CAL-RAG: Retrieval-Augmented Multi-Agent Generation for Content-Aware Layout Design

Najmeh Forouzandehmehr, Reza Yousefi Maragheh, Sriram Kollipara, Kai Zhao, Topojoy Biswas, Evren Korpeoglu, Kannan Achan
Walmart Global Tech
Sunnyvale, CA, USA
{najmeh.forouzandehmehr,reza.yousefimaraghehr,sriram.kollipara,kai.zhao,topojoy.biswas,EKorpeoglu,kannan.achan}@walmart.com

Abstract

Automated content-aware layout generation—the task of arranging visual elements such as text, logos, and underlays on a background canvas—remains a fundamental yet underexplored problem in intelligent design systems. While recent advances in deep generative models and large language models (LLMs) have shown promise in structured content generation, most existing approaches lack grounding in contextual design exemplars and fall short in handling semantic alignment and visual coherence. In this work, we introduce CAL-RAG, a Retrieval-Augmented, Agentic framework for content-aware layout generation that integrates multimodal retrieval, large language models, and collaborative agentic reasoning. Our system retrieves relevant layout examples from a structured knowledge base and invokes an LLM-based layout recommender to propose structured element placements. A vision-language grader agent evaluates the layout based on visual metrics, and a feedback agent provides targeted refinements, enabling iterative improvement. We implement our framework using LangGraph and evaluate on the PKU PosterLayout dataset, a benchmark rich in semantic and structural variability. CAL-RAG achieves state-of-the-art performance across multiple layout metrics—including underlay effectiveness, element alignment, and overlap—substantially outperforming strong baselines such as LayoutPrompter. Our results demonstrate that combining retrieval augmentation with agentic multi-step reasoning provides a scalable, interpretable, and high-fidelity solution for automated layout generation.

Keywords

Content-aware layout generation,Retrieval-Augmented Generation (RAG), Multi-agent systems, Large Language Models (LLMs), Vision-Language Models (VLMs), Creative AI

1 Introduction

Designing content-aware visual layouts is a fundamental challenge in computational creativity, with applications spanning digital advertising, web interfaces, and e-commerce presentation. The task involves arranging heterogeneous visual elements—such as text, logos, and underlays—on a canvas in a way that balances aesthetic principles with semantic intent. Despite the practical importance of this task, existing methods often fall short in handling the combinatorial, content-sensitive nature of layout generation.

Traditional approaches either relied on rule-based heuristics or framed layout design as an optimization problem over handcrafted alignment and spacing constraints [18]. These methods offer limited generalization, particularly in diverse or semantically rich design contexts. The advent of deep generative models—such as GANs, VAEs, and transformers—has enabled more expressive layout synthesis by learning from annotated layout corpora [1, 2, 7, 9, 10]. However, these models are typically data-hungry, brittle to out-of-distribution inputs, and often struggle to incorporate visual-semantic alignment at inference time.

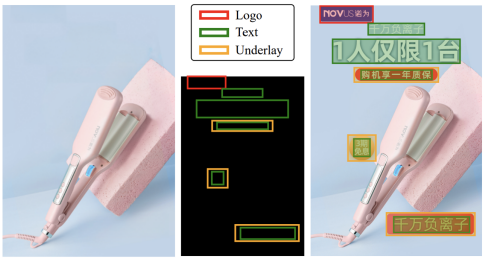

Figure 1: Example for content-aware layout generation: (a) Input canvas with background and content elements; (b) Generated layout based on visual and textual content awareness; (c) Final rendered presentation using the layout from (b).

More recently, Large Language Models (LLMs) and Large Vision-Language Models (LVLMs) have demonstrated remarkable zero-shot capabilities in generating structured outputs, such as HTML or JSON representations of layouts, conditioned on multimodal inputs [4, 11, 13, 17]. Yet, most LLM-driven methods rely on prompt engineering alone, lacking access to contextual design knowledge or external grounding signals. Retrieval-Augmented Generation (RAG) presents a promising solution by coupling generation with example-driven retrieval to enhance factuality and contextual alignment [3, 5, 7]. A notable prior work in this direction is the Retrieval-Augmented Layout Transformer (RALF), which retrieves nearest neighbor layout examples based on an input image and feeds these results into an autoregressive generator, achieving state-of-the-art performance in content-aware layout generation [7].

