DiagramGPT-Llama3:Enabling Editable, High-Fidelity Diagram Generation with Vision Large Language Models

Yongyuan Chen[1]*Minjie Hong[2]*, Boxu Wu[2,3]†, Xicheng Han[2]

[1]State Key Lab of CAD&CG, Zhejiang University
[2]School of Software Technology, Zhejiang University
[3]Daerwen AI

Abstract

The automation of diagram generation has gained significant attention in recent years. Previous studies mainly focused on generating diagrams from natural language, but often lacked support for user-friendly editing like drag-and-drop. This paper proposes a novel task: generating **editable**, high-fidelity diagrams from either text or raster images. It is also among the first to introduce diagram restoration and style transfer in this setting. To tackle these tasks, we constructed the **Diagram-mxGraph dataset**, covering restoration, text-to-diagram generation, and style transfer. We propose two core innovations: Fine-grained Adaptive Background Suppression (FABS) and Component-Aware Adaptive Loss (CAAL). Leveraging pre-trained Vision Transformers (ViTs) and the Diagram Adapter module, our method aligns diagram features with a Large Language Model (LLM) to output diagrams in editable mxGraph format.

1 Introduction

Automated diagram generation has attracted growing interest due to its utility in clearly and effectively visually representing complex information across domains like software engineering, data management, and business modeling. However, traditional methods are often time-consuming and require expertise with diagramming tools.

Recent advances in natural language processing (NLP) and large language models (LLMs) have significantly enabled converting text into structured diagrams. Prior work, such as DiagrammerGPT, leverages LLMs to guide layout generation in text-to-image (T2I) models (Zala et al. 2023), yet challenges in layout control and seamless text integration remain. Other efforts focus on specific tasks like converting hand-drawn UML diagrams (Conrardy and Cabot 2024) or generating data flow diagrams (Herwanto 2024), but they often lack generality and robustness.

Crucially, the tasks of diagram restoration and style transfer remain underexplored. Existing methods often fail to produce diagrams that are both editable and user-friendly, lacking features like drag-and-drop interaction.

To address these critical gaps, we propose a novel framework built on the Diagram-mxGraph dataset, targeting three key tasks: diagram restoration, text-to-diagram generation, and style transfer. Our approach seamlessly integrates pre-trained Vision Transformers (ViTs) and LLMs to generate mxGraph-compatible diagram code, enabling fine-grained control and editability in tools like draw.io[1].

As illustrated in Figure 2, our pipeline processes diagram images using a Diagram Adapter with Fine-grained Adaptive Background Suppression (FABS) and Diagram Attention. These components enhance feature extraction and align image features with LLM outputs. The system supports sliced image inputs and allows both image- and text-based diagram generation, producing high-fidelity, editable diagrams with preserved structure and styling.

The **contributions** of this work include:

- The creation of a new dataset for diagram generation tasks introduces the first dataset to include draggable and editable diagram data;
- The introduction of the tasks of diagram restoration and style transfer, effectively utilizing pre-trained LLMs and ViTs to accomplish these tasks;
- The introduction of the Diagram Adapter and an improved loss function, enhancing the accuracy and user-friendliness of generated diagrams.

2 Related Work

Recent advances in automated diagram generation leverage LLMs, ViTs, and vision-language models to overcome the limitations of traditional rule-based methods, which often lack editability and visual quality.

2.1 Diagram Generation

While text-to-image (T2I) models (Rombach et al. 2021; Ramesh et al. 2022; Yu et al. 2022; Chang et al. 2023; Dai et al. 2023) have shown success in natural image synthesis, they struggle with diagrams due to the need for precise spatial layout and symbolic elements. Traditional text-to-diagram approaches (Alashqar 2021; Zhu, Li, and Jin 2023; Yang and Sahraoui 2022; Ben Abdessalem Karaa et al. 2016; Narawita et al. 2017; Krishnan and Samuel 2010; Ghosh

*These authors contributed equally.
†Corresponding author.

[1]https://app.diagrams.net

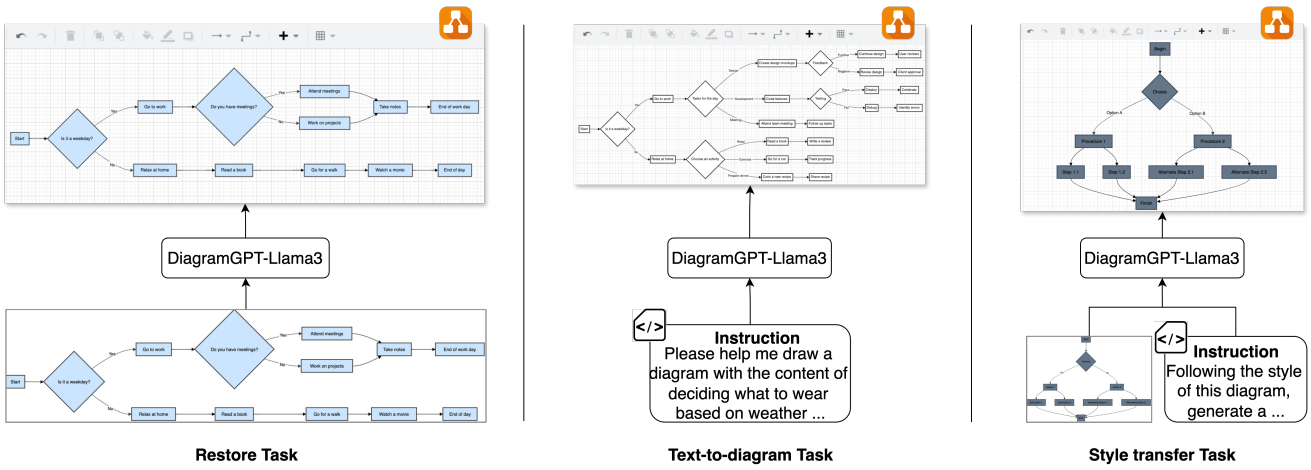

Figure 1: Examples of editable diagrams generated by DiagramGPT-Llama3 across tasks.Each column illustrates applications: diagrams generation,diagram restoration,and style transfer.The diagrams maintain drag-and-drop functionality,allowing users to modify components directly in compatible tools like draw.io. The pipeline demonstrates DiagramGPT-Llama3's ability to output high- fidelity, editable diagrams in mxGraph format,as shown by various diagram structures in the image.

