Multimodal Markup Document Models for Graphic Design Completion

Kotaro Kikuchi
kikuchi_kotaro_xa@cyberagent.co.jp
CyberAgent
Shibuya-ku, Tokyo, Japan

Ukyo Honda
honda_ukyo@cyberagent.co.jp
CyberAgent
Shibuya-ku, Tokyo, Japan

Naoto Inoue
naoto.inoue.0804@gmail.com
CyberAgent
Shibuya-ku, Tokyo, Japan

Mayu Otani
otani_mayu@cyberagent.co.jp
CyberAgent
Shibuya-ku, Tokyo, Japan

Edgar Simo-Serra
ess@waseda.jp
Waseda University
Shinjuku-ku, Tokyo, Japan

Kota Yamaguchi
yamaguchi_kota@cyberagent.co.jp
CyberAgent
Shibuya-ku, Tokyo, Japan

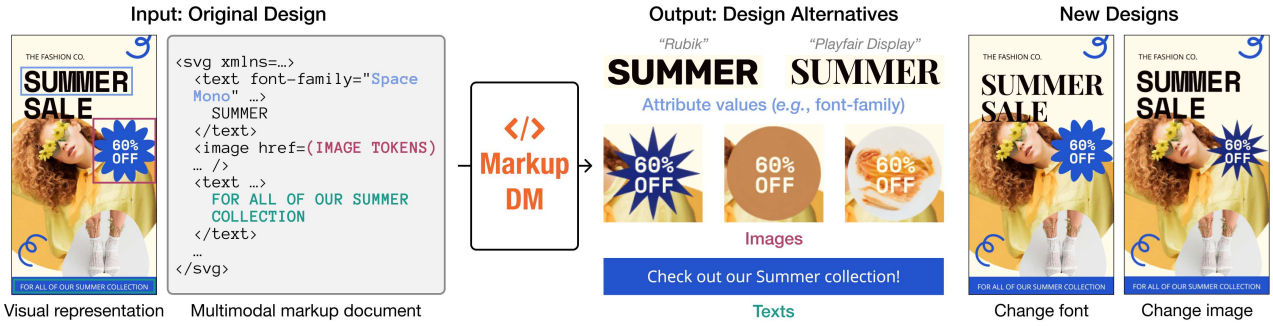

Figure 1: We present a multimodal markup document model (MarkupDM) for graphic design documents. Our model can generate alternative designs by inferring target spans, such as attribute values, images with transparency, and text, from the surrounding context.

Abstract

We introduce MarkupDM, a multimodal markup document model that represents graphic design as an interleaved multimodal document consisting of both markup language and images. Unlike existing holistic approaches that rely on an element-by-attribute grid representation, our representation accommodates variable-length elements, type-dependent attributes, and text content. Inspired by fill-in-the-middle training in code generation, we train the model to complete the missing part of a design document from its surrounding context, allowing it to treat various design tasks in a unified manner. Our model also supports image generation by predicting discrete image tokens through a specialized tokenizer with support for image transparency. We evaluate MarkupDM on three tasks, attribute value, image, and text completion, and demonstrate that it can produce plausible designs consistent with the given context. To further illustrate the flexibility of our approach, we evaluate our approach on a new instruction-guided design completion task where our instruction-tuned MarkupDM compares favorably to state-of-the-art image editing models, especially in textual completion. These findings suggest that multimodal language models with our document representation can serve as a versatile foundation for broad design automation.

CCS Concepts

• **Applied computing** → **Media arts**; • **Computing methodologies** → *Natural language generation*; *Image representations*.

Keywords

Graphic design, multimodal large language models, markup-based completion, instruction-guided design, design automation.

ACM Reference Format:
Kotaro Kikuchi, Ukyo Honda, Naoto Inoue, Mayu Otani, Edgar Simo-Serra, and Kota Yamaguchi. 2025. Multimodal Markup Document Models for Graphic Design Completion. In *Proceedings of the 33rd ACM International Conference on Multimedia (MM '25), October 27–31, 2025, Dublin, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3746027.3755420

1 Introduction

Graphic design is a visual medium for communicating information and ideas by organizing text, images, and other elements in an aesthetically pleasing way. It is critical in numerous applications, such as websites, advertisements, and printed materials, but creating high-quality designs typically requires specialized expertise and substantial time. Several studies employ machine learning techniques to automate design-related tasks, including layout generation [13, 16, 17, 28, 39, 40, 44], colorization [22, 35, 36], and

MM '25, Dublin, Ireland.

typography stylization [42, 53]. Beyond individual tasks, there have also been holistic modeling approaches for multiple design tasks by formulating graphic design as a grid representation of heterogeneous attributes (element type, position, size, and font information) for each element and then performing generation or completion tasks over this representation [18, 50]. Although these methods open the door to flexible foundational models for graphic design, they rely on a predefined grid structure and are inefficient for dealing with variable element lengths and type-dependent attributes.

To allow for more flexible application, we represent a graphic design as an interleaved multimodal document composed of markup language and images and then model it using multimodal large language models (LLMs). This resulting formulation is more human-readable and naturally accommodates variable-length elements, type-dependent attributes, and text content. Moreover, by employing the fill-in-the-middle training [1, 3], we can represent various design tasks in a unified manner by completing the missing part in a document from the surrounding context. We train our model, which we call the *Multimodal Markup Document Model (MarkupDM)*, on 19K graphic design templates. Our model converts image content into discrete tokens using a specialized image tokenizer designed to handle images with transparency in various sizes, allowing it to recognize and generate partial images that compose the overall design. We evaluate MarkupDM on three design completion tasks: generating missing attribute values, images, and text in graphic design templates. Results show that MarkupDM can produce plausible designs consistent with the given context, enabling exploration of various design alternatives (Fig. 1).

