TextLap: Customizing Language Models for Text-to-Layout Planning

Jian Chen[1]*, Ruiyi Zhang[2]†, Yufan Zhou[2], Jennifer Healey[2], Jiuxiang Gu[2], Zhiqiang Xu[3], Changyou Chen[1]

[1]University at Buffalo [2]Adobe Research [3]MBZUAI

Abstract

Automatic generation of graphical layouts is crucial for many real-world applications, including designing posters, flyers, advertisements, and graphical user interfaces. Given the incredible ability of Large language models (LLMs) in both natural language understanding and generation, we believe that we could customize an LLM to help people create compelling graphical layouts starting with only text instructions from the user. We call our method TextLap (text-based layout planning) [1]. It uses a curated instruction-based layout planning dataset (InsLap) to customize LLMs as a graphic designer. We demonstrate the effectiveness of TextLap and show that it outperforms strong baselines, including GPT-4 based methods, for image generation and graphical design benchmarks.

1 Introduction

Traditional layout generation tasks typically involve organizing text and graphical elements into a pleasing and professional looking arrangement. This is often time-consuming and difficult for users, especially those without training or design skill. Our goal is to condition an LLM so that it allows users to generate professional looking layouts by simply inputting text instructions. We are inspired by the rapid advancement of generative models for text (Ouyang et al., 2022; Achiam et al., 2023), images (Betker et al., 2023; Podell et al., 2023), and videos (Peebles and Xie, 2023) and recent works that create layouts using graphical elements such as category, position, or size as input (Li et al., 2020; Kikuchi et al., 2021; Jyothi et al., 2019; Inoue et al., 2023). We advance the field by customizing LLMs as text-guided layout generation models that require only a text description of the desired layout and optionally text descriptions for visual elements as input, enabling layout planning solely in the text modality. We believe that building an LLM-based layout generation offers a more user-friendly approach to achieving desired designs, allowing text instructions to guide the process—an aspect that is challenging to incorporate in traditional settings.

Creating 2D graphical layouts from text alone is challenging for several reasons. First, text descriptions of 2D arrangements are usually underspecified. There are multiple ways to describe a 2D layout, and there are multiple 2D layouts that can meet a description. Furthermore, comprehending 2D spatial relations (Xu et al., 2020) is challenging for LLMs due to their autoregressive nature and 1D positional embedding (Vaswani et al., 2017). Even understanding the meaning of coordinate numbers in a description is sometimes difficult for LLMs.

To enable the generation of 2D designs from text, we build an instruction-based layout planning (InstLap) dataset to customize large language models. We then use InstLap to train our Text-based Layout Planning (TextLap) model through supervised fine-tuning of an LLM, enabling it to understand 2D spatial relationships and generate or modify bounding box coordinates based on text prompts. Figure 1 illustrates example inputs and outputs of the proposed TextLap model.

As an LLM, TextLap enables users to iteratively refine layout designs through natural language conversations. Initial text instructions provide an initial draft, and users can further customize their designs by interacting with the model. TextLap generates text-coherent layouts in response to users' iterative requests, enabling them to quickly find the most suitable custom template for their design. We believe that TextLap significantly reduces the time required to create designs, enhances design efficiency, and plays an important role in assisting

*Work done at University at Buffalo.
†Corresponding Author
[1]Data and code are available at:
https://github.com/puar-playground/TextLap

Findings of the Association for Computational Linguistics: EACL 2024, pages 14275–14289

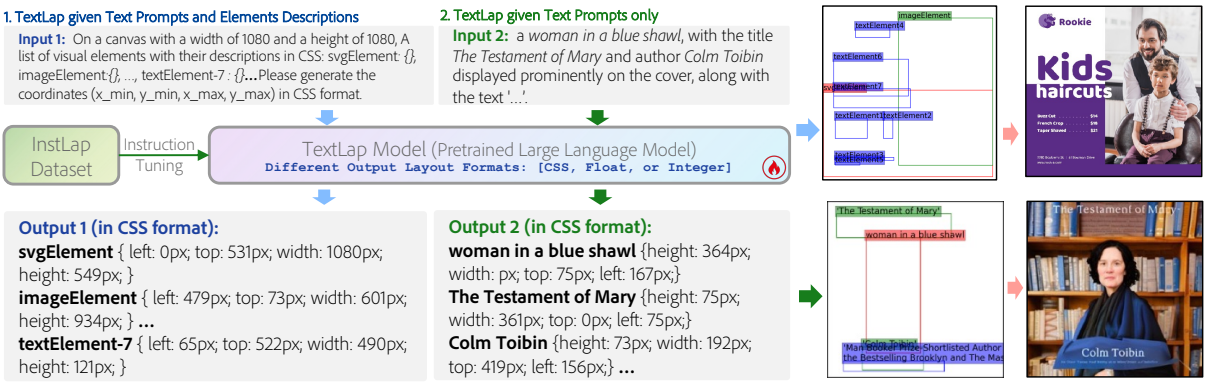


Figure 1: Overview of TextLap fine-tuned on InstLap. 1) TextLap can perform graphic designs and output coordinates given a list of elements including images, texts, and scalable vector graphics (SVG). The image is rendered accordingly. 2) TextLap can extract key elements from text prompts and provie their coordinates. The image can be rendered with image generation tools.

graphic designers. Moreover, TextLap also serves as a layout planning component for image generators, greatly enhancing the layout coherence of generated images according to text prompts. Our contributions are threefold:

- We proposed a novel text-to-layout tasks aiming to enhance design efficiency and designed evaluation dataset and protocals.
- We built an instruction tuning dataset named InstLap with human machine hybird annotations, enabling LLMs to perform text-to-layout planning.
- We trained a LLM-based layout planning model TextLap, and its empirical evaluations on benchmark datasets show its superior performance compared to GPT-4 models.

2 Related Work

Layout Generation Models Various approaches have been developed for controllable layout generation. Jiang et al. (2023); Arroyo et al. (2021); Gupta et al. (2021) utilize transformers to create auto-regressive models that produce layouts as a sequence of element attributes. Adversarial generative models (Kikuchi et al., 2021; Li et al., 2020) and diffusion models (Chen et al., 2024; Inoue et al., 2023; Zhang et al., 2023) have been used to enhance the quality of generation and unify various conditional generation tasks in a single model. TextDiffuser (Chen et al., 2023b) applies a layout transformer to generate layouts for keywords in text-rich images. (Jin et al., 2022; Hsu et al., 2023; Yu et al., 2022) generate background-aware layouts for posters by detecting smooth regions of the background image. (Yang et al., 2022) introduces a graph transformer architecture to generate layouts for images based on a scene graph that describes the relationship between visual objects. The visual layout serves as a crucial spatial control in image generation. GLIGEN (Li et al., 2023) integrates gated self-attention to enable spatial grounding capabilities in pre-trained diffusion models. Similarly, InstanceDiffusion (Wang et al., 2024) integrates locations and descriptions at the instance level into the generation process. TextLap is an LLM-based layout planning model that can be used to design various items, such as posters and book covers.