In this work, we propose CAL-RAG, a novel framework that integrates RAG with a collaborative, multi-agent system [6, 14–16] for content-aware layout generation. Our architecture is built upon three specialized agents: a layout recommender powered by an LLM, a vision-language grader agent, and a feedback agent that supports iterative refinement. The system is instantiated using LangGraph, and is trained and evaluated on the PKU PosterLayout dataset [8], a challenging benchmark containing semantically diverse poster designs.

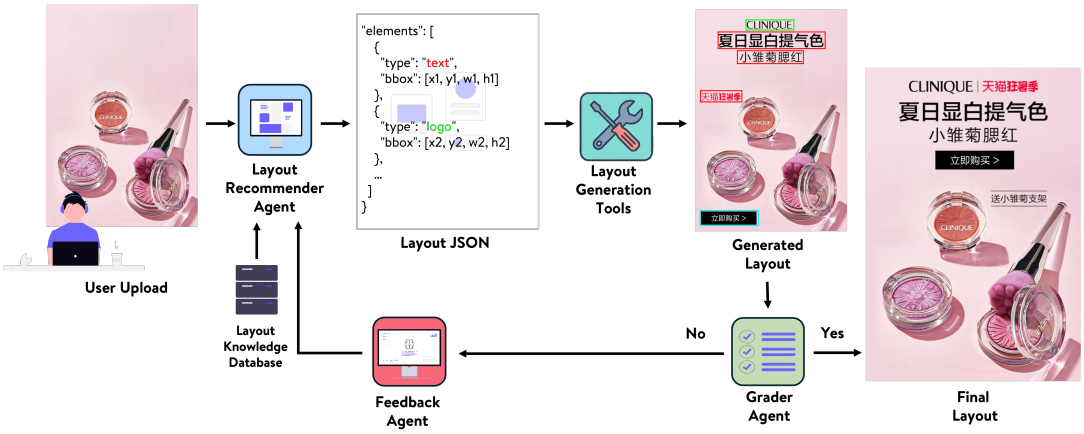

Figure 2: System Architecture Diagram of CAL-RAG.

Unlike prior work, CAL-RAG grounds layout decisions in a retrieval corpus of design exemplars, enabling better inductive bias and interpretability. The layout generator reasons over retrieved samples to produce structured layout hypotheses; these are then rigorously scored by the grader agent using geometric and visual coherence metrics. If a layout is rejected, the feedback agent intervenes with localized corrections, prompting the recommender for an updated design. Through this agentic iteration loop, CAL-RAG achieves state-of-the-art performance across multiple layout quality metrics, including underlay effectiveness, alignment, and element separation. Our contributions are threefold:

- We present a novel agentic RAG framework for layout generation that unifies retrieval, generation, evaluation, and revision within a single compositional system.
- We introduce a principled evaluation protocol based on geometric layout metrics and visual-semantic alignment on a challenging public benchmark.
- We demonstrate significant empirical gains over strong baselines, including LayoutPrompter and RALF, through extensive ablation and comparative analysis.

The core problem we tackle is: Given a background canvas image, how can we generate a layout that specifies the number, type, and position of content elements (text, logo, underlay) in a way that balances semantic coherence, aesthetic principles, inspired by prior design patterns. The input to our system is a background image, from which the type and number of content elements (e.g., text, logos, underlays) are automatically inferred. The output is a structured layout proposal, defined as a set of bounding boxes with associated element types.

2 Methodology

Our approach to content-aware layout generation (CAL-RAG) leverages a multi-agent system integrated with a Retrieval-Augmented Generation (RAG) framework. The overall system architecture comprises four key components: a Layout Recommender Agent, a Layout Generation Tool, a Grader Agent, and a Feedback Agent (see Figure. 2). This framework facilitates an iterative process of layout creation and refinement, aiming to produce high-quality, content-aware visual-textual presentations.