To further demonstrate the extensibility of our approach, we define a new task called instruction-guided graphic design completion, where the model completes a design based on a given instruction. This setup not only reflects the user's intent but also allows an emerging LLM agent [48] to control the design process, making it more adaptable to specific objectives or creative requirements. To this end, we extend the commonly used Crello dataset [50] to include 125K triplets of instructions, partial designs, and completed designs, resulting in the *Crello-Instruct dataset*. We then fine-tune MarkupDM on this dataset to adapt it to the instruction-guided task. Compared with state-of-the-art image editing models, our model demonstrates favorable performance on this task, particularly in textual completion. Our contributions are as follows:

- We formulate graphic design as an interleaved multimodal document consisting of markup language and images.
- We propose MarkupDM, a multimodal model that can generate both markup language and images, supported by a tailored image tokenizer capable of encoding variable-sized images with transparency into discrete tokens.
- We extend MarkupDM to an instruction-guided completion task by introducing the Crello-Instruct dataset, which comprises instruction-partial design-completed design triplets.
- We show empirically that both MarkupDM and its instruction-tuned variant can successfully complete graphic design documents, demonstrating advantages over existing methods.

2 Related Work

We first discuss existing approaches to *graphic design generation and completion*, covering both task-specific and holistic modeling methods. We then review recent advances in *multimodal large language models* that can recognize and generate images. Finally, we examine *instruction-guided image editing* methods, clarifying how our structured editing differs from purely image-based approaches.

2.1 Graphic Design Generation and Completion

Researchers have long studied computational support for graphic design tasks such as layout generation [5, 17, 23, 30, 31, 40], colorization [22, 35], typography stylization [42, 53], and general stylization [41]. Several studies share a common goal of inferring missing parts or alternative solutions from the existing context. For example, completing a layout from a partially specified layout is a common subtask in layout generation [17]. Zhao *et al.* [53] predict typographic styles in web design from both visual and semantic cues. Shao *et al.* [41] introduce a generative model for web page styling. Qiu *et al.* [35] propose a masked prediction approach for recoloring design documents based on color palette representations.

Different from the task-specific approaches, some studies aim to model entire design documents. CanvasVAE [50] is a variational autoencoder that generates heterogeneous attributes (type, position, size, and image content) for each element in a graphic design document. FlexDM [18] adopts a masked prediction strategy to capture relationships among elements and their attributes. Both methods estimate feature representations for images and text and then retrieve similar ones from a dataset. These methods, however, rely on a predefined element-by-attribute grid representation, which can be inefficient for variable-length elements and type-dependent attributes. There is also growing interest in generating stylized text over generated raster images [6, 19, 20, 49], focusing on producing high-quality overall designs.

Recent work has also applied large language models (LLMs) to design tasks [27–29, 39, 44]. Lin *et al.* [27] translate a text description into an intermediate representation to guide the subsequent layout generation. LayoutNUWA [44] formulates layout generation as a code generation task and leverages LLM knowledge to generate layout code. LaDeCo [29] uses multimodal LLMs to automatically place visual and textual elements in a layered manner.

Inspired by these studies, we propose a novel approach to holistic modeling by representing graphic design as an interleaved multimodal document. Unlike the grid-based methods [18, 50], our representation naturally accommodates variable-length elements, type-dependent attributes, and text content. We train a multimodal LLM on this document representation and support both text and image generation, in contrast to methods that assume images and text are provided [29] or retrieval-based methods [18, 50].

2.2 Multimodal Large Language Models

The recent success of large language models (LLMs) has led to the development of multimodal LLMs that can recognize and generate images [51]. Some approaches, such as DreamLLM [9], connect an LLM to an off-the-shelf pre-trained image encoder like CLIP [37] and a decoder such as Stable Diffusion [38]. However, these image encoders and decoders are not suitable for graphic design tasks

because they do not support images with transparency. They also require large-scale image-text datasets, which are difficult to collect in the graphic design domain, where textual descriptions often fail to capture the fine details of images, especially for decorative elements.

Another line of work in multimodal LLMs represents images as discrete tokens [1, 8, 45] using a pre-trained image tokenizer like VQGAN [10]. Publicly available tokenizers often do not support transparency, but they only require image data rather than large image-text datasets. We adopt this token-based approach and adapt it to handle images with transparency in graphic design. Furthermore, inspired by LLMs developed for code generation, we use a fill-in-the-middle training objective [1, 3] for our multimodal LLM. This objective enables the model to learn how to complete missing parts of a design from the surrounding context, serving as a flexible foundation for graphic design completion.

2.3 Instruction-guided Image Editing

Recent advances in image generation models have led to more practical applications of instruction-guided image editing. InstructPix2Pix [4] is a pioneering work in this field. The authors create a dataset by starting with manually created editing examples and then scaling them up using an off-the-shelf large language model and image generation model. MGIE [11] augments brief instructions with additional context derived from the embedded knowledge of pre-trained multimodal LLMs. HQ-Edit [15] enhances dataset quality through a tailored data creation pipeline that leverages advanced foundation models. More recently, proprietary models such as Gemini 2.0 Flash Experimental [21] and OpenAI's 4o Image Generation [33] have demonstrated impressive performance on image editing tasks. Concurrently, IDEA-Bench [26] proposes a comprehensive benchmark of professional design tasks, including image retouching and text insertion.

In contrast with the image-based approaches described above, we focus on instruction-guided editing within structured multimodal documents. This approach can improve the preservation of the original content while providing a more interpretable editing process. We build a new dataset specifically for this task and validate our model's performance with it.

3 Method

We begin by describing our multimodal document representation, then introduce our proposed MarkupDM model. Finally, we present our specialized image tokenizer, which supports images with transparency commonly used in graphic design. We illustrate an overview of our method in Figs. 2 and 3.