LLM-based Layout Generation Recent studies have introduced approaches based on language models to enable interactive layout generation and modification through chatting. TextDiffuser-2 (Chen et al., 2023a) fine-tunes a large language model to generate keywords for text rendering. (Yang et al., 2024) demonstrate that the integration of programming code in LLM training benefits the performance of LLM agents in various tasks. LayoutNUWA (Tang et al., 2023) leverages a large language model to generate layouts as HTML code. LayoutGPT (Feng et al., 2023) in-context visual demonstrations in CSS structures to enhance the visual planning skills of GPT-3.5/4 (Ouyang et al., 2022; Achiam et al., 2023) to plan layouts from text conditions. MuLan (Li et al., 2024) iteratively plans the layout of an image by breaking down the text prompt into a sequence of sub-tasks with an LLM, and then revises the image at each step based on feedback from a vision-language model (VLM). InstructScene (Lin and Mu, 2024) uses ChatGPT to filter the caption describing the layout elements. TextLap sets itself apart from prior research by delving into the text-to-layout task, which allows users to generate a layout design based on natural-language descriptions.

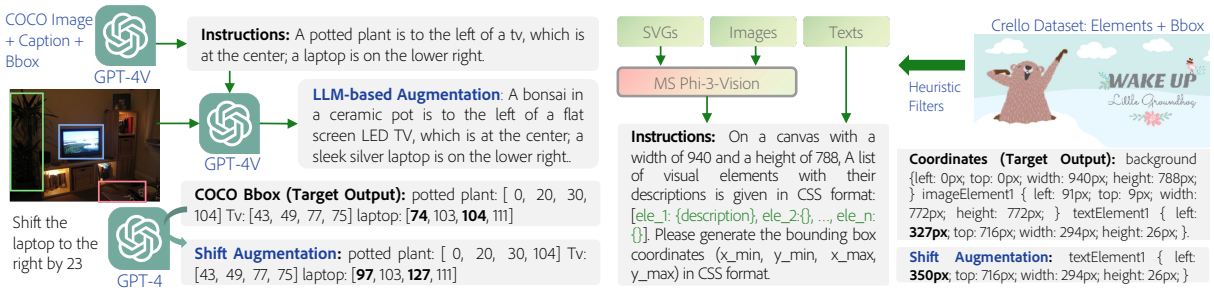


Figure 2: Overview of how to build the InstLap dataset. (a) shows how to build InstLap based on COCO dataset, which is composed of two data augmentations for input instructions and output layouts, respectively. (b) presents the how to incorporate Crello dataset into Instlap, where visual elements are first described by `Phi-3-Vision` and augmented into the text instructions.

3 Creating the InstLap Dataset

To customize a language model for layout planning, we developed the Instruction to Layout Planning (InstLap) dataset which comprises filtered and enhanced image-caption pairs from MS-COCO (2017) (Lin et al., 2014), a subset of the LAION datset (Schuhmann et al., 2022; Chen et al., 2023b), and Crelllo (Yamaguchi, 2021). Table 1 provides the statistics for the InstLap dataset, which includes human-annotated benchmarks, the original dataset, and augmented samples.

Domain	Task	Train	Test	Bench
Image	Layout Planning	27,246	1,255	–
	Keywords Aug.	22,364	1,039	–
	Layout Shift	22,364	–	–
Text	Layout Planning	4,873	–	502
	Text Split	4,873	–	–
Graphic	Layout Planning	16119	1954	–
	Layout Shift	16176	–	–

Table 1: Number of examples in each training and testing of the InstructLap datasets. Tasks refer to the original layout planning data processed and its different augmentation methods.

3.1 MS COCO Layout

We conducted data processing and filtering to remove abnormal samples, resulting in 27,246 text-to-layout pairs for layout planning tasks. We then generated textual descriptions for these layouts by prompting GPT-4V. we use different augmentation methods to obtain more instruction tuning data and split the dataset into training and testing for visual layout planning task.

Layout Normalization We standardized the size of the images in MS COCO by scaling the longest axis to a uniform length and centering it within a square canvas. This adjustment also involved recalibrating the bounding box coordinates to ensure that the scaling process did not distort the appearance of objects.

Object Selection In the annotations, small bounding boxes were discarded to simplify the layout. An area threshold $\tau_a = 0.1$ was applied to the largest box in an image to avoid layouts cluttered with numerous small objects. Furthermore, a threshold $\tau_l = 0.2$ was established for the maximum dimension (height or width) of all boxes in each image to remove background objects that are too small or irrelevant to the overall scene. These thresholds were manually selected.

Overlapping Filtering Overlapping bounding boxes were considered undesirable as they could introduce conflicting guidance in the same region, potentially leading to localized inaccuracies or "hallucinations" in the generated image. To address this issue, we calculated a pairwise matrix of Intersection over Union (IoU) scores for all bounding boxes within each image. A threshold $\tau_o = 0.01$ was selected as the maximum value to filter out images with significant overlaps of the boxes.

Crowd Filtering Images in MS COCO with the 'is_crowd' label in their annotations were removed. This label indicates cases where a bounding box encompasses multiple instances of the same class under a single label (*e.g.*, a densely populated area of people annotated with the singular label "person"). Such cases can potentially make LLMs confused when estimating the size of an individual.

3.2 Graphic Design Data

We also built InstLap based on Crello (Yamaguchi, 2021) for automatic graphic designs. Specifically, we first merge three types of elements into the background: i) tiny elements that occupy less than 1% of the canvas and ii) elements with more than 70% transparent pixels, which means that their shapes

are difficult to describe as bounding boxes. We then filter the dataset based on the number of remaining elements, discarding those with more than 10 elements resulting in 16,119 training layouts and 1,954 layouts for testing.

3.3 LLM-Assisted Annotation

Layout Prompt Generation We observed a significant inconsistency between object detection and caption annotations within the MS COCO dataset. Frequently, many objects either lack references in any of the five captions or are described using different terminology. For example, an object labeled as "person" in the annotation might be referred to as "woman" in the caption. As a result, caption annotations cannot reliably serve as prompts for layout generation. A detailed example is given in Appendix A.

To address this challenge, we capitalize on the observational and interpretative capabilities of large vision-language models, such as GPT-4V (Yang et al., 2023), LLaVA (Liu et al., 2023), and Phi-3-Vision (Abdin et al., 2024). We utilize these models to generate text prompts that describe given layouts, which consist of selected objects based on box annotations and the original image. To ensure consistency between the generated prompt and layout, we provide manually crafted examples to illustrate desired captions for specific layouts. Additionally, we instruct the model not to mention any objects in the image that are not included among the selected objects.

In the Crello dataset, the elements are too varied to be categorized or described using a predetermined set of labels. Consequently, we utilized the Phi-3-Vision model to create a brief description for every element in the dataset, including the background for content-aware graphic designs. In addition, we employed a heuristic method to create spatial descriptions for major elements in the design. All element captions, sizes, and spatial instructions are added in the prompt, providing contexts for LLM when generating vibrant layouts.

Object Augmentation The MS COCO dataset employs a set of labels of 80 classes, which would restrict the generalizability of LLM models. We utilize an LLM to augment the labels to facilitate open-set layout inference. Specifically, we provide carefully curated examples as context for GPT-4 (Achiam et al., 2023) and ask it to replace the single word tag of a box with a phrase that describes an object of comparable size and similar semantics, ensuring that the enhancement remains concise and natural.