2.1 Layout Recommender Agent

The process begins with the **Layout Recommender Agent**, which suggests an initial layout L for a given background image I_{bg}. This agent uses the *PKU PosterLayout* dataset as its layout knowledge base. This dataset consists of pairs of background image and ground truth layouts. Let us denote the set of these pairs by: $\mathcal{D} = \{(I_j, L_j), for\ j \in \mathcal{N}\}$, when $\mathcal{N}$ is set of indexes. After retrieving k similar examples from $\mathcal{D}$, the layout recommender uses them as few-shot references to guide an unconstrained generation process. Given these examples, it predicts the number of visual elements (e.g., logos, text blocks, overlays), their types, and the corresponding bounding boxes—producing a layout tailored to the input canvas. An example of the unconstrained layout generation task using the PKU dataset is shown in Figure 1.

1. Embedding and Retrieval. We assume each background image I can be transformed into a latent vector (embedding) via an embedding function: $\mathbf{E}(I) \in \mathbb{R}^d$. We use the CLIP image encoder for for $\mathbf{E}(.)$ [12]. Given an input background image I_{bg}, the agent computes its embedding $\mathbf{E}(I_{\text{bg}})$. For each candidate layout-image pair (I_i, L_i) in the knowledge base, it also has a corresponding image embedding $\mathbf{E}(I_i)$. The retrieval step is conducted by a cosine similarity score S between the background image and a reference image in the dataset:

$$S(I_{\text{bg}}, I_i) = \frac{\mathbf{E}(I_{\text{bg}}) \cdot \mathbf{E}(I_i)}{\|\mathbf{E}(I_{\text{bg}})\| \, \|\mathbf{E}(I_i)\|}.$$

CAL-RAG then selects set the top-k most similar reference layouts:

$$\mathcal{R} = \left\{(I_i, L_i); i \in \text{argtop}_k\{S(I_{\text{bg}}, I_m)\}_{m \in \mathcal{N}}\right\}.$$

Set $\mathcal{R}$ acts as source for "best practices" of bounding-box positioning, number of elements, and arrangement for the layout instance at hand.

2. Bounding Box Estimation. Let the agent decide on n elements to be placed in the poster layout, each denoted as e_j for $j = 1, \ldots, n$.

Each element e_j (e.g., text, logo, underlay) is represented by a bounding box:

$$B_j = \big(x_j,\ y_j,\ w_j,\ h_j\big),$$

where $(x_j,\ y_j)$ is the top-left coordinate of the bounding box and $(w_j,\ h_j)$ are the width and height, respectively.

The placement of these n bounding boxes constitutes a layout $L = \{B_1, B_2, \ldots, B_n\}$. To incorporate design principles such as balance and spatial harmony, the agent follows structured instructions and uses features from retrieved layouts $\mathcal{R}$. For example, it might minimize an overall *layout cost* $C(L)$ that accounts for alignment, overlap, and aesthetic metrics:

$$C(L) = \alpha_1 C_{\text{overlap}}(L) + \alpha_2 C_{\text{alignment}}(L) + \alpha_3 C_{\text{margins}}(L),$$

where $C_{\text{overlap}}(L)$ measures the total unwanted overlap among bounding boxes, $C_{\text{alignment}}(L)$ measures alignment deviations among box edges (e.g., difference in x-coordinates for left alignment), $C_{\text{margins}}(L)$ captures violations of spacing constraints from the design guidelines, $\alpha_1, \alpha_2, \alpha_3$ are weighting hyperparameters reflecting the importance of each principle.

Hence, the Layout Recommender Agent seeks:

$$L^* \ = \ \operatorname*{argmin}_{L} C(L),$$

subject to any constraints derived from the retrieved examples (e.g., recommended number of elements, recommended bounding box aspect ratios).

3. Structured Instructions and Output. To ensure reproducibility and clarity, the agent is given *structured instructions* that define how to: **Analyze** the retrieved layouts $\mathcal{R}$ and compare them with the test canvas I_{bg}; **Determine** the number of elements n and their types (e.g., text, logo); **Assign** bounding boxes $B_1, \ldots, B_n$ while adhering to spatial alignment and design heuristics.