et al. 2018; Zhou and Zhou 2004; Cheema, Tariq, and Pires 2023; Gulia and Choudhury 2016; Golovchinsky, Kamps, and Reichenberger 1995) rely on handcrafted rules, limiting scalability and output quality.

LLMs and ViTs have introduced diagram generation methods. DiagrammerGPT (Zala et al. 2023) uses LLMs to guide stable diffusion, but only produces non-editable raster images. Other approaches are either task-specific, like Herwanto's work on data flow diagrams (Herwanto 2024), or fail to preserve structure and accuracy, as shown in UML conversion tasks (Conrardy and Cabot 2024).

2.2 Vision-Language Models (VLMs)

VLMs combine vision and language understanding, with ViTs like ViT (Dosovitskiy 2020) and Swin (Liu et al. 2021) improving image representation. Cross-modal alignment has advanced with models like CAT (Lin et al. 2022) and multimodal LLMs such as LLaVA (Liu et al. 2023b) and Moe-LLaVA (Lin et al. 2024). MiniCPM-V (Yao et al. 2024) shows promise in visual reasoning tasks.

However, generating structured diagrams remains a challenge for VLMs due to the need for precise control over layout and semantics. Our work addresses this by integrating ViTs with LLMs to directly produce editable mxGraph code, enabling fine-grained, high-fidelity diagram generation.

3 Method

3.1 Dataset

Mermaid Code Generation: We used GPT4o-mini to construct prompts for generating Mermaid code, resulting in 2,475 Mermaid code samples.For Task 1 (diagram restoration), GPT4o-mini generated flowchart diagrams in JSON format, ensuring diversity in content and complexity. For Task 2 (text-to-diagram generation), we produced flowchart diagrams along with their corresponding textual descriptions. For Task 3 (diagram style transfer), GPT4o-mini generated pairs of stylistically similar flowchart diagrams, accompanied by textual descriptions for style transfer tasks.

mxGraph Code Generation: We implemented an automated conversion from Mermaid to mxGraph code by leveraging the open-source tool draw.io. Since Mermaid does not natively support conversion to mxGraph, the transformation results often included overlapping components, inconsistent font sizes, and disorganized connectors. We manually curated the data to remove diagrams with significant formatting issues and meticulously adjusted overlapping components to ensure clarity. The resulting mxGraph diagrams were standardized to a plain theme. To enhance visual diversity, we randomly applied color to 50% of the diagrams, selecting from a curated palette of 33 colors and programmatically embedding the chosen color in the style field of the mxGraph code.

mxGraph Dataset: We compiled the Diagram-mxGraph dataset, which contains a total of 2,475 entries and 3,290 diagrams, covering the three core tasks discussed above. Specifically:

- Task 1 (diagram restoration): 830 entries.
- Task 2 (text-to-diagram generation): 830 entries.
- Task 3 (diagram style transfer): 815 entries.

This dataset provides a comprehensive and versatile foundation for developing, evaluating, and benchmarking methods in diagram restoration, generation from text, and style transfer.

3.2 Diagram Adapter

The overall architecture of Diagram Adapter is plotted in Figure 2. Diagrams differ from natural images in their structured layouts, containing significant whitespace and distinct

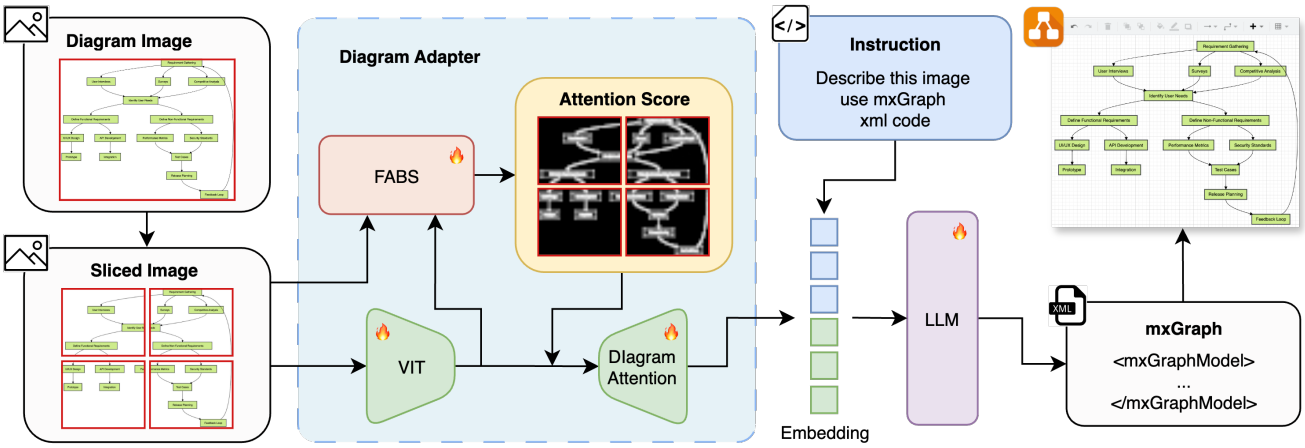

Figure 2: The pipeline of DiagramGPT-Llama3 for generating editable diagrams from diagram images and textual instructions. The process starts with a Diagram Image, which is processed by the Diagram Adapter module incorporating FABS and ViT components to produce an Attention Score. This score is used in Diagram Attention to align visual features with a Large Language Model (LLM), restoring the original diagram to a structured mxGraph XML code that renders an Editable Diagram. The resulting output is an editable, high-fidelity representation of the original diagram image, allowing for further customization and manipulation in compatible tools.