Layout Shift Augmentation We enrich the training data with instruction-following pairs to promote the model's understanding of the relationship between bounding-box coordinates and directions. We guide the model in moving an object in a specified direction for a random distance. To prevent overlaps, we ask the model to move objects at the boundary further from adjacent objects, *e.g.*, moving the leftmost object further to the left. This strategy boosts the model's spatial awareness and ensures the integrity of the data.

3.4 Visual Text Layout Data

We have curated a collection of 4,873 layout prompts specifically tailored for visual text analysis. These prompts were generated from TextDiffuser-2 training data samples (Chen et al., 2023a), which include caption-OCR pairs derived from the MARIO-10M dataset. We developed two distinct prompts for each layout to improve TextLap's ability to adhere to user instructions for keyword splitting and to autonomously handle keyword segmentation. One prompt explicitly demonstrates how keywords are split, providing clear guidance for the process. The other prompt delegates the segmentation task to TextLap, allowing the model to autonomously determine the optimal segmentation approach based on the layout context.

InstaLap Bench We build InstLap Bench following the TRINS-Gen dataset (Zhang et al., 2024b). Considering the difficulty of rendering too many words in a single image, we run Paddle OCR and filter out images with more than 10 OCR words, resulting in a curated set of 502 images. We hired annotators from Upwork to write human annotations for each given image, which describe both textual and visual objects. Each annotator has to explicitly describe words and their positions within the image. To this end, we introduce InstLap-Bench, the visually-rich document design benchmark meticulously annotated by humans.

4 Model Training

We construct the TextLap model by fine-tuning the Vicuna-1.5 7B (Chiang et al., 2023) model using the FastChat framework (Zheng et al., 2024). The model is trained on the InstLap-train set with both

object layouts and text layouts. Figure 2 presents examples of conversations from the InstLap-train dataset. We train the model using both the closed-set layout prompt and the prompt enhanced with object augmentations. The training aims to enable the model to extract objects and separate visual-text elements when no specific target is provided. Additionally, the model is trained to focus on user-specified keywords or modification instructions. Training details are included in Appendix B.1.

We use multiple text formats for layout representation, including lists of integer/float numbers and CSS structures. However, we notice that multi-digit coordinates are often tokenized into multiple tokens, potentially complicating the model's numerical understanding. We integrated discrete coordinates as single tokens into the tokenizer to mitigate this.

5 Experimental Results

5.1 Experiments Setup

We evaluate our model's capability to generate layouts for objects and visual text using the InstLap-Test split. All TextLap varaints utilize the same set of weights trained on the InstLap-Train set. We first assess the model's performance in close-set object layout generation, which feature a defined label set of 80 classes, and an open-set scenario that include LLM-augmented object descriptions. In both experiments, we present prompts with and without specific objects to evaluate the TextLap on identifying key objects within the text prompts.

Then, we conduct an experiment using the Crello dataset (Yamaguchi, 2021) for graphic design given a list of elements, including images, texts, and SVGs.

In addition, we evaluate our model for identifying visual-text elements and creating visual-text layouts using the InstLap-Bench split. The prompts in this dataset are significantly more complex than the 5k visual-text training samples in the InstLap-Train set. All experiments are implemented with PyTorch and performed on Nvidia A100 GPUs.

5.2 Visual Layout Generation

While image generation has greatly benefited from the continual expansion of diffusion models, controllability remains a challenge, particularly in scenarios involving multiple user-imposed conditions or constraints. Current generative models like DALL·E 3 (Betker et al., 2023) and SDXL (Podell et al., 2023) still struggle to generate samples that satisfy all conditions. These challenges underscore the limitations of relying solely on larger models, prompting the need to incorporate additional layout planning components to better address the problem.

Evaluation Metrics To assess the quality of the generated content, we employ six computational metrics: (1) Fréchet Inception Distance (FID) (Heusel et al., 2017): Assesses the similarity between distributions of generated layouts and validated layouts, capturing both the variety and fidelity of the generated layouts. We modified the network structure to adapt the open-set layout. The model is introduced in Appendix B.2. (2) MaxIoU (Maximum Intersection-over-Union): this metric assesses the overlap between the generated and target layouts by calculating the average IoU of the optimally matched element pairs. We adapt the original definition by (Kikuchi et al., 2021), focusing on matching layouts generated from the same prompt rather than from the same set of labels. For object layouts, we find the best-matched box among the boxes that share the same label. Finding exact matches is challenging for visual-text layouts, where box labels are segments of phrases that may be split differently. Therefore, we define the closest match using the cosine similarity of CLIP text features. (3) Failure rate: the proportion of layouts generated by the LLM in an invalid format that cannot be processed automatically. (4) Precision: Accuracy of correctly identified elements within all extracted elements. (5) Recall: The ratio of correctly identified elements within all ground truth elements. (6) F-score: The harmonic mean of Precision and Recall, balancing their contributions.

Settings To evaluate the effectiveness of text-to-layout models, we use text-to-image generation as a downstream task. We compare our method, TextLap, with LayoutGPT (Feng et al., 2023) and GPT-4-based baselines for close-set and open-set text-guided layout generation. The naive baseline is termed GPT-4, where we ask GPT-4 to generate a layout for a given prompt with three fixed examples to demonstrate the desired output format. We introduce two additional baselines, GPT-4 (R) and GPT-4 (rCSS), for our experiments designed to enhance in-context learning for the GPT-4 model. These baselines employ cosine similarity of CLIP text features to identify and retrieve the most relevant demonstration exam-

Methods	Text prompts with target objects			
	FID ↓	MaxIoU ↑	Fail % ↓	F-score ↑
GPT-4	382.0	0.292	0.956	0.989
GPT-4 (R)	26.40	0.452	1.116	0.979
GPT-4 (rCSS)	37.45	0.459	1.116	0.969
LayoutGPT	248.0	0.435	**0.000**	0.959
TextLap-S128	18.64	0.454	2.151	0.974
TextLap-D128	**13.54**	**0.475**	0.398	0.983
TextLap-D1024	14.43	0.456	0.398	0.973
TextLap-Float	15.09	**0.475**	0.159	0.996
TextLap-CSS	19.32	0.458	**0.000**	**0.998**
	Text prompts only			
GPT-4	504.0	0.005	98.566	0.012
GPT-4 (R)	30.01	0.417	4.382	0.856
GPT-4 (rCSS)	39.17	0.427	1.594	0.867
LayoutGPT	265.7	0.416	**0.000**	0.900
TextLap-S128	20.67	**0.452**	1.116	0.976
TextLap-D128	14.77	0.267	25.976	0.537
TextLap-D1024	16.27	0.347	9.163	0.716
TextLap-Float	**14.36**	0.424	3.426	0.877
TextLap-CSS	18.69	0.440	0.080	**0.979**

Table 2: Comparative results of close-set layout generation with 80-class COCO labels

Methods	Text prompts with target objects			
	FID ↓	MaxIoU ↑	Fail % ↓	F-score ↑
GPT-4	291.8	0.330	1.347	0.979
GPT-4 (R)	48.59	0.421	1.925	0.974
GPT-4 (rCSS)	41.73	0.456	0.481	0.972
LayoutGPT	261.4	0.383	**0.000**	0.824
TextLap-D128	**17.77**	**0.485**	1.059	0.972
TextLap-D1024	21.18	0.467	0.385	0.949
TextLap-Float	18.16	0.456	0.481	0.937
TextLap-CSS	19.48	0.464	**0.000**	**0.996**
	Text prompts only			
GPT-4	398.4	0.006	95.091	0.011
GPT-4 (R)	78.03	0.155	9.047	0.315
GPT-4 (rCSS)	88.66	0.093	2.406	0.197
LayoutGPT	269.4	0.326	**0.000**	0.712
TextLap-D128	**16.38**	0.322	18.383	0.642
TextLap-D1024	18.34	0.349	10.298	0.714
TextLap-Float	16.67	0.309	3.272	0.633
TextLap-CSS	19.44	**0.446**	0.192	**0.960**