2.2 Layout Generation Tool

After the **Layout Recommender Agent** produces its JSON output, the recommended bounding boxes $\{B_1, \ldots, B_n\}$ and associated element types (e.g., text, logo, underlay) are passed to the Layout Generation Tool. Let the background image be denoted by I_{bg}. The goal of this tool is to physically realize the abstract layout $L = \{B_j : j = 1, \ldots, n\}$ by compositing the specified elements onto I_{bg} according to their bounding box coordinates.

1. Compositing Framework. We model each visual element as a function or image C_j (e.g., text rendered as a small image or graphic, a logo image, etc.). Thus, the Layout Generation Tool aims to produce a final composed image I_{comp} by placing each C_j into the bounding box $B_j = (x_j,\ y_j,\ w_j,\ h_j)$ on the canvas of size (W, H). Formally, we define a compositing function

$$\mathcal{F}\big(I_{\text{bg}}, \{(C_j, B_j)\}_{j=1}^{n}\big) \ = \ I_{\text{comp}},$$

where $\mathcal{F}$ positions each element C_j in B_j and at location (x_j, y_j), scales it if necessary to fit width w_j and height h_j, and overlays or blends it (depending on the element type) onto I_{bg}.

2. Coordinate Transform for Placing Elements. For each element C_j, which may have an intrinsic resolution or size, we define a coordinate transform Φ_j that maps the local coordinates of C_j to the global coordinates of I_{bg} according to B_j. If C_j has a native size $(w_j^0,\ h_j^0)$, then placing it into the bounding box $B_j = (x_j, y_j, w_j, h_j)$ involves a scaling factor $(w_j/w_j^0, h_j/h_j^0)$.

3. Overlay and Blending Operations. To combine C_j with the background, we can use a simple overwrite or alpha-blending operation. For example, a standard "over" operator from alpha compositing can be defined for each pixel $\mathbf{q}$ in I_{bg}:

$$I_{\text{comp}}(\mathbf{q}) = \ (1 - \alpha_j(\mathbf{q}))\, I_{\text{bg}}(\mathbf{q}) \ + \ \alpha_j(\mathbf{q})\, C_j\big(\Phi_j^{-1}(\mathbf{q})\big),$$

where $\alpha_j(\mathbf{q}) \in [0, 1]$ is the opacity of element C_j at pixel $\mathbf{q}$. If C_j contains no transparency, $\alpha_j(\mathbf{q}) = 1$ for all valid $\mathbf{q}$ in the bounding box, effectively overwriting the background in that region.

4. Concrete Implementation. For n elements, the compositing function $\mathcal{F}$ applies the above placement and blending steps in a chosen order (e.g., from back to front). Hence, the final composition can be written as a recursive or sequential update:

$$I_{\text{comp}}^{(j)} \ = \ \text{Blend}\Big(I_{\text{comp}}^{(j-1)},\ C_j,\ B_j\Big),$$

The Layout Generation Tool transforms the abstract layout recommendations into a concrete visual representation that can be subsequently evaluated or refined by other agents in the system.

2.3 Grader Agent

Once the layout L is generated by the Layout Generation Tool, it is evaluated by the **Grader Agent**, which acts as a graphic design expert. This agent measures the quality and acceptability of L based on a set of predefined visual metrics. Let these metrics be represented by the function

$$\Gamma(L) \ = \ \big(\gamma_1(L),\ \gamma_2(L),\ \gamma_3(L),\ \gamma_4(L)\big),$$

where each component $\gamma_k(L)$ corresponds to a specific criterion:

Consistency and Cohesion ($\gamma_1(L)$):

$$\gamma_1(L) \ = \ \exp\big(-\sigma_{\text{colors}}(L)\big),$$

where $\sigma_{\text{colors}}(L)$ measures the standard deviation of dominant colors across all placed elements (lower deviation implies higher cohesion). Hence, $\gamma_1(L) \in (0, 1]$ increases as color usage becomes more harmonious.