components like geometric shapes, text, and arrows. To exploit these structural characteristics effectively, we introduce the Diagram Adapter, which integrates Fine-grained Adaptive Background Suppression (FABS) and a carefully tailored cross-attention mechanism for improved diagram generation. The Diagram Adapter utilizes MiniCPM-V as its vision-language model (VLM) framework, with each input image processed through a series of precisely defined steps.To effectively handle these unique properties, the Diagram Adapter incorporates several key innovations built upon the MiniCPM-V vision-language model framework. The system begins by processing input images through a Fine-grained Adaptive Background Suppression (FABS) module that intelligently distinguishes between foreground components and non-informative background regions. This preprocessing step helps reduce noise and focuses computational resources on semantically meaningful elements. The architecture then employs a specialized cross-attention mechanism designed to capture both the visual features of diagram components and their spatial relationships. This mechanism operates at multiple scales to handle both local details and global layout structures.This high-level process is summarized as follows in detail.

Image Slicing and Patch Embedding: For input images with original resolutions higher than the ViT model's trained input size (448×448), the image is sliced into up to nine sub-images to maintain resolution and detail. Each sub-image is divided into 14×14 patches, and each patch p_{ij} of size 14×14 is embedded as a feature vector $\mathbf{e}_{ij} \in R^d$ using the Vision Transformer (ViT):

$$\mathbf{e}_{ij} = ViT(p_{ij}), \quad \forall i, j \in \{1, \ldots, 14\}, \tag{1}$$

where d represents the embedding dimensionality of patch.

Fine-grained Adaptive Background Suppression (FABS): FABS computes two attention scores for each patch: the complexity score C_{ij} derived from Local Binary Pattern (LBP) entropy and a solid-color attention score S_{ij} based on pixel uniformity.The architecture of FABS is plotted in Figure 3.The calculation method is as follows:

- Complexity Score Calculation: For each patch p_{ij}, LBP and entropy are applied to generate the histogram $H_{ij} \in R^{10}$ with 10 bins (radius 1, 8 sampling points). Entropy is then calculated using the formula as:

$$C_{ij} = -\sum_{k=0}^{9} H_{ij}[k] \cdot \log_2(H_{ij}[k]) \cdot \frac{255}{\log_2(10)}, \tag{2}$$

  where $H_{ij}[k]$ represents the frequency of each LBP pattern in the histogram.
- Solid-Color Attention Score Calculation: For solid-color patches, the attention score is inversely proportional to the color frequency across all patches. Let f_{color} denote the calculated frequency of a specific color across all solid-color patches and its relative importance, then:

$$S_{ij} = (1 - f_{color}(c)) \cdot 255. \tag{3}$$

Gate Network for Attention Fusion: The Gate Network integrates the complexity and solid-color attention scores for each patch to produce a fused attention score, A_{ij}, that balances both types of attention. Given: all_pixel_values $\in R^{B\times3\times14\times(N\times14)}$: the RGB values of each patch (B is the batch size, N is the number of patches per image).LBP_Ent_AttnScore $C \in R^{B\times N}$: LBP-based complexity scores.SC_AttnScore $S \in R^{B\times N}$: solid-color attention scores.The Gate Network performs the following steps:

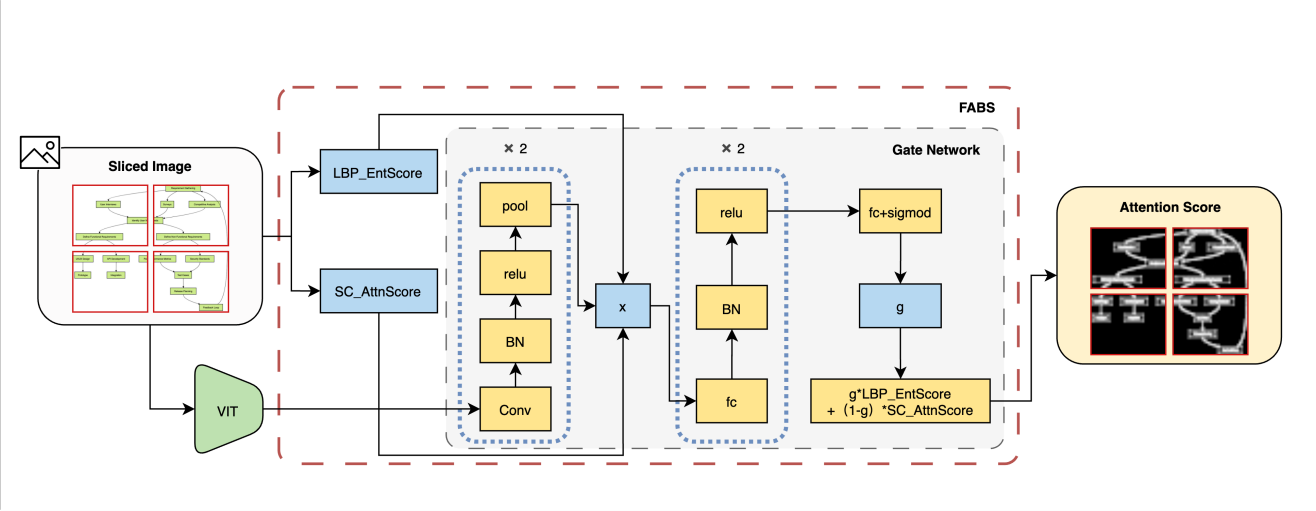

Figure 3: Pipeline of the Fine-grained Adaptive Background Suppression (FABS) framework. A sliced input image is processed by a Vision Transformer (ViT) to extract features. Complexity and color scores are computed and fused via a Gate Network to produce a final attention map, guiding precise feature alignment with an LLM for editable mxGraph diagram generation.

- Feature Extraction: Each input patch is processed through a sequence of convolutional and pooling layers, which produces a flattened feature vector $x \in R^{(B \cdot N) \times 288}$.
- Concatenation and Score Fusion: The flattened feature vector x is concatenated with C_{ij} and S_{ij}, yielding:

$$x_{input} = concat(x, C, S) \in R^{(B \cdot N) \times 290}. \quad (4)$$

  This combined vector is processed through a series of fully connected layers, producing a gating score $g \in R^{B \times N}$ that balances the two attention types.
- Final Attention Score Fusion: The final FABS attention score A_{ij} for each patch is calculated as:

$$A_{ij} = g \cdot C_{ij} + (1 - g) \cdot S_{ij}. \quad (5)$$

This fused attention score allows the model to dynamically prioritize patches based on their complexity, color distribution, and contextual relevance, thereby enhancing the model's ability to handle structured diagram content.

Diagram Attention with Cross-Attention Mechanism: The Diagram Attention aligns the patch embeddings $\{\mathbf{e}_{ij}\}$ with the LLM's embedding space. The cross-attention score matrix is modified to incorporate the FABS scores:

$$SA(Q, K) = softmax\left(\frac{QK^T}{\sqrt{d}}\right), \quad (6)$$

$$Attention(Q, K, V) = (w \cdot SA(Q, K) + (1 - w) \cdot A)\, V, \quad (7)$$

where $Q \in R^{query\_num \times D}$ is the query matrix, where $query\_num$ denotes the fixed number of queries, $K \in R^{L \times D}$ is the key matrix, $V \in R^{L \times D}$ is the value matrix, $A \in R^{query\_num \times L}$ is the FABS attention matrix, and $w = 0.9$ is a weighting factor that balances the FABS-enhanced attention with the softmax attention. The value of w is empirically determined;see the Appendix for details.

Language Model Processing and Diagram Code Generation: The aligned embeddings are input to the LLM, generating structured mxGraph code as:

$$mxGraphCode = LLM(\{\mathbf{e}_{ij} \mid DiagramAttention\}). \quad (8)$$

Visualization: The generated mxGraph code is rendered in draw.io, producing an editable diagram.

The entire process, from FABS-enhanced embedding to Diagram Attention, enables editable diagram generation optimized by pre-trained weights from MiniCPM-V.

3.3 Component-Aware Adaptive Loss (CAAL)

In this work, we propose a novel loss function, termed Component-Aware Adaptive Loss (CAAL), to improve the generation of mxGraph code in draw.io diagrams. The loss function emphasizes the importance of specific mxGraph fields—x, y, style, and id—by adjusting the weight of their corresponding loss terms to ensure that the generated diagrams are accurate, editable, and unique in structure.

Key Components of CAAL

Positional and Style Fields Enhancement: Accurate prediction of positional and style fields is essential for high-quality diagram generation, as these determine component layout and appearance. To emphasize their importance, we apply a scaled loss mechanism: the x/y coordinate losses are weighted by λ_1, and style field loss by λ_2, guiding the model

to balance positional accuracy and stylistic consistency. Additionally, spatial constraints penalize overlaps or poor spacing to ensure clean, coherent layouts. Together, these strategies enhance both the functional correctness and visual quality of the generated diagrams.

ID Field Regularization: The `id` field is crucial in mxGraph diagrams as a unique identifier for each individual component; duplicates can potentially lead to rendering errors or visual conflicts. To address this, we introduce a regularization mechanism that explicitly penalizes duplicate predictions with a loss scaled by λ_3, increasing exponentially with repeats. During inference, remaining duplicates are resolved via suffixes or reassignment strategies to ensure mxGraph compliance. Together with enhancements to both positional and style fields, this ensures the generated diagrams are both visually accurate and structurally valid.

Loss Function Formulation: The Component-Aware Adaptive Loss (CAAL) function modifies the cross-entropy loss to focus on the critical fields mentioned above. The loss function is defined as follows:

$$L_{CAAL} = L_{base} + \lambda_1 (L_x + L_y) + \lambda_2 L_{style} + \lambda_3 L_{id-duplicate}, \tag{9}$$

where L_{base} is the standard cross-entropy loss applied to all tokens in the mxGraph description.L_x, L_y, L_{style} represent the cross-entropy losses associated with the x, y, and style fields, respectively. $L_{id-duplicate}$ represents the penalization applied to any duplicate id values.λ_1, λ_2, and λ_3 are scaling factors applied to the respective loss components. In our experiments, we set $\lambda_1 = 10$, $\lambda_2 = 10$, and $\lambda_3 = 10$.