Table 3: Results on open-set layout generation with LLM-augmented labels

ples from the InstLap-Train dataset for each test prompt (Feng et al., 2023). They differ in their approach to representing layout coordinates: GPT-4 (R) uses lists of integers, while GPT-4 (rCSS) adopts a CSS-like structure with a maximum value of 128. We finetuned four TextLap variants with different coordinate representations: discrete coordinates on 128×128 and 1024×1024 canvases (TextLap-D128, TextLap-D1024), floating-point coordinates (TextLap-Float), and a CSS format with a max value of 128 (TextLap-CSS). Additionally, we trained a model with 128 new tokens for discrete coordinates, called TextLap-S128.

RQ1: Does simply scale up the LLM help for design problems? Table 2 presents the results of TextLap models alongside three GPT-4-based methods. In particular, TextLap models achieve significantly lower layout FID scores compared to GPT-4 baselines, highlighting the advantages of fine-tuning. In the open-set generation results, presented in Table 3, there is a notable decrease in the retrieval scores for GPT-4 baselines when objects are not specified in the prompt. This decline could be attributed to the fact that, in close-set experiments, the retrieval process often identifies examples with matching objects, implicitly providing the necessary keywords in context. This shows that task-specific models can potentially outperform larger models and simple scaling-up does not help for design problems.

RQ2: What is the impact of different layout representations? Among various TextLap variants, TextLap-CSS stands out, outperforming all baselines by a considerable margin and demonstrating high stability in object extraction. TextLap-S128 showed inferior performance, likely due to inadequate training data to effectively integrate new weights. GPT-4 (rCSS) also exhibits the lowest failure rates among the baseline methods, consistent with insights from previous studies by (Yang et al., 2024; Feng et al., 2023; Tang et al., 2023), which suggest that pre-trained large language models better understand programming patterns, possibly due to exposure to code snippets in their training data.

RQ3: Are special tokens of coordinates useful? TextLap-S128, which incorporates special coordinate tokens, performs well on closed-set data, with a lower failure rate in text prompt-only settings compared to TextLap-D128 and TextLap-Float. However, it struggles in open-set generation, likely due to the large number of new parameters requiring more training data and time. As shown in the loss curve in Figure 3, TextLap-S128 had higher loss after the same number of epochs and converged to a slightly higher loss even with extended training, indicating a need for more training to achieve similar performance. Fine-tuning existing tokens, which are well-trained in LLMs and handle arithmetic effectively, proves more efficient. Thus, adding special tokens is less effective than a CSS-based layout representation, especially for open-set tasks, where large-scale pre-training may help (Lv et al., 2023).

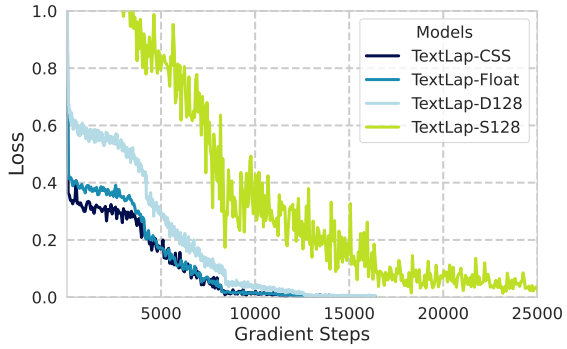


Figure 3: Loss curve on close-set layout generation with 80-class COCO labels.

5.3 Visual Text Layout Generation

We evaluate the ability of our models in generating layouts for visual-text using the InstLap-Bench set. Specifically, we compare TextLap-Float and TextLap-CSS with GPT-4 and GPT-4 (rCSS). Additionally, we benchmark our models against the large language model (LLM) employed in TextDiffuser2, which is fine-tuned on the same 5,000 visual-text layout training samples, utilizing the Vicuna-v1.5 7B checkpoint.

Methods	MaxIoU (suc) ↑	MaxIoU ↑	Fail % ↓
GPT-4	0.231	0.206	10.778
GPT-4 (rCSS)	0.252	0.241	4.192
TextDiffuser-2	0.166	0.165	0.80
TextLap-Float	0.209	0.096	54.09
TextLap-CSS	0.211	0.211	0.00

Table 4: Results on text layout generation on the InstLap-Bench.

Table 4 shows the results comparion between TextLap and GPT-4. Given the high failure rates of GPT-4 and TextLap-Float, we also present MaxIoU (suc), which represents the average MaxIoU across all successful generations. In particular, TextLap-CSS achieves the lowest failure rate and significantly outperforms TextDiffuser2 in MaxIoU. However, it fails to outperform the GPT-4 baselines. This limitation is likely attributed to insufficient training data, stemming from the limited number of visual-text samples in the InstLap-train dataset and the inconsistent distribution of layouts between InstLap-train and InstLap-Bench.

5.4 Automatic Graphic Designs

We conducted an experiment on content-aware graphic designs using the Crello dataset, where canvas sizes varies for each design. For CSS format layouts with integer coordinates, we include the canvas size in the prompt. We also normalize box coordinates to decimal values between 0 and 1 ("Float") and write layouts as JSON to leverage the model's code understanding. Figure 4 illustrates a generated example using CSS forat, where TextLap provides coordinates for each element given a text instruction and a list of elements.

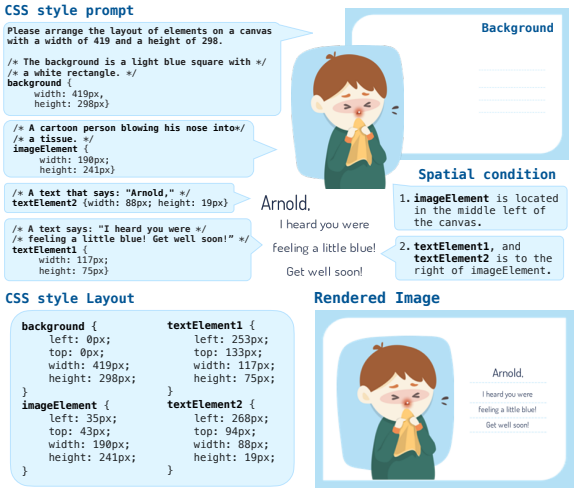


Figure 4: An example from InstLap that is built based on the Crello dataset.