Composition and Spacing ($\gamma_2(L)$):

$$\gamma_2(L) \ = \ 1 - \frac{\text{OverlapArea}(L)}{\text{TotalElementArea}(L)},$$

where $\text{OverlapArea}(L)$ sums the pairwise overlapping regions of bounding boxes, and $\text{TotalElementArea}(L)$ is the combined area of all bounding boxes (without subtracting overlaps). A well-spaced layout has minimal overlap, making $\gamma_2(L)$ closer to 1.

Product Visibility and Clarity ($\gamma_3(L)$):

$$\gamma_3(L) \ = \ 1 - \frac{\sum_{j=1}^{n} \mathbb{I}\{\text{Occluded}(B_j)\}}{n},$$

where $\mathbb{I}\{\text{Occluded}(B_j)\}$ is an indicator function that is 1 if bounding box B_j is significantly occluded (e.g., by another element) and 0 otherwise, and n is the total number of elements. Higher clarity yields fewer occlusions, pushing $\gamma_3(L)$ toward 1.

Let $\mathbf{t} = (t_1, t_2, t_3)$ be threshold values for these metrics. The Grader Agent outputs:

$$\text{Decision}(L) = \begin{cases} \texttt{Accept}, & \text{if } \gamma_k(L) \geq t_k \ \ \forall k = 1, 2, 3, \\ \texttt{Reject}, & \text{otherwise.} \end{cases}$$

Thus, the layout is `Accepted` only if it meets or exceeds *all* metric thresholds; otherwise, the Grader Agent outputs `Reject`.

2.4 Feedback Agent

If the **Grader Agent** rejects the generated layout L, the system invokes the **Feedback Agent** for targeted improvements. Acting as a graphic design expert, this agent analyzes the rejected layout to identify specific areas for refinement. In particular, it focuses on:

- **Placement** and **alignment** of elements for balanced composition,
- **Cohesiveness** and **proportions** of bounding boxes,
- Maintaining designated empty space in a certain canvas region $\Omega \subset \text{Canvas}$,
- Avoiding comments on *color*, *background*, or *item selection*.

Formally, let $\Phi(L)$ denote the *feedback function*, which encodes the agent's critique of L and prescribes modifications. That is,

$$\Phi(L) = \big(\delta_1(L), \delta_2(L), \ldots, \delta_n(L)\big),$$

where each $\delta_j(L)$ provides a corrective shift (e.g., in position, size, or alignment) for the j-th bounding box of L. Adhering to the requirement that Ω remains clear, any suggested change for $B_j \in L$ must satisfy the constraint $B_j \cap \Omega = \varnothing$, if B_j should not intrude on the empty space.

Once $\Phi(L)$ is computed, the layout is refined by the **Layout Recommender Agent**:

$$L^{(t+1)} = \text{Refine}\Big(L^{(t)}, \Phi\big(L^{(t)}\big)\Big),$$

where $L^{(t)}$ is the layout at iteration t and $\Phi\big(L^{(t)}\big)$ contains the feedback from the *Feedback Agent* regarding placement and alignment. This iterative process repeats until the **Grader Agent** finally accepts the layout or a maximum number of iterations is reached.

3 Experiments

3.1 Dataset and Evaluation Metrics

The RAG database and the evaluation of the proposed model are based on the PKU PosterLayout dataset, a new benchmark specifically designed for content-aware visual-textual presentation layout [8]. This dataset comprises 9,974 poster-layout pairs and 905 image canvases with annotations defining three primary element types: text, logo, and underlay User Query. Following the experimental framework of RALF, the dataset is split into training (7,735), validation (1,000), and testing (1,000) sets . For easier utilization, the dataset provides original posters, inpainted posters (where visual elements are removed), image canvases, and saliency maps.

The performance of the proposed model is evaluated using several layout quality metrics, including *underlay effectiveness* (both loose and strict), *overlay*, and *alignment* [11].