3.1 Document Representation

We represent graphic design as a multimodal markup document based on the SVG format[1], which naturally supports variable-length elements, type-dependent attributes, and text content. Unlike standard SVG, we replace image content with discrete image tokens generated by our image tokenizer (described later in Section 3.3). We show an example of the markup document representation in the following:

[1] https://www.w3.org/TR/SVG11/

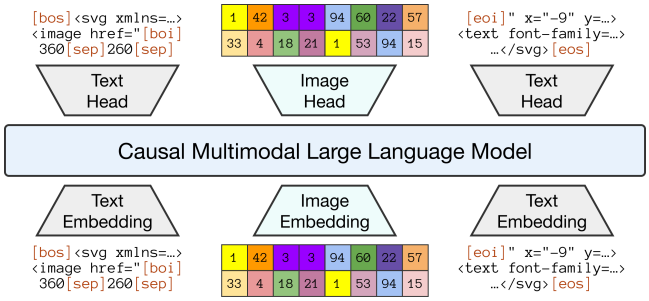

Figure 2: Our MarkupDM is based on causal multimodal LLM, with separate embedding layers and prediction heads dedicated to images and text tokens.

Multimodal markup document

```
[bos]<svg xmlns="http://www.w3.org/2000/svg" viewBox="0 0 419 298
 "width="419" height="298">
  <image href="[boi]360[sep]260[sep][img:1][img:42][img:3][img:94
  ]...[eoi]" x="-9" y="-9" width="436" height="315"/>
  <text font-family="Montserrat" font-size="30" font-weight="bold
  " fill="rgba(255, 255, 255, 1)" x="32" y="81">FAMILY</text>
...</svg>[eos]
```

The image content, *i.e.*, the value of `href` attribute in the `<image>` tag, starts with the special token `[boi]` and ends with `[eoi]`. The inside of these is separated by the special token `[sep]`, and each represents the width, height, and image tokens such as `[img:1]` obtained by our image tokenizer. This image representation is similar to the previous work on a multimodal LLM for simplified HTML documents [1], but differs in that the image size is also described as text and included in the target of generation.

3.2 Multimodal Markup Document Model

To incorporate the image representation described in Section 3.1, we build the multimodal markup document model (MarkupDM) by applying two extensions to the base LLM. First, we extend the vocabulary of the base LLM to include the additional special tokens, such as `[boi]`. Second, we add new modules dedicated to the image tokens, such as `[img:1]`, the embedding module, and the prediction head. In the embedding module, we first embed the image tokens via the frozen lookup table in our image decoder (Section 3.3). We then concatenate them with the positional encodings [43] and project them to the same dimension as the text embeddings. The prediction head for image tokens is similar to the one for text tokens, but uses a different set of parameters and vocabulary, *i.e.*, the codebook size in image tokenization.

We train our model based on the next token prediction in our sequences to which we randomly apply the fill-in-the-middle transformation [1, 3], allowing the model to predict the missing middle part from the prefix and suffix parts. During inference, our model must identify the modality of the next token due to the different prediction heads. To determine which modality to generate, we explicitly track whether the model is currently in the process of generating image tokens based on the generated text so far.

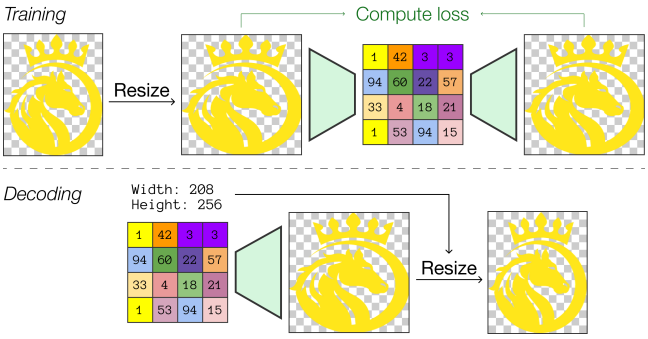

Figure 3: Our image tokenizer is trained by reconstructing images resized to a fixed size. When decoding, the image size is given in addition to the image tokens.

3.3 Specialized Image Tokenizer

Existing publicly available image tokenizers are typically designed for RGB images and thus do not support transparency in images, which is common in graphic design. To address this limitation, we develop a new image tokenizer by training an image autoencoder that encodes transparent images of varying sizes into discrete token maps at a $1/f$ resolution and decodes them back into the original images. While it is straightforward to vary the token size according to the image size, we found in preliminary experiments that this makes it difficult to train the markup language model at a later stage. Instead, we take a simple but effective approach of resizing the input image into a fixed square size. We follow the previous studies [10, 38] and take the same network architecture and training objectives for our autoencoder, with the only difference related to the alpha channel, *i.e.*, transparency. We set the number of input/output channels to four and consider L1 reconstruction loss for all channels. When calculating the loss based on RGB-based external models, *e.g.*, the perceptual loss [52], we convert generated RGBA images to RGB images by alpha compositing on a white background. We initialize our model with the weights of a pre-trained RGB image tokenizer. For the alpha channel weights, we use the mean values of the corresponding RGB weights.

4 Crello-Instruct Dataset

In addition to the document completion tasks and to showcase the extensibility of our approach, we introduce a new task called instruction-guided graphic design completion, which requires the model to complete a design based on a provided instruction. To create the benchmark dataset for this task, we extend the commonly used Crello dataset [50] to support instruction-guided completion. We refer to the resulting dataset as the *Crello-Instruct dataset*. A design template in the Crello dataset includes multimodal elements such as text, images, and other visual elements. We remove one of the elements to create a partial design, then use the specialized renderer[2] to generate rendered images of both the partial and original designs. Then, we feed the partial and original designs into the Qwen2.5-VL-7B-Instruct model [2] and ask it to generate an instruction to recreate the original design based on the partial design. Because the resulting instructions are often noisy, we use GPT-4o mini [32] to rate the quality of each triplet (instruction, partial design, and completed design) and filter out lower-quality samples. We then use the filtered dataset to train and evaluate our instruction-tuned model.