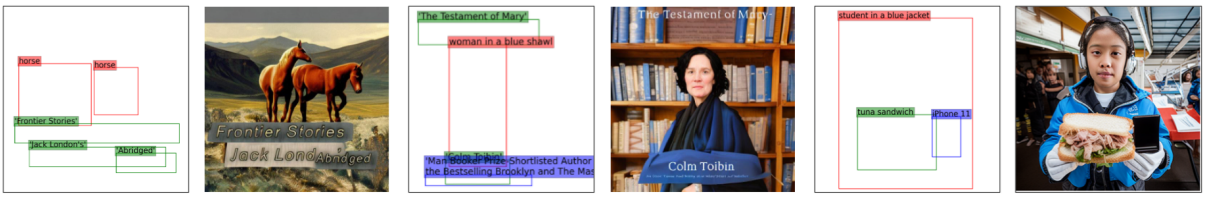
	MaxIoU↑	Precision↑	Recall↑	F-score↑
GPT-4 (CSS)	0.190	0.874	0.257	0.364
GPT-4 (rCSS)	0.209	0.934,	0.296,	0.406
GPT-4 (Float)	0.457	0.980	0.980	0.980
GPT-4 (rFloat)	0.440	0.984	0.984	0.984
TextLap-CSS	0.407	0.998	0.986	0.990
TextLap-Float	0.535	1.000	1.000	1.000

Table 5: Results on automatic graphic design on Crello.

We compare TextLap with four GPT-4 baselines that generate CSS and JSON-style layouts with both integer and float coordinates. SBERT retrieves demonstration layouts for the GPT-4 (rCSS, rFloat) baselines. Table 5, shows TextLap significantly outperforms its GPT-4 counterpart, demonstrating the effectiveness of instruction fine-tuning. Additionally, methods using float coordinates outperform those using integer coordinates, likely due to the complexity of adapting to different canvas sizes.

5.5 Impact of Layout on Text-to-Image Generation

We evaluated the effect of layout guidance on text-to-image generation using the COCO dataset. The experiment compared three models: Stable Diffusion 1.5, which serves as the layout-free baseline; TextLap + InstanceDiffusion (Wang et al., 2024),



Prompts: two horses in a mountain area with a title 'Frontier Stories' and author 'Jack London's' name, and a logo 'Abridged' at the bottom right.

Prompts: a woman in a blue shawl, with the title 'The Testament of Mary' and author 'Colm Toibin' displayed prominently on the cover, along with the text 'Man Booker Prize-Shortlisted Author of the Bestselling Brooklyn and The Master'.

Prompts: a student in a blue jacket is holding a tuna sandwich with their left hand and an iPhone 11 with their right hand.

Figure 5: Generated visual and textual layout planning examples. Layouts are provided by TextLap given text prompts and images are rendered by ARTIST (Zhang et al., 2024a) and InstanceDiffusion (Wang et al., 2024) respectively.

Method	Image FID	GPT-Preference
SD 1.5	58.09	-
InstDiff (TextLap)	58.92	72%
InstDiff (True Layout)	58.39	76%

Table 6: Text-to-image generation results on the COCO dataset. Layout-guided models are compared to the layout-free baseline, Stable Diffusion (SD) 1.5.

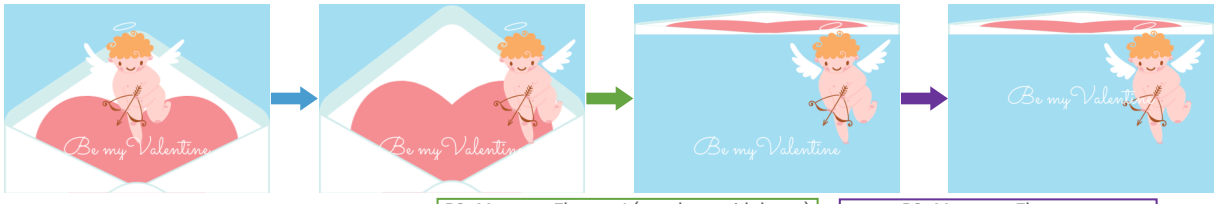
where layouts are generated by our TextLap model to guide InstanceDiffusion; and True Layout + InstanceDiffusion, where the actual COCO layout is used for guidance. The models were evaluated using two metrics. The first metric, Image FID, measures the quality of the generated images by comparing them to real images from the COCO dataset, with lower FID scores indicating better quality. The second metric, GPT-Preference, involves presenting three images and the text caption to GPT-4o: two images generated by a test model and Stable Diffusion 1.5 using the same prompt, and the real image used to generate the text caption of the layout. GPT-4o compares the generated images to determine which has a more similar layout to the real image and better coherence with the text caption. The results, shown in Table 6, indicate that layout guidance slightly impacts FID but improves text-image alignment. TextLap + InstanceDiffusion is preferred by GPT-4o 72% of the time, while the true layout model achieves 76%.

5.6 Qualitative Evaluation

Examples of Object Layout Generation We compare generated layouts between GPT-4 (rCSS) and InstLap-CSS and further evaluate them based on images rendered by InstanceDiffusion (Wang et al., 2024), as shown in Figure D.1. Both GPT-4 and InstLap can generate layouts that follow the constraints from the prompts. The advantages of InstLap include: (i) generated layouts can simplify the process of creating visually appealing images, indicating that these layouts are of high quality and more suitable for open-source rendering engines, as reflected in evaluation metrics such as MaxIOU; and (ii) InstLap offers customization based on user needs through instruction tuning, with a model size significantly smaller than GPT-4. Examples of text layout generation are provided in Appendix F.

Emergent Visual-Text Layout Generation As shown in Table 1, TextLap has been fine-tuned for text layout designs in TextDiffuser-2 (Chen et al., 2023a) and visual object designs on InstLap based on MS COCO 2017 (Caesar et al., 2018). It shows surprising generalization ability and emergent visual-text joint layout planning ability. The

Please arrange layout elements with given descriptions:
[{element_name: background, caption: The background is a solid light blue color., width: 1.0, height: 1.0}, {element_name: svgElement1, caption: The image shows a stylized representation of a heart within a geometric frame. The heart is filled with a solid color, and the frame is outlined in a contrasting color., width: 1.0, height: 0.866}, {element_name: svgElement2, caption: The image is a cartoon illustration of a cherub with curly hair, a pink body, and a brown bow and arrow., width: 0.382, height: 0.648}, {element_name: textElement, caption: A text that says: Be myValentine, width: 0.656, height: 0.128}]
Please ensure the following relationship: svgElement1 is located in the middle center of the canvas.



R1: Move svgElement2 (angel) to the right

R2: Move svgElement1 (envelope with heart) to the top edge of the canvas

R3: Move textElement ("Be my Valentine") upward by 80 px

Figure 6: Examples of interactive layout design, where user gives instructions to TextLap.

great generalizability comes from a carefully designed dataset building strategy as described in Section 3. Figure 5 shows examples of visual-text layout generation, and more examples can be found in Appendix F.

Interactive Layout Design Figure 6 and E.1 shows an example of an interactive layout design, where the user can comment on existing layout designs, and TextLap can generate a new layout, fulfilling user requests. This is another emergent ability, as TextLap is never fine-tuned on conversational data.

6 Conclusion

This study addresses the text-to-layout task by creating an instructional dataset built upon available resources with the assistance of GPT-4v to fine-tune a large language model for layout planning. The fine-tuned model, TextLap, outperforms GPT-4 in object layout planning and can generate layouts with both text and object when trained on text-only and object-only layouts. TextLap provides a framework to addressing real-world graphic design challenges by building instruction-following layout datasets. It is desired if the image rendering models can be finetuned or jointly trained with InstLap. However, it is beyond the scope of this paper as InstLap aims to unveiling the graphic design ability of large language models.