- **Underlay Effectiveness** measures how well the generated underlay elements support and decorate non-underlay content, such as text or logos. A predicted underlay is considered valid if it effectively overlaps with at least one non-underlay element. The loose variant (Und_ℓ) rewards underlays that significantly intersect with non-underlay elements, taking the highest degree of overlap as the score. The strict variant (Und_s) is more conservative: it only gives credit if a non-underlay element is entirely contained within an underlay. Und_s is then computed as the average score across all underlays.
- **Overlay** measures the average area of undesired overlap between non-underlay elements (e.g., text blocks, logos). It is computed as the ratio of the total overlapping area to the total layout area; lower values indicate better element separation and visual clarity.
- **Alignment** assesses how well elements are spatially aligned along common axes (e.g., left/right edges, centers). Misalignment is calculated as the average deviation between corresponding edge coordinates of adjacent elements. Lower scores indicate stronger geometric regularity and design coherence.

3.2 Overall Results

We compare the following methods in the experiments. CGL-GAN [19] is a non-autoregressive encoder–decoder model employing a Transformer architecture. The model takes in an empty layout or layout constraints to decode a layout representation. DS-GAN [18] is a non-autoregressive model that utilizes a CNN-LSTM architecture [8] LayoutPrompter, is a training-free method that leverages LLM prompting and in-context learning to tackle various layout generation tasks [11].

Table 1: Performance Comparison

Methods	Ove ↓	Ali ↓	Und_l ↑	Und_s ↑
Ground-Truth	0.0001	0.0002	0.9965	0.9912
Geometric Methods				
CGL-GAN	0.0605	0.0062	0.8624	0.4043
DS-GAN	0.0220	0.0046	0.8315	0.4320
LayoutPrompter	0.0036	0.0036	0.8986	0.8802
CAL-RAG(Ours)	**0.0023**	**0.002**	**1.0000**	**1.0000**

As shown in Table 1, CAL-RAG achieves state-of-the-art performance across all metrics, significantly outperforming LayoutPrompter and previous generation-based methods such as CGL-GAN and DS-GAN. We conducted evaluations following the PosterLayout setting [8] and included previously reported results for comparison. Notably, CAL-RAG attains perfect underlay effectiveness (1.0000) for both large and small elements, indicating a strong alignment between generated layouts and the intended spatial semantics of content elements. Moreover, our method achieves the lowest overlap (0.0023) and alignment (0.002) among all models, closely approaching the ground-truth layout statistics, demonstrating the advantage of incorporating agentic iteration and retrieval-based grounding. These results validate the effectiveness of our retrieval-augmented, multi-agent design in generating layouts that are not

Table 2: Ablation Results

Configuration	Ove ↓	Ali ↓	Und_l ↑	Und_s ↑
Layout Recommender only	0.00341	0.0031	0.891	0.8
+ Grader Agent	0.0029	0.0029	0.98	0.967
+ Feedback Agent	**0.0023**	**0.002**	**1**	**1**

only visually coherent but also semantically structured, setting a new benchmark for content-aware layout generation.

3.3 Ablation Study

To isolate the contributions of each component in the CAL-RAG architecture, we perform an ablation study by evaluating three progressively enhanced configurations: (1) Layout Recommender only, (2) Layout Recommender + Grader Agent, and (3) the full system with Grader and Feedback Agents. As shown in Table 2, the base configuration, Layout Recommender Agent achieves modest performance. This suggests that while retrieval-based generation provides a useful prior, it lacks the self-corrective mechanisms necessary for fine-grained layout optimization.

Introducing the Grader Agent significantly boosts performance, particularly in underlay effectiveness, confirming that the grader effectively filters suboptimal layouts based on learned visual metrics. When the Feedback Agent is incorporated, the system achieves state-of-the-art scores across all metrics, including perfect alignment with semantic expectations (1.0000) and minimal overlap (0.0023). This demonstrates the crucial role of agentic feedback in iteratively refining layouts, validating our core hypothesis that a multi-agent loop of generation, evaluation, and correction yields superior results over single-pass generation methods.

References

[1] D. M. Arroyo, J. Postels, and F. Tombari. 2021. Variational Transformer Networks for Layout Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 13642–13652.