[2]https://github.com/CyberAgentAILab/cr-renderer

Completed design

Partial 1: *Add a charcoal drawing of a horse's head in the bottom right corner of the image.*

Partial 2: *Replace "WORKSHOP ON" with "WORKSHOP ON CHARCOAL DRAWING".*

(a) Completed design and two partial designs with instructions.

A detailed pencil sketch of a horse's head and part of its neck. The horse is wearing a bridle with reins, ...

A zigzag pattern consisting of alternating straight and curved segments, ...

(b) Image elements with captions.

Figure 4: Examples of our Crello-Instruct dataset.

Additionally, we generate a caption for each non-textual element in the dataset with Qwen2.5-VL-7B-Instruct [2] to help the model understand the image content. In our document representation, we add an extra `caption` attribute in the `<image>` tag, placing it before the `href` attribute, so that the model predicts the caption first and then the actual image tokens [1]. We provide examples of instructions and captions in Fig. 4. The caption examples highlight the unique challenges of this dataset, which contains both semantically describable elements and abstract decorative ones, and the later often have noisy captions. Further details can be found in the supplementary material.

5 Experiments

We begin by evaluating our image tokenizer on an image reconstruction task. Next, we assess our multimodal markup language models on various graphic design completion tasks. Finally, we evaluate our instruction-tuned models on the instruction-guided completion task.

5.1 Image Reconstruction

5.1.1 Setup. We use an internal dataset of graphic design templates, which is similar to the Crello dataset [50]. Each template

Table 1: Quantitative comparison of image reconstruction for each tokenizer. The dagger symbol (†) indicates the score computed by setting the alpha value of every pixel to 1.0.

	MSE ↓		rFID ↓
	RGB ($\times 10^{-3}$)	Alpha ($\times 10^{-1}$)	RGB
LDM-VQ [38]	2.42	3.75†	6.34
Ours-RGB	**1.50**	3.75†	**1.65**
Ours	1.86	**0.03**	4.96

consists of an ordered set of elements, and each element is associated with an element category, geometric attributes, and design attributes. The template also includes global attributes such as canvas size. We use 800,000 RGBA images of non-textual elements from these design templates for training and 133,267 images from different templates for evaluation.

We finetune a baseline RGB tokenizer for 100,000 steps, following the techniques explained in Section 3.3, to adapt it to RGBA images. For the baseline tokenizer, we adopt the one from the Latent Diffusion Model (LDM-VQ) [38] trained on the OpenImages dataset [24], which is primarily composed of photographs. Specifically, we use the tokenizer with the scaling factor $f = 16$ and the codebook size $Z = 16{,}384$, balancing reconstruction quality and the resulting token length. For further analysis, we finetune the tokenizer solely on RGB images without additional techniques, referred to as *Ours-RGB*. As an additional baseline without the specialized tokenizer, we convert RGB images into RGBA using an off-the-shelf background removal tool, Rembg [12] with IS-Net [34].

We evaluate the tokenizers using mean squared error (MSE) for both the RGB and alpha channels, as well as reconstruction Fréchet Inception Distance (rFID) for RGB images, which measures the distance between the feature distributions of the original and reconstructed images. For RGB-based metrics, we convert the RGBA images generated by our tokenizer to RGB by alpha compositing them onto a white background.

5.1.2 Results. We show a quantitative comparison of image reconstruction in Table 1. Both of our tokenizers outperform the baseline in terms of RGB-based metrics thanks to their fine-tuning on images from the same domain. We also show qualitative comparisons in Fig. 5. As illustrated, the general background removal used for RGB-based reconstructions often fails, removing foreground objects either too aggressively or insufficiently. In contrast, our tokenizer successfully reconstructs RGBA images by leveraging the alpha information embedded in the discrete tokens.

5.2 Graphic Design Completion

5.2.1 Setup. We use the Crello dataset [50] (version 5.0.0), comprising 19,372 templates for training, 1,823 for validation, and 2,107 for testing. We then convert these templates into SVG format. During the conversion, we represent text elements with `<text>` tags and other elements with `<image>` tags. We omit attributes when they have default values. Also, because SVG does not support multi-line text within a single element, we split any text element into multiple elements whenever a new line appears.

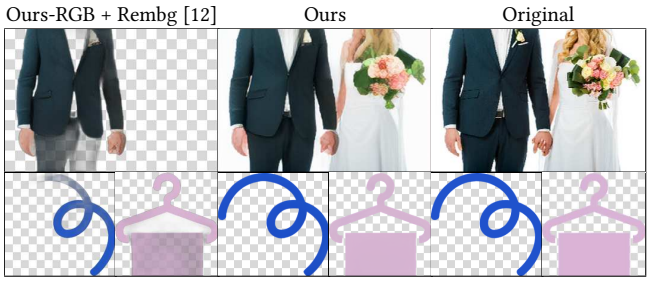

Figure 5: Image reconstruction results.

We train our MarkupDM with the fill-in-the-middle (FIM) objective [1, 3], which predicts a randomly selected middle span based on the prefix and suffix. In this setup, MarkupDM can infer the missing span from its preceding and following context. To demonstrate its effectiveness, we evaluate three tasks: attribute value completion, image completion, and text completion. Attribute value completion is represented as `<text x="[MASK]" ...>`, where `[MASK]` indicates the span to be filled. Image completion is represented as `<image href="[MASK]" .../>`, and text completion is represented as `<text ...>[MASK]</text>`. For attribute value completion, we focus on six attribute types: `x`, `y`, `width`, `height`, `font-family`, and `font-size`. Note that we do not train MarkupDM with task-specific supervision such as specialized FIM patterns; these tasks serve only for post-hoc evaluation.