7 Limitations

The limitations of the paper are caused by the design of the dataset and the model architecture. InstLap is a dataset created with complex heuristics and the help of large language models. There is still a quality gap between InstLap and high-quality human annotations. However, InstLap is much cheaper to build and can serve as a pioneer dataset to quickly verify whether language models can perform automatic graphic designs. TextLap uses the standard LLM architecture, and a special design with 2D spatial embedding layers should provide better performance. It is very expensive to pretrain such a model and is beyond the scope of this paper.

8 Ethics Statement

Our paper introduces a new instruction tuning dataset, which acknowledges the potential ethical implications inherent in using large language models for such applications.We have taken comprehensive steps to ensure that our research adheres to the highest ethical standards, particularly with respect to data privacy and responsible use of AI. This ethics statement reflects our dedication to conducting responsible research and our commitment to advancing the field of AI in a manner that respects individual privacy rights and promotes the ethical use of technology.

Acknowledgments

This work is partially supported by NSF AI Institute-2229873, NSF RI-2223292, NSF III-1747614, an Amazon research award, and an Adobe gift fund. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation, the Institute of Education Sciences, or the U.S. Department of Education.

References

Marah Abdin, Sam Ade Jacobs, Ammar Ahmad Awan, Jyoti Aneja, Ahmed Awadallah, Hany Awadalla, Nguyen Bach, Amit Bahree, Arash Bakhtiari, Harkirat Behl, et al. 2024. Phi-3 technical report: A highly capable language model locally on your phone. *arXiv preprint arXiv:2404.14219*.

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Diego Martin Arroyo, Janis Postels, and Federico Tombari. 2021. Variational transformer networks for layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13642–13652.

James Betker, Gabriel Goh, Li Jing, Tim Brooks, Jianfeng Wang, Linjie Li, Long Ouyang, Juntang Zhuang, Joyce Lee, Yufei Guo, et al. 2023. Improving image generation with better captions. *Computer Science. https://cdn. openai. com/papers/dall-e-3. pdf*, 2(3):8.

Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. 2018. Coco-stuff: Thing and stuff classes in context. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1209–1218.

Jian Chen, Ruiyi Zhang, Yufan Zhou, and Changyou Chen. 2024. Towards aligned layout generation via diffusion model with aesthetic constraints. In *The Twelfth International Conference on Learning Representations*.

Jingye Chen, Yupan Huang, Tengchao Lv, Lei Cui, Qifeng Chen, and Furu Wei. 2023a. Textdiffuser-2: Unleashing the power of language models for text rendering. *arXiv preprint arXiv:2311.16465*.

Jingye Chen, Yupan Huang, Tengchao Lv, Lei Cui, Qifeng Chen, and Furu Wei. 2023b. Textdiffuser: Diffusion models as text painters. *arXiv preprint arXiv:2305.10855*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Weixi Feng, Wanrong Zhu, Tsu-jui Fu, Varun Jampani, Arjun Akula, Xuehai He, Sugato Basu, Xin Eric Wang, and William Yang Wang. 2023. Layoutgpt: Compositional visual planning and generation with large language models. *arXiv preprint arXiv:2305.15393*.

Kamal Gupta, Justin Lazarow, Alessandro Achille, Larry S Davis, Vijay Mahadevan, and Abhinav Shrivastava. 2021. Layouttransformer: Layout generation and completion with self-attention. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1004–1014.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. 2017. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Hsiao Yuan Hsu, Xiangteng He, Yuxin Peng, Hao Kong, and Qing 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*, pages 6018–6026.

Naoto Inoue, Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2023. Layoutdm: Discrete diffusion model for controllable layout generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10167–10176.

Zhaoyun Jiang, Jiaqi Guo, Shizhao Sun, Huayu Deng, Zhongkai Wu, Vuksan Mijovic, Zijiang James Yang, Jian-Guang Lou, and Dongmei Zhang. 2023. Layoutformer++: Conditional graphic layout generation via constraint serialization and decoding space restriction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18403–18412.

Chuhao Jin, Hongteng Xu, Ruihua Song, and Zhiwu Lu. 2022. Text2poster: Laying out stylized texts on retrieved images. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4823–4827. IEEE.

Akash Abdu Jyothi, Thibaut Durand, Jiawei He, Leonid Sigal, and Greg Mori. 2019. Layoutvae: Stochastic scene layout generation from a label set. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9895–9904.

Kotaro Kikuchi, Edgar Simo-Serra, Mayu Otani, and Kota Yamaguchi. 2021. Constrained graphic layout generation via latent optimization. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 88–96.

Jianan Li, Jimei Yang, Jianming Zhang, Chang Liu, Christina Wang, and Tingfa Xu. 2020. Attribute-conditioned layout gan for automatic graphic design. *IEEE Transactions on Visualization and Computer Graphics*, 27(10):4039–4048.

Sen Li, Ruochen Wang, Cho-Jui Hsieh, Minhao Cheng, and Tianyi Zhou. 2024. Mulan: Multimodal-llm agent for progressive multi-object diffusion. *arXiv preprint arXiv:2402.12741*.

Yuheng Li, Haotian Liu, Qingyang Wu, Fangzhou Mu, Jianwei Yang, Jianfeng Gao, Chunyuan Li, and Yong Jae Lee. 2023. Gligen: Open-set grounded text-to-image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22511–22521.

Chenguo Lin and Yadong Mu. 2024. Instructscene: Instruction-driven 3d indoor scene synthesis with semantic graph prior. *arXiv preprint arXiv:2402.04717*.

Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023. Visual instruction tuning.

Tengchao Lv, Yupan Huang, Jingye Chen, Lei Cui, Shuming Ma, Yaoyao Chang, Shaohan Huang, Wenhui Wang, Li Dong, Weiyao Luo, et al. 2023. Kosmos-2.5: A multimodal literate model. *arXiv preprint arXiv:2309.11419*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

William Peebles and Saining Xie. 2023. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4195–4205.

Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. 2023. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR.

Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. 2022. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in Neural Information Processing Systems*, 35:25278–25294.

Zecheng Tang, Chenfei Wu, Juntao Li, and Nan Duan. 2023. Layoutnuwa: Revealing the hidden layout expertise of large language models. *arXiv preprint arXiv:2309.09506*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xudong Wang, Trevor Darrell, Sai Saketh Rambhatla, Rohit Girdhar, and Ishan Misra. 2024. Instancediffusion: Instance-level control for image generation. *arXiv preprint arXiv:2402.03290*.

Yiheng Xu, Minghao Li, Lei Cui, Shaohan Huang, Furu Wei, and Ming Zhou. 2020. Layoutlm: Pre-training of text and layout for document image understanding. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1192–1200.

Kota Yamaguchi. 2021. Canvasvae: Learning to generate vector graphic documents. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5481–5489.

Chiao-An Yang, Cheng-Yo Tan, Wan-Cyuan Fan, Cheng-Fu Yang, Meng-Lin Wu, and Yu-Chiang Frank Wang. 2022. Scene graph expansion for semantics-guided image outpainting. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15617–15626.

Ke Yang, Jiateng Liu, John Wu, Chaoqi Yang, Yi R Fung, Sha Li, Zixuan Huang, Xu Cao, Xingyao Wang, Yiquan Wang, et al. 2024. If llm is the wizard, then code is the wand: A survey on how code empowers large language models to serve as intelligent agents. *arXiv preprint arXiv:2401.00812*.

