andrew Zisserman, and Joao Carreira. Perceiver: General perception with iterative attention. In *ICML*, 2021. 2

[21] Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. LayoutVAE: Stochastic scene layout generation from a label set. In *CVPR*, 2019. 2, 6, 8

[22] Kotaro Kikuchi, Naoto Inoue, Mayu Otani, Edgar Simo-Serra, and Kota Yamaguchi. Generative colorization of structured mobile web pages. In *WACV*, 2023. 1

[23] Kotaro Kikuchi, Mayu Otani, Kota Yamaguchi, and Edgar Simo-Serra. Modeling visual containment for web page layout optimization. *Computer Graphics Forum*, 40(7), 2021. 1

[24] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. Constrained graphic layout generation via latent optimization. In *ACM MM*, 2021. 3, 6, 7, 8

[25] Iasonas Kokkinos. Ubernet: Training a universal convolutional neural network for low-, mid-, and high-level vision using diverse datasets and limited memory. In *CVPR*, 2017. 2

[26] Xiang Kong, Lu Jiang, Huiwen Chang, Han Zhang, Yuan Hao, Haifeng Gong, and Irfan Essa. BLT: bidirectional layout transformer for controllable layout generation. In *ECCV*, 2022. 1

[27] Hsin-Ying Lee, Weilong Yang, Lu Jiang, Madison Le, Irfan Essa, Haifeng Gong, and Ming-Hsuan Yang. Neural design network: Graphic layout generation with constraints. In *ECCV*, 2020. 3, 5, 6

[28] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, 2020. 6

[29] Chenhui Li, Peiying Zhang, and Changbo Wang. Harmonious textual layout generation over natural images via deep aesthetics learning. *IEEE TMM*, 2021. 3, 7, 8

[30] Jianan Li, Jimei Yang, Aaron Hertzmann, Jianming Zhang, and Tingfa Xu. LayoutGAN: Generating graphic layouts with wireframe discriminators. In *ICLR*, 2019. 1, 2, 5

[31] Jianan Li, Jimei Yang, Jianming Zhang, Chang Liu, Christina Wang, and Tingfa Xu. Attribute-conditioned layout gan for automatic graphic design. *IEEE TVCG*, 2020. 3

[32] Peizhao Li, Jiuxiang Gu, Jason Kuen, Vlad I Morariu, Handong Zhao, Rajiv Jain, Varun Manjunatha, and Hongfu Liu.

SelfDoc: Self-supervised document representation learning. In *CVPR*, 2021. 2

[33] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *CVPR*, 2019. 2

[34] Simon Lok and Steven Feiner. A survey of automated layout techniques for information presentations. In *SmartGraphics*, 2001. 2

[35] Raphael Gontijo Lopes, David Ha, Douglas Eck, and Jonathon Shlens. A learned representation for scalable vector graphics. In *CVPR*, 2019. 2

[36] Jiasen Lu, Christopher Clark, Rowan Zellers, Roozbeh Mottaghi, and Aniruddha Kembhavi. Unified-IO: A unified model for vision, language, and multi-modal tasks. In *ICLR*, 2023. 1, 2

[37] Jiasen Lu, Vedanuj Goswami, Marcus Rohrbach, Devi Parikh, and Stefan Lee. 12-in-1: Multi-task vision and language representation learning. In *CVPR*, 2020. 2

[38] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. DesignScape: Design with interactive layout suggestions. In *CHI*, 2015. 1

[39] Peter O'Donovan, Aseem Agarwala, and Aaron Hertzmann. Learning layouts for single-pagegraphic designs. *IEEE TVCG*, 20(8), 2014. 3

[40] Qianru Qiu, Xueting Wang, Mayu Otani, and Yuki Iwazaki. Color recommendation for vector graphic documents based on multi-palette representation. In *WACV*, 2023. 1

[41] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 5

[42] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 4, 5

[43] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Learning multiple visual domains with residual adapters. In *NeurIPS*, 2017. 2

[44] Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. Efficient parametrization of multi-domain deep neural networks. In *CVPR*, 2018. 2

[45] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning structured output representation using deep conditional generative models. 2015. 6

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 4

[47] Guolong Wang, Zheng Qin, Junchi Yan, and Liu Jiang. Learning to select elements for graphic design. In *ICMR*, 2020. 3, 7

[48] Peng Wang, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. Ofa: Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *ICML*, 2022. 2

[49] Yuxi Xie, Danqing Huang, Jinpeng Wang, and Chin-Yew Lin. CanvasEmb: Learning layout representation with large-scale pre-training for graphic design. In *ACM MM*, 2021. 2

[50] Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. LayoutLM: Pre-training of text and layout for document image understanding. In *KDD*, 2020. 2

[51] Yang Xu, Yiheng Xu, Tengchao Lv, Lei Cui, Furu Wei, Guoxin Wang, Yijuan Lu, Dinei Florencio, Cha Zhang, Wanxiang Che, et al. LayoutLMv2: Multi-modal pre-training for visually-rich document understanding. In *ACL*, 2020. 2

[52] Kota Yamaguchi. CanvasVAE: Learning to generate vector graphics documents. In *ICCV*, 2021. 1, 2, 5, 6

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

[54] Lin-Ping Yuan, Ziqi Zhou, Jian Zhao, Yiqiu Guo, Fan Du, and Huamin Qu. Infocolorizer: Interactive recommendation of color palettes for infographics. *IEEE TVCG*, 28(12), 2021. 1

[55] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *ECCV*, 2014. 2

[56] Nanxuan Zhao, Ying Cao, and Rynson WH Lau. Modeling fonts in context: Font prediction on web designs. *Comput. Graph. Forum*, 37(7), 2018. 1, 3, 7, 8

[57] Xinru Zheng, Xiaotian Qiao, Ying Cao, and Rynson WH Lau. Content-aware generative modeling of graphic design layouts. *ACM TOG*, 38(4), 2019. 3