We evaluate MarkupDM using several base language models, including StarCoderBase [25] with 1B, 3B, and 7B parameters, as well as Qwen2.5-7B [46] and Qwen2.5-Coder-7B [14]. We specifically select these models because they provide sufficiently long context lengths and employ the FIM objective during their pre-training. Both of the features are essential for our completion tasks where the model must handle multiple textual and visual elements and dynamically insert missing parts. For comparison with the approach of holistic yet grid-based graphic design generation approach (see Section 2.1), we also train FlexDM [18] on our dataset using random masking patterns, aiming to create similar experimental conditions. Note that during text and image completion tasks, FlexDM retrieves texts or images from the combined train and validation set instead of generating them directly.

We evaluate MarkupDM on between 12,559 and 25,435 target spans from the test templates, selecting the relevant spans for each task. To reduce inference time for image completion, we use 1,386 spans from the first 200 templates. We parse the text generated by MarkupDM and convert it to the same format used by FlexDM. We then compute accuracy over the quantized representation for attribute value completion, and cosine similarity over feature representations for text and image completion. More details are provided in the supplementary material.

5.2.2 Results for Attribute Value Completion. We show the quantitative results for attribute values (*X*, *Y*, *Width*, *Height*, *Font*, *F-Size*) in Table 2. Note that the scores for FlexDM and MarkupDM are not fully comparable, because they differ in formulation and available contextual cues. For example, MarkupDM can infer element sizes from the image dimensions, whereas FlexDM cannot. Nevertheless,

Table 2: Quantitative comparison for design completion tasks. The reported scores reflect accuracy for attribute values and cosine similarity for text and image completion. "Font" denotes the font family, and "F-Size" denotes the font size. "Mean" indicates the average score of all the completion tasks. FlexDM follows a different formulation than MarkupDM, so its scores are not directly comparable and are provided only for reference.

Model	Base LLM	X ↑	Y ↑	Width ↑	Height ↑	Font ↑	F-Size ↑	Text ↑	Image ↑	Mean ↑
FlexDM [18]	–	0.420	0.268	0.406	0.612	0.844	**0.851**	0.813	0.759	0.622
MarkupDM	Qwen2.5-7B	0.460	0.285	0.824	0.904	0.460	0.670	0.827	0.811	0.655
	Qwen2.5-Coder-7B	0.486	0.331	0.853	0.931	0.365	0.700	0.851	0.806	0.665
	StarCoderBase-1B	0.471	0.339	0.843	0.920	0.845	0.678	0.851	0.822	0.721
	StarCoderBase-3B	0.508	0.379	0.870	0.936	0.854	0.724	0.865	**0.823**	0.745
	StarCoderBase-7B	**0.526**	**0.404**	**0.882**	**0.951**	**0.867**	0.720	**0.874**	0.817	**0.755**

MarkupDM performs well in comparison, indicating that it successfully learns to fill graphic design templates. Among the MarkupDM variants, StarCoderBase-7B achieves the highest accuracy for most attributes. Comparing the results across different parameter sizes of StarCoderBase (1B, 3B, and 7B), we observe that larger models consistently perform better, as expected. Although Qwen2.5-based models also work with our approach, they tend to show lower performance, possibly due to limited exposure to SVG data during pre-training.

5.2.3 Results for Text Completion. We present the quantitative results for text completion in the *Text* column of Table 2. We observe that our model outperforms the baseline, and its performance improves as the model size increases. In the left and middle parts of Fig. 6, we show examples where the model successfully generates text that aligns grammatically with preceding or subsequent lines, or that serves a similar role to the ground truth text. Our model sometimes fails due to errors in image understanding or conflicting with other elements visually, *e.g.*, the rightmost example.

5.2.4 Results for Image Completion. The quantitative results for image completion in the *Image* column of Table 2 also demonstrate improved performance compared to the baseline. Unlike text completion, however, the variation in performance with respect to model size is relatively smaller. For deeper analysis, we investigate the effect of providing auxiliary caption information, which we introduced in Section 4. In Table 3, we observe no performance gain when training the model with captions. However, using ground-truth captions substantially improves image generation performance (the bottom row of Table 3), suggesting that the model struggles to accurately predict content, possibly due to limited training data. Qualitative results in Fig. 7 illustrate that our model can generate simpler design elements, such as underlays or buttons, by leveraging textual content or repetition patterns as hints. Our model has difficulty in producing main objects like the rightmost example or delicate visual harmonization with other elements. For example, in the middle example, the generated decoration slightly conflicts with the text element, highlighting the need for visual feedback.

5.3 Instruction-Guided Completion

5.3.1 Setup. We use the Crello-Instruct dataset as described in Section 4. Each sample is a triplet composed of an input document with one element missing, an instruction for completing that document, and a target document in which the missing element is filled in. The dataset includes 103,917 samples for training, 9,839 for validation, and 11,350 for testing.

Table 3: Image completion results using captions as auxiliary information. The baseline model is MarkupDM with StarCoderBase-7B.

Train with Caption	Test Input	Completion Target	Image ↑
–	Context	Image	0.817
✓	Context	Caption + Image	0.815
✓	Context + Caption	Image	0.857

We fine-tune the best variant of MarkupDM, *i.e.*, the one that uses StarCoderBase-7B as its base LLM, on this dataset, referring to the resulting model as *Instruct-MarkupDM*. For our baselines, we select two image editing methods: HQ-Edit [15] and Gemini 2.0 Flash Experimental (Gemini 2.0 FE) [21]. HQ-Edit is one of the latest open-source image editing models; we use both its original pre-trained model and a version further fine-tuned on our dataset. Gemini 2.0 FE is a proprietary model, which has recently demonstrated strong performance in terms of both image quality and instruction adherence.