Zhengyuan Yang, Linjie Li, Kevin Lin, Jianfeng Wang, Chung-Ching Lin, Zicheng Liu, and Lijuan Wang. 2023. The dawn of lmms: Preliminary explorations with gpt-4v (ision). *arXiv preprint arXiv:2309.17421*, 9(1):1.

Ning Yu, Chia-Chih Chen, Zeyuan Chen, Rui Meng, Gang Wu, Paul Josel, Juan Carlos Niebles, Caiming Xiong, and Ran Xu. 2022. Layoutdetr: detection transformer is a good multimodal layout designer. *arXiv preprint arXiv:2212.09877*.

Jianyi Zhang, Yufan Zhou, Jiuxiang Gu, Curtis Wigington, Tong Yu, Yiran Chen, Tong Sun, and Ruiyi Zhang. 2024a. Artist: Improving the generation of text-rich images by disentanglement. *arXiv preprint arXiv:2406.12044*.

Junyi Zhang, Jiaqi Guo, Shizhao Sun, Jian-Guang Lou, and Dongmei Zhang. 2023. Layoutdiffusion: Improving graphic layout generation by discrete diffusion probabilistic models. *arXiv preprint arXiv:2303.11589*.

Ruiyi Zhang, Yanzhe Zhang, Jian Chen, Yufan Zhou, Jiuxiang Gu, Changyou Chen, Nedim Lipka, and Tong Sun. 2024b. Trins: Towards multimodal language models that can read. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2024. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36.

Appendix

A Comparison between Annotations in COCO and InstLap

The key objective behind the development of the InstLap dataset is to ensure consistency in spatial relationships and object naming between captions and bounding box annotations. This alignment is critical for accurately mapping visual elements to descriptive captions, which is particularly important for tasks involving layout understanding.

To highlight the difference between datasets, consider the following five human-generated captions from the COCO dataset:

- "A picture of a dog laying on the ground."
- "Dog snoozing by a bike on the edge of a cobblestone street."
- "The white dog lays next to the bicycle on the sidewalk."
- "A white dog is sleeping on a street and a bicycle."
- "A puppy rests on the street next to a bicycle."

While these captions provide descriptive information, they lack the consistency between annotations. In contrast, the InstLap dataset provides coherent and detailed captions with precise layout annotations, enabling a tighter alignment between descriptions, spatial relationships, and object details:

- **Close-set caption:** "A dog is lying on the ground on the right with a bicycle parked to the left."
  Objects: [bicycle, dog]
- **Open-set caption:** "A sleepy labrador is lying on the ground on the right with a cherry red bicycle parked to the left."
  Objects: [cherry red bicycle, sleepy labrador]

B Implementation Detail

B.1 Training Detail

All models are trained using eight NVIDIA A100 80GB GPUs. We use the default configuration load from the pre-trained Vicuna model. In fine-tuning, we use a cosine annealing schedule with an initial learning rate of 2e-5 and a batch size of 32. This set of hyperparameters is adopted across all checkpoints.

B.2 Open-Set Layout Encoder for FID Score Calculation

The FID score evaluates the quality of generated data by measuring the distance between feature vectors of real and generated samples, which requests a feature encoder that captures the position and semantic details of layout elements in a single vector. Adapting from the close-set approach by (Kikuchi et al., 2021), we develop our feature encoder for open-set layouts. The model uses a transformer-based encoder-decoder backbone, which encodes a sequence of layout elements to a feature vector and uses the decoder to reconstruct the input sequence. The feature is also trained to discriminate between clean and noisy layouts. Specifically, each element $\mathbf{x} = [\mathbf{b}, \text{CLIP}(s)]$ is defined as a four dimension bounding box $\mathbf{b}$ consisting of the left, top, right, bottom coordinates and the CLIP (Radford et al., 2021) text feature of the phrase s of the element. The model is trained by three losses: A binary cross-entropy loss as the discrimination loss and a reconstruction loss consisting of a mean cosine distance loss for CLIP features and a mean square error loss for bounding boxes.

C Additional Quantitative Results

Table C.2 presents the results that combine open-set and close-set prompts, where TextLap models exhibit similar performance. This consolidation of results further underscores the consistent performance of TextLap models, showing that smaller LLMs customized for specific tasks can beat huge general LLMs, such as GPT4V.

Testing Methods	Captions with target objects					
	FID ↓	MaxIoU ↑	Fail % ↓	Precision ↑	Recall ↑	F-score ↑
GPT-4	382.0	0.292	0.956	0.989	0.989	0.989
GPT-4(r)	26.40	0.452	1.116	0.979	0.980	0.979
GPT-4(rCSS)	37.45	0.459	1.116	0.969	0.971	0.969
TextLap-S128	18.64	0.454	2.151	0.972	0.977	0.974
TextLap-D128	**13.54**	**0.475**	0.398	0.995	0.979	0.983
TextLap-D1024	14.43	0.456	0.398	0.993	0.966	0.973
TextLap-F	15.09	**0.475**	0.159	0.996	0.997	0.996
TextLap-CSS	19.32	0.458	**0.000**	**0.998**	**1.000**	**0.998**
	Captions only					
GPT-4	504.0	0.005	98.566	0.012	0.012	0.012
GPT-4(r)	30.01	0.417	4.382	0.838	0.900	0.856
GPT-4(rCSS)	39.17	0.427	1.594	0.842	0.925	0.867
TextLap-S128	20.67	**0.452**	1.116	0.977	0.980	0.976
TextLap-D128	14.77	0.267	25.976	0.538	0.540	0.537
TextLap-D1024	16.27	0.347	9.163	0.718	0.719	0.716
TextLap-F	**14.36**	0.424	3.426	0.878	0.882	0.877
TextLap-CSS	18.69	0.440	**0.080**	**0.983**	**0.981**	**0.979**

Table C.1: Comparative results of close-set layout generation with 80-class COCO labels

Testing Methods	Caption with target objects					
	FID ↓	MaxIoU ↑	Fail % ↓	Precision ↑	Recall ↑	F-score ↑
GPT-4	334.3	0.309	1.133	0.985	0.984	0.984
GPT-4(r)	27.99	0.438	1.482	0.977	0.978	0.977
GPT-4(rCSS)	32.54	0.458	0.828	0.970	0.972	0.971
TextLap-D128	**8.49**	**0.479**	0.697	0.991	0.973	0.978
TextLap-D1024	10.52	0.461	0.392	0.983	0.953	0.962
TextLap-F	9.96	0.466	0.305	0.969	0.970	0.969
TextLap-CSS	12.53	0.461	**0.000**	**0.997**	**0.999**	**0.997**
	Caption only					
GPT-4	394.1	0.005	96.992	0.011	0.013	0.011
GPT-4(r)	43.07	0.298	6.495	0.589	0.664	0.611
GPT-4(rCSS)	53.38	0.276	1.962	0.543	0.610	0.563
TextLap-D128	**8.46**	0.292	22.537	0.587	0.587	0.585
TextLap-D1024	10.75	0.348	9.677	0.718	0.717	0.715
TextLap-F	9.74	0.372	3.357	0.768	0.771	0.767
TextLap-CSS	12.46	**0.443**	**0.131**	**0.974**	**0.972**	**0.970**

Table C.2: Comparative results on layout generation with both close-set and open-set labels.