Detailed Explanation of Loss Terms

Base Loss: The base loss L_{base} is the cross-entropy loss for all other fields in the mxGraph code, computed as:

$$L_{base} = \sum_{i=1}^{N} \ell_{CE}(t_i, \hat{t}_i), \tag{10}$$

where t_i is the ground truth token, $\hat{t}_i$ is the predicted token, and ℓ_{CE} denotes the cross-entropy loss.

Positional and Style Loss: The x, y, and style fields are crucial for ensuring the diagram's layout and consistency. We increase their contribution to the overall loss by multiplying their cross-entropy losses by λ_1 and λ_2 as follows:

$$L_x = \sum_{i=1}^{N} \lambda_1 \times \ell_{CE}(x_i, \hat{x}_i), \tag{11}$$

$$L_y = \sum_{i=1}^{N} \lambda_1 \times \ell_{CE}(y_i, \hat{y}_i), \tag{12}$$

$$L_{style} = \sum_{i=1}^{N} \lambda_2 \times \ell_{CE}(s_i, \hat{s}_i). \tag{13}$$

where x_i, y_i, s_i are the ground truth values for the x, y, and style fields, and $\hat{x}_i, \hat{y}_i, \hat{s}_i$ are the corresponding predictions. The scaling factors λ_1 and λ_2 ensure that these fields receive greater attention during training.

ID Regularization Loss: The id field must remain unique for each component in the diagram. To penalize duplicate id values, we introduce a regularization term:

$$L_{id-duplicate} = \sum_{i=1}^{N} \lambda_3 \times I(\hat{id}_i = \hat{id}_j for i \neq j), \tag{14}$$

where $I(\cdot)$ is the indicator function that detects duplicate unique identifier id values, and N is the number of components. If a duplicate id is detected (i.e., $\hat{id}_i = \hat{id}_j$), a penalty of λ_3 times the cross-entropy loss is applied.

Impact of CAAL By applying the Component-Aware Adaptive Loss (CAAL), we significantly improve the model's ability to generate diagrams with accurate positioning, proper styling, and globally unique component identifiers. This key modification addresses the critical elements in diagram generation, making the generated diagrams more user-friendly, structurally sound, and easily editable in draw.io. The emphasis on the x, y, and style fields ensures that the visual quality of the generated diagrams is maintained, while the penalization of duplicate id values further enhances the structural integrity of the diagram.

4 Experiment

4.1 Experimental Setup

Datasets We use the **mxGraph dataset** with a total of 2,475 diagram samples, which is evenly divided into training, validation, and test sets for comprehensive model evaluation following an 8:1:1 split.

Evaluation Metrics

CLIPScore Following previous works (Cho, Zala, and Bansal 2023a; Saharia et al. 2022; Belouadi, Lauscher, and Eger 2023), we adopt CLIPScore as our primary metric to evaluate the similarity between the generated diagrams and the initial text descriptions or ground-truth diagrams. Specifically, we calculate **Img-Txt Score** to measure the similarity between the input text and the output diagram, and **Img-Img Score** to assess the similarity between the output diagram and the ground-truth diagram for comprehensive evaluation.

TaskScore We define TaskScore for each of the three tasks, using CLIP-based similarity measures tailored to each task's objective. For **Task 1** (Restore Task), which involves converting a non-editable diagram image into an editable version, we compute the CLIP similarity between the output image and the ground-truth diagram. **Task 2** (Text-to-Diagram Task) evaluates the generation of a diagram from a textual description; we calculate the average of the CLIP similarity between the input text and output diagram, and the similarity between the output and ground-truth diagrams. **Task 3** (Style Transfer Task) requires generating a diagram with a style similar to an input reference image; we compute the average of the CLIP similarities between the input text and output image, the output and ground-truth diagrams, and the output and reference style image to ensure a more comprehensive evaluation.

Methods	CLIPScore(%) ↑		TaskScore(%) ↑			Editable	Captioning(%) ↑	Overall(%) ↑
	Img-Txt	Img-Img	Task1	Task2	Task3			
Zero-shot								
Stable Diffusion v1.4	19.935	80.420	-	50.177	-	NO	5.427	38.989
VPGen	17.762	78.467	-	48.114	-	NO	1.333	36.419
AutomaTikZ	22.236	83.149	-	52.692	-	NO	1.797	39.968
Few-shot								
ChatGPT-4o-mini	**28.676**	84.084	91.385	54.813	**55.006**	**YES**	6.002	53.327
Fune-tuned								
Stable Diffusion v1.4	20.248	85.936	-	53.102	-	NO	6.133	41.354
DiagramGPT(Ours)	27.716	**87.235**	**95.298**	**58.821**	52.739	**YES**	**12.326**	**55.689**

Table 1: Quantitative results comparing DiagramGPT with baseline models on CLIPScore, TaskScore, and Captioning metrics, demonstrating DiagramGPT's superior performance across all tasks.

Methods	Task1(%)↑		Task2(%)↑		Task3(%)	
	accuracy	aesthetic	accuracy	aesthetic	accuracy	aesthetic
Stable Diffusion v1.4(finetuned)	-	-	24.581	28.722	-	-
VPGen	-	-	8.336	9.545	-	-
AutomaTikZ	-	-	17.818	19.395	-	-
ChatGPT-4o-mini(one-shot)	59.670	56.554	60.345	62.536	56.654	53.804
DiagramGPT(Ours)	**73.977**	**73.431**	**75.4**	**76.954**	**72.881**	**73.859**

Table 2: Human evaluation scores for accuracy and aesthetic quality across three tasks, showing DiagramGPT's consistently higher ratings compared to baseline models.