We evaluate each model's performance using four pixel-based metrics: MSE_{GT}, MSE_{Edit}, Alignment [15], and Coherence [15]. MSE_{GT} measures the pixel-wise difference between the predicted image and the ground truth image, while MSE_{Edit} measures the difference between the input and the predicted image. A lower MSE_{Edit} than the ground-truth score indicates that the model has under-edited the image, whereas a higher MSE_{Edit} suggests over-editing or adding irrelevant elements. Therefore, while a lower score indicates better performance for MSE_{GT}, MSE_{Edit} is considered better when its score is closer to the ground truth score. Alignment and Coherence [15] are both GPT-based evaluations: Alignment measures the degree to which the edited image satisfies the instruction in the context of the input image, and Coherence assesses the overall visual quality of the edited image, independent of the instruction. We employ GPT-4o mini [32] for both metrics, using the same prompts specified in the previous work [15].

5.3.2 Results. We present the quantitative results in Table 4 and the qualitative results in Fig. 8 for instruction-guided graphic design completion. Among the image-editing methods, HQ-Edit highlights

Figure 6: Text completion results. Each pair shows the predicted completion and the original design from left to right or top to bottom. The green boxes indicate the target text and some of them are zoomed in for better visibility.

Figure 7: Image completion results. Each triplet shows the input, the predicted completion, and the original design from left to right or top to bottom. The gray squares indicate the target image elements to be completed.

the importance of fine-tuning on our design dataset to bridge the domain gap from general image-editing datasets. By contrast, Gemini 2.0 FE achieves better performance than HQ-Edit even in zero-shot settings, presumably due to its strong instruction-following and image-generation capabilities. However, Gemini 2.0 FE sometimes applies overly aggressive visual edits or incorrect text edits (as shown in the top example in Fig. 8), leading to poor MSE scores.

Instruct-MarkupDM achieves the best MSE scores and a higher Coherence score, because it only adds the missing elements rather than altering existing ones, leaving most input designs intact. However, its Alignment score is lower than that of Gemini 2.0 FE, possibly reflecting less robust instruction-following and visual generation capabilities. As Fig. 8 illustrates, Instruct-MarkupDM generally handles text editing well but struggles with generating complex visual elements beyond simple colored backgrounds.

Given the recent success of text-to-image (T2I) models in generating high-quality images from text prompts, we also introduce a variant of our model, *Instruct-MarkupDM**, which generates additional captions for image elements to leverage external T2I models. Figure 9 shows the qualitative results with and without using T2I. Without T2I, the model produces vague and unclear objects, whereas with T2I, the images are more detailed and better aligned with the instructions. This result demonstrates that our model benefits from recent T2I models to generate high-quality images. While

Table 4: Quantitative comparison for instruction-guided graphic design completion.

Model	$\text{MSE}_{\text{GT}} \downarrow$	MSE_{Edit}	Align. ↑	Coher. ↑
HQ-Edit [15]	93.9	93.5	28.6	57.4
+ Finetune	43.9	43.1	51.1	62.3
Gemini 2.0 FE [21]	<u>33.5</u>	<u>31.6</u>	**72.3**	**69.4**
Instruct-MarkupDM	**10.0**	**6.7**	<u>60.5</u>	<u>69.3</u>
Ground Truth	0.0	8.2	85.2	71.8

Figure 8: Qualitative comparison for instruction-guided graphic design completion.

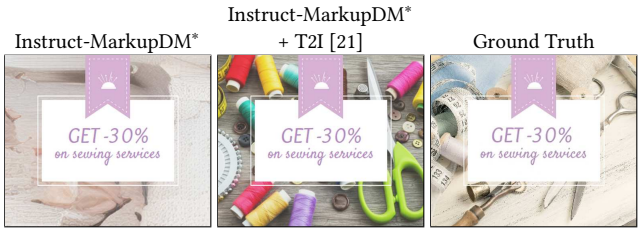

Figure 9: Qualitative results for instruction-guided completion with caption generation. The second column shows the result of using the external text-to-image model [21] to generate the image based on the predicted caption.

these models may struggle with images requiring transparency or extreme aspect ratios, our image tokenizer can handle these needs. Our findings suggest that external T2I models can compensate for our model's limited image-generation capabilities while achieving instruction-guided completion within editable, structured graphic design templates.

6 Limitations and Discussion

We presented MarkupDM, a multimodal markup document model that integrates a large language model trained using the fill-in-the-middle objective and a specialized image tokenizer for images of variable sizes with transparency. By treating graphic designs as interleaved multimodal documents, our approach unifies text and image token generation within a single framework. Experimental results indicate that MarkupDM effectively completes various graphic design tasks, including attribute value prediction, image generation, and text insertion, while preserving the contextual relationships among design elements. Further extension to instruction-guided design completion demonstrates the flexibility of our approach, where it achieves competitive performance compared with state-of-the-art image editing models.

Despite these promising results, our approach has several limitations. First, the model still struggles to generate complex or highly detailed images. As shown in Fig. 9, an external text-to-image model can generate primary image elements using predicted captions, but it is unclear whether it can produce decorative or background elements that visually harmonize with the surrounding content. Incorporating a more recent, powerful multimodal model such as Janus-Pro [7] could solve this issue, although it lacks native fill-in-the-middle capabilities. Additionally, given the rapid progress in foundation models, agentic approaches to design automation are another promising direction [47].

Second, our model faces challenges in visually intricate compositional tasks that require nuanced spatial reasoning, such as layering multiple objects or maintaining aesthetic coherence across various elements. Enhancing the model's spatial understanding may require domain-specific training or dedicated spatial modules.