D Qualitative Comparison with GPT-4 Results

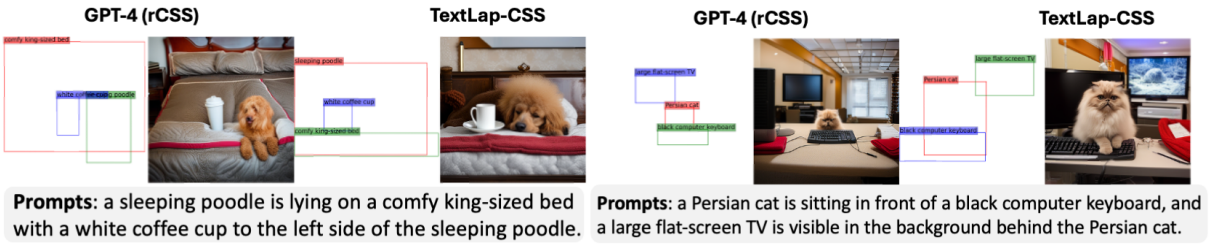


Figure D.1: Comparison between TextLap-CSS and GPT-4 (rCSS). Images are rendered by InstanceDiffusion (Wang et al., 2024) given layouts.

E Examples of Interactive Editing

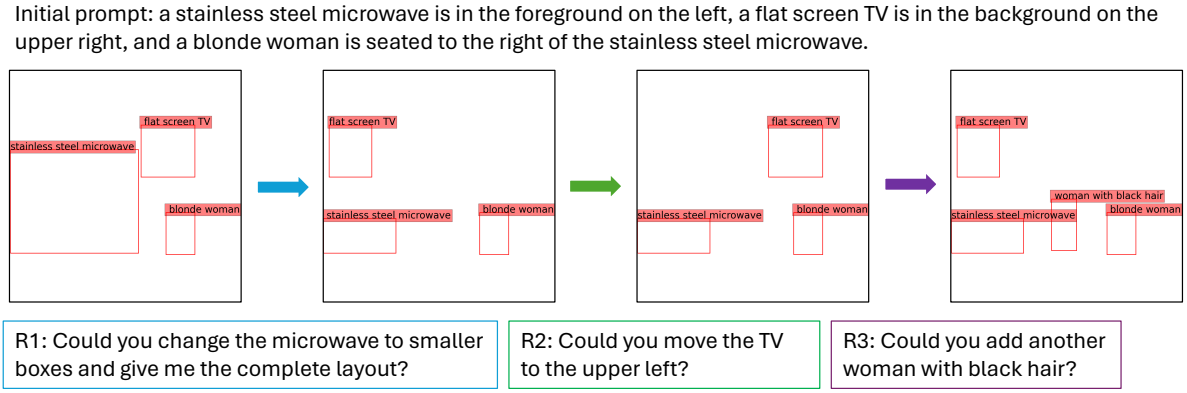


Figure E.1: Examples of interactive layout design on COCO dataset, where user gives instructions to TextLap.

F Examples of Text Layout Generation

Figure F.1 shows examples of text and visual layout generation, where InstCap can detect keywords, objects and plan their locations within the image.

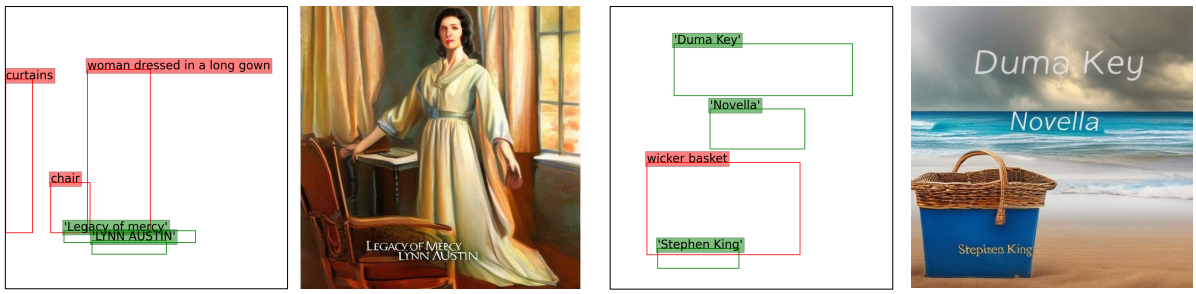
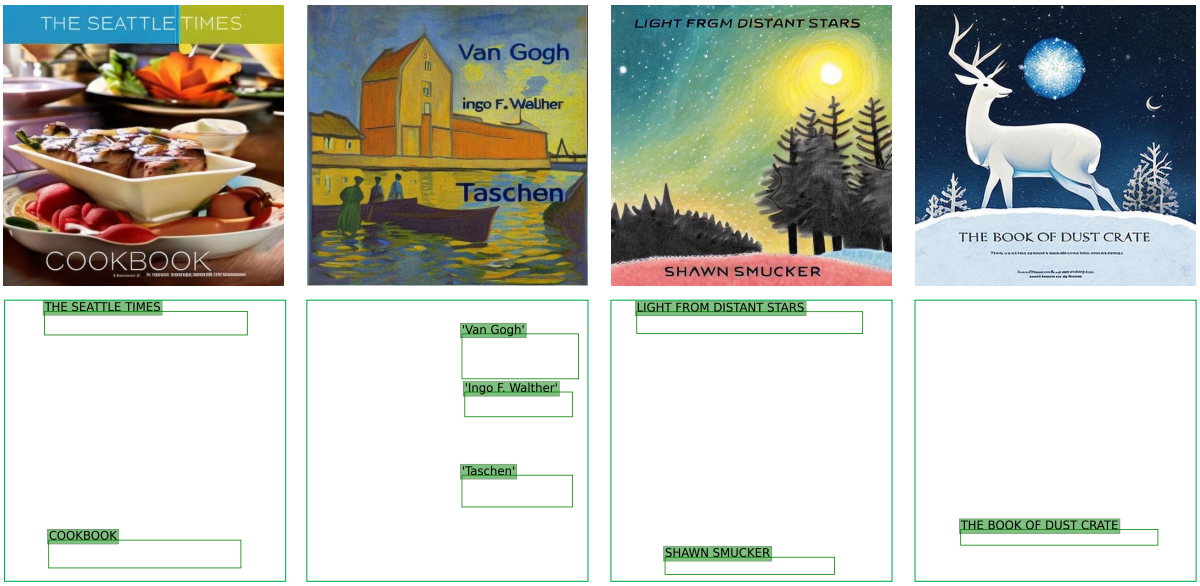


Figure F.1: Examples of generated layout including both visual and textual elements by TextLap. Green boxes are text elements and Red boxes are visual elements of the generated layouts.



THE SEATTLE TIMES COOKBOOK' is the major text and there are various cooking ingredients in the background.

the title 'LIGHT FROM DISTANT STARS' and the author name 'SHAWN SMUCKER' are displayed on the book cover.

an impressionist painting of a drawbridge and a sunken canoe in the center, with the title 'Van Gogh' and author 'Ingo F. Walther' on top, and 'Taschen' at the bottom.

a big white deer on the left, with a snowy and starry sky. 'THE BOOK OF DUST CRATE' is the title of the special edition book.

Figure F.2: Examples of designing text layouts where visual elements are generated based on the prompts.