[2] S. Chai, L. Zhuang, and F. Yan. 2023. LayoutDM: Transformer-Based Diffusion Model for Layout Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 18349–18358.

[3] J. Chen, H. Lin, X. Han, and L. Sun. 2024. Benchmarking Large Language Models in Retrieval-Augmented Generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 17754–17762.

[4] W. Feng, W. Zhu, T. J. Fu, V. Jampani, A. Akula, X. He, S. Basu, X. E. Wang, and W. Y. Wang. 2023. LayoutGPT: Compositional visual planning and generation with large language models. In *Advances in Neural Information Processing Systems*, Vol. 36. 18225–18250.

[5] Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv preprint arXiv:2312.10997* 2 (18 Dec 2023).

[6] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. 2023. CRITIC: Large Language Models Can Self-Correct with Tool-Interactive Critiquing. *arXiv preprint* arXiv:2305.11738 (May 2023). doi:10.48550/arXiv.2305.11738

[7] D. Horita, N. Inoue, K. Kikuchi, K. Yamaguchi, and K. Aizawa. 2024. Retrieval-Augmented Layout Transformer for Content-Aware Layout Generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 67–76.

[8] H. Y. Hsu, X. He, Y. Peng, H. Kong, and Q. Zhang. 2023. PosterLayout: A New Benchmark and Approach for Content-Aware Visual-Textual Presentation Layout. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6018–6026.

[9] A. A. Jyothi, T. Durand, J. He, L. Sigal, and G. Mori. 2019. LayoutVAE: Stochastic Scene Layout Generation from a Label Set. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 9895–9904.

[10] J. Li, J. Yang, A. Hertzmann, J. Zhang, and T. Xu. 2019. LayoutGAN: Generating graphic layouts with wireframe discriminators. *arXiv preprint* arXiv:1901.06767 (Jan 21 2019).

[11] J. Lin, J. Guo, S. Sun, Z. Yang, J. G. Lou, and D. Zhang. 2023. LayoutPrompter: Awaken the design ability of large language models. In *Advances in Neural Information Processing Systems*, Vol. 36. 43852–43879.

[12] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, and Gretchen Krueger. 2021. Learning Transferable Visual Models from Natural Language Supervision. In *Proceedings of the 38th International Conference on Machine Learning (ICML)*. PMLR, 8748–8763.

[13] J. Seol, S. Kim, and J. Yoo. 2024. PosterLLAMA: Bridging design ability of language model to contents-aware layout generation. *arXiv preprint* arXiv:2404.00995 (Apr 1 2024).

[14] Aditi Singh, Abul Ehtesham, Saket Kumar, and Tala Talaei Khoei. 2025. Agentic Retrieval-Augmented Generation: A Survey on Agentic RAG. *arXiv preprint* arXiv:2501.09136 (January 2025). doi:10.48550/arXiv.2501.09136

[15] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. 2023. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation. *arXiv preprint* arXiv:2308.08155 (August 2023). doi:10.48550/arXiv.2308.08155

[16] Shi-Qi Yan, Jia-Chen Gu, Yun Zhu, and Zhen-Hua Ling. 2024. Corrective Retrieval Augmented Generation. arXiv:2401.15884 [cs.CL] https://arxiv.org/abs/2401.15884

[17] T. Yang, Y. Luo, Z. Qi, Y. Wu, Y. Shan, and C. W. Chen. 2024. PosterLLaVA: Constructing a Unified Multi-Modal Layout Generator with LLM. *arXiv preprint arXiv:2406.02884* (5 Jun 2024).

[18] J. Zhang, R. Yoshihashi, S. Kitada, A. Osanai, and Y. Nakashima. 2024. VASCAR: Content-Aware Layout Generation via Visual-Aware Self-Correction. *arXiv preprint arXiv:2412.04237* (5 Dec 2024).

[19] M. Zhou, C. Xu, Y. Ma, T. Ge, Y. Jiang, and W. Xu. 2022. Composition-Aware Graphic Layout GAN for Visual-Textual Presentation Designs. *arXiv preprint arXiv:2205.00303* (30 Apr 2022).