Finally, our current model primarily focuses on the generation of new elements rather than refining or editing existing elements in detail or creating entire documents from scratch. Although it can insert a missing component, full-fledged editing of already-placed objects (including detailed manipulations of shape and texture) remains outside its scope. Addressing these limitations in future work will likely involve larger and more diverse datasets. We hope our findings encourage further research on multimodal LLMs for design tasks and motivate the development of more sophisticated, user-driven design automation techniques.

References

[1] Armen Aghajanyan, Bernie Huang, Candace Ross, Vladimir Karpukhin, Hu Xu, Naman Goyal, Dmytro Okhonko, Mandar Joshi, Gargi Ghosh, Mike Lewis, and

Luke Zettlemoyer. 2022. CM3: A Causal Masked Multimodal Model of the Internet. *arXiv preprint arXiv:2201.07520* (2022).

[2] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Mingkun Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. 2025. Qwen2.5-VL Technical Report. *arXiv preprint arXiv:2502.13923* (2025).

[3] Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient Training of Language Models to Fill in the Middle. *arXiv preprint arXiv:2207.14255* (2022).

[4] Tim Brooks, Aleksander Holynski, and Alexei A Efros. 2023. Instructpix2pix: Learning to follow image editing instructions. In *CVPR*.

[5] Shang Chai, Liansheng Zhuang, Fengying Yan, and Zihan Zhou. 2023. Two-stage Content-Aware Layout Generation for Poster Designs. In *ACM MM*.

[6] Haoyu Chen, Xiaojie Xu, Wenbo Li, Jingjing Ren, Tian Ye, Songhua Liu, Ying-Cong Chen, Lei Zhu, and Xinchao Wang. 2025. POSTA: A Go-to Framework for Customized Artistic Poster Generation. In *CVPR*.

[7] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. 2025. Janus-Pro: Unified Multimodal Understanding and Generation with Data and Model Scaling.

[8] Ethan Chern, Jiadi Su, Yan Ma, and Pengfei Liu. 2024. ANOLE: An Open, Autoregressive, Native Large Multimodal Models for Interleaved Image-Text Generation. *arXiv preprint arXiv:2407.06135* (2024).

[9] Runpei Dong, Chunrui Han, Yuang Peng, Zekun Qi, Zheng Ge, Jinrong Yang, Liang Zhao, Jianjian Sun, Hongyu Zhou, Haoran Wei, Xiangwen Kong, Xiangyu Zhang, Kaisheng Ma, and Li Yi. 2024. DreamLLM: Synergistic Multimodal Comprehension and Creation. In *ICLR*.

[10] Patrick Esser, Robin Rombach, and Bjorn Ommer. 2021. Taming Transformers for High-Resolution Image Synthesis. In *CVPR*.

[11] Tsu-Jui Fu, Wenze Hu, Xianzhi Du, William Yang Wang, Yinfei Yang, and Zhe Gan. 2024. Guiding Instruction-based Image Editing via Multimodal Large Language Models.

[12] Daniel Gatis. 2020. Rembg. https://github.com/danielgatis/rembg. (accessed 2024-08-27).

[13] Daichi Horita, Naoto Inoue, Kotaro Kikuchi, Kota Yamaguchi, and Kiyoharu Aizawa. 2024. Retrieval-Augmented Layout Transformer for Content-Aware Layout Generation. In *CVPR*.

[14] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. 2024. Qwen2.5-Coder Technical Report. *arXiv preprint arXiv:2409.12186* (2024).

[15] Mude Hui, Siwei Yang, Bingchen Zhao, Yichun Shi, Heng Wang, Peng Wang, Cihang Xie, and Yuyin Zhou. 2025. HQ-Edit: A High-Quality Dataset for Instruction-based Image Editing. In *ICLR*.

[16] Mude Hui, Zhizheng Zhang, Xiaoyi Zhang, Wenxuan Xie, Yuwang Wang, and Yan Lu. 2023. Unifying Layout Generation With a Decoupled Diffusion Model. In *CVPR*.

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

[18] Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2023. Towards Flexible Multi-Modal Document Models. In *CVPR*.

[19] Naoto Inoue, Kento Masui, Wataru Shimoda, and Kota Yamaguchi. 2024. OpenCOLE: Towards Reproducible Automatic Graphic Design Generation. In *CVPRW*.

[20] Peidong Jia, Chenxuan Li, Yuhui Yuan, Zeyu Liu, Yichao Shen, Bohan Chen, Xingru Chen, Yinglin Zheng, Dong Chen, Ji Li, Xiaodong Xie, Shanghang Zhang, and Baining Guo. 2024. COLE: A Hierarchical Generation Framework for Multi-Layered and Editable Graphic Design. *arXiv preprint arXiv:2311.16974* (2024).

[21] Kat Kampf and Nicole Brichtova. 2025. Experiment with Gemini 2.0 Flash Native Image Generation. https://developers.googleblog.com/en/experiment-with-gemini-20-flash-native-image-generation/ (accessed 2025-03-31).

[22] Kotaro Kikuchi, Naoto Inoue, Mayu Otani, Edgar Simo-Serra, and Kota Yamaguchi. 2023. Generative Colorization of Structured Mobile Web Pages. In *WACV*.

[23] Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2021. Constrained Graphic Layout Generation via Latent Optimization. In *ACM MM*.

[24] Alina Kuznetsova, Hassan Rom, Neil Alldrin, Jasper Uijlings, Ivan Krasin, Jordi Pont-Tuset, Shahab Kamali, Stefan Popov, Matteo Malloci, Alexander Kolesnikov, Tom Duerig, and Vittorio Ferrari. 2020. The Open Images Dataset V4. *IJCV* 128, 7 (2020).