Captioning In line with prior works (Zala et al. 2023; Hong et al. 2018; Hinz, Heinrich, and Wermter 2020; Cho, Zala, and Bansal 2023b), we use captioning as an additional metric to assess diagram quality. Each generated diagram is automatically captioned using LLaVA 1.5 (Liu et al. 2023a), then compared with ground-truth captions using CIDEr (Vedantam, Zitnick, and Parikh 2014) and BERTScore (Zhang et al. 2020).

Baseline Models Our experiments cover three tasks, of which two—diagram restoration and diagram style transfer—are newly proposed in this work. To the best of our knowledge, no prior research has directly addressed these tasks, so we compare these tasks against a one-shot baseline using ChatGPT-4o-mini. For the text-to-diagram generation task, we compare our method with several existing state-of-the-art models: **Stable Diffusion v1.4** (Rombach et al. 2022), **VPGen** (Chiang et al. 2023), and **AutomaTikZ** (Belouadi, Lauscher, and Eger 2023). AutomaTikZ is specifically designed to generate TikZ code for creating various diagrams and flowcharts, while VPGen transforms structured data into visualized diagrams. We exclude DiagrammerGPT (Alashqar 2021) from our comparisons, as its code is not fully open-sourced; DiagrammerGPT employs GPT-based layout planning, integrated with GLIGEN for diagram generation. For Stable Diffusion v1.4, we experiment with both zero-shot and fine-tuned variants (fine-tuned on our Diagram-mxGraph dataset)to evaluate performance.

4.2 Implementation Details

Our framework builds on MiniCPM-V (Yao et al. 2024). We employ SigLIP (Zhai et al. 2023) as the ViT module, transforming each image into a sequence of 96 embeddings. The diagramAttention component also generates embeddings, and we use LLaMA 3.0-8B (Dubey et al. 2024) as the LLM. Initial weights of SigLIP, LLaMA, and diagramAttention are inherited from MiniCPM-V, while the gate network in FABS is randomly initialized. For CLIPScore, we use the CLIP variant clip-vit-large-patch14-336 (Radford et al. 2021).

4.3 Experimental Results

Quantitative Analysis We evaluated the performance of DiagramGPT and several baselines on three distinct tasks using comprehensive multiple metrics, including CLIP-Score, TaskScore, and Captioning accuracy. The results are summarized in Table 1.

In terms of CLIPScore, DiagramGPT achieved high scores, particularly for *Img-Img* similarity with 87.235,further indicating its effectiveness in producing diagrams that closely resemble ground-truth diagrams. DiagramGPT also performed well in *Img-Txt* similarity, scoring 27.716, second only to ChatGPT-4o-mini's 28.676.

For TaskScore, DiagramGPT demonstrated superior performance across all tasks. It achieved 95.298 on Task 1 (Restore Task), 58.821 on Task 2 (Text-to-Diagram), and 52.739 on Task 3 (Style Transfer). These scores highlight the model's capability in accurately transforming input into

Methods	CLIPScore(%) ↑		TaskScore(%) ↑				Captioning(%) ↑	
	Img-Txt	Img-Img	Task1	Task2	Task3	Overall	BERTScore	CIDEr
basemodel	27.054	86.898	95.780	58.047	52.190	66.854	88.99	4.54
basemodel+FABS	26.967	87.221	**95.809**	58.412	52.170	66.988	89.21	3.44
basemodel+CAAL	**27.805**	86.784	95.094	58.229	**52.887**	66.966	89.20	9.16
basemodel+FABS+CAAL	27.716	**87.235**	95.298	**58.821**	52.739	**67.188**	**89.26**	**9.56**

Table 3: Ablation study results examining the impact of FABS and CAAL modules on performance, indicating that both modules contribute positively to DiagramGPT's effectiveness in generating high-fidelity diagrams.

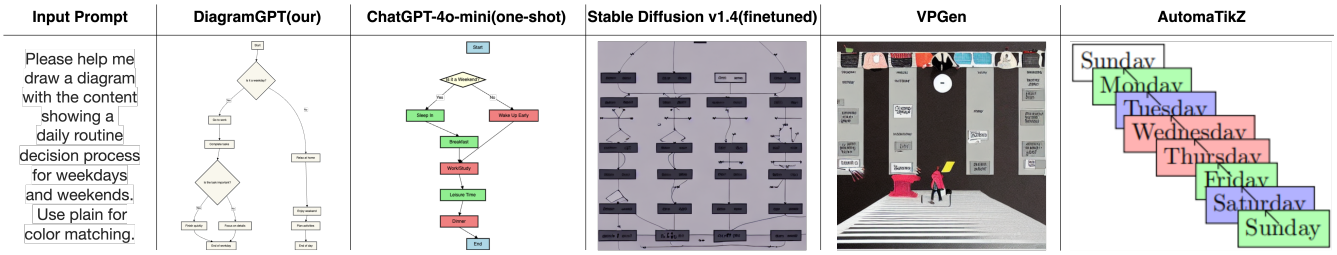

Figure 4: Comparison of generated diagrams across different models. The input prompt requests a daily routine decision process for weekdays and weekends, with a plain color scheme. DiagramGPT produces a clear and structured flowchart closely aligned with the prompt. Other models, including ChatGPT-4o-mini, Stable Diffusion v1.4, VPGen, and AutomaTikZ, show varying levels of adherence to the prompt, with differences in structure, clarity, and color scheme.

editable, semantically faithful diagrams. In addition, DiagramGPT significantly outperformed other models in Captioning with a score of 12.326, reflecting the semantic alignment of generated diagrams with the input prompts.