[25] Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. 2023. StarCoder: May the Source Be with You! *TMLR* (2023).

[26] Chen Liang, Lianghua Huang, Jingwu Fang, Huanzhang Dou, Wei Wang, Zhi-Fan Wu, Yupeng Shi, Junge Zhang, Xin Zhao, and Yu Liu. 2024. IDEA-Bench: How Far are Generative Models from Professional Designing?

[27] Jiawei Lin, Jiaqi Guo, Shizhao Sun, Weijiang Xu, Ting Liu, Jian-Guang Lou, and Dongmei Zhang. 2023. A Parse-Then-Place Approach for Generating Graphic Layouts from Textual Descriptions. In *ICCV*.

[28] Jiawei Lin, Jiaqi Guo, Shizhao Sun, Zijiang James Yang, Jian-Guang Lou, and Dongmei Zhang. 2023. LayoutPrompter: awaken the design ability of large language models. In *NeurIPS*.

[29] Jiawei Lin, Shizhao Sun, Danqing Huang, Ting Liu, Ji Li, and Jiang Bian. 2025. From Elements to Design: A Layered Approach for Automatic Graphic Design Composition. In *CVPR*.

[30] Jinpeng Lin, Min Zhou, Ye Ma, Yifan Gao, Chenxi Fei, Yangjian Chen, Zhang Yu, and Tiezheng Ge. 2023. AutoPoster: A Highly Automatic and Content-aware Design System for Advertising Poster Generation. In *ACM MM*.

[31] David D. Nguyen, Surya Nepal, and Salil S. Kanhere. 2021. Diverse Multimedia Layout Generation with Multi Choice Learning. In *ACM MM*.

[32] OpenAI. 2024. GPT-4o mini: advancing cost-efficient intelligence. https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/ (accessed 2025-03-31).

[33] OpenAI. 2025. Introducing 4o Image Generation. https://openai.com/index/introducing-4o-image-generation/ (accessed 2025-03-31).

[34] Xuebin Qin, Hang Dai, Xiaobin Hu, Deng-Ping Fan, Ling Shao, and Luc Van Gool. 2022. Highly Accurate Dichotomous Image Segmentation. In *ECCV*.

[35] Qianru Qiu, Xueting Wang, and Mayu Otani. 2023. Multimodal Color Recommendation in Vector Graphic Documents. In *ACM MM*.

[36] Qianru Qiu, Xueting Wang, Mayu Otani, and Yuki Iwazaki. 2023. Color Recommendation for Vector Graphic Documents based on Multi-Palette Representation. In *WACV*.

[37] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In *ICML*.

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. 2022. High-Resolution Image Synthesis with Latent Diffusion Models. In *CVPR*.

[39] Jaejung Seol, Seojun Kim, and Jaejun Yoo. 2024. PosterLlama: Bridging Design Ability of Langauge Model to Contents-Aware Layout Generation. In *ECCV*.

[40] Mohammad Amin Shabani, Zhaowen Wang, Difan Liu, Nanxuan Zhao, Jimei Yang, and Yasutaka Furukawa. 2024. Visual Layout Composer: Image-Vector Dual Diffusion Model for Design Layout Generation. In *CVPR*.

[41] Zirui Shao, Feiyu Gao, Hangdi Xing, Zepeng Zhu, Zhi Yu, Jiajun Bu, Qi Zheng, and Cong Yao. 2024. WebRPG: Automatic Web Rendering Parameters Generation for Visual Presentation. In *ECCV*.

[42] Wataru Shimoda, Daichi Haraguchi, Seiichi Uchida, and Kota Yamaguchi. 2024. Towards Diverse and Consistent Typography Generation. In *WACV*.

[43] Matthew Tancik, Pratul P. Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T. Barron, and Ren Ng. 2020. Fourier Features Let Networks Learn High Frequency Functions in Low Dimensional Domains. In *NeurIPS*.

[44] Zecheng Tang, Chenfei Wu, Juntao Li, and Nan Duan. 2024. LayoutNUWA: Revealing the Hidden Layout Expertise of Large Language Models. In *ICLR*.

[45] Chameleon Team. 2024. Chameleon: Mixed-Modal Early-Fusion Foundation Models. *arXiv preprint arXiv:2405.09818* (2024).

[46] Qwen Team. 2024. Qwen2.5 Technical Report. *arXiv preprint arXiv:2412.15115* (2024).

[47] Heng Wang, Yotaro Shimose, and Shingo Takamatsu. 2025. BannerAgency: Advertising Banner Design with Multimodal LLM Agents. *arXiv preprint arXiv:2503.11060* (2025).

[48] Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, et al. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science* (2024).

[49] Shaodong Wang, Yunyang Ge, Liuhan Chen, Haiyang Zhou, Qian Wang, Xinhua Cheng, and Li Yuan. 2024. Prompt2Poster: Automatically Artistic Chinese Poster Creation from Prompt Only. In *ACM MM*.

[50] Kota Yamaguchi. 2021. CanvasVAE: Learning to Generate Vector Graphic Documents. In *ICCV*.

[51] Shukang Yin, Chaoyou Fu, Sirui Zhao, Ke Li, Xing Sun, Tong Xu, and Enhong Chen. 2024. A Survey on Multimodal Large Language Models. *arXiv preprint arXiv:2306.13549* (2024).

[52] Richard Zhang, Phillip Isola, Alexei A. Efros, Eli Shechtman, and Oliver Wang. 2018. The Unreasonable Effectiveness of Deep Features as a Perceptual Metric. In *CVPR*.

[53] Nanxuan Zhao, Ying Cao, and Rynson W.H. Lau. 2018. Modeling Fonts in Context: Font Prediction on Web Designs. *Computer Graphics Forum* 37 (2018). Issue 7.