Qualitative Analysis A qualitative comparison of DiagramGPT with baseline models is shown in 4. DiagramGPT generated diagrams that are visually coherent, editable, and accurately reflect the provided prompts. For example, DiagramGPT's output captures the specific structure and layout as described in the prompt, while other models like Stable Diffusion and VPGen produced outputs that lacked coherence and did not fully adhere to the specified diagram structure. These comparisons underline DiagramGPT's advantage in both fidelity to the prompt and aesthetic quality.

Human Evaluation To further assess the quality of the generated diagrams, we conducted a human evaluation with 22 participants. Each participant rated the diagrams based on two criteria: (1) **Accuracy**, measuring how well the generated diagram represents the intended structure and content; and (2) **Aesthetic**, evaluating the visual appeal of the diagrams. Ratings were given on a scale from 0 to 100, where 0 indicates "strongly disagree" and 100 indicates "strongly agree." We randomly selected 30 samples from the mxGraph dataset, with 10 samples for each task.

As shown in Table 2, DiagramGPT outperformed baseline models in *accuracy* and *aesthetic* ratings for all tasks. For instance, DiagramGPT achieved an accuracy score of 73.977 and an aesthetic score of 73.431 on Task 1, significantly higher than other models. These results confirm that DiagramGPT produces diagrams that are both accurate and visually appealing,a strong choice for generating high-quality diagrams across different styles and content.

4.4 Ablation Study

To investigate the FABS and CAAL modules, we conducted an ablation study by removing each component from DiagramGPT.Table 3 presents the results of this analysis.

Removing the FABS module slightly decreased the *Img-Img* and *Overall* scores, which implies that FABS helps in differentiating background elements from foreground features, thus enhancing visual coherence. The removal of CAAL led to a more noticeable reduction in TaskScore for Task 3, further indicating that CAAL plays a vital role in ensuring precise component placement and maintaining style consistency across generated diagrams.

Combining FABS and CAAL yielded the best performance, achieving a TaskScore of 67.188 and a competitive Captioning CIDEr score of 9.56. These findings highlight that FABS and CAAL effectively complement each other, enabling DiagramGPT to produce precisely editable, high-quality diagrams that maintain both structural and aesthetic fidelity to the original input specifications.

5 Conclusion

We proposed a novel method for automated diagram generation, covering diagram restoration, text-to-diagram generation, and style transfer. Leveraging Fine-grained Adaptive Background Suppression (FABS) and Component-Aware Adaptive Loss (CAAL), our approach achieves SOTA results on metrics like CLIPScore, TaskScore, and captioning accuracy. Extensive experiments on the mxGraph dataset show that DiagramGPT generates high-quality, editable diagrams aligned with both text and visual inputs, significantly outperforming existing models and setting a strong baseline for future research directions and applications.

Acknowledgments

This research was supported by The National Nature Science Foundation of China (Grant Nos: 62402417, 62273301, 62273302), in part by "Pioneer" and "Leading Goose" R&D Program of Zhejiang (Grant No. 2025C02026), in part by the Key R&D Program of Ningbo (Grant Nos: 2024Z115, 2025Z035), in part by Yongjiang Talent Introduction Programme (Grant No: 2023A-197-G).

References

Alashqar, A. M. 2021. Automatic generation of uml diagrams from scenario-based user requirements. *Jordanian Journal of Computers and Information Technology*, 7(2).

Belouadi, J.; Lauscher, A.; and Eger, S. 2023. Automatikz: Text-guided synthesis of scientific vector graphics with tikz. *arXiv preprint arXiv:2310.00367*.

Ben Abdessalem Karaa, W.; Ben Azzouz, Z.; Singh, A.; Dey, N.; S. Ashour, A.; and Ben Ghazala, H. 2016. Automatic builder of class diagram (ABCD): an application of UML generation from functional requirements. *Software: Practice and Experience*, 46(11): 1443–1458.

Chang, H.; Zhang, H.; Barber, J.; Maschinot, A.; Lezama, J.; Jiang, L.; Yang, M.-H.; Murphy, K.; Freeman, W. T.; Rubinstein, M.; Li, Y.; and Krishnan, D. 2023. Muse: Text-To-Image Generation via Masked Generative Transformers. In *ICML*.

Cheema, S. M.; Tariq, S.; and Pires, I. M. 2023. A natural language interface for automatic generation of data flow diagram using web extraction techniques. *Journal of King Saud University-Computer and Information Sciences*, 35(2): 626–640.

Chiang, W.-L.; Li, Z.; Lin, Z.; Sheng, Y.; Wu, Z.; Zhang, H.; Zheng, L.; Zhuang, S.; Zhuang, Y.; Gonzalez, J. E.; et al. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality. *See https://vicuna. lmsys. org (accessed 14 April 2023)*, 2(3): 6.

Cho, J.; Zala, A.; and Bansal, M. 2023a. Dall-eval: Probing the reasoning skills and social biases of text-to-image generation models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 3043–3054.

Cho, J.; Zala, A.; and Bansal, M. 2023b. DALL-Eval: Probing the Reasoning Skills and Social Biases of Text-to-Image Generation Models. In *ICCV*.

Conrardy, A.; and Cabot, J. 2024. From Image to UML: First Results of Image Based UML Diagram Generation Using LLMs. *arXiv preprint arXiv:2404.11376*.

Dai, X.; Hou, J.; Ma, C.-Y.; Tsai, S.; Wang, J.; Wang, R.; Zhang, P.; Vandenhende, S.; Wang, X.; Dubey, A.; Yu, M.; Kadian, A.; Radenovic, F.; Mahajan, D.; Li, K.; Zhao, Y.; Petrovic, V.; Singh, M. K.; Motwani, S.; Wen, Y.; Song, Y.; Sumbaly, R.; Ramanathan, V.; He, Z.; Vajda, P.; and Parikh, D. 2023. Emu: Enhancing Image Generation Models Using Photogenic Needles in a Haystack. arXiv:2309.15807.

Dosovitskiy, A. 2020. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

Dubey, A.; Jauhri, A.; Pandey, A.; Kadian, A.; Al-Dahle, A.; Letman, A.; Mathur, A.; Schelten, A.; Yang, A.; Fan, A.; et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Ghosh, S.; Mukherjee, P.; Chakraborty, B.; and Bashar, R. 2018. Automated generation of er diagram from a given text in natural language. In *2018 International Conference on Machine Learning and Data Engineering (iCMLDE)*, 91–96. IEEE.

Golovchinsky, G.; Kamps, T.; and Reichenberger, K. 1995. Subverting structure: Data-driven diagram generation. In *Proceedings Visualization'95*, 217–223. IEEE.

Gulia, S.; and Choudhury, T. 2016. An efficient automated design to generate UML diagram from Natural Language Specifications. In *2016 6th international conference-cloud system and big data engineering (Confluence)*, 641–648. IEEE.

Herwanto, G. B. 2024. Automating Data Flow Diagram Generation from User Stories Using Large Language Models. In *7th Workshop on Natural Language Processing for Requirements Engineering*.

Hinz, T.; Heinrich, S.; and Wermter, S. 2020. Semantic Object Accuracy for Generative Text-to-Image Synthesis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1–1.

Hong, S.; Yang, D.; Choi, J.; and Lee, H. 2018. Inferring Semantic Layout for Hierarchical Text-to-Image Synthesis. In *CVPR*.

Krishnan, H.; and Samuel, P. 2010. Relative Extraction Methodology for class diagram generation using dependency graph. In *2010 International conference on communication control and computing technologies*, 815–820. IEEE.

Lin, B.; Tang, Z.; Ye, Y.; Cui, J.; Zhu, B.; Jin, P.; Zhang, J.; Ning, M.; and Yuan, L. 2024. Moe-llava: Mixture of experts for large vision-language models. *arXiv preprint arXiv:2401.15947*.

Lin, H.; Cheng, X.; Wu, X.; and Shen, D. 2022. Cat: Cross attention in vision transformer. In *2022 IEEE international conference on multimedia and expo (ICME)*, 1–6. IEEE.

Liu, H.; Li, C.; Li, Y.; and Lee, Y. J. 2023a. Improved Baselines with Visual Instruction Tuning.

Liu, H.; Li, C.; Wu, Q.; and Lee, Y. J. 2023b. Visual Instruction Tuning.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, 10012–10022.

Narawita, C. R.; et al. 2017. UML generator-use case and class diagram generation from text requirements. *The International Journal on Advances in ICT for Emerging Regions*, 10(1).

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Ramesh, A.; Dhariwal, P.; Nichol, A.; Chu, C.; and Chen, M. 2022. Hierarchical Text-Conditional Image Generation with CLIP Latents. arXiv:2204.06125.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2021. High-Resolution Image Synthesis with Latent Diffusion Models. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 10674–10685.

Rombach, R.; Blattmann, A.; Lorenz, D.; Esser, P.; and Ommer, B. 2022. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 10684–10695.

Saharia, C.; Chan, W.; Saxena, S.; Li, L.; Whang, J.; Denton, E. L.; Ghasemipour, K.; Gontijo Lopes, R.; Karagol Ayan, B.; Salimans, T.; et al. 2022. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35: 36479–36494.

Vedantam, R.; Zitnick, C. L.; and Parikh, D. 2014. CIDEr: Consensus-based image description evaluation. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 4566–4575.

Yang, S.; and Sahraoui, H. 2022. Towards automatically extracting UML class diagrams from natural language specifications. In *Proceedings of the 25th International Conference on Model Driven Engineering Languages and Systems: Companion Proceedings*, 396–403.

Yao, Y.; Yu, T.; Zhang, A.; Wang, C.; Cui, J.; Zhu, H.; Cai, T.; Li, H.; Zhao, W.; He, Z.; et al. 2024. MiniCPM-V: A GPT-4V Level MLLM on Your Phone. *arXiv preprint arXiv:2408.01800*.

Yu, J.; Xu, Y.; Koh, J. Y.; Luong, T.; Baid, G.; Wang, Z.; Vasudevan, V.; Ku, A.; Yang, Y.; Ayan, B. K.; Hutchinson, B.; Han, W.; Parekh, Z.; Li, X.; Zhang, H.; Baldridge, J.; and Wu, Y. 2022. Scaling Autoregressive Models for Content-Rich Text-to-Image Generation. *Transactions on Machine Learning Research*.

Zala, A.; Lin, H.; Cho, J.; and Bansal, M. 2023. DiagrammerGPT: Generating Open-Domain, Open-Platform Diagrams via LLM Planning. *arXiv preprint arXiv:2310.12128*.

Zhai, X.; Mustafa, B.; Kolesnikov, A.; and Beyer, L. 2023. Sigmoid loss for language image pre-training. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 11975–11986.

Zhang, T.; Kishore, V.; Wu, F.; Weinberger, K. Q.; and Artzi, Y. 2020. BERTScore: Evaluating Text Generation with BERT. In *Proceedings of the International Conference on Learning Representations*.

Zhou, X.; and Zhou, N. 2004. Auto-generation of class diagram from free-text functional specifications and domain ontology. *Artificial Intelligence*, 26.

Zhu, R.; Li, W.; and Jin, C. 2023. TAG: UML Activity Diagram Deeply Supervised Generation from Business Textural Specification. In *2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER)*, 956–961. IEEE.